3G Handy Guide:

Building Web Sites



BIBLIOTEX
Digital Library

3G HANDY GUIDE:
BUILDING WEB SITES



www.bibliotex.com

3G HANDY GUIDE: BUILDING WEB SITES



www.bibliotex.com
email: info@bibliotex.com

e-book Edition 2020

ISBN: 978-1-98464-157-1 (e-book)

This book contains information obtained from highly regarded resources. Reprinted material sources are indicated. Copyright for individual articles remains with the authors as indicated and published under Creative Commons License. A Wide variety of references are listed. Reasonable efforts have been made to publish reliable data and views articulated in the chapters are those of the individual contributors, and not necessarily those of the editors or publishers. Editors or publishers are not responsible for the accuracy of the information in the published chapters or consequences of their use. The publisher assumes no responsibility for any damage or grievance to the persons or property arising out of the use of any materials, instructions, methods or thoughts in the book. The editors and the publisher have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission has not been obtained. If any copyright holder has not been acknowledged, please write to us so we may rectify.

Notice: Registered trademark of products or corporate names are used only for explanation and identification without intent of infringement.

© 2020 3G E-learning LLC

In Collaboration with 3G E-Learning LLC. Originally Published in printed book format by 3G E-Learning LLC with ISBN 978-1-98463-778-9

EDITORIAL BOARD



Aleksandar Mratinković was born on May 5, 1988. in Arandjelovac, Serbia. He has graduated on Economic high school (2007), The College of Tourism in Belgrade (2013), and also has a master degree of Psychology (Faculty of Philosophy, University of Novi Sad). He has been engaged in different fields of psychology (Developmental Psychology, Clinical Psychology, Educational Psychology and Industrial Psychology) and has published several scientific works.



Dan Piestun (PhD) is currently a startup entrepreneur in Israel working on the interface of Agriculture and Biomedical Sciences and was formerly president-CEO of the National Institute of Agricultural Research (INIA) in Uruguay. Dan is a widely published scientist who has received many honours during his career including being a two-time recipient of the Amit Golda Meir Prize from the Hebrew University of Jerusalem, his areas of expertise includes stem cell molecular biology, plant and animal genetics and bioinformatics. Dan's passion for applied science and technological solutions did not stop him from pursuing a deep connection to the farmer, his family and nature. Among some of his interest and practices counts enjoying working as a beekeeper and onboard fishing.



Hazem Shawky Fouda has a PhD. in Agriculture Sciences, obtained his PhD. From the Faculty of Agriculture, Alexandria University in 2008, He is working in Cotton Arbitration & Testing General Organization (CATGO).



Felecia Killings is the Founder and CEO of LiyahAmore Publishing, a publishing company committed to providing technical and educational services and products to Christian Authors. She operates as the Senior Editor and Writer, the Senior Writing Coach, the Content Marketing Specialist, Editor-in-Chief to the company's quarterly magazine, the Executive and Host of an international virtual network, and the Executive Director of the company's online school for Authors. She is a former high-school English instructor and professional development professor. She possesses a Master of Arts degree in Education and a Bachelor's degree in English and African American studies.



Dr. Sandra El Hajj, Ph.D. in Health Sciences from Nova Southeastern University, Florida, USA is a health professional specialized in Preventive and Global Health. With her 12 years of education obtained from one of the most prominent universities in Beirut, in addition to two leading universities in the State of Florida (USA), Dr. Sandra made sure to incorporate interdisciplinary and multicultural approaches in her work. Her long years of studies helped her create her own miniature world of knowledge linking together the healthcare field with Medical Research, Statistics, Food Technology, Environmental & Occupational Health, Preventive Health and most noteworthy her precious last degree of Global Health. Till today, she is the first and only doctor specialized in Global Health in the Middle East area.



Fozia Parveen has a Dphil in Sustainable Water Engineering from the University of Oxford. Prior to this she has received MS in Environmental Sciences from National University of Science and Technology (NUST), Islamabad Pakistan and BS in Environmental Sciences from Fatima Jinnah Women University (FJWU), Rawalpindi.



Igor Krunic 2003-2007 in the School of Economics. After graduating in 2007, he went on to study at The College of Tourism, at the University of Belgrade where he got his bachelor degree in 2010. He was active as a third-year student representative in the student parliament.Then he went on the Faculty of science, at the University of Novi Sad where he successfully defended his master's thesis in 2013. The crown of his study was the work titled Opportunities for development of cultural tourism in Cacak". Later on, he became part of a multinational company where he got promoted to a deputy director of logistic. Nowadays he is a consultant and writer of academic subjects in the field of tourism.



Dr. Jovan Pehcevski obtained his PhD in Computer Science from RMIT University in Melbourne, Australia in 2007. His research interests include big data, business intelligence and predictive analytics, data and information science, information retrieval, XML, web services and service-oriented architectures, and relational and NoSQL database systems. He has published over 30 journal and conference papers and he also serves as a journal and conference reviewer. He is currently working as a Dean and Associate Professor at European University in Skopje, Macedonia.



Dr. Tanjina Nur finished her PhD in Civil and Environmental Engineering in 2014 from University of Technology Sydney (UTS). Now she is working as Post-Doctoral Researcher in the Centre for Technology in Water and Wastewater (CTWW) and published about eight International journal papers with 80 citations. Her research interest is wastewater treatment technology using adsorption process.



Stephen obtained his PhD from the University of North Carolina at Charlotte in 2013 where his graduate research focused on cancer immunology and the tumor microenvironment. He received postdoctoral training in regenerative and translational medicine, specifically gastrointestinal tissue engineering, at the Wake Forest Institute of Regenerative Medicine. Currently, Stephen is an instructor for anatomy and physiology and biology at Forsyth Technical Community College.



Michelle holds a Masters of Business Administration from the University of Phoenix, with a concentration in Human Resources Management. She is a professional author and has had numerous articles published in the Henry County Times and has written and revised several employee handbooks for various YMCA organizations throughout the United States.

TABLE OF CONTENTS

Preface ix

Chapter 1 Designing and Planning Web Pages 1

Introduction ... 2
Web Designing Procedure ... 3
- Web Browsers ... 4
- Web Technologies ... 6
- Elements of Web Page ... 11
- Static Web Page ... 17
- Dynamic Web page ... 18

Planning The Criteria and Quality For Websites ... 23
- Criteria for Websites ... 24
- Quality for Websites ... 26
- Features of Website ... 34

Knowledge Check ... 43
References ... 46

Chapter 2 Creating Web Pages With HTML 47

Introduction ... 48
Fundamental Of HTML ... 48
- HTML Tags ... 50
- Naming Files and Folders ... 56
- Create a Simple Web Page with HTML ... 59
- HTML Attribute ... 63
- Core Attributes ... 64

HTML STRUCTURE ... 71
- DOCTYPE ... 71
- HTML Character Entities ... 73
- HTML Color ... 82
- HTML Lists ... 84

HTML Links 86
Knowledge Check 91
References 94

Chapter 3 Essential of Web API 97

Introduction 98
Functionality of Application Programing Interface 99
Explanation of API 100
Web APIs 105
Web API Controller 109
API Usages and Testing Procedure 116
Operating systems 117
Remote APIs 118
Use in Web APIs 119
API Testing Strategy 121
Challenges with API Testing 123
Start API testing 125
Configure Web API 126
Knowledge Check 132
References 136

Chapter 4 Cascading Style Sheets (CSS) 137

Introduction 138
Synopsis of CSS 139
Advantages of CSS 144
Visibility 145
Positioning 146
Layers 149
Pseudo Classes 150
Function of Cascading Style Sheets 158
Inline Styles 158
Embedded Style Sheets 160
Conflicting Styles 163
Linking External Style Sheets 165
Positioning Elements 168
Backgrounds 173
Element Dimensions 175
Box Model and Text Flow 177

Knowledge Check 187
References 191

Chapter 5 JavaScript 193

Introduction 194
Features And Uses of Javascript 195
Features of JavaScript 196
JavaScript Syntax 197
JavaScript Data types 199
Operators 201
Arithmetic operators 204
Bitwise Operators 205
Functions In Javascript 214
Calling a Function 215
Function Parameters 215
The Return Statement 216
Nested Functions 218
Function () Constructor 219
Knowledge Check 224
References 228

Chapter 6 Programing Languages 229

Introduction 230
PHP 231
Characteristics of PHP 232
Variables in PHP 236
PHP File Handling 240
ASP.NET 245
ASP.NET Architecture and its Components 245
ASP.NET Page Life Cycle 247
ASP.NET Page Life Cycle Events 248
ASP.NET - First Example 249
Personalization 254
Java Basics 259
Features of Java 261
Simple Program of Java 264
Java Security 265

Knowledge Check 271
References 274

Index **275**

PREFACE

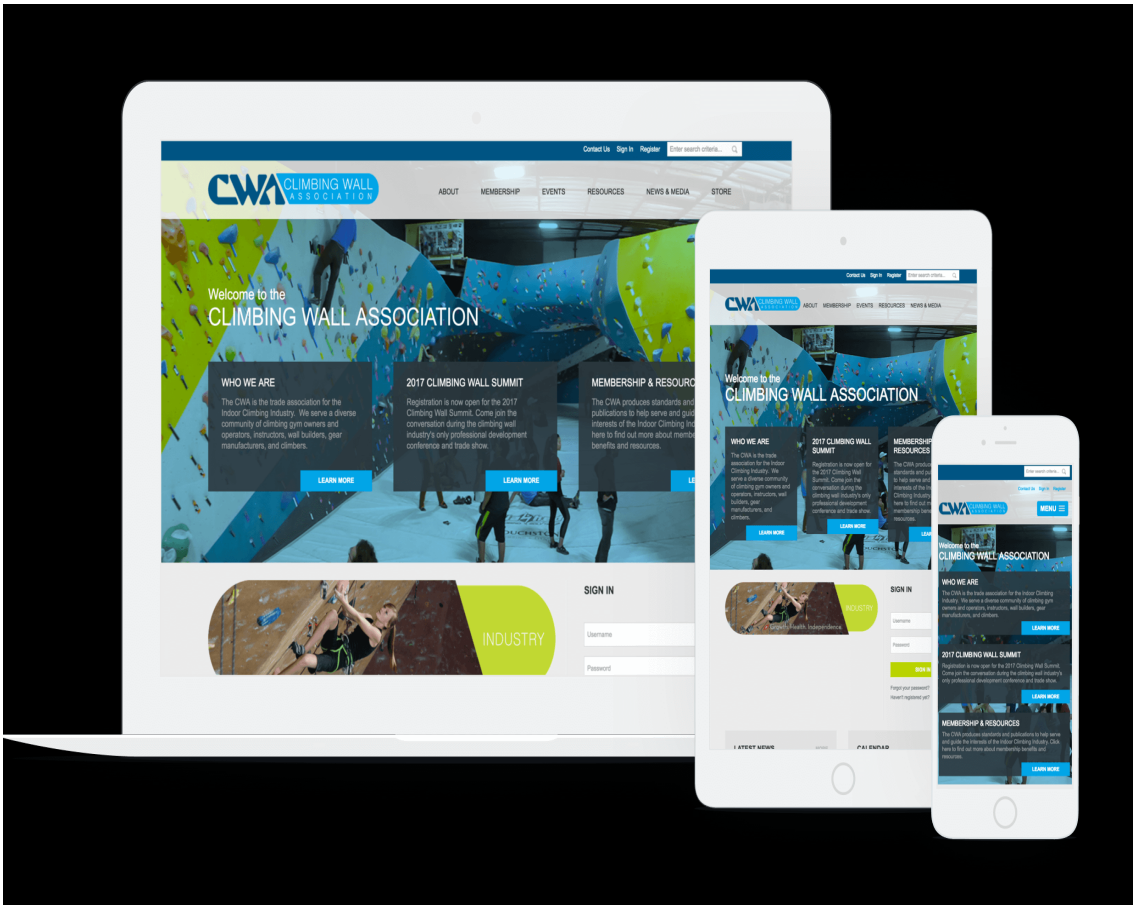
In today digital world, website is need for every organization of any field and web technology is playing vital role in business development as its technology and working procedures are also changing speedily. So, up-to-date designing and development techniques in website making its wider scope and the course content are also growing gradually. Web design is used as a general term to describe any of the various tasks involved in creating a web page. More specifically, it refers to jobs focused on building the front-end of a web page. The web contains numerous pages, presenting information using different technologies and associated together with hyperlinks. There are two basic aspects to any web page found on the Internet. The first is a presentation that the user interacts with, usually visually, while the second is a back-end that includes information for non-human browsers.

Web designing encompasses many different skills and disciplines in the production and maintenance of websites including web graphic design; interface design; authoring, standardized coding using proprietary software; user experience design. Web Designing manages the way toward building up a site. A website specialist use pictures, content, html5, Responsive website composition, CSS3, data engineering, hues, text styles and other intuitive components to deliver a site page. Website specialists use markup language for the most part HTML for making structure and CSS for presentation. The web has gone through many changes over the past few decades, but HTML has always been the fundamental language used to develop webpages. Interestingly, while websites have become more advanced and interactive, HTML has actually gotten simpler. If you compare the source of an HTML5 page with a similar page written in HTML 4.01 or XHTML 1.0, the HTML5 page would probably contain less code. This is because modern HTML relies on cascading style sheets or JavaScript to format nearly all the elements within a page.

This Handy Guide introduces students to basic web design using HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets). Throughout the book students are introduced to planning and designing effective web pages; implementing web pages by writing HTML and CSS code; enhancing web pages with the use of page layout techniques, text formatting, graphics, images, and multimedia; and producing a functional, multi-page website. In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

CHAPTER 1

DESIGNING AND PLANNING WEB PAGES



INTRODUCTION

A web page or webpage is a document commonly written in HTML (Hypertext Markup Language) that is accessible through the Internet or other networks using an Internet browser. A web page is accessed by entering a URL address and may contain text, graphics, and hyperlinks to other web pages and files. The page you are reading now is an example of a web page.

The term web page usually refers to what is visible, but may also refer to the contents of the source code itself, which is usually a text file containing hypertext written in HTML or a comparable markup language. Most current web browsers include the ability to view the source code. Web browsers will frequently have to access multiple web resource elements, such as style sheets, scripts, and images, while presenting each web page.

SCENARIO: BAR AND BEVERAGE SERVICE OPERATION

Designing is a process of creating a plan and originating the development of a specific product. Once the product is designed, its design is used to start the production. Design often refers to the creation of a product prototype. In most cases the meaning of "product design" refers only to the product's appearance rather than to its construction, architecture and technical specifications.

Web design itself refers to the process of creating a web page's appearance and to the choice of a right color scheme, page layout, fonts and more. Every single web page in a website has different content, but all the pages are using a similar graphic design. Often sites will use website templates, which contain all the basic elements of web design - the website's CSS style, buttons, backgrounds, borders and various graphic elements like hover images, bullets and header banners. When the website template is applied to the website, all the pages assume its appearance, using the same styles, background and other graphical elements.

What skills will you need?

In order to work effectively in web pages designing and planning, you must know:

- Web designing procedure
- Planning the criteria and quality for websites

WEB DESIGNING PROCEDURE

Web design is used as a general term to describe any of the various tasks involved in creating a web page. More specifically, it refers to jobs focused on building the front-end of a web page.

The web consists of myriad pages, presenting information using different technologies and linked together with hyperlinks. There are two basic aspects to any web page found on the Internet. The first is a presentation that the user interacts with, usually visually, while the second is a back-end that includes information for non-human browsers.

The basic markup language used to tell a browser how to present information is called the HyperText Markup Language (HTML). A stricter version of HTML is also widely used, known as eXtensible HyperText Markup Language (XHTML). Using HTML or XHTML, a web designer is able to tell a browser how a web page should appear. In the last few years, there has been a push towards separating the underlying structure of a web-page (using HTML) from the visual presentation of the site (using Cascading Style Sheets or CSS). This approach has a number of major benefits in both the short and long term, and it is gathering popularity as time progresses.

From a technical standpoint, the act of web design can be quite difficult. Unlike more traditional print media, HTML has a number of variable factors. To begin with, not all browsers interpret HTML according to the standards created by the standard-setting body — the World Wide Web Consortium, also known as W3. This means that while a page will appear as the designer wishes it to in one browser, it may appear completely differently in another. There are numerous fixes and work-arounds to try to circumvent browser-specific bugs, but it is a tenuous business at best.

Another major limiting factor is the plethora of formats a site might be viewed in. While graphic designers know exactly how large the piece of paper they are printing on will be, a web designer must account for different monitor sizes, different display settings, and even browsers for non-sighted users. Combined, these concerns often leave a design professional struggling to incorporate enough dynamism to make a web page attractive on a range of browser sizes, while creating a layout static enough to allow for the use of images and other necessarily fixed-size components.

In addition to XHTML and CSS, designers often use a number of database driven languages to allow for more dynamism and interactivity on their websites. While useful with smaller sites, such languages become a virtual necessity on any site presenting huge amounts of data.

The possibilities for web design are virtually limitless, although at one point, they were quite constrained by the boundaries of the browser itself. With the advent and flexibility of embedded technologies, these boundaries have been all but removed, allowing for a versatility and dynamism that challenges the imagination of anyone interested in designing for the Internet.

Web Browsers

A web browser is a software program that allows a user to locate, access, and display web pages. In common usage, a web browser is usually shortened to "browser." Browsers are used primarily for displaying and accessing websites on the internet, as well as other content created using languages such as Hypertext Markup Language (HTML) and Extensible Markup Language (XML).

Browsers translate web pages and websites delivered using Hypertext Transfer Protocol (HTTP) into human-readable content. They also have the ability to display other protocols and prefixes, such as secure HTTP (HTTPS), **File Transfer Protocol (FTP)**, email handling (mailto:), and files (file:).

In addition, most browsers also support external plug-ins required to display active content, such as in-page video, audio and game content.

File Transfer Protocol is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

A variety of web browsers are available with different features, and are designed to run on different operating systems. Common browsers include Internet Explorer from Microsoft, Firefox from Mozilla, Google Chrome, Safari from Apple, and Opera. All major browsers have mobile versions that are lightweight versions for accessing the web on mobile devices.

Web browsers date back to the late 1980s when an English scientist, Tim Berners-Lee, first developed the ideas that led to the World Wide Web (WWW). This consisted of a series of pages created using the HTML language and joined or linked together with pointers called hyperlinks. Following this was the need for a program that could access and display the HTML pages

correctly – the browser. In 1993, a new browser known as Mosaic was developed, which soon gained widespread usage due to its graphical-interface capability. Marc Andreesen, a member of the Mosaic development team, left in 1994 to develop his own commercial browser based on Mosaic. He called it Netscape Navigator, and it quickly captured over 90 percent of the nascent browser market.

It soon faced stiff competition in 1995 from Microsoft's Internet Explorer, which was freely bundled with Windows 95 (and later versions of Windows). It was pointless to buy Navigator when Internet Explorer was free, and as a result, Navigator (and Netscape) were driven into the ground.

But while Mosaic and Netscape are no longer around, the age of the browser was launched and continues to this day, as more and more applications move to the web.

The Advantages & Importance of Web Browsers

The Web is essentially a publishing medium that shares characteristics with more traditional publishing media. Web browsers have a similar function to print publishers. They conform to certain standards that ensure that a website designed in any part of the world looks the way its creators expect it to. Web browsers developed along with the Web and there are now at least five widely used ones. Web browsers are not the same thing as search engines, which find information.

Web Standards

Web standards form the foundation of the Web publishing. Web standards are technical specifications that dictate the methods for building and designing websites. Sites complying with these standards use valid Hypertext Markup Language, valid character encoding, metadata and character embedding, as well as valid server settings. If a website doesn't follow these standards, there is no guarantee that it will display fully in all browsers, and users may find that they can't perform some site transactions unless they have a specific browser version. On the other hand, when browser developers don't follow these standards, Web developers face incompatibility issues. Web sites and Web browsers need to follow the same set of rules to achieve universal uniformity of appearance.

Browser Incompatibility

Prior to the introduction of Web standards, Web developers faced the problem of browser incompatibility. This lack of a "one website fits all browsers" added to significant costs to website development. Developers on low budgets were forced to build sites that locked out some users as they had to choose which browsers a website would be compatible with and the ones they would ignore. Also, some

browsers didn't support the Web standards specifications, so developers questioned the logic of using the standards if the browsers didn't use them as well. The development of Web standards in 1998, attempted to iron out this issue and get site and browser developers to use the same standards.

Browser Development

Mosaic was the original Web browser. The company developed Netscape Navigator, which quickly just became known as Netscape. In 1999, Microsoft launched Internet Explorer, which took over Netscape's market share, largely due to most PCs coming bundled with Microsoft software. In 2002, Mozilla introduced its free open source browser that anyone could download. Mozilla is the old Netscape in a new form, and in 2004 the Firefox version appeared. Google's Chrome browser is one of the newest browsers. Apple Macs come with the Safari browser, but Mac users are free to use one of the other browsers.

Browser Qualities

One of the most important qualities a browser should have is the ability to load Web pages quickly regardless of how dynamic the content is. Browsers must be able to load graphic images and videos rapidly and not wait for minutes while the page downloads. It is important for users to know that they are not stuck with the default browser that comes with the computer. Read reviews of browsers, and try out alternatives with some favorite websites to see how well the browser performs. Use the Acid3 test to how quickly each browser loads a page.

◑ Web Technologies

Web technology is a collective name for technology primarily for the world wide web. This does, however, tend to focus more specifically towards technology that aides in the creation, maintenance and development of web-based applications.

From a developers point of view, these are things like languages and frameworks, data storage mechanisms, services and products to allow us to create platforms and applications for the web.

ASP.NET is a Microsoft framework for web development that allows a developer to use languages like C# and VB.NET to build websites and web applications (a web application is a website that offers functionality, storing data - text, images, etc. - that can be processed and return a useful output to you, or that you can come back to later, such as Facebook).

You can find web frameworks for Java, Python and a few other languages so that developers can use a whole array of technologies, rather than being pinned

down to one. These are known as server-side technologies which lives and runs on a server somewhere.

On the other side, you have client-side web technologies, which include things like HTML, CSS, JavaScript (which includes JavaScript Frameworks like jQuery, Angular and React). These execute run from the web page that a user loads, but actually run on the users (clients) computer.

However, web technology is not limited to the developers. Web browsers, for example, are in the category of web technology as it is technology that allows a user to access the web. FTP may also be considered a web technology as it is designed to send files over the Internet.

Each separate technology is fairly limited on its own, and tends to require the dual use of at least one other such technology. Therefore we can conclude that all of the components that make up a site are interdependent on one another. This is some common terminology use in today's modern web world:



REMEMBER

Web technologies is a general term referring to the many languages and multimedia packages that are used in conjunction with one another, to produce dynamic web sites such as this one.

1. Browsers: Browsers are the interpreters of the web. They request information and then when they receive it, they show us on the page in a format we can see and understand.

- Google Chrome - Currently, the most popular browser brought to you by Google
- Safari - Apple's web browser
- Firefox - Open-source browser supported by the Mozilla Foundation
- Internet Explorer - Microsoft's browser. You will most often hear web developers complain about this one.

2. HTML: HTML is a markup language. It provides the structure of a website so that web browsers know what to show.

HTML (Hyper Text Mark-up Language) is the glue that holds together every web site. Like building a house, you always build a strong foundation first. For any site, HTML is that foundation. HTML is an open source language

(i.e. not owned by anyone), which is easy to learn, and requires no fancy (or expensive!) packages to start using it. All you need is something to type with, such as Windows Notepad, and a lot of time and patience.

HTML works on a 'tag' system, where each tag effects the content placed within that tag, e.g.

<TAG>What the tag effects**</TAG>**.

Although relatively limited by itself, it is the flexibility of HTML that allows web sites to grow in complexity. Like the foundation of your house, HTML is robust enough to support many kinds of languages integrated within your HTML pages.

3. CSS: CSS is a Cascading Style Sheet. CSS let's web designers change colors, fonts, animations, and transitions on the web. They make the web look good.

- LESS - a CSS pre-compiler to make working with CSS easier and add functionality
- SASS - a CSS pre-compiler to make working with CSS easier and add functionality

4. Programming Languages: Programming languages are ways to communicate to computers and tell them what to do. There are many different programming languages just like there are many different lingual languages (English, Spanish, French, Chinese, etc). One is not better than the other. Developers typically are just proficient at a couple so they promote those more than others. Below are just some of the languages and links to their homepages

- Javascript - used by all web browsers, Meteor, and lots of other frameworks
- Coffeescript - is a kind of "dialect" of javascript. It is viewed as simpler and easier on your eyes as a developer but it complies (converts) back into javascript
- Python -used by the Django framework and used in a lot of mathematical calculations
- Ruby - used by the Ruby on Rails framework
- PHP - used by Wordpress
- Go - newer language, built for speed.
- Objective-C - the programming language behind iOS (your iPhone), lead by Apple
- Swift - Apple's newest programming language
- Java - Used by Android (Google) and a lot of desktop applications.

5. Frameworks: Frameworks are built to make building and working with programming languages easier. Frameworks typically take all the difficult, repetitive tasks in setting up a new web application and either does them for you or make

them very easy for you to do.

- Meteor - a full-stack (front and back end) javascript framework
- Node.js - a server-side javascript framework
- Ruby on Rails - a full-stack framework built using ruby
- Django - a full-stack framework built using python
- Ionic - a mobile framework
- Phonegap / Cordova - a mobile framework that exposes native api's of iOS and Android for use when writing javascript
- Bootstrap - a UI (user interface) framework for building with HTML/CSS/Javascript
- Foundation - a UI framework for building with HTML/CSS/Javascript
- Wordpress - a CMS (content management system) built on PHP. Currently, about 20% of all websites run on this framework
- Drupal - a CMS framework built using PHP.
- .NET - a full-stack framework built by Microsoft
- Angular.js - a front-end javascript framework.
- Ember.js - a front-end javascript framework.
- Backbone.js - a front-end javascript framework.

6. Libraries: Libraries are groupings of code snippets to enable a large amount of functionality without having to write it all by yourself. Libraries typically also go through the trouble to make sure the code is efficient and works well across browsers and devices (not always the case, but typically they do).

- jQuery
- Underscore

7. Databases: Databases are where all your data is stored. It's like a bunch of filing cabinets with folders filled with files. Databases come mainly in two flavors: SQL and NoSQL. SQL provides more structure which helps with making sure all the data is correct and validated. NoSQL provides a lot of flexibility for building and maintaining applications.

- MongoDB - is an open-sourced NoSQL database and is currently the only database supported by Meteor.
- Redis - is the most popular key-value store. It is lighting fast for retrieving data but doesn't allow for much depth in the data storage.
- PostgreSQL - is a popular open-sourced SQL database.
- MySQL - is another popular open-sourced SQL database. MySQL is used in **Wordpress** websites.

- Oracle - is an enterprise SQL database.
- SQL Server - is an SQL server manager created by Microsoft.

WordPress is a content management system based on PHP and MySQL that is usually used with the MySQL or MariaDB database servers but can also use the SQLite database engine.

8. Client (or Client-side): A client is one user of an application. It's you and me when we visit http://google.com. Client's can be desktop computers, tablets, or mobile devices. There are typically multiple clients interacting with the same application stored on a server.

9. Server (or Server-side): A server is where the application code is typically stored. Requests are made to the server from clients, and the server will gather the appropriate information and respond to those requests.

10. Front-end: The front-end is comprised of HTML, CSS, and Javascript. This is how and where the website is shown to users.

11. Back-end: The back-end is comprised of your server and database. It's the place where functions, methods, and data manipulation happens that you don't want the clients to see.

12. Protocols: Protocols are standardized instructions for how to pass information back and forth between computers and devices.

- HTTP - This protocol is how each website gets to your browser. Whenever you type a website like "http://google.com" this protocol requests the website from google's server and then receives a response with the HTML, CSS, and javascript of the website.
- DDP - is a new protocol created in connection with Meteor. The DDP protocol uses websockets to create a consistent connection between the client and the server. This constant connection lets websites and data on those websites update in real-time without refreshing your browser.
- REST - is a protocol mainly used for API's. It has standard methods like GET, POST, and PUT that let information be exchanged between applications.

13. API: An API is an application programming interface.

It is created by the developer of an application to allow other developers to use some of the application's functionality without sharing code. Developers expose "end points" which are like inputs and outputs of the application. Using an API can control access with API keys. Examples of good API's are those created by Facebook, Twitter, and Google for their web services.

14. Data formats: Data formats are the structure of how data is stored.

- JSON - is quickly becoming the most popular data format
- XML - was the main data format early in the web days and predominantly used by Microsoft systems
- CSV - is data formatted by commas. Excel data is typically formatted this way.

This was not meant to be an all-encompassing list, but rather a way to talk about all the great technologies we have at our finger tips.

◑ Elements of Web Page

While every designer may have a different plan when it comes to building a website, they do have a common checklist. No matter how you try to avoid it, there are a few elements every website should (and usually does!) include.

From plenty of whitespace and great images, to search functionality and clear calls-to-action, these common elements are the things that users expect when it comes to using a site with ease. Today we're taking a look at ten elements you should prioritize on your website, perfectly designed examples of each, and tips on how to use each in your next website design project.

If you make or edit online content, you are a content creator. As a content creator, you'll need to strike that balance between SEO (search engine optimization) and a positive user experience. To make sure everyone is on the same page, here is a list of the different elements that combine to create an engaging and well optimized piece of content.

The Elements of a Web Page are:

The Page Title

The Title of a web page shows up in in the browser tab. It is very important to explain concisely what the page is about in about eight or fewer words. In the HTML, the Title tag looks like this <title>This is the title of the page</title>.

The difference between the page Title and the Headline.

The Title is something that is seen only in the browser tab and is placed in the HTML. But the Headline is part of the content of the page. This is an example of a headline («Which Companies Have the Best Reputations») that is at the top of a blog post:



Compelling Titles show up in search. Compelling headlines get people to read the content.

People click on your articles and other content in search results because the title was good. A good article with a bad headline will attract far less page views. Page titles should be engaging and persuasive so that searchers are drawn to them. Copyblogger's guide on creating magnetic headlines is a great resource to learn more about creating titles for your web pages. Here is an example of a "clickbait" headline. It makes you curious.

You might also find this article not only informative but amusing, and this one is very good as well.

Headlines drive clicks and rankings

Several experiments have shown that a page with a higher CTR (Click Through Rate) quickly rises in search results. In other words, the more people that click on your Title in search results, and the longer they stay on the page, the higher your page may rise in search results.

- People click on compelling page titles in search results.
- The best titles and headlines have the search phrase (key phrase) in them.
- Clickbait headlines can be ridiculous, but they work if not overdone.

You don't need to write ridiculous headlines that compel people to click, but keep the theory in mind when you do create headlines.

The Content of the Page

The main content of a page may be text, video, graphics or images, or something else. Search engines are getting to the point they can understand images and video to the point they can text.

Keywords

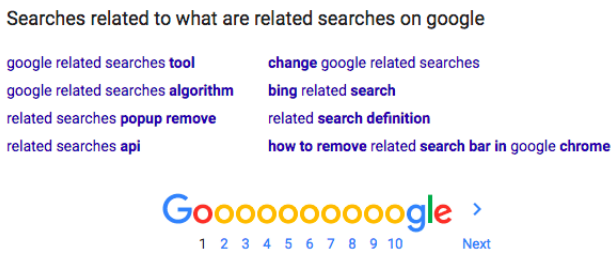
For text-based content, keywords help search engines understand. Keywords / search phrases are becoming less and less important as artificial intelligence rises. Keywords (also known as search phrases) are the words people use to search anything on Google or other search engines like Bing, YouTube, and even Facebook (yes, Facebook is a search engine). For voice-based search they are spoken, usually as part of a longer phrase.

As a content creator, you need to identify the keywords that your audience is using to search for the answers to their problems on Google, and make those keywords a part of your content strategy. If your content is primarily video or images it may be less important, but it can help.

Related Search Phrases Help You Spy on People's Intent

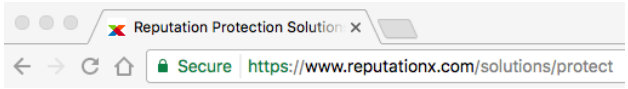
At the bottom of search results in Google you will often find "related searches". Related searches can be used to understand what people are searching for, for key phrase research, and to improve the quality of your articles by including additional information you may not have thought of that people want.

Example: Here is an image of the Related Searches for the search term "related searches":



Page Titles Tell Search Engines the Main Point

The Title of your page is usually (but not always) what Google puts into search results. Here is the Title of one of our pages in the browser tab (the top tab in your browser).



Here is how Google portrays it in search results.

Reputation Protection Solutions - Reputation X
https://www.reputationx.com/solutions/protect
Reputation protection services that defend both people and brands against negative search results.

The page title tells Google and the readers about the content of a page, in this case "Reputation Protection Solutions". The Title tag is one of the most important on-page SEO factors and has a direct impact on your site's search engine rankings.

In the HTML code of the web page, the Title tag looks like this:

<title>Reputation Protection Solutions</title>

Use the key phrase early in the title tag

From an SEO perspective, you not only want to make sure the search phrase you are targeting the page for is in the Title tag, but make sure your target keyword is used *as early in the title as possible* to not only give it a strong chance of ranking higher but also to make it more eye-catching for the user.

Beware the Character Limit

The character limit for page Titles is 70 characters for desktop and 78 characters for mobile search results. So, try keeping your titles to under 70 characters so that they're displayed properly in search results.

Click-through rate (CTR) is the ratio of users who click on a specific link to the number of total users who view a page, email, or advertisement.

The Meta Description Helps People Decide to Click

The Meta Description of your page is the text that appears right under your page title in search results. You cannot see the Meta Description in the content of a web page, but you can see it in search results. The Meta Description is not a search ranking factor for Google, but it does help

people decide whether to click on your page or not in search results.

The meta description is important from a user's point of view because it lets people know the subject of the page when they see it in search results. A compelling description can persuade searchers to click on your page which improves its average **click through rate (CTR).** This means it's important to "sell" the link to your audience in search results - you can do this by writing a description that makes the reader really want to click on it.

The Meta Description HTML looks like this:

```
5    <meta name="description" content="Reputation protection services that defend
both people and brands against negative search results. ">
```

Google often uses the page Meta Description in search results, but not always.



H1 Headline

H1 is the HTML tag for the main headline of your page (usually the title). Generally speaking, every web page should have one headline with an H1 tag and should contain the primary target keyword. But having more than one shouldn't hurt your SEO. It's more important to nest H3 headlines inside of H2 headlines inside of an H1 headline.

In most WordPress themes, the title of your page is wrapped in H1 tag by default.

Google considers the H1 headline a strong indicator of the content of a page. Which is why using your primary keyword in an H1 headline gives you a better chance of ranking for it in search results.

Body Content

The main content of your post is called body content. It should not only be optimized for search engines but should also be engaging for the readers (most important). Here are a few tips for achieving both of these objectives:

Compelling Intro

Write a compelling introductory paragraph that's no longer than 100-150 words (unless necessary). Read this detailed guide for more.

Include Search Phrase in First 100 Words

If it makes sense, try to include your primary keyword in the first 100 words of the body content. This can help rank the page better.

Use Short Paragraphs

If possible, break down your content into short paragraphs to make it easier for the readers to skim through it. Most people don't read the content all the way through, they skim it (sorry Writers).

Use H2 Sub-Headings with Related Search Terms

Organize your content into different segments using multiple sub-headings (H2 Headlines). The H2 headlines should include "related search phrases" often found at the bottom of a search engine results page if possible. This acts as an additional hook for search engines to grasp and increases the elevance of your page for not only your main search phrase but similar ones as well.

Embolden Important Points

Use bold, italics and underlines to highlight important points.

Use Bullet Points

Here are some reasons to use bullet points:

- They help people skim content quickly
- Search engines like them and often show them in results
- They break up the text to make it less daunting to read

Keep Engagement with Other Content

Use images, snapshots and embedded videos frequently to make your content more engaging for the readers. People get bored easily.

Use Synonyms

Use synonyms and other terminologies related to your target keyword throughout your body content for better optimization. Again, related searches that appear at the bottom of the search page for your primary keyword can help with this. So can your trusty thesaurus.

Keyword Tool Tip

You can also use tools like Ubersuggest and LSI Graph to find hundreds of synonyms and related terminologies that you can include in your body content along with your primary keyword. Remember that you are writing for people first, so don't stuff search phrases into an article that really doesn't need them. Balance is key.

◑ Static Web Page

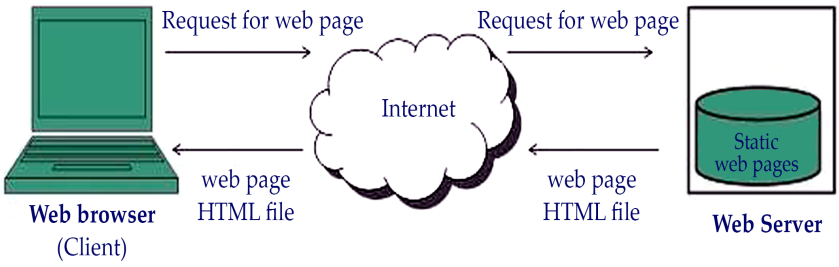
A static Web page is a page that is built using HTML code and features the same presentation and content, regardless of user identity or other factors. Static Web pages are easier to code and assemble than dynamic Web pages, which may feature customizable content according to a user's identity or other factors.

In a sense, a static Web page is just a simple purveyor of information. Designers often use a combination of text and images, controlled by HTML tags, to render something that is a lot like a newspaper page. It has typesetting and layout, but it does not change from one load to another.

Another way to understand static Web pages is to contrast them with dynamic Web pages. The latter have controls and forms that are deep coded, so that the page displays differently to different users or in different situations. For example, a dynamic Web page may access a database to find out about a user's identification and history, or to display custom items like the user's name or his/her collected preferences. In contrast, a static Web page does not provide this kind of customization.

Static web pages are also known as flat or stationary web page. They are loaded on the client's browser as exactly they are stored on the web server. Such web pages contain only static information. User can only read the information but can't do any modification or interact with the information.

Static web pages are created using only HTML. Static web pages are only used when the information is no more required to be modified.



◑ Dynamic Web page

A dynamic web page is a page that changes based on the user. It responds to the user's needs, and provides relevant information to meet them, by accessing information in a connected database. This type of web page allows users to go beyond reading text and looking at graphics. It allows for an interactive experience, with the user being in control of the information he views.

There are many types of dynamic web pages. One of the most popular is Google Earth™. Most map websites are dynamic; users input the address they are traveling to, and the page returns with directions and a map to the location.

Another example of a dynamic web page is a search engine. Web site searches, where the user is looking for information on just one site, are also dynamic pages. Other examples include email websites, banking sites where users can access their personal financial information, and sites that base their information on location, such as a national weather page. Any website that offers information based on any data the user provides is a dynamic page.

Dynamic web pages are often very simple, as far as their actual design. They offer a place for the user to type in data, and some information and basic graphics. After the user types in their data, this changes. The web page server accesses a database of information, searching for what the user has requested. Once the information is found, it is brought back to the web page, and displayed.

Creating a dynamic web page requires a certain level of previous programming experience. Having a basic understanding of a couple programming languages is essential. Javascript is a popular language to use for creating these types of web pages. Creating a web page that interacts with users can be difficult, especially for beginners to web page design and programming. It requires understanding and being able to use complicated code, and can be difficult to work with if you do not have a lot of previous experience.

Creating a user-friendly dynamic web page may be challenging, but it provides a great way for website owners to better meet the needs of people browsing their site. It allows people to find what they need in a quick and easy way, while keeping track of what information is being accessed, which give owners a way to access information about how their site is being used. Using this information, site owners can improve and expand their site in a way that benefits site visitors, increasing the success of the web site.

A dynamic web page is a web page that displays different content each time it's viewed. For example, the page may change with the time of day, the user that accesses the webpage, or the type of user interaction. There are two types of dynamic web pages.

Client-Side Scripting

Web pages that change in response to an action within that web page, such as a mouse or a keyboard action, use client-side scripting.

Client-side scripts generate client-side content. Client-side content is content that's generated on the user's computer rather than the server. In these cases, the user's web browser would download the web page content from the server, process the code that's embedded in the web page, and then display the updated content to the user.

Scripting languages such as JavaScript and Flash allow a web page to respond to client-side events.

Client-side scripting is a process by which a web-based computer program runs on the user's computer rather than on the computer hosting the website. Specifically, it runs through a type of program known as a script, which is handled by the user's web browser. The main reason for client-side scripting is to allow a web page to be created specifically around the user's own data and options, rather than being a fixed page that always appears the same to every user.

The need for client-side scripting derives from the way web pages work. Originally, pages were written in standard HTML and were known as static pages; that is, every computer, and in turn every user, saw exactly the same page. This made websites very limited in dealing with large sets of data, such as a railway schedule. With static pages, the only solution was to print the schedule in full and let the user hunt down the relevant detail.

This problem was solved by the development of dynamic web pages. These can adapt to meet a specific situation, such as if a user is searching for a journey between two stations within a specific time period. The result of the search is displayed through a dynamic web page, which is automatically created for that query.

There are two ways to carry out the processing needed to generate the dynamic web page. Server-side scripting means that the computer hosting the website, known as a server, carries out all the work and then generates an HTML page to send to the user's computer. One method for doing this is known as VBScript, which is why occasionally users may carry out a search or other dynamic request on a website and get back a page listing a "VBScript database error."

The second method, client-side scripting, involves embedding special code in the web page. The user's web browser then uses this code to carry out the necessary processing, such as searching a behind-the-scenes database on the website, and produce a page from the results. The best known type of code used in this way is called Javascript. The biggest drawback of client-side scripting is the security implications. In theory, the code could order the browser to carry out all

sorts of actions, including accessing data on the computer itself. Web browsers use a variety of security measures to prevent this being abused, such as limiting the code to accessing only the browser. Flaws in these security measures are one of the main sources of opportunity for virus creators.

Server-Side Scripting

Web pages that change when a web page is loaded or visited use server-side scripting. Server-side content is content that's generated when a web page is loaded. For example, login pages, forums, submission forms, and shopping carts, all use server-side scripting since those web pages change according to what is submitted to it.

Scripting languages such as PHP, ASP, ASP.NET, JSP, ColdFusion and Perl allow a web page to respond to submission events.

Server-side scripting is a term primarily used with regards to serving custom content via the hypertext transfer protocol (HTTP) on a web server by having the server execute small programs. These programs, usually written in a scripting language, are executed by the server when a client request arrives. Depending on the parameters given by the client at the time of the request, the script then generates a web page for the client. Web pages created in this way are often referred to as dynamic pages. This is in contrast to a client-side script that is sent from the server to the requesting client and then executed.

Internet Engineering Task Force (IETF) is an open standards organization, which develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite (TCP/IP).

The first server-side scripting techniques came about in order to deliver content that is unique to the user making the request, as well as save time by reducing the manual editing of the hypertext markup language (HTML) documents. The server-side scripts were often times small, executable files containing a series of commands to pass to the operating system. The web daemon, the software running the web server, would use these shell scripts to further execute an additional program residing on the host computer. This general technique was then defined in 1993 in the common gateway interface (CGI) standard developed by the **Internet Engineering Task Force (IETF).**

In most cases, these early CGI scripts were used to send database responses from the server back to the requesting client.

As interpreted scripting languages were developed, such as Perl, and PHP: Hypertext Preprocessor (PHP), server-side scripting methods also evolved. HTTP daemons were updated to include use extensions that allowed for these various scripting languages to be called from the web daemon itself, instead of being passed to the operating system on the host. With this addition, bits of scripting language could be included within the HTML document. As the web server reads through a document before sending it out across the web, it checks for and executes any of the script within the document.

Since the content called up by server-side scripting resides in a database, it can be virtually anything. Content such as product descriptions, price variations, weblog entries, images, and even formatting is stored in the database. It is also possible to nest one server-side script snippet within another, where the first script grabs certain data from the database, but also calls a second script which accesses secondary data. This is useful, for example, when delivering some fairly custom standard content with additional nested content that may be time sensitive or as yet unknown, such as comments on a weblog entry. Web site maintenance, then, involves updating data in the database, which will then affect every page on the website with the included script to call that data.

Many of the server-side scripting techniques have been further developed into what are known as content management systems (CMS). PHP is probably the most frequently used language for this purpose, running at the core of many CMS implementations. Here, a user operating a website running on a CMS will edit her HTML documents to include what are called tags. The tags are essentially cues to the CMS to tell it what script to include in the document. The CMS can then be customized by creating additional scripts and their associated tags.

LEARNING ACTIVITY: THE DIFFERENCE BETWEEN DYNAMIC AND STATIC WEB PAGES

Some websites online bring up-to-the-minute, customized content, from the latest sports scores to updates from your friends on social media pages. Others have information that remains constant no matter who's viewing the site and when they stop by. Sites that change over time are generally known as dynamic sites, while sites that do not change are static.

Read more details on:

https://smallbusiness.chron.com/difference-between-dynamic-static-pages-69951.html

What are the importance of web technologies in web page designing?

Define your Direction: On a separate piece of paper, explain the significance of the different types of the terms according to web designing procedure:

Terms	***Significance***
Click through rate (CTR)	
Search engine	
Hypertext transfer protocol (HTTP)	
Common gateway interface (CGI)	
Internet Engineering Task Force (IETF)	



PLANNING THE CRITERIA AND QUALITY FOR WEBSITES

A website is a page or collection of pages on the World Wide Web that contains specific information which was all provided by one person or entity and traces back to a common Uniform Resource Locator (URL). Facebook and MySpace are examples of websites.

Building a website without a plan is like constructing a building without blueprints. Things end up in the wrong place, features are overlooked, and the situation is ripe for miscommunication between website builder and client. Planning your website ahead of time will give it clear direction as well as prevent missed deadlines and backtracking. If you are a web designer working with clients, then this guide will help both of you to plan properly. If you are a business owner or employee of an organization, then this guide will help lay the groundwork for your coming website.

The purpose of an information-centric website is to convey specific, helpful information to a specific user/audience so that the reader learns something new or understands a topic better. These websites showcase entertaining information for visitors.

A website is a collection of publicly accessible, interlinked Web pages that share a single domain name. Websites can be created and maintained by an individual, group, business or organization to serve a variety of purposes. Together, all publicly accessible websites constitute the **World Wide Web**.

Websites come in a nearly endless variety, including educational sites, news sites, porn sites, forums, social media sites, e-commerce sites, and so on. The pages within a website are usually a mix of text and other media. That said, there are no rules dictating the form of a website. A person could create a website of nothing but black and white photos of roses, or the word “cat” linked to another Web page with the word “mouse.” However, many sites follow a standard pattern of a homepage that links off to other categories and content within the website. Originally, websites were categorized by their top-level domains. Some examples include: Government agency websites = .gov Educational institutions’ websites = .edu Nonprofit organizations’ websites = .org Commercial websites = .com information sites = .info Although these top-level domains extensions still exist, they say little about a website’s actual content. In the modern day internet, the “.com” extension is by far the most popular domain, a long with many other country-specific extensions.

◑ Criteria for Websites

These six criteria deal with the content of Web sites rather than the graphics or site design. Apply these criteria when you research on the internet.

1. Authority. Authority reveals that the person, institution or agency responsible for a site has the qualifications and knowledge to do so. Evaluating a web site for authority:

- Authorship: It should be clear who developed the site.
- Contact information should be clearly provided: e-mail address, snail mail address, phone number, and fax number.
- Credentials: the author should state qualifications, credentials, or personal background that gives them authority to present information.
- Check to see if the site supported by an organization or a commercial body

World Wide Web (WWW), is an information system where documents and other web resources are identified by Uniform Resource Locators (URLs, such as https://www.example.com/), which may be interlinked by hypertext, and are accessible over the Internet.

2. Purpose. The purpose of the information presented in the site should be clear. Some sites are meant to inform, persuade, state an opinion, entertain, or parody something or someone. Evaluating a web site for purpose:

- Does the content support the purpose of the site?
- Is the information geared to a specific audience (students, scholars, general reader)?
- Is the site organized and focused?
- Are the outside links appropriate for the site?
- Does the site evaluate the links?
- Check the domain of the site. The URL may indicate its purpose.

3. Coverage. It is difficult to assess the extent of coverage since depth in a site, through the use of links, can be infinite. One author may claim comprehensive coverage of a topic while another may cover just one aspect of a topic. Evaluating a web site for coverage:

- Does the site claim to be selective or comprehensive?

- Are the topics explored in depth?
- Compare the value of the site's information compared to other similar sites.
- Do the links go to outside sites rather than its own?
- Does the site provide information with no relevant outside links?

4. Currency. Currency of the site refers to: 1) how current the information presented is, and 2) how often the site is updated or maintained. It is important to know when a site was created, when it was last updated, and if all of the links are current. Evaluating a web site for currency involves finding the date information was:

- First written
- Placed on the web
- Last revised

Then ask if:

- Links are up-to-date
- Links provided should be reliable. Dead links or references to sites that have moved are not useful.
- Information provided so trend related that its usefulness is limited to a certain time period?
- The site been under construction for some time?

5. Objectivity. Objectivity of the site should be clear. Beware of sites that contain bias or do not admit its bias freely. Objective sites present information with a minimum of bias. Evaluating a web site for objectivity:

- Is the information presented with a particular bias?
- Does the information try to sway the audience?
- Does site advertising conflict with the content?
- Is the site trying to explain, inform, persuade, or sell something?

6. Accuracy. There are few standards to verify the accuracy of information on the web. It is the responsibility of the reader to assess the information presented. Evaluating a web site for accuracy:

- Reliability: Is the author affiliated with a known, respectable institution?
- References: do statistics and other factual information receive proper references as to their origin?
- Does the reading you have already done on the subject make the information seem accurate?
- Is the information comparable to other sites on the same topic?
- Does the text follow basic rules of grammar, spelling and composition?

- Is a bibliography or reference list included?

◑ Quality for Websites

A high quality website is one that provides relevant, useful content and a good user experience, however, there are many individual factors that need to be considered beyond that summarization. Spending a lot of money on a site is no guarantee of it being classed as a high quality one. You can actually create one for very little financial investment (as little as £250 probably) so long as you invest thought and time into it. Though assessing quality is a subjective task, as what one person's judges as low quality another might judge as high quality, there are multiple indicators that people and search engines use when making judgments. If a site can answer 'yes' to more than 90% of the below questions then it most cases it can be said to be a high quality website. However, even a single 'no' (slow to load, duplicate content, too many ads, etc.) can be sufficient for it to get classed as low quality.



REMEMBER

A website is the face of every business and it must be unique and relevant to the organization. Also it must be able to engage the viewer and should be easy to navigate. The website is also your professional image; it must have relevant and easy to access information.

The webpage designs should be responsive .Also, in the present age, it is important to have mobile friendly websites. Your website must also contain inks to your social media pages and also relevant FAQ, this keeps the viewer well informed about your business and the product. Most enterprises seek professional guidance to develop an efficient website.

Here are the top 22 key elements of a high quality website that you should be sure to consider:

1. Relevance and context. Developing well written and informative content for the user is one of the biggest key factors in creating a high quality website.

Quality content is original, purposeful, and correctly optimized information that people and search engines are driven to read, view, and share.

According to SearchMetrics, Google's algorithm recognizes high quality, relevant content, and rewards it with higher rankings.

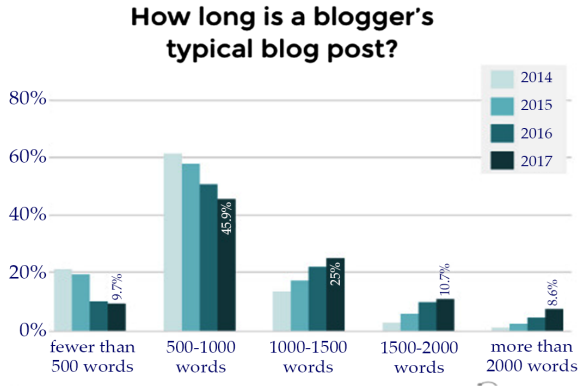
Focus on creating only the best high quality content that

you can. It will help you rank better and delight your customers.

2. Content length. Focus on developing longer content. The ideal blog post length from and SEO perspective is between 2,000 and 2,450+ words long.

However, the ideal blog post length from a reader's perspective is only 1,600 words long.

Sites with more words in the copy occupy higher ranking positions but it is important to find a balance between SEO and user readability.



The perfect blend will vary depending on your niche, competition, and audience.

3. Grammar and spelling. Ask yourself this question: Why would a search engine show a page of content with grammar and spelling errors higher in the rankings when other pages of error free content exist?

The answer: *They won't!*

Grammar and spelling mistakes make you look bad in the eyes of your customer, and the search engines may even penalize you for it. Always read your copy at least 2 times before you publish it or hire an editor to proofread and edit your post if you need help.

Flawless copy makes your website look professional and will earn you better rankings.

4. Readability. Readability is the ease in which text can be read and understood.

Use shorter sentences, paragraphs, and active web forms. Remove all clutter, unnecessary words and limit the use of adverbs and adjectives.

Use the Flesch reading ease formula to determine the readability of your text before publishing.

5. Formatting. 79% of users always scan web pages, according to a Nielsen study.

Not only are that but visitors less likely to read a post with poor formatting. High

quality content is easier to read, and suitable for scanning and skimming.

Use H1, H2, H3, etc. tags, number lists, and bullet points to break down your content.

Keep sentences and paragraphs short. Use **bold** and *italics* to highlight important parts so that they are highly visible as people scan.

6. Images and video. Include images or videos in every piece of content that you publish.

Web pages with images and videos are more engaging for visitors and rank better in Google too (according to SearchMetrics).

Keep in mind that web pages in top rankings have an average of 7 images on their page so be sure to use at least a few.

7. Expertise. High quality pages and websites need enough expertise to be authoritative and trustworthy on their topic.

The expertise of the author is a *critical factor* for any content to be considered high quality.

People want to read posts from experts that can dig into a topic and explain it. Focus on writing detailed, well-researched posts and give examples to support your points.

8. Social media shares. High quality websites have social media buttons present on their pages.



Place your social media sharing icons visibly on the page and include call-to-action for people to share.

9. Internal and external links. Linking to valuable internal and external resources not only delights your readers but will also help you rank better.

9 out of 10 sites at position 1 in SERPS have at least one internal self-referencing link.

Focus on building a nice internal link architecture. The URL that you link to and

the anchor text need to be relevant to your content.

Never link to unrelated pages or you may be penalized by Google and it leads to poor user experience overall.

10. Quality of comments. Content with a high number of comments is perceived as high quality.

On the other hand, spammy unrelated comments might hurt your rankings and make you look bad in the eyes of your visitors.

Make sure you moderate your comments and leave thoughtful responses to engage your users to do the same.

Quality comments help you rank better and engages your readers with the content.

11. Limit advertisements. While advertisements may be a nice form of income for you, they aren't popular with your visitors.

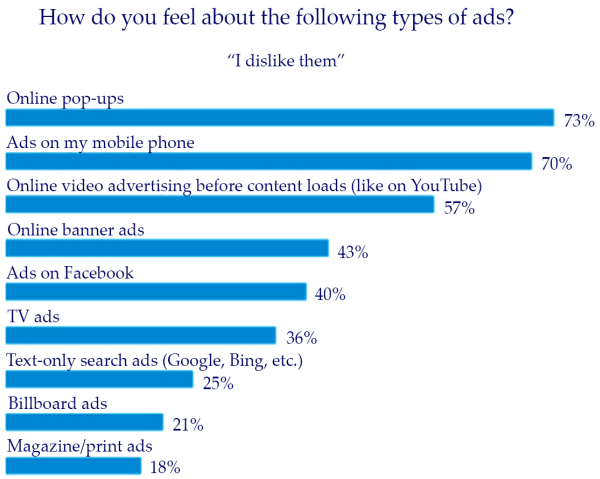
How much do you rely on ads to make a profit?

If it's just a small percentage, I recommend getting rid of them altogether.

If you're an ecommerce site or have a website that makes money from other revenue streams, ads aren't always necessary.

But say you run a blog and ads are your primary income. In that case, you'll need to keep them as limited as possible.

Take a look at the types of ads people dislike the most:



Take these numbers into consideration.

Avoid popup ads, and use minimal banner ads.

Although 43% of customers still dislike banners, it's not as high of a number compared to some other options.

12. Customer service that's easily accessible. If someone visiting your site has a question or problem, they shouldn't have to hunt for customer service options.

This should be readily available.

When customer service is unreachable, it makes the visitor feel uneasy.

Especially if it's during normal business hours.

They even have a recommended option.

People love to have choices.

Not everyone wants to pick up the phone.

It's great when companies have customer service available via online chat.

If you can swing it, give it a try.

13. All your contact information. This should go without saying, but you'd be surprised how often I can't find contact information on websites.

When I see that, I think it's sketchy.

What are they trying to hide by withholding their phone number?

Make sure your site has:

- physical address
- email address
- phone number
- links to social pages (Facebook, YouTube, LinkedIn)

Failure to do so will make your page appear untrustworthy.

14. Reviews and testimonials. Showcasing customer testimonials on your website helps generate social proof.

This is especially true if you can get a testimonial from an expert in your industry.

You should also have a place on your site where customers can leave reviews.

While good reviews are obviously what you're looking for, some unfavorable comments may actually boost your credibility as well.

If all customer feedback on your website is positive, it may appear fake.

Even if some people didn't have the best experience with your business, allowing them to leave a review for others to read will establish trust.

It also helps prove you're an actual business and not a scam.

Interact with the customers who left a review on your site.

This will help build credibility as well.

15. Security badges. What kind of security measures are you taking to protect users who visit your website?

Showcase those badges on each page.



Studies show that people trust the Norton AntiVirus seal the most compared to other badges.

If you use Norton, proudly display that badge on your site.

If you're looking for services to improve your site security, Norton may not be a bad place to start based on this information.

16. Validation from other media sources. Have you been featured in a magazine, newspaper, or on a website?

Any positive press about your company should be proudly displayed on your site.

If established media sources have verified your business, it will increase your legitimacy in the eyes of anyone who visits your website.

Find a good spot on your page to add any videos, screenshots, or links to all those stories.

17. Awards and achievements. Your website is a great place to show off any awards or achievements.

Whether it's local, regional, or national, anything helps.

Even if you won an award a couple of years ago, put it up on your website.

Showcasing awards from the past shows you've been credible for a while.

It establishes your company's history over time.

Companies that have been in business for longer periods tend to be well established and appear more credible than those that just started.

If you've been operating since 1950, don't be afraid to plaster that fact on your website.

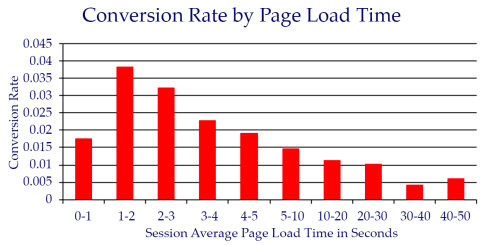
18. Ease of navigation. Customers shouldn't struggle to find what they're looking for on your website.

The menu options should be limited so it's not too overwhelming.

Adding a search bar so your readers can look for something specific is a great way to improve your navigation as well.

All of this helps enhance the user experience, which helps with your credibility score.

19. Page loading speed. The faster your page loads, the higher your conversion rates will be.



It's that simple.

Web hosting service is a type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web.

Don't try to find the cheapest **web hosting** service on the market. You get what you pay for.

It's worth it to pay a little extra to avoid technical glitches and always have fast loading times.

20. Clearly state all policies. Don't assume website visitors know your company policies.

All of these should be clearly stated on your website.

This will help you from a legal perspective as well in case there is a dispute.

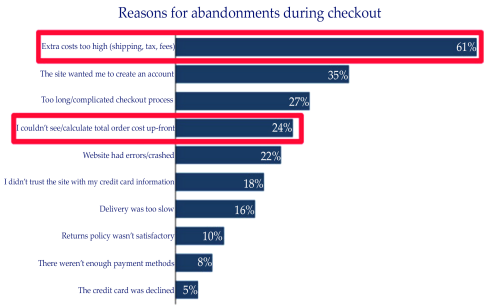
Make sure things such as your return policy or money back guarantee are outlined in detail.

If you're an ecommerce business, consumers may be hesitant to shop if they don't think you stand behind your product.

21. Be upfront about your prices. Don't try to sneak hidden fees past your customers.

It's shady.

Let's look at the top reasons for shopping cart abandonment:



Don't wait until the last minute to tell customers you're charging them tax, shipping, or other fees.

You should have all your prices clearly listed on the website.

Adding extra costs in the shopping cart could make the customer think you're trying to sneak one by them.

It's just not good business practice.

You also shouldn't say things like "Contact us for pricing."

Why wouldn't you just list your prices?

What are you trying to hide?

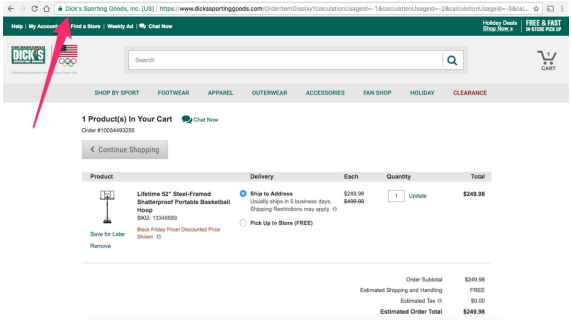
Those are questions that will go through the customer's mind if you do that.

22. Secure the checkout process. Refer back to that graph we just looked at regarding shopping cart abandonment.

Note 18% of those respondents said they didn't trust the website with their credit card information.

In addition to adding security badges to your page, you have to make sure your checkout procedure is secure.

Look at this example from Dick's Sporting Goods:



The secure link will make their customers feel comfortable about entering their personal information, including a credit card number.

◑ Features of Website

Nowadays, website are being used by companies to advance their business and mark their corporate identities in the global market with several products to the customers. The medium of website emerged as effective and most efficient tools of communication. The image of website solely depends upon the layout, structure and compilation of company's content. Each of the websites planned in a precise manner to cater specific need. The web designing holds the key and important aspect of conducting global business.

Web designing differs according to the nature of business. The definite designing based on the need and demand of business helps in segregation of one field from another. This provides advantage to the users as well as web designers as they have compile content in accordance to need. The potential of website depends on the traffic it invites. The well-structured websites invites more traffic while fuzzy websites repel most people from the site.

Color, typography, illustration, and interaction

Web pages usually include information such as the colors of text and backgrounds and very often contain links to images and other types of media to be included in the final view. Layout, typographic and color-scheme information is provided by Cascading Style Sheet (CSS) instructions, which can either be embedded in the HTML or can be provided by a separate file, which is referenced from within the HTML.

The latter case is especially relevant where one lengthy style sheet is relevant to a whole website: due to the way HTTP works, the browser may only download it once from the web server and use the cached copy for the whole site.

Images are stored on the web server as separate files, but again HTTP allows for the fact that once a web page is downloaded to a browser, it is quite likely that related files such as images and style sheets will be requested as it is processed. An HTTP 1.1 web server will maintain a connection with the browser until all related resources have been requested and provided. Web browsers usually render images along with the text and other material on the displayed web page.

Scrolling

Web pages will often require more screen space than is available for a particular display resolution.

Most modern browsers will place a scrollbar (a sliding tool at the side of the screen that allows the user to move the page up or down, or side-to-side) in the window to allow the user to see all content.

Scrolling horizontally is less prevalent than vertical scrolling, not only because such pages often do not print properly, but because it inconveniences the user more so than vertical scrolling would (because lines are horizontal; scrolling back and forth for every line is more inconvenient than scrolling after reading a whole screen; most computer keyboards have page up and down keys, and almost all computer mice have vertical scroll wheels, the horizontal scrolling equivalents are rare). When web pages are stored in a common directory of a web server, they become a website.

Web pages do not have a fixed length as in a paper page, and they can vary in length. The width of a web page varies depending on the size of the display so it leads to web pages of different lengths. For long web pages, information flow and presentation is quite critical. If the web page is longer and the information on the top is undesirable to the user, the probability of reading further down is low. However, both longer and shorter web pages have their own pros and cons.

The initial viewing area of a web page is known as being “above page fold”. The content above the page fold is important as users use it to evaluate if they have come to the right page. It is important to have content above the page fold that keeps the user interested enough that they scroll down. The information foraging theory describes that once a user has deemed the part above the fold of a page valuable they are more likely to deem the rest of the page valuable.

Build Basic Webpage Templates

Commercial websites are built from templates. You or your designer will create a template that constructs each part of a typical webpage, with a “hole” in the center for the unique page content. This takes many hours to build from scratch, but it’s worth it. Now you can create page after page from the template. For each webpage you’ll insert a page title, meta tag content, a headline, and the text content, each in its appropriate spot. Have fun!

But let me take this a step further. Take a look at the sample webpage from my site. I’ve simplified it here, but the article content is surrounded by four sections, each of which is shown when a web browser comes to the web page:

- **top.ssi** — inserts the masthead graphic, a banner ad, and some of the “tabs” navigation system at the top of the page. This is a separate file, called “top.ssi” that is inserted at the top.
- **menu.ssi** — inserts the complex left-side menu plus a database search feature.

- **bottom.ssi** — inserts a subscription form for my newsletter, plus more navigation links, copyright and trademark information.
- **right.ssi** — inserts cover shots of my books, plus links to purchase my e-books and affiliate links to products and services in the field of web marketing and e-commerce.

Each of these files is called a Server Side Include (SSI) file. On the webpage a single line of code calls one of these files and places it where it belongs on the page. Here's what it the code looks like:

<!–#include virtual="/ssi/top.ssi"–>

The beauty of this kind of modular system is that a site built with SSIs can be modified or completely altered by just changing one of the SSI files and uploading it to the server. Now *all* the webpages in the entire system reflect the change. When I discovered how to do this it cut my maintenance time dramatically. Yes, it takes a learning curve to make it work, but it's well worth the time you spend!

It is possible, of course, to use a template for your pages that doesn't employ SSIs. But if you anticipate a site that could grow to more that 8 to 10 pages, you're much better off building your site with SSIs. If your designer doesn't know how to use SSIs, find another designer.

Modern websites control the font sizes and colors using Cascading Style Sheets (CSS). When you change the font size on a single master CSS file, it changes the fonts and colors in *all* your webpages. Cool! Make sure your website designer builds webpages using a single CSS file, since it saves maintenance costs in the long run.

Server Side Includes (SSI) is a simple interpreted server-side scripting language used almost exclusively for the web.

The design decisions that you need to consider here are many, since they involve every detail of the look and feel of your basic template. Hopefully, you'll decide to employ both **Server Side Includes (SSIs)** and Cascading Style Sheets (CSS) that make your entire site easy to modify and maintain. Also consider features available with XHTML.

Saving

While one is viewing a web page, a copy of it is saved locally; the copy is being viewed.

Depending on the browser settings, the copy may be deleted at any time, or stored indefinitely, sometimes without the user realizing it.

Most GUI browsers provide options for saving a web page more permanently. These may include

- save the rendered text without formatting or images, with hyperlinks reduced to plain text;
- save the HTML as it was served: overall structure is preserved, but some links may be broken;
- save the HTML with relative links changed to absolute ones so that hyperlinks are preserved;
- save the entire web page: all images and other resources including stylesheets and scripts are downloaded and saved in a new folder alongside the HTML, with links referring to the local copies: Other relative links are changed to absolute;
- save the HTML as well as all images and other resources into a single MHTML file. This is supported by Internet Explorer and Opera. Other browsers may support it if a suitable plugin is installed.

In addition to the option to print the currently viewed web page to a printer, most operating systems allow applications such as web browsers to "print to a file" which can be viewed or printed later. Some web pages are designed, for example by use of CSS, so that hyperlinks, menus and other navigation items, which would be useless on paper, are rendered into print with this in mind. Sometimes, the destination addresses of hyperlinks may be shown explicitly, either within the body of the page or listed at the end of the printed version. It can be specified in CSS that non-functional menus, navigational blocks and other items ought to simply be absent from the printed version.

Search Engine Savvy Navigation Systems

Navigation systems are built to help actual humans find their way around your website. But these navigation systems had better be designed carefully or the search engines will throw up their hands in disgust, with the result that actual humans will never *get* to your website. Search engines need a chain of hypertext links — starting at your homepage — that will take them, page by page, to every webpage in your entire site. But let me explain two common navigation design problems that can disrupt search engine indexing of your site:

- **Frames** produce a navigation system where the menu on the left scrolls independently of the page content on the right. Unfortunately, frames can wreak havoc with search engines. (a) Unless you are careful to include <NOFRAMES> tags, search engines may not be able to find the content pages. (b) Even if search engines do find your content pages, these pages can show up in response to a search engine query all by themselves, without the navigation system and links necessary for a visitor to find the rest of your website. Don't use frames. If your current site has frames, make plans to rebuild the site without them. A menu constructed from SSIs is just as easy to maintain — even easier, once you learn how to do it.
- **JavaScript and Flash** are programming languages that can make very classy, animated menu systems. For example, a menu item might have a drop-down sub-menu that will wow your visitors (you hope). The problem is that if JavaScript and Flash systems replace plain hyperlinks, the search engine may not be able to find the underlying pages. Most search engines have posters on their walls saying, "I don't do Flash." Stubborn creatures, these search engines. One solution: retain your fancy menus, but include hypertext links at the bottom of the page to your sectional pages, with links on your sectional pages to all the subpages in that section. You can also submit a site map webpage to the search engines that contains a link to every page on your site.

What are the design decisions regarding search engines? A commitment to design (a) each web page and (b) **the site navigation system** with search engines in mind. This is a marketing, not a techie priority, so you may have to insist that your website designers work with search engines on their minds.

URLs

A Uniform Resource Locator (URL), or web address, is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it. A URL is a specific type of Uniform Resource Identifier (URI), although many people use the two terms interchangeably. A URL implies the means to access an indicated resource and is denoted by a protocol or an access mechanism, which is not true of every URI. URLs occur most commonly to reference web pages (http), but are also used for file transfer (ftp), email (mailto), database access (JDBC), and many other applications.

Web pages are typically becoming more dynamic. A dynamic web page is one that is created by the web server when it is requested by the visiting user's browser. These types of web pages typically do not have a permalink, or a static URL, associated with them. This practice is intended to reduce the amount of

static pages in lieu of storing the relevant web page information in a database. This can be seen in forums, online shopping websites, and on Wikipedia. Some search engines may have a hard time indexing a web page that is dynamic, so static web pages can be provided in those instances.

LEARNING ACTIVITY: STEPS FOR PLANNING A NEW WEBSITE

One of the most important stages and oftentimes consuming tasks of building a website is the Planning Phase. If you neglect to plan a project thoroughly, the odds of an unhappy finished project are at a greater risk.

Planning a new website? Here are nine steps to follow that will ensure your new website delivers effectively for your business and delights your users.

Read more details on:

https://www.hallaminternet.com/9-steps-for-planning-a-new-website/

Give an overview on search engine savvy navigation systems.

Define your Direction: On a separate piece of paper, explain the outlets of the different types of the terms according to criteria and quality of websites planning:

Terms	***Outlets***
Uniform Resource Locator (URL)	
World Wide Web	
Social media buttons	
Cascading Style Sheet (CSS)	
Server Side Includes (SSIs)	

ROLE MODEL



MARC LOWELL ANDREESSEN

He is the co-author of Mosaic, the first widely used Web browser; co-founder of Netscape; and co-founder and general partner of Silicon Valley venture capital firm Andreessen Horowitz.

BIOGRAPHY

Marc Lowell Andreessen (born July 9, 1971) is an American entrepreneur, investor, and software engineer. He is the co-author of Mosaic, the first widely used Web browser; co-founder of Netscape; and co-founder and general partner of Silicon Valley venture capital firm Andreessen Horowitz. He founded and later sold the software company Opsware to Hewlett-Packard. Andreessen is also a co-founder of Ning, a company that provides a platform for social networking websites. He sits on the board of directors of Facebook, eBay, and Hewlett Packard Enterprise, among others. Andreessen was one of six inductees in the World Wide Web Hall of Fame announced at the First International Conference on the World-Wide Web in 1994.

Marc Andreessen has been called many things in his young life—from boy wonder and "Golden Geek" to visionary and "Internet Evangelist"—all because he helped develop an Internet browser called Netscape Navigator. In the days when the Internet and World Wide Web were in their infancy, Andreessen's ability to simplify the information superhighway not only made him millions of dollars but thrust him into cutthroat competition with huge rival Microsoft. Although he later quit Netscape after it was acquired by America Online (AOL), Andreessen turned his attention to forming a new company that would continue to bring the Internet to millions of people. "Right now, today, with a little luck and brains and timing, any kid with a computer can do what Netscape has done. There are no barriers to entry anymore. Any kid can spark a revolution."

Mosaic, the Monster Browser

Andreessen and Bina demonstrated Mosaic's capabilities in January 1993, proud of their accomplishments but having no idea that their program, called a "browser," would literally change the world. What Mosaic did was help users browse, or navigate, through the thousands and eventually millions of Web addresses and locations available on the Internet. At the time the Web was still a vast, untapped wilderness; while some folks had begun using computers at work, the boom for home users had yet to begin.

Marc Andreessen developed his first computer program while he was in sixth grade at his school's computer lab. It was for a virtual calculator. Unfortunately for the budding software designer, power outages erased the program before he could save or copy it.

A few months after finishing Mosaic, Andreessen received his bachelor's degree in computer science from the University of Illinois. Mosaic was then offered over the Internet, and within the first year was downloaded by two million users. After graduation, Andreessen took a job in southern California's famed Silicon Valley at Enterprise Integration Technologies, a company that developed Internet security products. While there, he received an e-mail from a former computer science professor at Stanford University named Jim Clark.

Clark had left Stanford to form Silicon Graphics, Inc., a billion-dollar computer hardware and software graphics company. He had recently left Silicon Graphics and wanted to meet with Andreessen to discuss a new business venture. Andreessen agreed, and the two ended up on Clark's yacht discussing Mosaic and its importance to the Web and beyond. With Andreessen's technical expertise, Clark's business background, and start-up funds of $4 million, the two agreed to form a partnership. They launched Mosaic Communications Corporation in April 1994.

KNOWLEDGE CHECK

1. **Write the advantages and importance of web browsers.**
2. **Elaborate the useful elements of web page.**
3. **How to Meta description helps people decide to click on web page? Explain.**
4. **Give an overview on static and dynamic web page**
5. **What are the differences between client and server-side scripting?**
6. **Determine the features of website.**
7. **What is a CMS in web design?**
   a) Content Management System
   b) Creative Management System
   c) Content Mixing System
   d) Creatives Managerial System
8. **To make your website mobile friendly, you can make your website**
   a) Responsive
   b) Reactive
   c) Fast Loading
   d) Light
9. **What does CSS stand for**
   a) Current Style Sheets
   b) Current Sheets Style
   c) Cascading Style Sheets
   d) Cascading Sheets Style
10. **Which of the following statements is false**
    a) You can make a website without using HTML
    b) You can make a website without using PHP
    c) You can make a website without using CSS
    d) You can make a website without using Javascript

11. What is WordPress

a) It is a software used to press text

b) It is a text formatting software

c) It is a CMS (Content Management System)

d) It is mail service

12. Which of the following is true about Javascript

a) It is a server side scripting language

b) It is client side scripting language

c) It is a Software

d) It is a database

13. Which of the following is true about PHP

a) It is a server side scripting language

b) It is client side scripting language

c) It is a Software

d) It is a database

14. Which of the following is true

a) You need a server to host your website files

b) You don't need a server to host your website files

c) You can create a website without using HTML

d) You can't create a website without a CMS

15. Which of the following software's could be used to build a website

a) Power Point

b) Excel

c) Dream Weaver

d) ERP

16. Which of the following is not a CMS

a) WordPress

b) Drupal

c) Magneto

d) SAP

17. Which of the following statements is false about hosting

a) Shared hosting is cheaper than dedicated hosting

b) Shared hosting is safer than dedicated hosting

c) Dedicated hosting is safer than shared hosting

d) Though more expensive than shared hosting, Dedicated hosting is more secure

18. Which of the following is not a web hosting company

a) Hostgator

b) Blue Host

c) WPX Hosting

d) Facebook

19. Which of the following statements is true

a) The web designer shouldn't just be concerned about the looks but also about user interface

b) Usability is very important in web design

c) a and b

d) None of the above

20. Which types of criteria and quality are requires for websites development?

REFERENCES

1. Blanck, Peter. eQuality: The Struggle for Web Accessibility by Persons with Cognitive Disabilities, Cambridge Disability Law and Policy Series, 2015.
2. Byrne, Jim. 60 hot to touch Accessible Web Design tips – the tips no web developer can live without!, Jim Byrne, 2006, (ISBN: 978-1-4116-6729-7).
3. Chisholm, and May. Universal Design for Web Applications: Web Applications That Reach Everyone, O'Reilly Media, 2008.
4. Duckett, Jon. Accessible XHTML and CSS Web Sites Problem Design Solution, Wrox, 2005.
5. Gay, Greg et al.Introduction to Web Accessibility, Ryerson University Pressbooks, 2019.
6. Hassell, Jonathan. Including your missing 20% by embedding web and mobile accessibility, BSI British Standards Institution, 2014.
7. Horton, Sarah. Access by Design: A Guide to Universal Usability for Web Designers, New Riders Publishing, 2005.
8. Hricko, Mary (Editor). Design and Implementation of Web-Enabled Teaching Tools, Idea Group Publishing, 2002.
9. Kirkpatrick, Andrew et al. Web Accessibility: Web Standards and Regulatory Compliance, friends of ED, 2006.
10. Paciello, Michael G. Web Accessibility for People With Disabilities, CMP Books, 2000.
11. Pickering, Heydon. Apps For All: Coding Accessible Web Applications, Smashing Magazine GmbH, 2014.
12. Slatin, John M. and Rush, Sharron. Maximum Accessibility: Making Your Web Site More Usable for Everyone, Addison Wesley Professional, 2002.
13. Sydik, Jeremy. Design Accessible Web Sites, Pragmatic Bookshelf, 2007.
14. Thatcher, Jim et al. Web Accessibility: Web Standards and Regulatory Compliance, friends of ED, 2006.
15. Watson, Leonie et al. Practical Approaches For Designing Accessible Websites, Smashing Magazine GmbH, 2015. (ISBN (EPUB: 978-3-945749-23-4).

CHAPTER 2

CREATING WEB PAGES WITH HTML



INTRODUCTION

HyperText Markup Language (HTML) is a type of computer language that is used to create pages that can be posted on the Internet or sent via email. Although it might seem complex to many people, it is considered to be a relatively simple language. All text, graphics, and design elements a page designed with this language are "tagged" with codes that instruct the web browser or email program how to display those elements. The tags also provide layout and formatting information so that the web page or email will look as close as possible to the way its designer intended it to look. For the novice website designer or anyone else who needs to know a certain code or wants to learn how to create an entire website, there are many software utilities, programs and websites that can assist him or her in writing HTML code.

SCENARIO: THE BASICS OF HYPERTEXT MARKUP LANGUAGE

HTML defines the structure and layout of a Web document by using a variety of tags and attributes. The correct structure for an HTML document starts with <HTML><HEAD>(enter here what document is about)<BODY> and ends with </BODY></HTML>. All the information you'd like to include in your Web page fits in between the <BODY> and </BODY> tags.

What skills will you need?

In order to work effectively and creating web pages with html, you must know:

- Fundamental of html
- Html structure

FUNDAMENTAL OF HTML

HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser page. The markup tells the Web browser how to display a Web page's words and images for the user. Each individual markup code is referred to as an element (but many people also refer to it as a tag). Some elements come in pairs that indicate when some display effect is to begin and when it is to end.

HTML is the language used to create webpages. "Hypertext" refers to the hyperlinks that an HTML page may contain. "Markup language" refers to the way

tags are used to define the page layout and elements within the page.

Below is an example of HTML used to define a basic webpage with a title and a single paragraph of text.

```
<!doctype html>
<html>
<head>
<title>TechTerms.com</title>
</head>
<body>
<p>This is an example of a paragraph in HTML.</p>
</body>
</html>
```

The first line defines what type of contents the document contains. "<!doctype html>" means the page is written in HTML5. Properly formatted HTML pages should include <html>, <head>, and <body> tags, which are all included in the example above. The page title, metadata, and links to referenced files are placed between the <head> tags. The actual contents of the page go between the <body> tags.

The web has gone through many changes over the past few decades, but HTML has always been the fundamental language used to develop webpages. Interestingly, while websites have become more advanced and interactive, HTML has actually gotten simpler. If you compare the source of an HTML5 page with a similar page written in HTML 4.01 or XHTML 1.0, the HTML5 page would probably contain less code. This is because modern HTML relies on cascading style sheets or JavaScript to format nearly all the elements within a page.

- HTML is the standard markup language for creating Web pages.
- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

Example

```
<!DOCTYPE html>
```

```
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

The <!DOCTYPE html> declaration defines this document to be HTML5

The <html> element is the root element of an HTML page

The <head> element contains meta information about the document

The <title> element specifies a title for the document

The <body> element contains the visible page content

The <h1> element defines a large heading

The <p> element defines a paragraph

◑ HTML Tags

Any document starts with a heading. You can use different sizes for your headings. HTML also has six levels of headings, which use the elements <h1>, <h2>, <h3>, <h4>, <h5>, and <h6>. While displaying any heading, browser adds one line before and one line after that heading.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Heading Example</title>
</head>
<body>
```

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

This will produce following result:

This is heading 1

This is heading 2

This is heading 3

This is heading 4

This is heading 5

This is heading 6

Paragraph Tag

The tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag as shown below in the example:

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Paragraph Example</title>
</head>
<body>
<p>Here is a first paragraph of text.</p>
<p>Here is a second paragraph of text.</p>
```

```
<p>Here is a third paragraph of text.</p>

</body>

</html>
```

This will produce following result:

Here is a first paragraph of text.

Here is a second paragraph of text.

Here is a third paragraph of text.

Line Break Tag

Whenever you use the <br /> element, anything following it starts from the next line. This tag is an example of an empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <br /> tag has a space between the characters br and the forward slash. If you omit this space, older browsers will have trouble rendering the line break, while if you miss the forward slash character and just use <br> it is not valid in XHTML

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Line Break  Example</title>

</head>

<body>

<p>Hello<br />

You delivered your assignment ontime.<br />

Thanks<br />

Mahnaz</p>

</body>
```

```
</html>
```

This will produce following result:

Hello

You delivered your assignment ontime.

Thanks

Mahnaz

Centering Content

You can use tag to put any content in the center of the page or any table cell.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>Centring Content Example</title>
</head>
<body>
<p>This text is not in the center.</p>
<center>
<p>This text is in the center.</p>
</center>
</body>
</html>
```

This will produce following result:

This text is not in the center.

This text is in the center.

Horizontal Lines

Horizontal lines are used to visually break up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly.

For example you may want to give a line between two paragraphs as in the given example below:

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Horizontal Line Example</title>

</head>

<body>

<p>This is paragraph one and should be on top</p>

<hr />

<p>This is paragraph two and should be at bottom</p>

</body>

</html>
```

This will produce following result:

This is paragraph one and should be on top

This is paragraph two and should be at bottom

Again <hr /> tag is an example of the empty element, where you do not need opening and closing tags, as there is nothing to go in between them.

The <hr /> element has a space between the characters hr and the forward slash. If you omit this space, older browsers will have trouble **rendering** the Horizontal line, while if you miss the forward slash character and just use <hr> it is not valid in XHTML

Preserve Formatting

Sometimes you want your text to follow the exact format of how it is written in the HTML document. In those cases, you can use the preformatted tag <pre>.

Any text between the opening <pre> tag and the closing </pre> tag will preserve the formatting of the source document.

Rendering is the automatic process of generating a photorealistic or non-photorealistic image from a 2D or 3D model (or models in what collectively could be called a scene file) by means of computer programs.

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Preserve Formatting Example</title>

</head>

<body>

<pre>

function testFunction( strText ){

   alert (strText)

}

</pre>

</body>

</html>
```

This will produce following result:

```
function testFunction( strText ){

   alert (strText)

}
```

Try using same code without keeping it inside <pre>...</

pre> tags

Nonbreaking Spaces

Suppose you want to use the phrase "12 Angry Men." Here you would not want a browser to split the "12, Angry" and "Men" across two lines:

An example of this technique appears in the movie "12 Angry Men."

In cases where you do not want the client browser to break text, you should use a nonbreaking space entity; instead of a normal space. For example, when coding the "12 Angry Men" in a paragraph, you should use something similar to the following code:

Example

<!DOCTYPE html>

<html>

<head>

<title>Nonbreaking Spaces Example</title>

</head>

<body>

An example of this technique appears in the movie "12 Angry Men."

</body>

</html>

◑ Naming Files and Folders

If the page you are creating is going to be the home page for your site, you should name it index.html.

You might think of a home page as the page visitors arrive at first when they come to your site. And you are correct. But how does a browser know that when your visitors enter "pat.com" (assuming for the moment that you are Pat and pat.com is your site), which of the many HTML pages at your site to open first?

In other words, index.html serves as the home page within any folder. And in your site's root folder (your URL), the index.html file serves as the site home page.

These tips are intended to save you time as you create and edit your Web pages.

- HTML file names should end in the suffix ".html" or ".htm". Whichever suffix you decide to use, please talk to your team members and use the same suffix on all your project html pages. Otherwise it is hard to remember which is which and your project will likely have more broken links in it. Example:

myhomepage.html

In addition, use a consistent and standard suffix on other file formats indicating the format (.jpg, .gif, .wav, . . .).

- The file name should be no more than 32 characters, including the ".html" or ".htm" file suffix. Because your filename will become part of your Web page's URL, it is better to use a slightly longer file name that will have an easier URL to remember than a jumble of unpronounceable letters. For example: Which of these would be easier to remember and give to a friend as a URL: hurricanes.html or hrrcns.html ? More examples:

kennedy.html

shakespeare.htm

- The first character of the file name should be a letter.
- **File names and folder names** should contain **only letters, digits, and underscores—no spaces, punctuation, or funny characters.** We want to really emphasize this because people who do not follow these guidelines end up spending much more time fixing broken links, which can be a source of frustration. This is a «conservative» guideline with the goal that you should not have to spend extra time redoing your links if you move your files from one type of computer



REMEMBER

Browsers detect your home page by looking for a file called index.html, which plays a special role. By default, when a browser is directed to a URL (like davidkarlins.com), the browser looks for a file called index.html.

to another. This has been an issue for some past LInC participants when moving or copying their LInC project Web pages from the LInC class Web site to their school Web site.

- Whenver possible, make relative links for links from your project Web pages to other Web pages that are in that same project. This will make it possible to move your LInC project Web site to your school Web server without having to fix dozens of links. A relative link to a file in the same folder as the web page you are editing looks like: "shakespeare.html". In contrast, an "absolute" link looks like: "http://ed.fnal.gov/lincon/projects/essays/shakespeare.shtml". You can see that if your project was copied onto a different web server, you would have to fix all those absolute links. You will need to use absolute links when linking to Web pages for external resources that are not inside your project folder.
- We **strongly** suggest using all lower case letters because people make fewer mistakes. Web authors (such as yourself) make fewer mistakes when creating links to your pages. Web readers make fewer mistakes when typing in the addresses to access your pages.
- Use relatively short names that are indicative of the content of the page. Nothing "cute" because you probably won't remember it six months from now when you need to edit it.
- Also name your files with images, sounds, etc., with names that will help you remember what is in them. Participants who name their images something like picture1.jpg through picture20.jpg are frequently frustrated later because they have to open five or ten files later to find the image they want to edit. If you instead name your images something more descriptive, they will be easier to find later. Examples:

lightbulb.gif

book.jpg

bluebullet.gif

redline.jpg

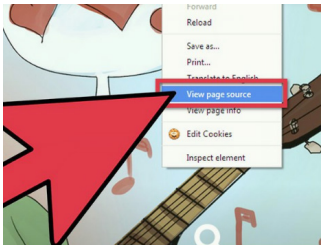
stopsign.gif

- Be sure to create a graphics folder (directory) inside the folder (directory) you are using for all your Web pages. It should be titled "graphics"--all lower case letters. Past participants who did not do this have told us they wish they did because it was harder to keep their files organized with so many pictures in with the Web pages.
- Name your anchors in your pages something descriptive. An anchor

named "#assessment" is easier to link to and tell others about than an anchor named "#anchor82904872".

◑ Create a Simple Web Page with HTML

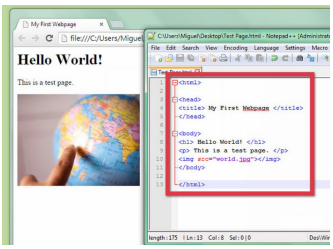
There are many good websites where you can learn HTML, the basics of creating a website by using HTML. With this basic introduction, you'll soon be able to make a webpage from scratch!



Using HTML

1. Understand what HTML is. HTML (Hyper Text Markup Language) is the coding language that makes web-pages, it is the skeleton of every web-page that exists. To see what it looks like, go into Internet Explorer, navigate to any website and right-click on the web-page selecting «View Source». You will see a page of code, and that is HTML. That code is what your browser sees and interprets into a human friendly web-page.

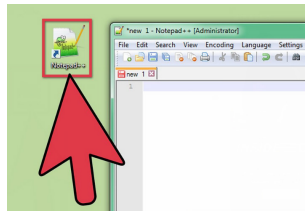
- In Google Chrome press F12.
- In Mozilla Firefox, press Ctrl + U to view the page's source code.
- In Safari, select View→View Source (or Option + Command + U).
- In Internet Explorer, the VIEW menu option is "source".



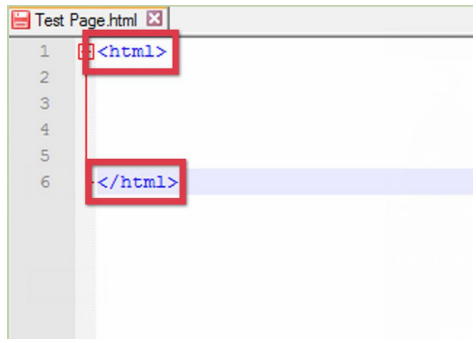
2. Make your initial web page as simple as possible**.** If you don›t, you›ll likely get overwhelmed by the syntax and the **script languages**.

Scripting language is a computer language with a series of commands within a file that is capable of being executed without being compiled.

- It's important to remember that you'll be writing your information between an opening HTML tag and a closing HTML tag. An opening tag looks like this: <___>. A closing tag looks like this: </___>. Eventually, the ___ is replaced with a code.



Go to Start, then "Programs" and then "Accessories." and click "Notepad." It is a lot easier if you use Notepad++ once you choose the HTML language, everything you write will be automatically connected with different colors - that way it will be a lot easier to correct possible mistakes.



Tell the browser what language you are using. On the first line, type <!DOCTYPE>, this should always be the first tag in your code. On the second line add <html>. It is the second tag you write that tells the computer you're starting a web-page. It will also be closed last, so at the end of the document, close it off by typing this: </html>. This ends the web page.



Add the heading of the page as shown.



Give your page a title. A title is important because it gives your users an idea on what the page is all about. Also, when users bookmark your site, that title is all they will see in their bookmark list. The title for HTML code is <title>. Close it off at the end of your title by writing </title> The title is going to show on the tab, don't expect it to be the title of the actual website.



Work on the body of the page**.** Type <body> to open the body tag. Then close the body tag by typing </body>. The bulk of the information for your web-page goes between <body> and </body>.

- To give your web-page a background color, you can add a style to the body. To add a background color, add this tag <body style="background-color:silver">. You can try a different color or even a hex code. The

words in the quotation marks are known as "attributes." They must be surrounded by quotation marks!



Write some text between the body tags.

- To make the text go to the next line (like pressing "Enter" on your keyboard), write
- Want to add a marquee, otherwise known as a word that moves across a screen? Simply type <marquee>TEXT GOES HERE</marquee>.



Add some pictures. If you want to put a picture from the Internet onto your web page, the HTML code for pictures is <img src="URL"> . The closing tag is: . However, the closing tag is optional.



Check to make sure all of your tags are closed. Your webpage should look something like this:

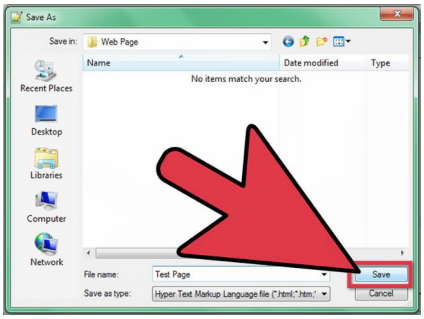
<title>My Web-page</title>

<body bgcolor="yellow">

I love Rose because

<marquee>It's the best flower in the world!</marquee>

<img src="http://www.rose.com/skins/common/images/roseimg_logo_5.gif"></body>



Save your work. Go to "save as", put a filename with an .html extension (such as "testfile.html") and choose "all files" or "text" under file type. It won't work if both are not done. Go find the page wherever you saved it, double click it, and your default web browser should open up your very own web-page.

◑ HTML Attribute

We have seen few HTML tags and their usage like heading tags <h1>, <h2>, paragraph tag <p> and other tags. We used them so far in their simplest form, but most of the HTML tags can also have attributes, which are extra bits of information.

An attribute is used to define the characteristics of an HTML element and is placed inside the element's opening tag. All attributes are made up of two parts: a name and a value:

- The name is the property you want to set. For example, the paragraph <p> element in the example carries an attribute whose name is align, which you can use to indicate the alignment of paragraph on the page.
- The value is what you want the value of the property to be set and always put within quotations. The below example shows three possible values of align attribute: left, center and right.

Attribute names and attribute values are case-insensitive. However, the World

Wide Web Consortium (W3C) recommends lowercase attributes/attribute values in their HTML 4 recommendation.

Example

```
<!DOCTYPE html>

<html>

<head>

<title>Align Attribute  Example</title>

</head>

<body>

<p align="left">This is left aligned</p>

<p align="center">This is center aligned</p>

<p align="right">This is right aligned</p>

</body>

</html>
```

This will display following result:

This is left aligned

This is center aligned

This is right aligned

◑ Core Attributes

The four core attributes that can be used on the majority of HTML elements (although not all) are:

- id
- title
- class
- style

The id Attribute

The **id** attribute of an HTML tag can be used to uniquely identify any element within an HTML page. There are two primary reasons that you might want to use an id attribute on an element:

- If an element carries an id attribute as a unique identifier it is possible to identify just that element and its content.
- If you have two elements of the same name within a **Web page** (or style sheet), you can use the id attribute to distinguish between elements that have the same name.

Web page is a document that is suitable for the World Wide Web and web browsers. A web browser displays a web page on a monitor or mobile device.

Let's use the id attribute to distinguish between two paragraph elements as shown below.

Example

```
<p id="html">This para explains what is HTML</p>
<p id="css">This para explains what is Cascading Style Sheet</p>
```

The title Attribute

The **title** attribute gives a suggested title for the element. They syntax for the **title** attribute is similar as explained for **id** attribute:

The behavior of this attribute will depend upon the element that carries it, although it is often displayed as a tooltip when cursor comes over the element or while the element is loading.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>The title Attribute Example</title>
</head>
<body>
<h3 title="Hello HTML!">Titled Heading Tag Example</h3>
```

```
</body>
</html>
```

This will produce following result:

Titled Heading Tag Example

Now try to bring your cursor over "Titled Heading Tag Example" and you will see that whatever title you used in your code is coming out as a tooltip of the cursor.

The class Attribute

The **class** attribute is used to associate an element with a style sheet, and specifies the class of element. You will learn more about the use of the class attribute when you will learn Cascading Style Sheet (CSS). So for now you can avoid it.

The value of the attribute may also be a space-separated list of class names. For example:

```
class="className1 className2 className3"
```

The style Attribute

The style attribute allows you to specify Cascading Style Sheet (CSS) rules within the element.

```
<!DOCTYPE html>
<html>
<head>
<title>The style Attribute</title>
</head>
<body>
<p style="font-family:arial; color:#FF0000;">Some text...</p>
</body>
</html>
```

This will produce following result:

Some text...

At this point of time, we are not learning CSS, so just let's proceed without bothering much about CSS. Here you need to understand what are HTML attributes and how they can be used while formatting content.

Internationalization Attributes

There are three internationalization attributes, which are available for most (although not all) XHTML elements.

- dir
- lang
- xml:lang

The dir Attribute

The **dir** attribute allows you to indicate to the browser the direction in which the text should flow. The dir attribute can take one of two values, as you can see in the table that follows:

Value	Meaning
ltr	Left to right (the default value)
rtl	Right to left (for languages such as Hebrew or Arabic that are read right to left)

Example

```
<!DOCTYPE html>

<html dir="rtl">

<head>

<title>Display Directions</title>

</head>

<body>

This is how IE 5 renders right-to-left directed text.

</body>

</html>
```

This will produce following result:

This is how IE 5 renders right-to-left directed text.

When *dir* attribute is used within the <html> tag, it determines how text will be presented within the entire document. When used within another tag, it

controls the text›s direction for just the content of that tag.

The lang Attribute

The lang attribute allows you to indicate the main language used in a document, but this attribute was kept in HTML only for backwards compatibility with earlier versions of HTML. This attribute has been replaced by the xml:lang attribute in new XHTML documents.

The values of the *lang* attribute are ISO-639 standard two-character language codes. Check HTML Language Codes: ISO 639 for a complete list of language codes.

Example

<!DOCTYPE html>

<html lang="en">

<head>

<title>English Language Page</title>

</head>

<body>

This page is using English Language

</body>

</html>

The xml:lang Attribute

The *xml:lang* attribute is the XHTML replacement for the *lang* attribute. The value of the *xml:lang* attribute should be an ISO-639 country.

Generic Attributes

Here's a table of some other attributes that are readily usable with many of the HTML tags.

Attribute	Options	Function
align	right, left, center	Horizontally aligns tags
valign	top, middle, bottom	Vertically aligns tags within an HTML element.
bgcolor	numeric, hexidecimal, RGB values	Places a background color behind an element

background	URL	Places a background image behind an element
id	User Defined	Names an element for use with Cascading Style Sheets.
class	User Defined	Classifies an element for use with Cascading Style Sheets.
width	Numeric Value	Specifies the width of tables, images, or table cells.
height	Numeric Value	Specifies the height of tables, images, or table cells.
title	User Defined	"Pop-up" title of the elements.

We will see related examples as we will proceed to study other HTML tags. For a complete list of HTML Tags and related attributes.

LEARNING ACTIVITY: GETTING STARTED WITH HTML

HTML is a language, which makes it possible to present information (e.g. scientific research) on the Internet. What you see when you view a page on the Internet is your browser's interpretation of HTML. To see the HTML code of a page on the Internet, simply click "View" in the top menu of your browser and choose "Source".

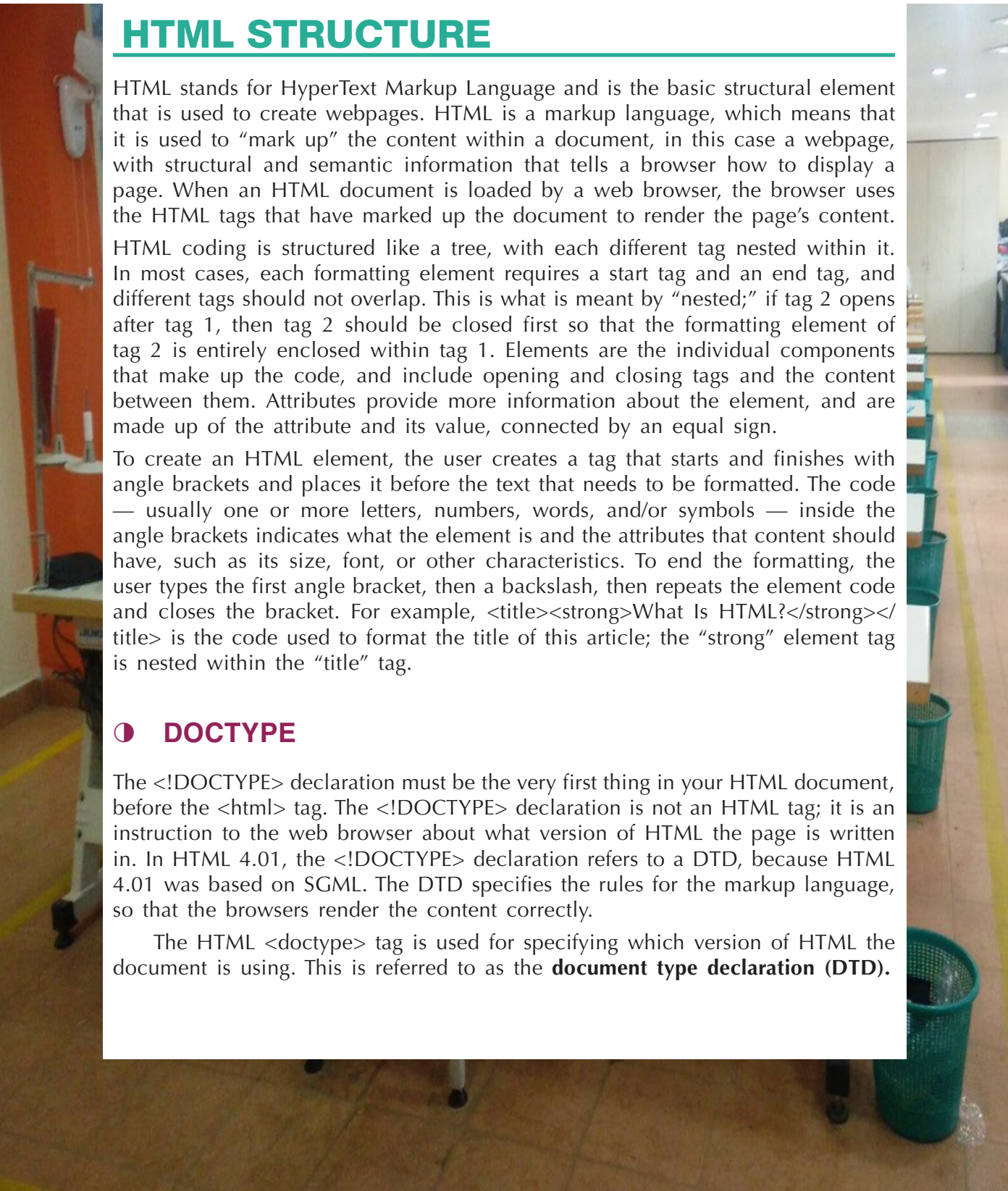
Read more details on:

https://horwitz.website/web-app-html-introduction/

Determine and execute the core attributes of html with a suitable example.

Define your Direction: On a separate piece of paper, explain the importance of the different types of the terms according to fundamental of html:

Terms	***Importance***
Source Document	
Notepad	
Internet browser	
Coding language	
World Wide Web Consortium	



HTML STRUCTURE

HTML stands for HyperText Markup Language and is the basic structural element that is used to create webpages. HTML is a markup language, which means that it is used to "mark up" the content within a document, in this case a webpage, with structural and semantic information that tells a browser how to display a page. When an HTML document is loaded by a web browser, the browser uses the HTML tags that have marked up the document to render the page's content.

HTML coding is structured like a tree, with each different tag nested within it. In most cases, each formatting element requires a start tag and an end tag, and different tags should not overlap. This is what is meant by "nested;" if tag 2 opens after tag 1, then tag 2 should be closed first so that the formatting element of tag 2 is entirely enclosed within tag 1. Elements are the individual components that make up the code, and include opening and closing tags and the content between them. Attributes provide more information about the element, and are made up of the attribute and its value, connected by an equal sign.

To create an HTML element, the user creates a tag that starts and finishes with angle brackets and places it before the text that needs to be formatted. The code — usually one or more letters, numbers, words, and/or symbols — inside the angle brackets indicates what the element is and the attributes that content should have, such as its size, font, or other characteristics. To end the formatting, the user types the first angle bracket, then a backslash, then repeats the element code and closes the bracket. For example, <title>What Is HTML?</title> is the code used to format the title of this article; the "strong" element tag is nested within the "title" tag.

◑ DOCTYPE

The <!DOCTYPE> declaration must be the very first thing in your HTML document, before the <html> tag. The <!DOCTYPE> declaration is not an HTML tag; it is an instruction to the web browser about what version of HTML the page is written in. In HTML 4.01, the <!DOCTYPE> declaration refers to a DTD, because HTML 4.01 was based on SGML. The DTD specifies the rules for the markup language, so that the browsers render the content correctly.

The HTML <doctype> tag is used for specifying which version of HTML the document is using. This is referred to as the **document type declaration (DTD).**

Document type declaration is an instruction that associates a particular SGML with a document type definition or XML document.

Example

```
<!DOCTYPE html>
<html>
<head>
<title>HTML doctype Tag</title>
</head>
<body>
<p>doctype declaration <doctype> is mentioned at the starting of every HTML document.</p>
</body>
</html>
```

This will produce following result:

doctype declaration <doctype> is mentioned at the starting of every HTML document.

Declaration

HTML 4.01 has 3 possible doctypes: HTML 4 Strict, HTML 4 Transitional, and HTML 4 Frameset. Every HTML document you create should have one of these three DTDs.

HTML 4 Strict

This document type includes all HTML elements except those that have been deprecated, and those that appear in frameset documents.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
        "https://www.w3.org/TR/html4/strict.dtd">
```

HTML 4 Transitional

This document type includes all HTML elements including those that have been deprecated.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

"https://www.w3.org/TR/html4/loose.dtd">;

HTML 4 Frameset

This document type includes all HTML elements in the transitional DTD as well as those in framed document.

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"

"https://www.w3.org/TR/html4/frameset.dtd">

HTML 5 Declaration

In HTML5 there is only one declaration i.e.

<!DOCTYPE html>

◑ HTML Character Entities

Some characters have a special meaning in HTML, like the less than sign (<) that defines the start of an HTML tag. If we want the browser to actually display these characters we must insert character entities in place of the actual characters themselves.

The Most Common Character Entities:

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
"	quotation mark	"	"
'	apostrophe	' (does not work in IE)	'

A character entity has three parts: an ampersand (&), an entity name or an entity number, and finally a semicolon (;). The & means we are beginning a special character, the; means ending a special character and the letters in between are sort of an abbreviation for what it's for. To display a less than sign in an HTML document we must write: < or < The advantage of using a name instead of a number is that a name is easier to remember. The disadvantage is that not all browsers support the newest entity names, while the support for entity numbers is very good in almost all browsers.

Non-breaking Space

The most common character entity in HTML is the non-breaking space .. Normally HTML will truncate spaces in your text. If you add 10 spaces in your text, HTML will remove 9 of them. To add spaces to your text, use the character entity.

This Code	Would Display
This code would appear as this.	This code would appear as this.
This code would appear with three extra spaces.	This code would appear with three extra spaces.

Html Element

An HTML element usually consists of a start tag and end tag, with the content inserted in between:

<tagname>Content goes here...</tagname>

The HTML element is everything from the start tag to the end tag:

My first paragraph.

Nested HTML Elements

HTML elements can be nested (elements can contain elements).

All HTML documents consist of nested HTML elements.

This example contains four HTML elements:

Example

<!DOCTYPE html>

<html>

<body>

<h1>My First Heading</h1>

My first paragraph.

</body>

</html>

Example Explained

The <html> element defines the whole document.

It has a start tag <html> and an end tag </html>.

The element content is another HTML element (the <body> element).

<html>

<body>

<h1>My First Heading</h1>

My first paragraph.

</body>

</html>

The <body> element defines the document body.

It has a start tag <body> and an end tag </body>.

The element content is two other HTML elements (<h1> and).

<body>

<h1>My First Heading</h1>

My first paragraph.

</body>

The <h1> element defines a heading.

It has a start tag <h1> and an end tag </h1>.

The element content is: My First Heading.

<h1>My First Heading</h1>

The element defines a paragraph.

It has a start tag and an end tag .

The element content is: My first paragraph.

My first paragraph.

Do Not Forget the End Tag

Some HTML elements will display correctly, even if you forget the end tag:

Example

<html>

<body>

This is a paragraph

This is a paragraph

</body>

</html>

Empty HTML Elements

HTML elements with no content are called empty elements.

 is an empty element without a closing tag (the <br> tag defines a line break).

Empty elements can be "closed" in the opening tag like this: <br />.

HTML5 does not require empty elements to be closed. But if you want stricter **validation**, or if you need to make your document readable by XML parsers, you must close all HTML elements properly.

Use Lowercase Tags

HTML tags are not case sensitive: <P> means the same as <p>.

The HTML5 standard does not require lowercase tags, but W3C recommends lowercase in HTML, and demands lowercase for stricter document types like XHTML.

Validation is the process of establishing documentary evidence demonstrating that a procedure, process, or activity carried out in testing and then production maintains the desired level of compliance at all stages.

Head Element

HTML Head tag is the first child of HTML Element. It Contains all information related to webpage, css links, scripts and meta tags. The HTML <head> element provides general information (metadata) about the document, including its title and links to its scripts and style sheets.

The HTML <head> Elemen

The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag. HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, links, scripts, and other meta information.

The following tags describe metadata: <title>, <style>, <meta>, <link>, <script>, and <base>.

The HTML <title> Element

The <title> element defines the title of the document, and is required in all HTML/XHTML documents.

The <title> element:

- defines a title in the browser tab
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine results

A simple HTML document:

Example

```
<!DOCTYPE html>
<html>

<head>
  <title>Page Title</title>
</head>

<body>
The content of the document.....
</body>

</html>
```

The HTML <style> Element

The <style> element is used to define style information for a single HTML page:

Example

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element

The <link> element is used to link to external style sheets:

Example

<link rel="stylesheet" href="mystyle.css">

The HTML <meta> Element

The <meta> element is used to specify which character set is used, page description, keywords, author, and other metadata.

Metadata is used by browsers (how to display content), by search engines (keywords), and other web services.

Define the character set used:

<meta charset="UTF-8">

Define a description of your web page:

<meta name="description" content="Free Web tutorials">

Define keywords for search engines:

<meta name="keywords" content="HTML, CSS, XML, JavaScript">

Define the author of a page:

<meta name="author" content="John Doe">

Refresh document every 30 seconds:

<meta http-equiv="refresh" content="30">

Example of <meta> tags:

<meta charset="UTF-8">

<meta name="description" content="Free Web tutorials">

<meta name="keywords" content="HTML,CSS,XML,JavaScript">

<meta name="author" content="John Doe">

The HTML <script> Element

The <script> element is used to define client-side JavaScripts.

This JavaScript writes “Hello JavaScript!” into an HTML element with id=”demo”:

Example

```
<script>
function myFunction {
    document.getElementById(“demo”).innerHTML = “Hello JavaScript!”;
}
</script>
```

The HTML <base> Element

The <base> element specifies the base URL and base target for all relative URLs in a page:

Example

```
<base href=”https://www.w3schools.com/images/” target=”_blank”>
```

Language Attribute.

The lang attribute specifies the language of the element’s content. Common examples are “en” for English, “es” for Spanish, “fr” for France and so on. The lang attribute is part of the Global Attributes, and can be used on any HTML element.

Some French text in a paragraph:

```
<p lang=”fr”>Ceci est un paragraphe.</p>
```

The lang Attribute specifies the natural language of the content of a web page. An attribute on the html tag sets the language for all the text on the page. If part of the page uses text in a different language, you can add a language attribute with a different value to the element that surrounds that content.

Body Element

The <body> element contains the entire content of a webpage. It must be the second element inside of the parent <html> element, following only the <head> element.

<!DOCTYPE html> <html> <head> . . . </head> <body> <!-- Page content goes here --> </body> </html>

The <body> element may be the most important HTML element there is. The contents of the <body> element are what actually get displayed to the user visiting your web page or viewing your document. And yet, it is very simple. The <body> must be a direct child of <html>, it must come after <head> (if <head> is present, which it doesn't have to be), there can be only one of it, and … that's it.

body { font-family: "Adobe Garamond Premier Pro", Garamond, "Linux Libertine", Baskerville, Georgia, serif; font-weight: normal: font-size: 18px; color: #111111; background-color: #fefefe; padding: 1em; width: 95vw; max-width: 950px; margin: 0 auto; }

The example shows the basic structure of a webpage, including the body block.

```
<!DOCTYPE html>

<html>

  <head>

    <title>The beauty of life</title>

  </head>

  <body>

    <h1>The beauty of life</h1>

    <p>This morning I woke up feeling different...</p>

  </body>

</html>
```

Targeting the <body>

In content management systems, it is a common practice to place a list of content-

specific CSS classes into the <body> element. This allows designers and JavaScript developers to target pages based on specific content attributes, such as category, tags, or content type.

<body class="post post-single post-template-default tag-html tag-element tag-document category-markup category-html post-123 with-comments social-sharing-on logged-in">

These classes are added programmatically by different components and plugins, and provide an easy way for developers to target styling or functionality at highly specific sets of pages (and if they never get used, they aren't causing any problems).

◑ HTML Color

Colors are very important to give a good look and feel to your website. You can specify colors on page level using <body> tag or you can set colors for individual tags using bgcolor attribute.

The <body> tag has following attributes which can be used to set different colors:

- **bgcolor -** sets a color for the background of the page.
- **text -** sets a color for the body text.
- **alink -** sets a color for active links or selected links.
- **link -** sets a color for linked text.
- **vlink -** sets a color for *visited links* - that is, for linked text that you have already clicked on.

HTML Color Coding Methods

There are following three different methods to set colors in your web page:

- **Color names -** You can specify color names directly like green, blue or red.
- **Hex codes -** A six-digit code representing the amount of red, green, and blue that makes up the color.
- **Color decimal or percentage values -** This value is specified using the rgb( ) property.

HTML Colors - Color Names

You can sepecify direct a color name to set text or background color. W3C has listed 16 basic color names that will validate with an HTML validator but there

are over 200 different color names supported by major browsers.

W3C Standard 16 Colors

Here is the list of W3C Standard 16 Colors names and it is recommended to use them.

Black	Gray	Silver	White
Yellow	Lime	Aqua	Fuchsia
Red	Green	Blue	Purple
Maroon	Olive	Navy	Teal

HTML Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits(RR) represent a red value, the next two are a green value(GG), and the last are the blue value(BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Paintshop Pro or MS Paint.

Each hexadecimal code will be preceded by a pound or hash sign #. Following is a list of few colors using hexadecimal notation.

Color	Color HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

HTML Colors - RGB Values

This color value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.

Following is a list to show few colors using RGB values.



REMEMBER

All the browsers does not support rgb() property of color so it is recommended not to use it.

Color	Color RGB
	rgb(0,0,0)
	rgb(255,0,0)
	rgb(0,255,0)
	rgb(0,0,255)
	rgb(255,255,0)
	rgb(0,255,255)
	rgb(255,0,255)
	rgb(192,192,192)
	rgb(255,255,255)

◑ HTML Lists

HTML lists are used to present list of information in well-formed and semantic way. There are three different types of list in HTML and each one has a specific purpose and meaning:

- Unordered list — Used to group a set of related items, in no particular order.
- Ordered list — Used to group a set of related items, in a specific order.
- Description list — Used to display a list of terms and their descriptions.

HTML Unordered Lists

An unordered list created using the <ul> tag, and each list item starts with the <li> tag.

The list items in unordered lists are marked with bullets (small black circles), by default.

Example

```
<ul>
  <li>Chocolate Cake</li>
```

```
<li>Black Forest Cake</li>
<li>Pineapple Cake</li>
</ul>
```

The output of the above example will look something like this:

- Chocolate cake
- Black forest cake
- Pineapple cake

HTML Ordered Lists

An ordered list, created using the <ol> tag, and each list item starts with the <li> tag. Ordered list contain information where order should be emphasized.

The list items in ordered lists are marked with numbers.

Example

```
<ol>
<li>Mix ingredients</li>
<li>Bake in oven for an hour</li>
<li>Allow to stand for ten minutes</li>
</ol>
```

The output of the above example will look something like this:

- Mix ingredients
- Bake in over for an hour
- Allow to stand for ten minutes

HTML Definition Lists

A definition list is a list of items, with a description of each item.

The definition list created using <dl> tag. The <dl> tag is used in conjunction with <dt> - defines the item in the list, and <dd> describes the item in the list:

Example

```
<dl>
  <dt>Bread</dt>
  <dd>A baked food made of flour.</dd>
  <dt>Coffee</dt>
  <dd>A drink made from roasted coffee beans.</dd>
</dl>
```

The output of the above example will look something like this:

Bread

A baked food made of flour

Coffee

A drink made from roasted coffee beans

◑ HTML Links

Links are used to "link" a visitor from one area to another. There area many types of links:

- Local: A page on the same server or directory
- Internal: A section on the current page or document
- External: A page or site on a different server or directory
- Download: A file for the visitor to download
- E-mail: Opens the visitor's e-mail program

Local Links

A Local link uses a page name (including sub-directories if needed) as the target. It is "local" to the current server.

```
<a href="Page_in_the_same_server.html">
```

Click here to go to the local page

Internal Links

Internal links can also be called page jump, u can make this jump with two simple steps

First step:

assign the place of the page where u want the pointer to move, and bookmark it by adding this cod in it

<A NAME="ur bookmark name"></A>

u will need the "ur bookmark name" in the next step.

Second step:

u need to add a link in anyplace in ur page to let the user move to ur bookmark point from it

<A HREF="#ur bookmark name "> Add the text to be displayed and clicked by the user </A></P>

The A stands for Anchor. In general, the anchor tag tells the browser to anchor or to attach to something else. Every Anchor tag must have a closing or end tag (</A>) to signal the end of the anchor. HREF stands for Hypertext REFerence. It means that "this is where the link is going to.

Example:

lets add a link to move the user to the top of the current page

first step: we add the bookmark code at the top of the page

<A NAME="top"></A>

second step: add the link anywhere in ur page

<A HREF=”#top “> click here to go to the top of this page </A></P>

and this is my link:
click here to go to the top of this page

the link will take u few lines above since we are not far from the top of this page

External Links

To link to any page in the world,u need the (URL) of the page you want to link to.

 text to click on

i want to link to my blog home page, so i will use this code

 Click here to visit my blog, enjoy!

and this is my link

Click here to visit my blog, enjoy!

Download Links

File links are used for allowing a visitor to download a file. These links are set up exactly the same as the local or external links. Instead of “pointing” to another page or site, it points to a file. When the user clicks on this link, the browser knows it is a file and will ask the visitor if they want to download the file. The types of files available to be used for download depends on your online server. A common and most accepted type is a ZIP file.

Click here to d ownload this file

E-mail Links

The e-mail link is for receiving e-mail and feedback from visitors. This link will prompt the browser's e-mail program to start and place the e-mail address in automatically.

click here ti send me e-mail

The subject line of an e-mail can be filled in automatically by adding a SUBJECT property:

click here ti send me e-mail

the cc and bcc can be added in the same way replacing their name instead of the subject word.

Link Colors

Links, by default, show up in different colors from the rest of the text on your web page. These settings are found in the individual browser settings. To over-ride these settings, you can declare your own link colors in the opening BODY tag:

- LINK: Color of an non-activated link. (default blue)
- VLINK: Color of a previously visited link. (default red)
- ALINK: Color of a currently active link. (default orange)

<body link= "#0000ff" vlink= "#ff0000" alink= "#ff8429" > ur HTML code here </body>

Using the hex color codes, you can choose the colors to suit your page. The ALINK is not used much. There aren't many instances when there is an open link at the same time as the current page. Be sure you are just inserting these properties into the current opening BODY tag.Do NOT create a second opening BODY tag. Any given HTML document can have only one BODY tag set. Any more than one set will result with page errors.

LEARNING ACTIVITY: BASIC STRUCTURE OF AN HTML DOCUMENT

An HTML document has two* main parts:

- Head. The head element contains title and Meta data of a web document.
- Body. The body element contains the information that you want to display on a web page.

Read more details on:

http://www.scriptingmaster.com/html/basic-structure-HTML-document.asp

Where do you use html definition lists in web page?

Define your Direction: On a separate piece of paper, explain the use of the different types of the terms according to html structure:

Terms	***Use***
Document type declaration	
Style sheets	
Metadata	
Hexadecimal code	
Hexadecimal notation	

KNOWLEDGE CHECK

1. Explain the various Html Tags.

2. How to create an HTML document? Explain.

3. How to naming files and folders in HTML document?

4. What do you understand by the HTML attribute and DOCTYPE?

5. Give an overview on head element and body element.

6. The common element which describe the web page, is?

a) heading
b) paragraph
c) list
d) All of these

7. HTML stands for?

a) Hyper Text Markup Language
b) High Text Markup Language
c) Hyper Tabular Markup Language
d) None of these

8. Which of the following tag is used to mark a beginning of paragraph?

a) <TD>
b)
c) <P>
d) <TR>

9. From which tag descriptive list starts?

a) <LL>
b) <DD>
c) <DL>
d) <DS>

10. Correct HTML tag for the largest heading is

a) <head>
b) <h6>
c) <heading>
d) <h1>

11. HTML is what type of language?

a) Scripting Language
b) Markup Language
c) Programming Language
d) Network Protocol

12. HTML uses

a) User defined tags
b) Pre-specified tags
c) Fixed tags defined by the language
d) Tags only for linking

13. Fundamental HTML Block is known as .....

a) HTML Body
b) HTML Tag
c) HTML Attribute
d) HTML Element

14. Apart from <b> tag, what other tag makes text bold ?

a) <fat>
b)
c) <black>
d) <emp>

15. What should be the first tag in any HTML document?

a) <head>
b) <title>
c) <html>
d) <document

16. Which attribute specifies a unique alphanumeric identifier to be associated with an element?

a) class
b) id
c) article
d) html

17. The _____ attribute specifies an inline style associated with an element, which determines the rendering of the affected element.

a) dir

b) style

c) class

d) article

18. Which attribute is used to provide an advisory text about an element or its contents?

a) tooltip

b) dir

c) title

d) head

19. The _____ attribute sets the text direction as related to the lang attribute.

a) lang

b) sub

c) dir

d) ds

20. Where are you use HTML lists and HTML links when creating a web page?

REFERENCES

1. Colorimetry", Third Edition, Commission Internationale de l'Eclairage, CIE Publication 15:2004, ISBN 3-901-906-33-9. Available at http://www.cie.co.at/publ/abst/15-2004.html
2. CSS3 Color Module," Tantek Çelik, Chris Lilley, 28 October 2010, W3C Proposed Recommendation. Available at http://www.w3.org/TR/2010/PR-css3-color-20101028/
3. CSS3 module: lists," Tantek Çelik, Ian Hickson, 7 November 2002, W3C working draft (work in progress). Available at http://www.w3.org/TR/2002/WD-css3-lists-20021107
4. CSS3 Speech Module", David Raggett, Daniel Glazman, Claudio Santambrogio, Daniel Weck, 19 April 2011, W3C Working Draft (work in progress). Available at http://www.w3.org/TR/2011/WD-css3-speech-20110419/
5. Document Object Model (DOM) Level 3 Core Specification", A. Le Hors, P. Le Hégaret, et al. (eds.), 7 April 2004, W3C Recommendation. Available at http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/.
6. Extensible Markup Language (XML) 1.0 (fifth edition)" T. Bray, J. Paoli, C.M. Sperberg-McQueen, Eve Maler, François Yergeau, editors, 26 November 2008. Available at http://www.w3.org/TR/REC-xml/.
7. Information Technology - Universal Multiple- Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:2003. Useful roadmap of the BMP and plane 1 documents show which scripts sit at which numeric ranges.
8. Mathematical Markup Language (MathML) Version 3.0", D. Carlisle, P. Ion, R. Miner, 21 October 2010 Available at http://www.w3.org/TR/2010/REC-MathML3-20101021/
9. Media Queries", Håkon Wium Lie, Tantek Çelik, Daniel Glazman, Anne van Kesteren, 27 July 2010 (Work in progress.) Available at http://www.w3.org/TR/2010/CR-css3-mediaqueries-20100727/
10. Namespaces in XML 1.0 (third edition)", T. Bray, D. Hollander, A. Layman, R. Tobin, H. S. Thompson, 8 December 2009 Available at: http://www.w3.org/TR/2009/REC-xml-names-20091208/.
11. Portable Network Graphics (PNG) Specification (Second Edition)", David Duce, ed., 10 November 2003. Available at http://www.w3.org/TR/PNG/.
12. Scalable Vector Graphics (SVG) 1.1 Specification", J. Ferraiolo, et.al. 14 January 2003 Available at http://www.w3.org/TR/2003/REC-SVG11-20030114

13. Selectors", D. Glazman, T. Çelik, I. Hickson, 15 December 2009 Available at http://www.w3.org/TR/2009/PR-css3-selectors-20091215/
14. Specification ICC.1:2004-10 (Profile version 4.2.0.0) Image technology colour management – Architecture, profile format, and data structure. Available at http://www.color.org/icc_specs2.html
15. Tags for Identifying Languages", A. Phillips, M. Davis, September 2009. Available at http://www.rfc-editor.org/rfc/bcp/bcp47.txt.
16. The latest version of the specification is available at http://www.w3.org/TR/html4/. The Recommendation defines three document type definitions: Strict, Transitional, and Frameset, all reachable from the Recommendation.
17. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification", L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle, 16 April 2002 Available at http://www.w3.org/TR/2002/REC-P3P-20020416
18. Unicode Bidirectional Algorithm", Mark Davis. (Unicode Standard Annex #9.) 27 September 2010. Available at http://www.unicode.org/reports/tr9/
19. Uniform Resource Identifier (URI): Generic Syntax," T. Berners-Lee, R. Fielding, L. Masinter, January 2005. Available at http://www.ietf.org/rfc/rfc3986.
20. User Agent Accessibility Guidelines 1.0." Ian Jacobs, Jon Gunderson, Eric Hansen (editors). 17 December 2002. Available at http://www.w3.org/TR/2002/REC-UAAG10-20021217
21. Web Content Accessibility Guidelines (WCAG) 2.0", Ben Caldwell, Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden, 11 December 2008.

CHAPTER 3

ESSENTIAL OF WEB API



INTRODUCTION

In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

To put it in simple terms, API is some kind of interface which has a set of functions that allow programmers to access specific features or data of an application, operating system or other services.

Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc. For example, Twitter's REST APIs provide programmatic access to read and write data using which we can integrate twitter's capabilities into our own application.

SCENARIO: PHENOMENA OF APPLICATION PROGRAMMING INTERFACE (API)

API stands for Application Programming Interface, and it lets developers integrate any two parts of an application or any different applications together. It consists of various elements such as functions, protocols, and tools that allow developers to build applications. A common goal of all types of APIs is to accelerate the development of applications by providing a part of its functionality out-of-the-box, so developers do not have to implement it themselves. There are APIs for all types of systems, including operating systems, libraries, and the Web.

What skills will you need?

In order to work effectively in essential of web API, you must know:

- Functionality of application programing interface
- API usages and testing procedure

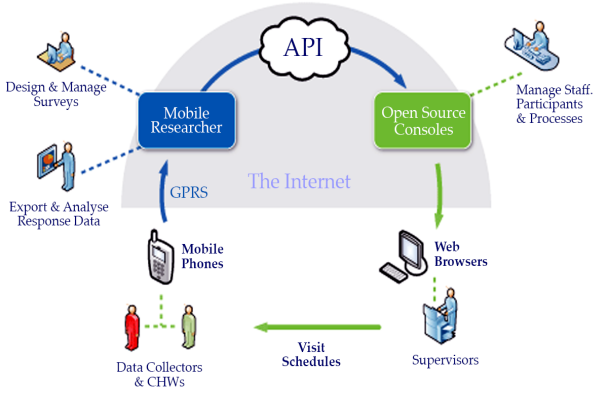
FUNCTIONALITY OF APPLICATION PROGRAMING INTERFACE

An application programming interface or API is a set of data structures, protocols, routines and tools for accessing a web-based software application. It provides all the building blocks for developing programs with ease. An application programming interface may be language dependent or language independent. A language dependent API is available only in a particular programming language. Language independent APIs are not bound to a particular language, system or process. They can be called from several programming languages.

APIs run silently in the background. They are completely invisible to software users and web surfers. They're primary role is to provide a channel for applications to work with each other to make sure the user gets the required functionality and information.

For example, when a user buys a product online and enters his or her credit card information, the product website uses an API to send the credit card information to another application. The application verifies whether the information is correct or not. It then authorizes the product website to issue the purchased product after the information is verified and payment is confirmed. A website has many such applications that work together using APIs.

Application programming interfaces ensure that software developers do not have to start building applications from scratch. There is no need to build one application that performs every function. The application can transfer certain responsibilities to a remote application. Thus, APIs are similar to SaaS (Software as a Service).



Language used

An API can be:

- Language-dependent, meaning it is only available by using the syntax and elements of a particular language, which makes the API more convenient to use.
- Language-independent, written so that it can be called from several programming languages. This is a desirable feature for a service-oriented API that is not bound to a specific process or system and may be provided as remote procedure calls or web services. For example, a website that allows users to review local restaurants is able to layer their reviews over maps taken from Google Maps, because Google Maps has an API that facilitates this functionality. Google Maps' API controls what information a third-party site can use and how they can use it.

The term API may be used to refer to a complete interface, a single function, or even a set of APIs provided by an organization. Thus, the scope of meaning is usually determined by the context of usage.

◑ Explanation of API

Prototype is generally used to evaluate a new design to enhance precision by system analysts and users.

An API may describe the ways in which a particular task is performed. In procedural languages like C language the action is usually mediated by a function call. Hence the API usually includes a description of all the functions/ routines it provides. For instance: the math.h include file for the C language contains the definition of the function prototypes of the mathematical functions available in the C language library for mathematical processing (usually called libm). This file describes how to use the functions included in the given library: the function **prototype** is a signature that describes the number and types of the parameters to be passed to the functions and the type of the return value.

The behavior of the functions is usually described in more details in a human readable format in printed books or in electronic formats like the man pages: e.g. on UNIX systems the command man 3 sqrt will present the signature of the function sqrt in the form:

SYNOPSIS

#include <math.h>

double sqrt(double X);

float sqrtf(float X);

DESCRIPTION

DESCRIPTION

sqrt computes the positive square root of the argument. ...

RETURNS

On success, the square root is returned. If X is real and

positive...

That means that the function returns the square root of a positive floating point number (single or double precision) as another floating point number. Hence the API in this case can be interpreted as the collection of the include files used by the C language and its human readable description provided by the man pages.

Documentation

Many program development environments provide the documentation associated with an API in some digital format,

e.g. perl comes with the tool perldoc:

```
$ perldoc -f sqrt
sqrt EXPR
sqrt #Return the square root of EXPR. If EXPR is omitted,
returns
#square root of $_. Only works on non-negative
operands, unless
#you've loaded the standard Math::Complex module.
```

python comes with the tool pydoc:

```
$ pydoc math.sqrt
Help on built-in function sqrt in math:
math.sqrt = sqrt(...)
sqrt(x)
```

```
 Return the square root of x.
ruby comes with the tool ri:
 $ ri Math::sqrt
---------------------------------------------------------- Math::sqrt
 Math.sqrt(numeric) => float
------------------------------------------------------------------------
 Returns the non-negative square root of _numeric_.
```

Java comes with the documentation organized in HTML pages (JavaDoc format), while Microsoft distributes the API documentation for its languages (Visual C++, C#, Visual Basic, F#, etc...) embedded in Visual Studio's help system.

API in object-oriented languages

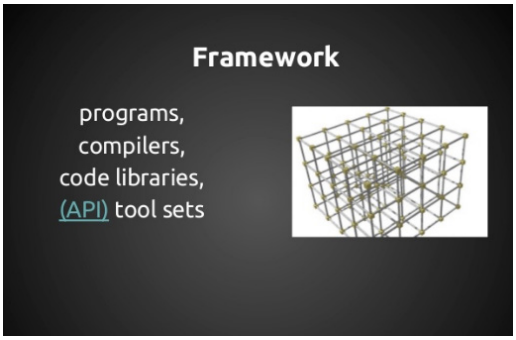
In object-oriented languages, an API usually includes a description of a set of class definitions, with a set of behaviors associated with those classes. This abstract concept is associated with the real functionality exposed, or made available, by the classes that are implemented in terms of class methods (or more generally by all its public components hence all public methods, but also possibly including any internal entity made public, like fields, constants, nested objects, enums...). The API in this case can be conceived as the totality of all the methods publicly exposed by the classes (usually called the class interface). This means that the API prescribes the methods by which one interacts with/handles the objects derived from the class definitions. More generally, one can see the API as the collection of all the kinds of objects one can derive from the class definitions, and their associated possible behaviors. Again: the use is mediated by the public methods, but in this interpretation, the methods are seen as a technical detail of how the behavior is implemented. For instance: a class representing a Stack can simply expose publicly two methods push() (to add a new item to the stack), and pop() (to extract the last item, ideally placed on top of the stack).

In this case the API can be interpreted as the two methods pop() and push(), or, more generally, as the idea that one can use an item of type Stack that implements the behavior of a stack: a pile exposing its top to add/remove elements. The second interpretation appears more appropriate in the spirit of object orientation. This concept can be carried to the point where a class interface in an API has no methods at all, but only behaviors associated with it. For instance, the Java language and Lisp (programming language) API include the interface Serializable, which is a marker interface that requires that each class that implements it should behave in a serialized fashion. This does not require to have any public method, but rather requires that any class that implements it to have a representation that can be saved (serialized) at any time (this is typically true for any class containing

simple data and no link to external resources, like an open connection to a file, a remote system, or an external device). Similarly the behavior of an object in a concurrent (multi-threaded) environment is not necessarily determined by specific methods, belonging to the interface implemented, but still belongs to the API for that Class of objects, and should be described in the documentation. In this sense, in object-oriented languages, the API defines a set of object behaviors, possibly mediated by a set of class methods. In such languages, the API is still distributed as a library. For example, the Java language libraries include a set of APIs that are provided in the form of the JDK used by the developers to build new Java programs. The JDK includes the documentation of the API in JavaDoc notation. The quality of the documentation associated with an API is often a factor determining its success in terms of ease of use.

API libraries and frameworks

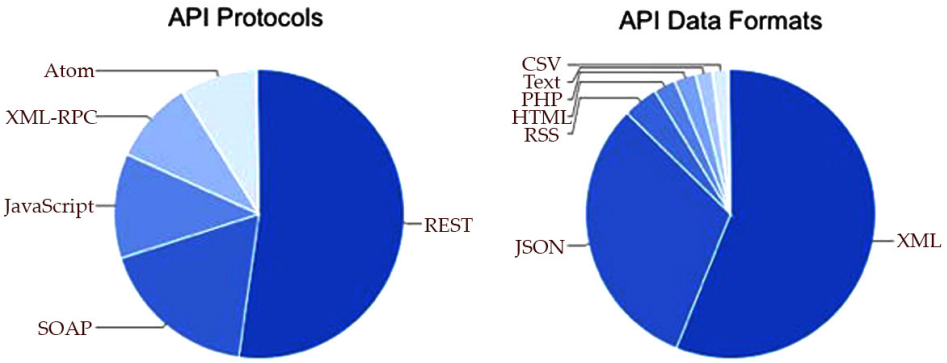
An API is usually related to a software library: the API describes and prescribes the expected behavior while the library is an actual implementation of this set of rules. A single API can have multiple implementations (or none, being abstract) in the form of different libraries that share the same programming interface. An API can also be related to a software framework: a framework can be based on several libraries implementing several APIs, but unlike the normal use of an API, the access to the behavior built into the framework is mediated by extending its content with new classes plugged into the framework itself. Moreover the overall program flow of control can be out of the control of the caller, and in the hands of the framework via inversion of control or a similar mechanisms.



API and protocols

An API can also be an implementation of a protocol. In general the difference between an API and a protocol is that the protocol defines a standard way to exchange requests and responses based on a common transport and agreeing

on a data/message exchange format, while an API (not implementing a protocol) is usually implemented as a library to be used directly: hence there can be no transport involved (no information physically transferred from/to some remote machine), but rather only simple information exchange via function calls (local to the machine where the elaboration takes place) and data is exchanged in formats expressed in a specific language. When an API implements a protocol it can be based on proxy methods for remote invocations that underneath rely on the communication protocol. The role of the API can be exactly to hide the detail of the transport protocol. E.g.: RMI is an API that implements the JRMP protocol or the IIOP as RMI-IIOP. Protocols are usually shared between different technologies (system based on given computer programming languages in a given operating system) and usually allow the different technologies to exchange information, acting as an abstraction/mediation level between the two worlds. While APIs can be specific to a given technology: hence the APIs of a given language cannot be used in other languages, unless the function calls are wrapped with specific adaptation libraries.



Object API and protocols

An object API can prescribe a specific object exchange format, an object exchange protocol can define a way to transfer the same kind of information in a message sent to a remote system. When a message is exchanged via a protocol between two different platforms using objects on both sides, the object in a programming language can be transformed (marshalled and unmarshalled) in an object in a remote and different language: so, e.g., a program written in Java invokes a service via SOAP or IIOP written in C# both programs use APIs for remote invocation (each locally to the machine where they are working) to (remotely) exchange information that they both convert from/to an object in local memory.



API Sharing and Reuse via Virtual Machine

Some languages like those running in a virtual machine (e.g. .NET CLI compliant languages in the Common Language Runtime and JVM compliant languages in the Java Virtual Machine) can share APIs. In this case the virtual machine enables the language interoperation thanks to the common denominator of the virtual machine that abstracts from the specific language using an intermediate byte code and its language binding. Hence this approach maximizes the code reuse potential for all the existing libraries and related APIs.

◑ Web APIs

When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. While "Web API" is virtually a synonym for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based services towards more direct Representational State Transfer (REST) style communications. Web APIs allow the combination of multiple services into new applications known as mashups.



REMEMBER

Instead when a similar object is exchanged via an API local to a single machine the object is effectively exchanged (or a reference to it) in memory: e.g. via the memory allocated by a single process, or among multiple processes using shared memory or other sharing technologies like tuple spaces.



Web use to share content

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

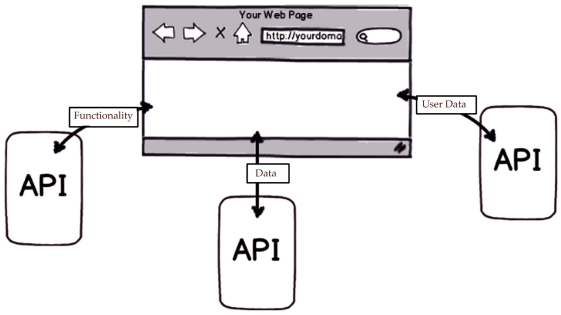
1. Photos can be shared from sites like Flickr and Photobucket to social network sites like Facebook and Myspace.
2. Content can be embedded, e.g. embedding a presentation from SlideShare on a LinkedIn profile.
3. Content can be dynamically posted. Sharing live comments made on Twitter with a Facebook account, for example, is enabled by their APIs.
4. Video content can be embedded on sites which are served by another host.
5. User information can be shared from web communities to outside applications, delivering new functionality to the web community that shares its user data via an open API. One of the best examples of this is the Facebook Application platform. Another is the Open Social platform.



Implementations

The POSIX standard defines an API that allows a wide range of common computing functions to be written in a way such that they may operate on many different systems (Mac OS X, and various **Berkeley Software Distributions (BSDs)** implement this interface); however, making use of this requires re-compiling for each platform. A compatible API, on the other hand, allows compiled object code to function without any changes to the system implementing that API. This is beneficial to both software providers (where they may distribute existing software on new systems without producing and distributing upgrades) and users (where they may install older software on their new systems without purchasing upgrades), although this generally requires that various software libraries implement the necessary APIs as well.

Berkeley Software Distribution (BSD) is a prominent version of the Unix operating system that was developed and distributed by the Computer Systems Research Group (CSRG) from the University of California at Berkeley between 1977 and 1995.



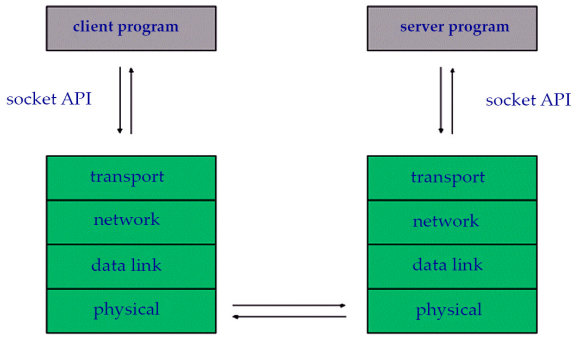
Apple Inc. has shown less concern, breaking compatibility or implementing an API in a slower "emulation mode"; this allows greater freedom in development, at the cost of making older software obsolete. Among Unix-like operating systems, there are many related but incompatible operating systems running on a common hardware platform (particularly Intel 80386-compatible systems). There have been several attempts to standardize the API such that software vendors may distribute one binary application for all these systems; however, to date, none of these have met with much success. The Linux Standard Base is attempting to do this for the Linux platform, while many of the BSD Unixes, such as FreeBSD, NetBSD, and OpenBSD, implement various

levels of API compatibility for both backward compatibility (allowing programs written for older versions to run on newer distributions of the system) and cross-platform compatibility (allowing execution of foreign code without recompiling).

Release policies

The two options for releasing API are:

1. Protecting information on APIs from the general public. For example, Sony used to make its official PlayStation 2 API available only to licensed PlayStation developers. This enabled Sony to control who wrote PlayStation 2 games. This gives companies quality control privileges and can provide them with potential licensing revenue streams.
2. Making APIs freely available. For example, Microsoft makes the Microsoft Windows API public, and Apple releases its APIs Carbon and Cocoa, so that software can be written for their platforms. A mix of the two behaviors can be used as well.



APIs and Copyrights

In 2010 Oracle sued Google for having distributed a new implementation of Java embedded in the Android operating system. The litigation concerns also the fact that Google did not acquire any permission to reproduce the Java API. Instead a similar permission was given to the OpenJDK project. However, Hon. William Alsup ruled on May 31, 2012, that APIs cannot be copyrighted.

API examples

- ASPI for SCSI device interfacing
- Carbon and Cocoa for the Macintosh
- DirectX for Microsoft Windows

- EHLLAPI
- Java APIs
- ODBC for Microsoft Windows
- OpenAL cross-platform sound API
- OpenCL cross-platform API for general-purpose computing for CPUs & GPUs
- OpenGL cross-platform graphics API
- OpenMP API that supports multi-platform shared memory multiprocessing programming in C, C++ and Fortran on many architectures, including **Unix** and Microsoft Windows platforms.
- Simple DirectMedia Layer (SDL)
- Talend integrates its data management with BPM from Bonita Open Solution

UNIX was originally meant to be a convenient platform for programmers developing software to be run on it and on other systems, rather than for non-programmers.

◑ Web API Controller

Web API Controller is similar to ASP.NET MVC controller. It handles incoming HTTP requests and send response back to the caller.

Web API controller is a class which can be created under the Controllers folder or any other folder under your project›s root folder. The name of a controller class must end with «Controller» and it must be derived from System.Web.Http.ApiControlle**r** class. All the public methods of the controller are called action methods.

The following is a simple controller class added by visual studio by default when we created a new Web API project in the Create Web API Project section.

Example: Simple Web API Controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
```

```
namespace MyWebAPI.Controllers
{
    public class ValuesController : ApiController
    {
        // GET: api/student
        public IEnumerable<string> Get()
        {
            return new string[] { “value1”, “value2” };
        }

        // GET: api/student/5
        public string Get(int id)
        {
            return “value”;
        }

        // POST: api/student
        public void Post([FromBody]string value)
        {
        }

        // PUT: api/student/5
        public void Put(int id, [FromBody]string value)
        {
        }

        // DELETE: api/student/5
        public void Delete(int id)
        {
        }
    }
}
```

As you can see in the above example, ValuesController class is derived from ApiController and includes multiple action methods whose names match with HTTP verbs like Get, Post, Put and Delete.

Based on the incoming request URL and HTTP verb (GET/POST/PUT/PATCH/DELETE), Web API decides which Web API controller and action method to execute e.g. Get() method will handle HTTP GET request, Post() method will handle HTTP POST request, Put() mehtod will handle HTTP PUT request and Delete() method will handle HTTP DELETE request for the above Web API.

Web API Controller Impression

If you want to write methods that do not start with an HTTP verb then you can apply the appropriate http verb attribute on the method such as HttpGet, HttpPost, HttpPut etc. same as MVC controller.

Example: Simple Web API Controller

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;

namespace MyWebAPI.Controllers
{
    public class ValuesController : ApiController
    {
        [HttpGet]
        public IEnumerable<string> Values()
        {
            return new string[] { “value1”, “value2” };
        }

        [HttpGet]
        public string Value(int id)
        {
            return “value”;
```

```
        }

        [HttpPost]
        public void SaveNewValue([FromBody]string value)
        {
        }

        [HttpPut]
        public void UpdateValue(int id, [FromBody]string value)
        {
        }

        [HttpDelete]
        public void RemoveValue(int id)
        {
        }
    }
}
```

Web API Controller Characteristics

- It must be derived from System.Web.Http.ApiController class.
- It can be created under any folder in the project's root folder. However, it is recommended to create controller classes in the Controllers folder as per the convention.
- Action method name can be the same as HTTP verb name or it can start with HTTP verb with any suffix (case in-sensitive) or you can apply Http verb attributes to method.
- Return type of an action method can be any primitive or complex type.

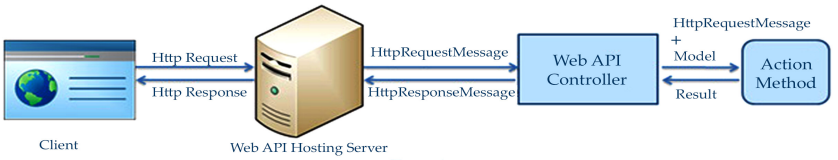
Action Method Naming Conventions

Name of the action methods in the Web API controller plays an important role. Action method name can be the same as HTTP verbs like Get, Post, Put, Patch or Delete as shown in the Web API Controller example above. However, you can append any suffix with HTTP verbs for more readability. For example, Get method can be GetAllNames(), GetStudents() or any other name which starts with Get.

The following table lists possible action method names for each HTTP method:

HTTP Method	Possible Web API Action Method Name	Usage
GET	Get() get() GET() GetAllStudent() *any name starting with Get *	Retrieves data.
POST	Post() post() POST() PostNewStudent() *any name starting with Post*	Inserts new record.
PUT	Put() put() PUT() PutStudent() *any name starting with Put*	Updates existing record.
PATCH	Patch() patch() PATCH() PatchStudent() *any name starting with Patch*	Updates record partially.
DELETE	Delete() delete() DELETE() DeleteStudent() *any name starting with Delete*	Deletes record.

The following figure illustrates the overall request/response pipeline.



LEARNING ACTIVITY: CREATING WEB API IN ASP.NET

Let's create a Web API with the latest version of ASP.NET Core and Entity Framework Core.

In this guide, we'll use Wide World Importers database to create a Web API.

REST APIs provide at least the following operations:

- GET
- POST
- PUT
- DELETE

Read more details on:

https://www.codingame.com/playgrounds/35462/creating-web-api-in-asp-net-core-2-0/part-1---web-api

What are the characteristics of web API controller?

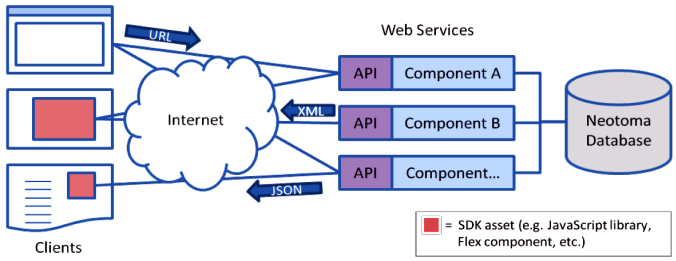
Define your Direction: On a separate piece of paper, explain the significance of the different types of the terms according to functionality of application programing interface:

Terms	***Significance***
Software as a Service	
Google Map	
Programming language	
Java Virtual Machine	
Procedural languages	

API USAGES AND TESTING PROCEDURE

Basically, an API is a set of protocols, subroutines and tools to develop software programs. Like a graphical interface, the objective of an API is to make the developer's work easier. To achieve this, an API provides remote calls, object classes, data structures or variables, which are all the blocks required for a developer to use certain technologies in his application, reducing the cognitive load to which he is exposed. The maintenance and development of applications that use APIs, becomes more expensive and time consuming, without proper documentation. This, should contain all the relevant information about the services available in the API, as well as the way they work, so the developer knows what the expected response when interacting with it is.

Despite the three components introduced being different from each other, there is a strong connection between an API, library and framework. An API describes a behavior while a library contains all the implementation of the rules used in this behavior. Regarding the API-framework relation, an API is the public side of a framework, which means, from all the methods implemented in a framework, only the ones relevant to the user will be presented in an API.



Library

A library is composed by a set of rules used by an application, this means the library contains all the implementation for the expected behavior. Example given, when a programmer wants to develop an application, he can use a library to make the system calls instead of implementing them. The code available in the library is independent from the application itself and is structured in a way that it can be used and reused by multiple applications, without these being connected between each other. These libraries, allow the code to be shared by multiple developers, which makes it easy when developing new software projects.

Framework

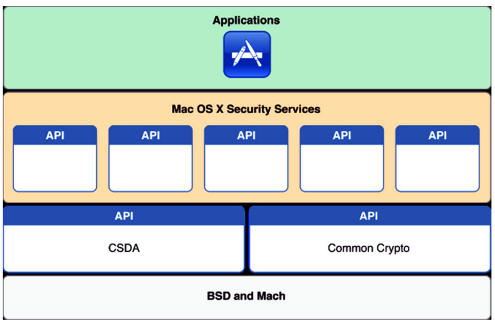
Last but not least, a framework is composed by a group of libraries that allows to execute a bigger operation. These frameworks can be seen as development platforms, which should be adapted to the developer's requirements. In the end, a framework can be seen as the skeleton of the application, for example, considering a team that is developing a website for a company, by using a framework, they can focus their efforts on specific code for the website itself instead of all the handling and management of the requests.

◑ Operating systems

An API can specify the interface between an application and the operating system. POSIX, for example, specifies a set of common APIs that aim to enable an application written for a POSIX conformant operating system to be compiled for another POSIX conformant operating system. Linux and Berkeley Software Distribution are examples of operating systems that implement the POSIX APIs. Microsoft has shown a strong commitment to a backward-compatible API, particularly within their Windows API (Win32) library, so older applications may run on newer versions of Windows using an executable-specific setting called "Compatibility Mode". An API differs from an application binary interface (ABI) in that an API is source code based while an ABI is binary based. For instance, POSIX provides APIs, while the Linux Standard Base provides an ABI.

- Software developers have demonstrated that many different programs can take advantage of a number of similar modules.

For example, word processors, spreadsheets, presentation software, and database software, although they accomplish quite different tasks, all need to be able to permit the user to save work in progress into a data file, with the user being able to select the disk drive, folder name, file name, and in some cases file format.



Every version of the Macintosh OS and every version of Windows has included, as part of the operating system, a module that provides that functionality.

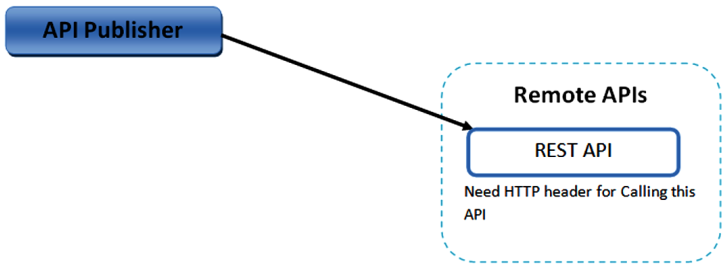
- Software developers writing application programs do not have to write their own module that performs such functions. Instead, they write their software to use the operating system's module.
- Some operating systems provide modules that support graphical user interfaces, other operating systems provide modules that support command line user interfaces, and some provide both.

Remote APIs

Remote APIs allow clients connecting from a separate tier to communicate with the Alfresco content application server. Remote APIs are based on web services and RESTful and CMIS protocols, and are language agnostic, allowing you to develop against these APIs using a range of languages including Java, PHP, Ruby, and .NET.

There are several Remote APIs available, including:

- **Web services** - object-oriented API using SOAP and supporting a range of content services including authentication, query, node creation and update, access control, and actions
- **RESTful** - HTTP-based resource oriented interfaces used by the Surf framework and Alfresco Share
- **CMIS** - a standard that defines web services and REST-based bindings for working with CMIS compliant repositories



The remote API is composed by approximately one hundred specific functions and one generic function that can be called from a C/C++ application, a Python script, a Java application, a Matlab/Octave program, an Urbi script, or a Lua script. The remote API functions are interacting with via socket communication in a way that reduces lag and network load to a great extent. All this happens in a hidden fashion to the user. The remote API can let one or several external applications

interact with V-REP in a synchronous* or asynchronous* way (asynchronous by default), and even remote control of the simulator is supported (e.g. remotely loading a scene, starting, pausing or stopping a simulation for instance).

The remote API functionality comes in 2 separate entities that are interacting via socket communication:

- The client side (i.e. your application): the remote API on the client side is available for many different programming languages. Currently following languages are supported: C/C++, Python, Java, Matlab, Octave, Urbi and Lua. Bindings for other languages can easily be created by yourself. Instructions on how to enable the remote API on the client side are given here.
- The server side: the remote API on the server side is implemented via a plugin that should be loaded by default: v_repExtRemoteApi.dll, libv_repExtRemoteApi.dylib or libv_repExtRemoteApi.so. The plugin project files are located in installation directory, in the "programming" folder. Instructions on how to enabled the remote API on the server side are given here.



REMEMBER

A remote API allowing to control a simulation (or the simulator itself) from an external application or a remote hardware (e.g. real robot, remote computer, etc.).

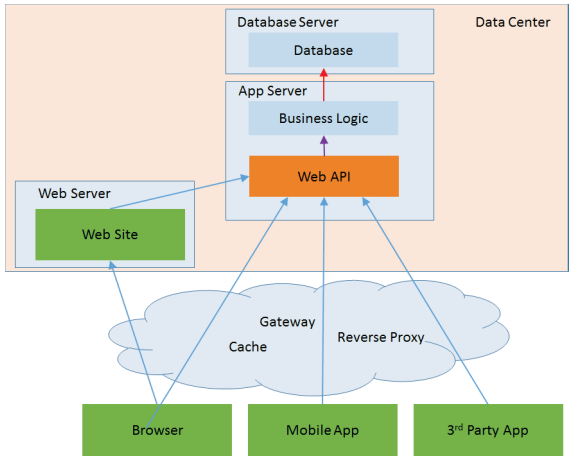
◑ Use in Web APIs

An API approach is an architectural approach that revolves around providing programmable interfaces to a set of services to different applications serving different types of consumers. When used in the context of web development, an API is typically defined as a set of Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, which is usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. An example might be a shipping company API that can be added to an eCommerce-focused web site, to facilitate ordering shipping services and automatically include current shipping rates, without the site developer having to enter the shipper's rate table into a web database. While "web API" historically has been

Simple Object Access Protocol (SOAP) is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks.

virtually synonymous for web service, the recent trend (so-called Web 2.0) has been moving away from **Simple Object Access Protocol (SOAP)** based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and resource-oriented architecture (ROA).

Part of this trend is related to the Semantic Web movement toward Resource Description Framework (RDF), a concept to promote web-based ontology engineering technologies. Web APIs allow the combination of multiple APIs into new applications known as mashups. In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

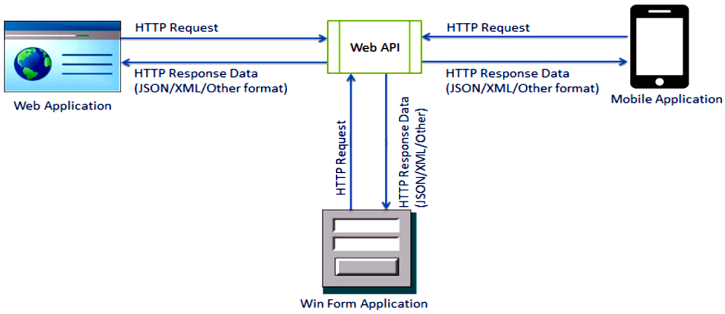


Web API as the name suggests, is an API over the web which can be accessed using HTTP protocol. It is a concept and not a technology. We can build Web API using different technologies such as Java, .NET etc. For example, Twitter's REST APIs provide programmatic access to read and write data using which we can integrate twitter's capabilities into our own application.

Web API in ASP.NET

The ASP.NET Web API is an extensible framework for

building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc. It works more or less the same way as ASP.NET MVC web application except that it sends data as a response instead of html view. It is like a webservice or WCF service but the exception is that it only supports HTTP protocol.



ASP.NET Web API Characteristics:

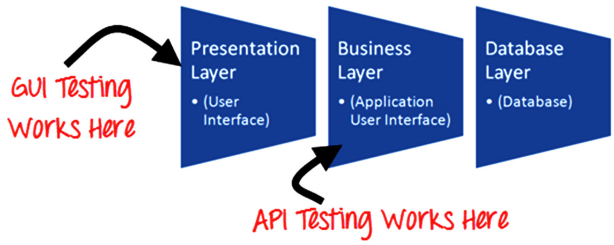
- ASP.NET Web API is an ideal platform for building RESTful services.
- ASP.NET Web API is built on top of ASP.NET and supports ASP.NET request/response pipeline
- ASP.NET Web API maps HTTP verbs to method names.
- ASP.NET Web API supports different formats of response data. Built-in support for JSON, XML, BSON format.
- ASP.NET Web API can be hosted in IIS, Self-hosted or other web server that supports .NET 4.0+.
- ASP.NET Web API framework includes new HttpClient to communicate with Web API server. HttpClient can be used in ASP.MVC server side, Windows Form application, Console application or other apps.

◑ API Testing Strategy

API testing is a type of software testing that involves testing application programming interfaces (APIs) directly and as part of integration testing to determine if they meet expectations for functionality, reliability, performance, and security. Since APIs lack a GUI, API testing is performed at the message layer. API testing is now considered critical for automating testing because APIs now serve as the primary interface to application logic and because GUI tests are difficult to maintain with the short release cycles and frequent changes commonly used with Agile software development and DevOps).

The API Testing is performed for the system, which has a collection of API that ought to be tested. During Testing, a test of following things is looked at.

- Exploring boundary conditions and ensuring that the test harness varies parameters of the API calls in ways that verify functionality and expose failures.
- Generating more value added parameter combinations to verify the calls with two or more parameters.
- Verifying the behaviour of the API which is considering the external environment conditions such as files, peripheral devices, and so forth.
- Verifying the Sequence of API calls and check if the API's produce useful results from successive calls.



Instead of using standard user inputs (keyboard) and outputs, in API Testing, you use software to send calls to the API, get output, and note down the system's response.

API Testing requires an application to interact with API. In order to test an API, you will need to

- Use Testing Tool to drive the API
- Write your own code to test the API

Set-up of API Test environment

- API testing is different than other testing types as GUI is not available, and yet you are required to setup initial environment that invokes API with required set of parameters and then finally examines the test result.
- Hence, Setting up a testing environment for API testing seems a little complex.
- Database and server should be configured as per the application requirements.
- Once the installation is done, API Function should be called to check whether that API is working.

Common Tests performed on API's

- Return Value based on input condition - The return value from the API's are checked based on the input condition.
- Verify if the API's does not return anything.
- Verify if the API triggers some other event or calls another API. The Events output should be tracked and verified.
- Verify if the API is updating any data structure.

◑ Challenges with API Testing

An API is an interface intended to be interpreted by a computer. Since the interface boundary has to be technology agnostic, the inputs and outputs are normally text-formatted messages. Text messages may be easy for a human to read in order to interpret the inputs and outputs, but in-depth knowledge of the internals of the application is required to sufficiently test it.

This is the difference between black-box testing and white-box testing. In black-box testing, the tester is only concerned with the functionality of the overall application, exercising only the inputs and checking the outputs.

In white-box testing, the tester is much more concerned with internal operations of the application, exercising the individual paths the data can travel through the application.

API testing normally includes only the white-box testing approach.

Another difficulty is that testers are used to working from a set of provided business requirements. However, business requirements normally describe use cases at a fairly high level, and do not go down to the level of individual application functions.

It is not unusual that **business analysts** are not even aware of low level detail and the amount of granularity that everything has to be broken down into, and are not aware that software is able to be tested at this level.

Business analyst is someone who analyzes an organization or business domain and documents its business or processes or systems, assessing the business model or its integration with technology.

Typically, project managers are not concerned with assigning time specifically to developing a rich and detailed API, let alone testing it. Paradoxically, if everything is done correctly, the API is the only place where business rules are coded and enforced. The end-user GUI only displays results and any error messages provided by the API.

Sadly, because of the above problems, developers are often left with testing their own code, if there is any testing at this level at all. And due to time pressures, unit tests often only glance at the positive tests – the happy path – and more in depth testing is left until later in the project.

Of course, looking at only the positive paths through the application is not really testing, but only checking that the software fulfils business requirements under the most optimal conditions.

The GUI usually exposes only part of the overall functionality and by design testers are not able to go really deep into the application code, especially when it comes to testing all the negative test cases. This leaves the internals of the application possibly exposed to all sorts of breakage and attacks by end-users, accidental or malicious.

Lastly, there are situations where the API is the final product – a public API. A simple Internet search will reveal there are thousands of public APIs for anything from GPS and mapping solutions to locating a radio station that is playing your favorite song right now. However, these APIs expose functionality that the developer hopes or thinks other developers/ users will want. In this case, the tester's role needs also to include a user advocate role, to make sure the offering is complete – everything that any user may need is all there. This is in addition to the multitude of hats, the tester already wears: functional testing, security testing, load testing, etc.

Advantages of API Testing

Putting more effort into API testing leads to a much healthier final product. Ensuring that all data access (read and write) goes only through the API significantly simplifies security and compliance testing and thereby certification, since there is only one interface.

Ensuring that all the required business rules are being enforced at the API tier allows time for much more complete user-experience tests once the UI is released, and not having to concentrate on testing every single business rule and path through the application near the end of the project. Ensuring that the API offers complete functionality allows for easy future expansion of the application as new business needs arise.

◑ Start API testing

Once you've set up your API testing environment, make an API call right away to make sure nothing is broken before you go forward to start your more thorough testing.

You can start combining your application data with your API tests to ensure that the API performs as expected against possible known input configurations. See this REST Testing example for beginners to help with the visualization.

Next, you need to organize yourself around the API test. Start by asking yourself these questions:

- Who is your target audience? Who is your API consumer?
- What environment/s should the API typically be used?
- What aspects are you testing?
- What problems are we testing for?
- What are your priorities to test?
- What is supposed to happen in normal circumstances?
- What could potentially happen in abnormal circumstances?
- What is defined as a Pass or a Fail? What data is the desired output? What is the chain of events?
- What other APIs could this API interact with?
- Who on your team is in charge of testing what?

After you've created these testing boundaries and requirements, you need to decide what you want to test your API for. These are some of the common kinds of API testing:

- Functionality testing — the API technically works.
- Usability testing — the API is easy to work with.
- Reliability testing — the API can be consistently connected to and lead to consistent results.
- Load testing — the API can handle a large amount of calls.
- Creativity testing — the API can handle being used in different ways.
- Security testing — the API has defined security requirements including authentication, permissions and access controls. See some API security tips for protectiving vital data.
- Proficiency testing — the API increases what developers are able to do.
- API documentation testing — also called discovery testing, the API documentation easily guides the user.

Tests will surely vary but here are common API test examples:

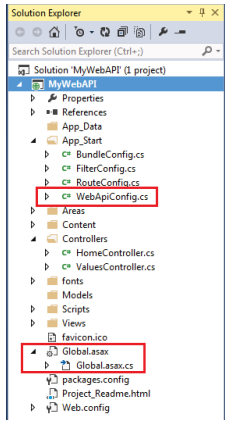
- Checking API return values based on the input condition.
- Verifying if the API doesn't return anything at all or the wrong results.
- Verifying if the API triggers some other event or calls another API.
- Verifying if the API is updating any **data structures**.

Data structure is a data organization, management, and storage format that enables efficient access and modification.

◑ Configure Web API

Web API supports code based configuration. It cannot be configured in web.config file. We can configure Web API to customize the behaviour of Web API hosting infrastructure and components such as routes, formatters, filters, DependencyResolver, MessageHandlers, ParamterBindingRules, properties, services etc.

We created a simple Web API project in the Create Web API Project section. Web API project includes default WebApiConfig class in the App_Start folder and also includes Global.asax as shown below.



Global.asax

public class WebAPIApplication : System.Web.HttpApplication

```
{
    protected void Application_Start()
    {
        GlobalConfiguration.Configure(WebApiConfig.Register);

        //other configuration
    }
}
```

WebApiConfig

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {

        config.MapHttpAttributeRoutes();

        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );

        // configure additional webapi settings here..
    }
}
```

Web API configuration process starts when the application starts. It calls GlobalConfiguration.Configure(WebApiConfig.Register) in the Application_Start method. The Configure() method requires the callback method where Web API has been configured in code. By default this is the static WebApiConfig.Register() method.

As you can see above, WebApiConfig.Register() method includes a parameter of HttpConfiguration type which is then used to configure the Web API. The HttpConfiguration is the main class which includes following properties using which you can override the default behaviour of Web API.

Property	Description
DependencyResolver	Gets or sets the dependency resolver for dependency injection.
Filters	Gets or sets the filters.
Formatters	Gets or sets the media-type formatters.
IncludeErrorDetailPolicy	Gets or sets a value indicating whether error details should be included in error messages.
MessageHandlers	Gets or sets the message handlers.
ParameterBindingRules	Gets the collection of rules for how parameters should be bound.
Properties	Gets the properties associated with this Web API instance.
Routes	Gets the collection of routes configured for the Web API.
Services	Gets the Web API services.

LEARNING ACTIVITY: SELENIUM FRAMEWORK FOR URL BASED API TESTING

In order to provide effective and efficient software testing services, IT enterprises globally are implementing various automation solutions. At Evoke, we have been actively offering automation solutions for numerous customer engagements across multiple industry verticals.

Read more details on:

https://www.evoketechnologies.com/blog/selenium-framework-url-based-api-testing/

How to configure web API? Define with a suitable example.

Define your Direction: On a separate piece of paper, explain the effect of the different types of the terms according to API usages and testing procedure:

Terms	***Effect***
Windows API	
Application Binary Interface	
Spreadsheets	
Database Software	
Representational State Transfer	

CASE STUDY

FEW GOOD REASONS FOR WHY YOU NEED WEB APIS FOR ASP.NET APPLICATION DEVELOPMENT

Whether you want to build a website or web application, a web API is mandatory. So, what actually is an API and when do you require it? API, Application Programming Interface resembles the online web services that are used by the apps at the client side to retrieve and update information. An API would hold on to a central logic of an application irrespective of number of client apps utilizing it.

In order to better understand this, let us consider an example, where there are number of mobile apps supported by different platforms and a web app. If there is no centralized web API, then each client app will have its own embedded business logic which would enable it to connect directly to the database in order to retrieve, update and manipulate the data. Hence, implementing any change to the client apps will be a complex process as each of the app would be required to maintain in sync with each other. Moreover, in case of any update, it would be required to make changes to each and every app individually. Hence, this requires more efforts, takes more time and costs and can also result in lot of bugs or errors in the application.

So, a better way is to use a central API to get, update or change the data. It will be easy to maintain the apps and it would be required to make changes only at one place.

In case of ASP.NET, a web API is an easy way to implement RESTful web services using .NET framework. The RESTful web services are those that use HTTP as the underlying method for communication. An ASP.NET web API is basically defined as a framework that enables development of HTTP services to reach out to client entities like browsers, devices or tablets. ASP.NET Web API can be used with MVC for any type of application. Hence, .NET web APIs are very important for ASP.NET web application development.

Whatever be the type of ASP.Net website you opt to develop whether MVC or Web Forms website, AJAX is required. A web API can help you develop ASP.NET application via AJAX. Using a web API framework, one can easily create services that can run on various entities. Hence, web API makes it easier for the developers to build an ASP.NET application that is compatible with any browser and almost any device. With Web API, you get access to entire features of HTTP like URIs, request/response headers, content formatting, caching etc., and hence, as compared to WCF rest services (which require defining extra config settings for different devices), it is much easier to develop ASP.NET web applications using RESTful web services through Web APIs.

Back End for Native Mobile Applications:

If you are looking for a back end to develop native applications for mobile devices that do not support SOAP, ASP.NET Web API can serve your purpose. Almost any native application running on mobile device other than the Windows one can use ASP.NET Web API as backend. Hence, a web API is good for using with native applications which require web services but not SOAP support.

Develop AJAX based Web Applications:

ASP.NET web API is an ideal choice for development of client web applications that heavily rely on AJAX and do not require extensive configuration settings like WCF REST services.

Light Weight and Easy Creation of Services:

The Web API supports a light architecture powering HTTP services to reach broader range of clients. As compared to WCF, it is much easier and quicker to create services using ASP.NET Web API.

Hence, Web APIs can be helpful in various significant ways in web application development especially when it is an ASP.NET web application.

So, are you looking for ASP.NET web application development based on Web API technology, then look out for experts to help you. At Brainvire, we have experienced .NET developers who rich experience into ASP.NET application development using the latest technologies and tools. Approach us for a free quote and more details about our services.

KNOWLEDGE CHECK

1. **Define the role of API in object-oriented languages.**
2. **Discuss on API libraries and frameworks.**
3. **Give an overview on API and protocols**
4. **What do you understand by the remote APIs?**
5. **Identify the challenges in API testing.**
6. **In order to transfer data between a database and an application written in the Java programming language, the JDBC API provides which of these methods?**

   a) Methods on the ResultSet class for retrieving SQL SELECT results as Java types.

   b) Methods on the PreparedStatement class for sending Java types as SQL statement parameters.

   c) Methods on the CallableStatement class for retrieving SQL OUT parameters as Java types.

   d) All mentioned above

7. **Which of the following component of HTTP request is used to identify the resource on server?**

   a) VERB

   b) URI

   c) HTTP Version

   d) Request Header

8. **What are the tools used for API testing?**

   a) SoapUI Pro

   b) PostMan

   c) Alertsite API monitoring

   d) All mentioned above

9. **Which of the following is a best practice for designing a secure RESTful web service?**

   a) No sensitive data in URL - Never use username, password or session token in

URL , these values should be passed to Web Service via POST method.

b) Restriction on Method execution - Allow restricted use of methods like GET, POST, DELETE. GET method should not be able to delete data.

c) Both of the above.

d) None of the above.

10. Request Body - Contains Message content or Resource representation.

a) True

b) False

11. Web API supports which of the following protocol?

a) TCP

b) HTTP

c) Soap

d) All of the above

12. Web API supports which of the following request/response data formats by default?

a) JSON

b) XML

c) BSON

d) All of the above

13. Which of the following .NET framework supports Web API?

a) .NET 2.0

b) .NET 3.0

c) .NET 3.5

d) .NET 4.0

14. Which of the following statement is TRUE?

a) Web API can be configured using web.config.

b) Web API can only be configured by code.

c) Web API can be configured using app.config.

d) None of the above

15. Web API uses which of the following open-source library for JSON serialization?

a) Json.NET

b) JsonFormatter.NET

c) GetJson.NET

d) None of the above

16. Web API controller must be derived from _____.

a) Controller class

b) ApiController class

c) WebApiController class

d) WebController class

17. Which of the following types of routing is supported in Web API 2?

a) Attribute Routing

b) Convention-based Routing

c) All of the above

d) None of the above

18. Which of the following types are valid response types of Web API 2 action method?

a) HttpResponseMessage

b) IHttpActionResult

c) Custom types

d) All of the above

19. Which of the following action method names are valid to handle HTTP GET request?

a) Get

b) GetAllStudents

c) GetStudent

d) All of the above

20. What is the strategy of API testing? Also define the starting procedure of API testing.

REFERENCES

1. Alexa Top 500 Global Sites". www.alexa.com. Retrieved 2015-11-04.
2. An Online Platform for Web APIs and Service Mashups". IEEE Internet Computing. 12 (5). Sep–Oct 2008. doi:10.1109/MIC.2008.92.
3. API Development with Lumen (Laravel) Microframework". Retrieved 2016-09-19.
4. API Directory". ProgrammableWeb. Retrieved 2015-11-03.
5. APIs.io - the API search engine". apis.io. Retrieved 2016-03-14.So make it clearly understandable & share your knowledge.
6. APIs.json". apisjson.org. Retrieved 2016-03-14.
7. Benslimane, D.; Dustdar, S.; Sheth, A. (2008). "Services Mashups: The New Generation of Web Applications". IEEE Internet Computing. 10 (5): 13–15. doi:10.1109/MIC.2008.110.
8. Benslimane, Djamal; Schahram Dustdar, and Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications" (http:// dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1& path=dsonline/2008/09&file=w5gei.xml&xsl=article.xsl). IEEE Internet Computing, vol. 12, no. 5. Institute of Electrical and Electronics Engineers. pp. 13–15. .
9. Clarifai API: Large Scale Visual Recognition". developer.clarifai.com. Retrieved 2015-11-04.
10. Difference between Web API and Web Service?". programmers.stackexchange.com. Retrieved 2015-11-03.
11. Disqus – The Web's Community of Communities". Disqus. Retrieved 2015-11-04.
12. Microsoft (October 2001). "Run Older Programs On Windows XP" (http://www.microsoft.com/windowsxp/using/helpandsupport/ learnmore/appcompat.mspx). Microsoft. p. 4. .
13. Mulloy, Brian. Web API Design - Crafting Interfaces that Developers Love (PDF). apigee. p. 11.
14. Niccolai, James (2008-04-23), "So What Is an Enterprise Mashup, Anyway?" (http://www.pcworld.com/businesscenter/article/145039/ so_what_is_an_enterprise_mashup_anyway.html), PC World, "Dynamic Community content via APIs". October 26, 2009.
15. REST vs SOAP, the difference between soap and rest : spf13.com". spf13 is Steve Francia. 2010-01-15. Retrieved 2015-11-02.

CHAPTER 4

CASCADING STYLE SHEETS (CSS)



INTRODUCTION

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects. CSS is easy to learn and understand but it provides a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

SCENARIO: CSS SPECIFICATIONS

CSS is the standard and preferred mechanism for formatting HTML pages. Conforming with the separation of concerns design pattern and best practice, cascading style sheets provide a central location in which information about what various fonts, foreground colors, background colors, italicization and emphasization should be applied to various HTML elements within a webpage. Cascading style sheets can also control how various parts of a page, such as the header, footer, body, article content, sections and asides, are laid out on the page. This is extremely helpful when content must be laid out in a dramatically different fashion depending upon whether it is being viewed on a desktop, tablet or a smartphone.

What skills will you need?

In order to work effectively in cascading style sheets, you must know:

- Synopsis of CSS
- Function of cascading style sheets

SYNOPSIS OF CSS

Cascading Style Sheet (CSS) is a way to design a website, or a group of websites, so that they have a consistent look and feel, and so that their look and feel is easy to change. By using CSS to design a website, the web developer gains a greater degree of control over how the site appears.

A web developer can use a CSS file to control the look of a website in three main ways. The first way is called inline, referring to the fact that the code is placed right into the line of the website code. For example, a web developer might want to make a particular sentence appear in bold, red type so that it stands out. She could use CSS to set the style of that sentence to bold and red using inline code. The benefit of this method is that it allows a quick and easy change to a particular part of a web page.

Another way that a web developer can use CSS is to make rules for a single web page. In this case, the developer would use what is called embedded CSS. The developer can, for example, make each new paragraph indent and each header in bold. The embedded instructions are usually placed at the top of the web page's code.

This allows the developer to change the embedded code once and have the effects take place throughout the entire page. If he decided to put all headers in italics rather than bold text, he could simply change the style coding, and all the headers on that page would change. This has an advantage over the inline method in that it covers the entire web page, and changes can be made to the entire page at once.

The final common type of CSS is what is known as an external CSS. A web developer will write the code to apply to an entire group of web pages, a whole website, or even multiple web sites. These rules can include things like background color, text color, word spacing, and other elements of page layout, just like the previous two examples of CSS.

The difference is that these instructions are not for a single section of the page, or just one web page, but for an entire website. The advantage is that the look and feel of an entire website can be changed at one time by making changes to the external style sheet. If the designer wants to try a new background color or a new font for the entire website, she can do so with the change of a few lines in the external code, rather than going to each page individually and making changes there.

The disadvantages to both embedded and external CSS is that there is more time involved in creating a large style sheet, which will cover a great deal of layout, rather than making smaller changes to individual lines. The web designer

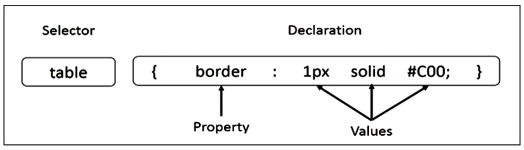
must decide if it is more efficient in the long run to make individual inline instructions, or to make an embedded or external CSS to take care of many design features at the same time.

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts −

- **Selector** − A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property** − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be *color*, *border* etc.
- **Value** − Values are assigned to properties. For example, *color* property can have value either *red* or *#F1F1F1* etc.

You can put CSS Style Rule Syntax as follows −

selector { property: value }



Example − You can define a table border as follows −

table{ border :1px solid #C00; }

Here table is a selector and border is a property and given value *1px solid #C00* is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

The Type Selectors

This is the same selector we have seen above. Again, one more example to give a color to all level 1 headings −

```
h1 {
  color: #36CFFF;

}
```

The Universal Selectors

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type –

```
* {
    color: #000000;
}
```

This rule renders the content of every element in our document in black.

The Descendant Selectors

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to element only when it lies inside <ul> tag.

```
ul em {
    color: #000000;
}
```

The Class Selectors

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
  color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to *black* in our document. You can make it a bit more particular. For example –

```
h1.black {
    color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to *black*.

You can apply more than one class selectors to given element. Consider the following example –

```
<p class = “center bold”>
```

This para will be styled by the classes *center* and *bold*.

</p>

The ID Selectors

You can define style rules based on the *id* attribute of the elements. All the elements having that *id* will be formatted according to the defined rule.

```
#black {
   color: #000000;
}
```

This rule renders the content in black for every element with *id* attribute set to *black* in our document. You can make it a bit more particular. For example –

```
h1#black {
   color: #000000;
}
```

This rule renders the content in black for only <h1> elements with *id* attribute set to *black*.

The true power of *id* selectors is when they are used as the foundation for descendant selectors, For example –

```
#black h2 {
   color: #000000;
}
```

In this example all level 2 headings will be displayed in black color when those headings will lie with in tags having *id* attribute set to *black*.

The Child Selectors

You have seen the **descendant selectors**. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example –

```
body > p {
  color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like or <td> would not have any effect of this rule.

The Attribute Selectors

You can also apply styles to HTML elements with particular attributes. The style rule below will match all the input elements having a type attribute with a value of *text* −

```
input[type = "text"] {
   color: #000000;
}
```

The advantage to this method is that the <input type = "submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- **p[lang]** − Selects all paragraph elements with a *lang* attribute.
- **p[lang="fr"]** − Selects all paragraph elements whose *lang* attribute has a value of exactly «fr".
- **p[lang~="fr"]** − Selects all paragraph elements whose *lang* attribute contains the word «fr".
- **p[lang|="en"]** − Selects all paragraph elements whose *lang* attribute contains values that are exactly «en», or begin with «en-».

Multiple Style Rules

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example −

```
h1 {
   color: #36C;
   font-weight: normal;
   letter-spacing: .4em;
   margin-bottom: 1em;
```



REMEMBER

Descendant selector is made up of two or more selectors separated by white space.

```
    text-transform: lowercase;
}
```

Here all the property and value pairs are separated by a semicolon (;). You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

Grouping Selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example –

```
h1, h2, h3 {
  color: #36C;
  font-weight: normal;
  letter-spacing: .4em;
  margin-bottom: 1em;
  text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine the various *id* selectors together as shown below –

```
#content, #footer, #supplement {
  position: absolute;
  left: 510px;
  width: 200px;
}
```

◑ Advantages of CSS

- CSS saves time - You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element

and apply it to as many web pages as you want.

- Pages load faster - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- Easy maintenance - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.
- Superior styles to HTML - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- Multiple Device Compatibility - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.
- Global web standards – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.

Visibility

A property called visibility allows you to hide an element from view. You can use this property along with JavaScript to create very complex menu and very complex webpage layouts.

You may choose to use the visibility property to hide error messages that are only displayed if the user needs to see them, or to hide answers to a quiz until the user selects an option.

The visibility property can take the values listed in the table that follows –



REMEMBER

Remember that the source code will still contain whatever is in the invisible paragraph, so you should not use this to hide sensitive information such as credit card details or passwords.

Sr.No.	Value & Description
1	**visible** The box and its contents are shown to the user.
2	**hidden** The box and its content are made invisible, although they still affect the layout of the page.
3	**collapse** This is for use only with dynamic table columns and row effects.

Here is an example –

```
<html>
   <head>
   </head>

   <body>
      <p>
         This paragraph should be visible in normal way.
      </p>

      <p style = "visibility:hidden;">
         This paragraph should not be visible.
      </p>
   </body>
</html>
```

It will produce the following result –

This paragraph should be visible in normal way.

◑ Positioning

CSS helps you to position your HTML element. You can put any HTML element at whatever location you like. You can specify whether you want the element positioned relative to its natural position in the page or absolute based on its parent element.

Now, we will see all the CSS positioning related properties with examples –

Relative Positioning

Relative positioning changes the position of the HTML element relative to where it normally appears. So "left:20" adds 20 pixels to the element's LEFT position.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.
- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

You can use *bottom* or *right* values as well in the same way as *top* and *left*.

Here is the example −

```
<html>
   <head>
   </head>

   <body>
     <div style = "position:relative; left:80px; top:2px; background-color:yellow;">
          This div has relative positioning.
      </div>
   </body>
</html>
```

It will produce the following result −

This div has relative positioning.

Absolute Positioning

An element with position: absolute is positioned at the specified coordinates relative to your screen top-left corner.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.
- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

You can use *bottom* or *right* values as well in the same way as top and left.

Here is an example −

Live Demo

```
<html>
   <head>
   </head>

   <body>
    <div style = "position:absolute; left:80px; top:20px; background-color:yellow;">
         This div has absolute positioning.
      </div>
   </body>
</html>
```

Fixed Positioning

Fixed positioning allows you to fix the position of an element to a particular spot on the page, regardless of scrolling. Specified coordinates will be relative to the browser window.

You can use two values *top* and *left* along with the *position* property to move an HTML element anywhere in the HTML document.

- Move Left - Use a negative value for *left*.
- Move Right - Use a positive value for *left*.
- Move Up - Use a negative value for *top*.
- Move Down - Use a positive value for *top*.

You can use *bottom* or *right* values as well in the same way as *top* and *left*.

Here is an example −

```
<html>
   <head>
   </head>

   <body>
     <div style = "position:fixed; left:80px; top:20px; background-color:yellow;">
```

```
        This div has fixed positioning.
      </div>
    </body>
</html>
```

This div has fixed positioning.

Layers

CSS gives you opportunity to create layers of various divisions. The CSS layers refer to applying the *z-index* property to elements that overlap with each other.

The z-index property is used along with the *position* property to create an effect of layers. You can specify which element should come on top and which element should come at bottom.

A z-index property can help you to create more complex webpage layouts. Following is the example which shows how to create layers in CSS.

```
<html>
   <head>
   </head>

   <body>
      <div style = "background-color:red;
         width:300px;
         height:100px;
         position:relative;
         top:10px;
         left:80px;
         z-index:2">
      </div>

      <div style = "background-color:yellow;
         width:300px;
         height:100px;
         position:relative;
         top:-60px;
```

```
            left:35px;
            z-index:1;">
        </div>

        <div style = "background-color:green;
            width:300px;
            height:100px;
            position:relative;
            top:-220px;
            left:120px;
            z-index:3;">
        </div>
    </body>
</html>
```

It will produce the following result –



◑ Pseudo Classes

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions.

CSS pseudo-classes are used to add special effects to some selectors. You do not need to use **JavaScript** or any other script to use those effects.

A simple syntax of pseudo-classes is as follows –

```
selector:pseudo-class {property: value}
```

CSS classes can also be used with pseudo-classes –

```
selector.class:pseudo-class {property: value}
```

The most commonly used pseudo-classes are as follows –

Sr.No.	Value & Description
1	**:link** Use this class to add special style to an unvisited link.
2	**:visited** Use this class to add special style to a visited link.
3	**:hover** Use this class to add special style to an element when you mouse over it.
4	**:active** Use this class to add special style to an active element.
5	**:focus** Use this class to add special style to an element while the element has focus.
6	**:first-child** Use this class to add special style to an element that is the first child of some other element.
7	**:lang** Use this class to specify a language to use in a specified element.

While defining pseudo-classes in a <style>...</style> block, following points should be noted –

- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.
- a:active MUST come after a:hover in the CSS definition in order to be effective.
- Pseudo-class names are not case-sensitive.
- Pseudo-class are different from CSS classes but they can be combined.

The: link pseudo-class

The following example demonstrates how to use the: link class to set the link color. Possible values could be any color name in any valid format.

```
<html>
   <head>
```

```
        <style type = "text/css">
            a:link {color:#000000}
        </style>
    </head>

    <body>
        <a href = "">Black Link</a>
    </body>
</html>
```

It will produce the following black link –

Black Link

The: visited pseudo-class

The following is the example which demonstrates how to use the: visited class to set the color of visited links. Possible values could be any color name in any valid format.

```
<html>
    <head>
        <style type = "text/css">
            a:visited {color: #006600}
        </style>
    </head>

    <body>
        <a href = "">Click this link</a>
    </body>
</html>
```

This will produce following link. Once you will click this link, it will change its color to green.

Click this link

The: hover pseudo-class

The following example demonstrates how to use the: hover class to change the color of links when we bring a mouse pointer over that link. Possible values could

be any color name in any valid format.

```
<html>
   <head>
      <style type = "text/css">
         a:hover {color: #FFCC00}
      </style>
   </head>

   <body>
      <a href = "">Bring Mouse Here</a>
   </body>
</html>
```

It will produce the following link. Now you bring your mouse over this link and you will see that it changes its color to yellow.

Bring Mouse Here

The: active pseudo-class

The following example demonstrates how to use the: active class to change the color of active links. Possible values could be any color name in any valid format.

```
<html>
   <head>
      <style type = "text/css">
         a:active {color: #FF00CC}
      </style>
   </head>

   <body>
      <a href = "">Click This Link</a>
   </body>
</html>
```

It will produce the following link. When a user clicks it, the color changes to pink.

Click this Link

The: focus pseudo-class

The following example demonstrates how to use the: focus class to change the color of focused links. Possible values could be any color name in any valid format.

```
<html>
   <head>
      <style type = "text/css">
         a:focus {color: #0000FF}
      </style>
   </head>

   <body>
      <a href = "">Click this Link</a>
   </body>
</html>
```

It will produce the following link. When this link gets focused, its color changes to orange. The color changes back when it loses focus.

Click this Link

The: first-child pseudo-class

The: first-child pseudo-class matches a specified element that is the first child of another element and adds special style to that element that is the first child of some other element.

To make: first-child work in IE <!DOCTYPE> must be declared at the top of document.

For example, to indent the first paragraph of all elements, you could use this definition –

```
<html>
   <head>
      <style type = "text/css">
         div > p:first-child {
            text-indent: 25px;
         }
      </style>
   </head>
```

```
   <body>

      <div>
         <p>First paragraph in div. This paragraph will be indented</p>
         <p>Second paragraph in div. This paragraph will not be indented</p>
      </div>
      <p>But it will not match the paragraph in this HTML:</p>

      <div>
         <h3>Heading</h3>
       <p>The first paragraph inside the div. This paragraph will not be effected.</
p>
      </div>

   </body>
</html>
```

It will produce the following result –

First paragraph in div. This paragraph will be indented

Second paragraph in div. This paragraph will not be indented

But it will not match the paragraph in this HTML:

Heading

The first paragraph inside the div. This paragraph will not be effected.

The: lang pseudo-class

The language pseudo-class: lang, allows constructing selectors based on the language setting for specific tags.

This class is useful in documents that must appeal to multiple languages that have different conventions for certain language constructs. For example, the French language typically uses angle brackets (< and >) for quoting purposes, while the English language uses quote marks (‘ and ‘).

In a document that needs to address this difference, you can use the: lang pseudo-class to change the quote marks appropriately. The following code changes the <blockquote> tag appropriately for the language being used –

```
<html>
   <head>
      <style type = "text/css">

         /* Two levels of quotes for two languages*/
         :lang(en) { quotes: '"' '"'  "'"  "'"; }
         :lang(fr) { quotes: "<<" ">>" "<" ">"; }
      </style>
   </head>

   <body>
      <p>...<q lang = "fr">A quote in a paragraph</q>...</p>
   </body>
</html>
```

The: lang selectors will apply to all the elements in the document. However, not all elements make use of the quotes property, so the effect will be transparent for most elements.

It will produce the following result –

..<<A quote in a paragraph>>...

LEARNING ACTIVITY: TYPES OF CASCADING STYLE SHEETS

CSS stands for Cascading Style Sheets. It is generally used to display HTML in various views. It is used to design the view for HTML. CSS is a combination of a selector and a declaration.

Read more details on:

https://www.c-sharpcorner.com/UploadFile/e6a884/types-of-css/

Discuss on active pseudo-class and also write a program on it.

Define your Direction: On a separate piece of paper, explain the consequence of the different types of the terms according to synopsis of CSS:

Terms	***Consequence***
Web Developer	
Web Pages	
Style sheets	
CSS pseudo-classes	
Language Pseudo-Class	

FUNCTION OF CASCADING STYLE SHEETS

Cascading style sheets are used to format the layout of Web pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML.

CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once. For example, a Web developer may want to increase the default text size from 10pt to 12pt for fifty pages of a Web site. If the pages all reference the same style sheet, the text size only needs to be changed on the style sheet and all the pages will show the larger text.

While CSS is great for creating text styles, it is helpful for formatting other aspects of Web page layout as well. For example, CSS can be used to define the **cell padding** of table cells, the style, thickness, and color of a table's border, and the padding around images or other objects. CSS gives Web developers more exact control over how Web pages will look than HTML does. This is why most Web pages today incorporate cascading style sheets.

◑ Inline Styles

A Web developer can declare document styles in many ways. This method presents inline styles that declare an individual element's format using the XHTML attribute style. Figure 1 applies inline styles to p elements to alter their font size and color.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 1: inline.html -->
6 <!-- Using inline styles -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Inline Styles</title>
```

```
10 </head>
11 <body>
12 <p>This text does not have any style applied to it.</p>
13
14 <!-- The style attribute allows you to declare -->
15 <!-- inline styles. Separate multiple style properties -->
16 <!-- with a semicolon. -->
17 <p style = “font-size: 20pt”> This text has the
18 <em>font-size</em> style applied to it, making it 20pt.
19 </p>
20
21 <p style = “font-size: 20pt; color: #6666ff”>
22 This text has the <em>font-size</em> and
23 <em>color</em> styles applied to it, making it
24 20pt. and light blue.</p>
25 </body>
26 </html>
```

Cell padding is the space between cell borders and the content within a cell.

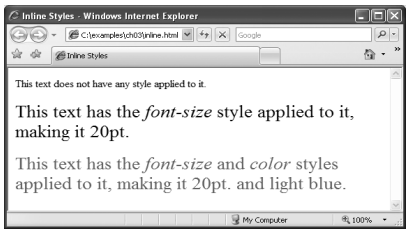


Figure 1: Using inline styles.

The first inline style declaration appears in line 20. Attribute style specifies the style for an element. Each CSS property (the font-size property in this case) is followed by a colon and a value. In line 20, we declare this particular p element to use 20-point font size. Line 21 uses element em to “emphasize” text, which most browsers do by making the font italic.

Line 24 specifies the two properties, font-size and color, separated by a semicolon. In this line, we set the given paragraph’s color to blue, using the hexadecimal

code #0000ff. Color names may be used in place of hexadecimal codes.

◑ Embedded Style Sheets

A second technique for using style sheets is embedded style sheets. Embedded style sheets enable a Web-page programmer to embed an entire CSS document in an XHTML document's head section. Figure 2 creates an embedded style sheet containing four styles.

```
<?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 2: embedded.html -->
6 <!-- Embedded style sheets. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Style Sheets</title>
10
11 <!-- this begins the style sheet section -->
12 <style type = "text/css">
13 em { font-weight: bold;
14 color: black }
15 h1 { font-family: tahoma, helvetica, sans-serif }
16 p { font-size: 12pt;
17 font-family: arial, sans-serif }
18 .special { color: #6666ff }
19 </style>
20 </head>
21 <body>
22 <!-- this class attribute applies the .special style -->
23 <h1 class = "special">Deitel &amp;amp; Associates, Inc.</h1>
24
25 <p>Deitel &amp;amp; Associates, Inc. is an internationally
26 recognized corporate training and publishing organization
```

```
27 specializing in programming languages, Internet/World
28 Wide Web technology and object technology education.
29 The company provides courses on Java, C++, Visual Basic,
30 C#, C, Internet and World Wide Web programming, Object
31 Technology, and more.</p>
32
33 <h1>Clients</h1>
34 <p class = “special”> The company’s clients include many
35 <em>Fortune 1000 companies</em>, government agencies,
36 branches of the military and business organizations.
37 Through its publishing partnership with Prentice Hall,
38 Deitel &amp;amp; Associates, Inc. publishes leading-edge
39 programming textbooks, professional books, interactive
40 web-based multimedia Cyber Classrooms, satellite
41 courses and World Wide Web courses.</p>
42 </body>
43 </html>
```

MIME (Multi-Purpose Internet Mail Extensions) is an extension of the original Internet e-mail protocol that lets people use the protocol to exchange different kinds of data files on the Internet: audio, video, images, application programs, and other kinds, as well as the ASCII text handled in the original protocol, the Simple Mail Transport Protocol (SMTP).

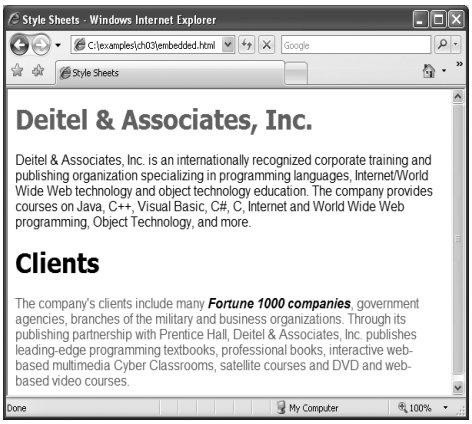


Figure 2: Embedded style sheets.

The style element (lines 13–24) defines the embedded style sheet. Styles placed in the head apply to matching elements wherever they appear in the entire document. The style element's type attribute specifies the **Multipurpose Internet Mail Extensions (MIME)** type that describes a file's content. CSS documents use the MIME type text/ css. Other MIME types include image/gif (for GIF images) and text/javascript.

The body of the style sheet (lines 15–22) declares the CSS rules for the style sheet. We declare rules for em (lines 15–16), h1 (line 18) and p (line 20) elements. When the browser renders this document, it applies the properties defined in these rules to every element to which the rule applies. For example, the rule in lines 15–16 will be applied to all em elements (in this example, there is one in line 43). The body of each rule is enclosed in curly braces ({and}).

Line 22 declares a style class named special. Style classes define styles that can be applied to any type of element. In this example, we declare class special, which sets color to blue. We can apply this style to elements of any type, whereas the other rules in this style sheet apply only to specific element types (i.e., em, h1 or p). Style class declarations are preceded by a period. We will discuss how to apply a style class momentarily.

CSS rules in embedded style sheets use the same syntax as inline styles; the property name is followed by a colon (:) and the value of the property. Multiple properties are separated by semicolons (;). In the rule for em elements, the color property specifies the color of the text, and property background-color specifies the background color of the element.

The font-family property (line 18) specifies the name of the font to use. In this case, we use the arial font. The second value, sans-serif, is a generic font family. Not all users have the same fonts installed on their computers, so Web-page programmers often specify a comma-separated list of fonts to use for a particular style. The browser attempts to use the fonts in the order they appear in the list. Many Web-page programmers end a font list with a generic font family name in case the other fonts are not installed on the user's computer. In this example, if the arial font is not found on the system, the browser instead will display a generic sans-serif font, such as helvetica or verdana. Other generic font families include serif (e.g., times new roman, Georgia), cursive (e.g., script), fantasy (e.g., critter) and monospace (e.g., courier, fixedsys).

The font-size property (line 20) specifies a 14-point font. Other possible measurements in addition to pt (point). Relative values— xxsmall, x-small, small, smaller, medium, large, larger, x-large and xx-large— also can be used. Generally, relative values for font-size are preferred over point sizes because a programmer does not know the specific measurements of the display for each client. Relative font-size values permit moreflexible viewing of Web pages. For example, a user may wish to view a Web page on a handheld device with a small screen.

Specifying an 18- point font size in a style sheet will prevent such a user from seeing more than one or two characters at a time. However, if a relative font size is specified, such as large or larger, the actual size is determined by the browser that displays the font. Using relative sizes also makes pages more accessible to users with disabilities. Users with impaired vision, for example, may configure their browser to use a larger default font, upon which all relative sizes are based. Text that the programmer specifies to be smaller than the main text still displays in a smaller size font, yet it is clearly visible to each user.

Line 30 uses attribute class in an h1 element to apply a style class—in this case class special (declared as .special in the style sheet). When the browser renders the h1 element, note that the text appears on screen with the properties of both an h1 element (arial or sans-serif font defined in line 18) and the .special style class applied (the color blue defined in line 22).

The formatting for the p element and the .special class are applied to the text in lines 42–49. All the styles applied to an element (the parent or ancestor element) also apply to the element's nested elements (child or descendant elements). The em element nested in the p element in line 43 inherits the style from the p element (namely, the 14-point font size in line 20), but retains its italic style. The em element has its own color property, so it overrides the color property of the special class.

◑ Conflicting Styles

Cascading style sheets are "cascading" because styles may be defined by a user, an programmer or a user agent (e.g., a Web browser). Styles "cascade," or flow together, such that the ultimate appearance of elements on a page results from combining styles defined in several ways. Styles defined by the user take precedence over styles defined by the user agent, and styles defined by programmers take precedence over styles defined by the user. Styles defined for parent elements are also inherited by child (nested) elements. In this section, we discuss the rules for resolving conflicts between styles defined for elements and styles inherited from parent and ancestor elements

Figure 2 presented an example of inheritance in which a child em element inherited the font-size property from its parent p element. However, in Figure 2, the child em element had a color property that conflicted with (i.e., had a different value than) the color property of its parent p element. Properties defined for child and descendant elements have a greater specificity than properties defined for parent and ancestor elements.

Line 14 applies property text-decoration to all a elements whose class attribute is set to nodec. The text-decoration property applies decorations to text within



REMEMBER

Comments may be placed in any type of CSS code (i.e., inline styles, embedded style sheets and external style sheets) and always start with /* and end with */. Text between these delimiters is ignored by the browser.

an element. By default, browsers underline the text of an a (anchor) element. Here, we set the text-decoration property to none to indicate that the browser should not underline hyperlinks. Other possible values for text-decoration include overline, linethrough, underline and blink. [Note: blink is not supported by Internet Explorer.] The .nodec appended to a is an extension of class styles; this style will apply only to an elements that specify nodec in their class attribute.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
45
<!-- Fig. 3: advanced.html -->
6 <!-- Inheritance in style sheets. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>More Styles</title>
10 <style type = "text/css">
11 body { font-family: arial, helvetica, sans-serif }
12 a.nodec { text-decoration: none }
13 a:hover { text-decoration: underline }
14 li em { font-weight: bold }
15 h1, em { text-decoration: underline }
16 ul { margin-left: 20px }
17 ul ul { font-size: .8em }
18 </style>
19 </head>
20 <body>
21 <h1>Shopping list for Monday:</h1>
22
23 <ul>
24 <li>Milk</li>
25 <li>Bread
```

```
26 <ul>
27 <li>White bread</li>
28 <li>Rye bread</li>
29 <li>Whole wheat bread</li>
30 </ul>
31 </li>
32 <li>Rice</li>
33 <li>Potatoes</li>
34 <li>Pizza <em>with mushrooms</em></li>
35 </ul>
36
37 <p><em>Go to the</em>
38 <a class = “nodec” href = “http://www.deitel.com”>
39 Grocery store</a>
40 </p>
41 </body>
42 </html>
```

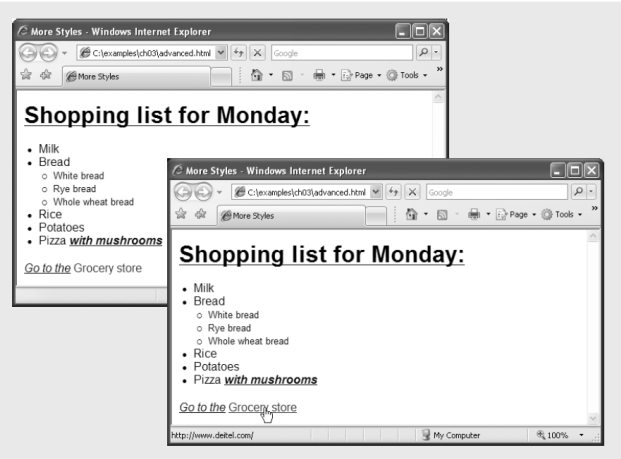


Figure 3: Inheritance in style sheets.

◑ Linking External Style Sheets

Style sheets are a convenient way to create a document with a uniform theme.

With external style sheets (i.e., separate documents that contain only CSS rules), Web-page programmers can provide a uniform look and feel to an entire Web site. Different pages on a site can all use the same style sheet. When changes to the styles are required, the Web-page programmer needs to modify only a single CSS file to make style changes across the entire Web site. Figure 4 presents an external style sheet. Lines 1–2 are CSS comments. Like XHTML comments, CSS comments describe the content of a CSS document.

```
1 /* Fig. 4: styles.css */
2 /* External stylesheet */
3
4 body { font-family: arial, helvetica, sans-serif }
5
6a.nodec {text-decoration: none }
7
8a:hover { text-decoration: underline }
9
10 li em { font-weight: bold }
11
12 h1, em { text-decoration: underline }
13
14 ul { margin-left: 20px }
15
16 ul ul { font-size: .8em; }
```

Figure 4. External style sheet.

Figure 5 contains an XHTML document that references the external style sheet in Figure.4. Lines 11–12 (Figure 5) show a link element that uses the rel attribute to specify a relationship between the current document and another document. In this case, we declare the linked document to be a stylesheet for this document. The type attribute specifies the MIME type as text/css. The href attribute provides the URL for the document containing the style sheet. In this case, styles.css is in the same directory as external.html.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5<!-- Fig. 5: external.html -->
```

```
6 <!-- Linking an external style sheet. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Linking External Style Sheets</title>
10 <link rel = "stylesheet" type = "text/css"
11 href = "styles.css" />
12 </head>
13 <body>
14 <h1>Shopping list for <em>Monday</em>:</h1>
15
16 <ul>
17 <li>Milk</li>
18 <li>Bread
19 <ul>
20 <li>White bread</li>
21 <li>Rye bread</li>
22 <li>Whole wheat bread</li>
23 </ul>
24 </li>
25 <li>Rice</li>
26 <li>Potatoes</li>
27 <li>Pizza <em>with mushrooms</em></li>
28 </ul>
29
30 <p><em>Go to the</em>
31 <a class = "nodec" href = "http://www.deitel.com">
32 Grocery store</a>
33 </p>
34 </body>
35 </html>
```

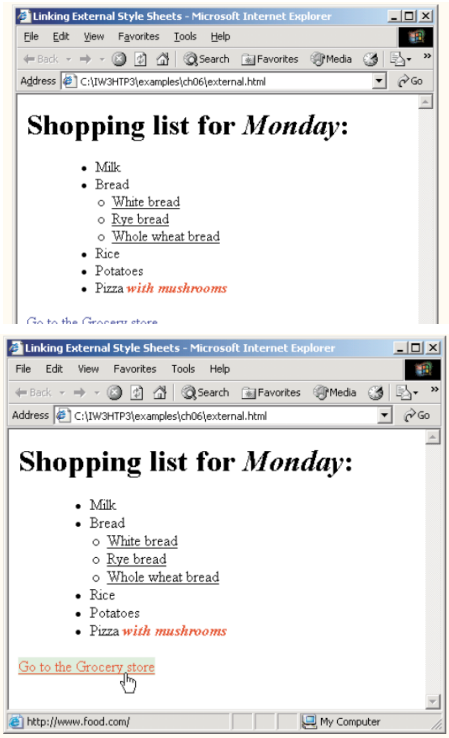


Figure 5: Linking an external style sheet.

◑ Positioning Elements

Before CSS, controlling the positioning of elements in an XHTML document was difficult—the browser determined positioning. CSS introduced the position property and a capability called absolute positioning, which gives programmer greater control over how document elements are displayed. Figure 6 demonstrates absolute positioning.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 6: positioning.html -->
6 <!-- Absolute positioning of elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
```

```
8 <head>
9 <title>Absolute Positioning</title>
10 <style type = “text/css”>
11 .bgimg { position: absolute;
12 top: 0px;
13 left: 0px;
14 z-index: 1 }
15 .fgimg { position: absolute;
16 top: 25px;
17 left: 100px;
18 z-index: 2 }
19 .text { position: absolute;
20 top: 25px;
21 left: 100px;
22 z-index: 3;
23 font-size: 20pt;
24 font-family: tahoma, geneva, sans-serif }
25 </style>
26 </head>
27 <body>
28 <p><img src = “bgimg.gif” class = “bgimg”
29 alt = “First positioned image” /></p>
30
31 <p><img src = “fgimg.gif” class = “fgimg”
32 alt = “Second positioned image” /></p>
33
34 <p class = “text”>Positioned Text</p>
35 </body>
36 </html>
```

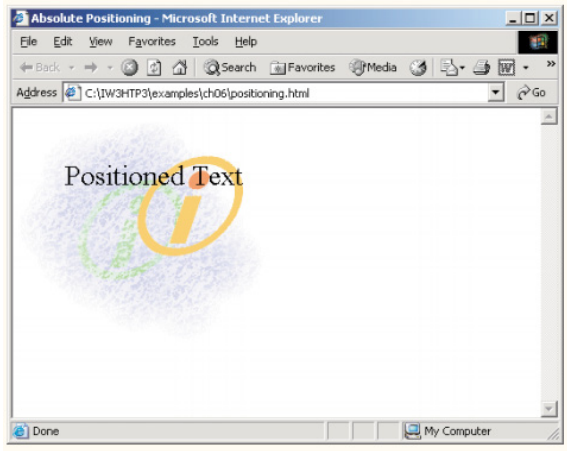


Figure 6: Absolute positioning of elements with CSS.

Lines 15–17 position the first img element (i.gif) on the page. Specifying an element's position as absolute removes the element from the normal flow of elements on the page, instead positioning it according to the distance from the top, left, right or bottom margins of its containing **block-level element** (i.e., an element such as body or p). Here, we position the element to be 0 pixels away from both the top and left margins of the p element (lines 15–17).

Block-level element is any element that starts on a new line, like a paragraph, and takes up the full width of a page, from left to right.

The z-index attribute allows you to layer overlapping elements properly. Elements that have higher z-index values are displayed in front of elements with lower z-index values. In this example, i.gif has the lowest z-index (1), so it displays in the background. The img element in lines 20–22 (circle.gif) has a z-index of 2, so it displays in front of i.gif. The p element in lines 18–19 (Positioned Text) has a z-index of 3, so it displays in front of the other two. If you do not specify a z-index or if elements have the same z-index value, the elements are placed from background to foreground in the order they are encountered in the document.

Absolute positioning is not the only way to specify page layout. Figure 7 demonstrates relative positioning, in which elements are positioned relative to other elements.

1 <?xml version = "1.0" encoding = "utf-8"?>

```
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5<!-- Fig. 7: positioning2.html -->
6 <!-- Relative positioning of elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Relative Positioning</title>
10 <style type = "text/css">
11 p { font-size: 1.3em;
12 font-family: verdana, arial, sans-serif }
13 span { color: red;
14 font-size: .6em;
15 height: 1em }
16 .super { position: relative;
17 top: -1ex }
18 .sub { position: relative;
19 bottom: -1ex }
20 .shiftleft { position: relative;
21 left: -1ex }
22 .shiftright { position: relative;
23 right: -1ex }
24 </style>
25 </head>
```

```
26 <body>
27 <p>The text at the end of this sentence
28 <span class = “super”>is in superscript</span>.</p>
29
30 <p>The text at the end of this sentence
31 <span class = “sub”>is in subscript</span>.</p>
32
33 <p>The text at the end of this sentence
34 <span class = “shiftleft”>is shifted left</span>.</p>
35
36 <p>The text at the end of this sentence
37 <span class = “shiftright”>is shifted right</span>.</p>
38 </body>
39 </html>
```

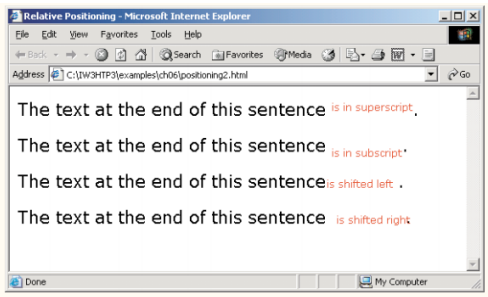


Figure 7: Relative positioning of elements with CSS.

Setting the position property to relative, as in class super (lines 21–22),lays out the element on the page and offsets it by the specified top, bottom, left or right value. Unlike absolute positioning, relative positioning keeps elements in the general flow of elements on the page, so positioning is relative to other elements in the flow. Recall that ex (line 22) is the x-height of a font, a relative length measurement typically equal to the height of a lowercase x.

We introduce the span element in line 39. Element span is a grouping element—it does not apply any inherent formatting to its contents. Its primary purpose is to apply CSS rules or id attributes to a block of text. Element span is an inline-level element—it is displayed inline with other text and with no line breaks. Lines 17–19 define the CSS rule for span. A similar element is the div element, which also applies no inherent styles but is displayed on its own line, with margins above and below (a block-level element).

◑ Backgrounds

CSS provides control over the element backgrounds. We introduced the background-color property. CSS also can add background images to documents. Figure 8 adds a corporate logo to the bottom-right corner of the document. This logo stays fixed in the corner even when the user scrolls up or down the screen.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5 <!-- Fig. 8: background.html -->
6 <!-- Adding background images and indentation. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Background Images</title>
10 <style type = "text/css">
11 body { background-image: url(logo.gif);
12 background-position: bottom right;
13 background-repeat: no-repeat;
14 background-attachment: fixed;
15 background-color: #eeeeee }
16 p { font-size: 18pt;
17 color: #1144AA;
18 text-indent: 1em;
19 font-family: arial, sans-serif; }
20 .dark { font-weight: bold }
21 </style>
```

```
22 </head>
23 <body>
24 <p>
25 This example uses the background-image,
26 background-position and background-attachment
27 styles to place the <span class = “dark”>Deitel
28 &amp;amp; Associates, Inc.</span> logo in the bottom,
29 right corner of the page. Notice how the logo
30 stays in the proper position when you resize the
31 browser window. The background-color fills in where
32 there is no image.
33 </p>
34 </body>
35 </html>
```

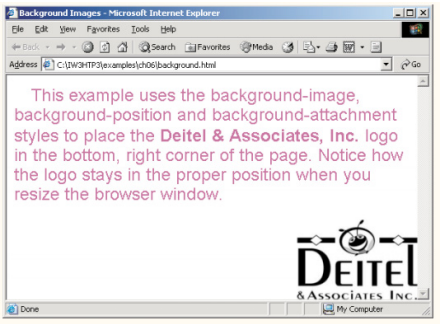


Figure 8: Background image added with CSS.

The background-image property (line 14) specifies the image URL for the image logo.gif in the format url(fileLocation). The Web-page programmer also can set the background-color property in case the image is not found. The background-position property (line 15) places the image on the page. The keywords top, bottom, center, left and right are used individually or in combination for vertical and horizontal positioning. An image can be positioned using lengths by specifying the horizontal length followed by the vertical length. For example, to position the image as horizontally centered (positioned at 50% of the distance across the screen) and 30 pixels from the top, use

background-position: 50% 30px;

The background-repeat property (line 16) controls the tiling of the background image. Tiling places multiple copies of the image next to each other to fill the background. Here, we set the tiling to no-repeat to display only one copy of the background image. The background-repeat property can be set to repeat (the default) to tile the image vertically and horizontally, repeat-x to tile the image only horizontally or repeat-y to tile the image only vertically.

The final property setting, background-attachment: fixed (line 17), fixes the image in the position specified by background-position. Scrolling the browser window will not move the image from its position. The default value, scroll, moves the image as the user scrolls through the document.

Line 21 indents the first line of text in the element by the specified amount, in this case 1em. A programmer might use this property to create a Web page that reads more like a novel, in which the first line of every paragraph is indented.

Line 24 uses the font-weight property to specify the "boldness" of text. Possible values are bold, normal (the default), bolder (bolder than bold text) and lighter (lighter than normal text). Boldness also can be specified with multiples of 100, from 100 to 900 (e.g., 100, 200, …, 900). Text specified as normal is equivalent to 400, and bold text is equivalent to 700. However, many systems do not have fonts that can scale with this level of precision, so using the values from 100 to 900 might not display the desired effect. Another CSS property that formats text is the font-style property, which allows the developer to set text to none, italic or oblique (oblique will default to italic if the system does not support oblique text).

◑ Element Dimensions

In addition to positioning elements, CSS rules can specify the actual dimensions of each page element. Figure 9 demonstrates how to set the dimensions of elements.

```
1 <?xml version = "1.0" encoding = "utf-8"?>

2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

4

5 <!-- Fig. 9: width.html -->

6 <!-- Element dimensions and text alignment. -->

7 <html xmlns = "http://www.w3.org/1999/xhtml">

8 <head>
```

```
9 <title>Box Dimensions</title>
10 <style type = “text/css”>
11 div { background-color: #aaccff;
12 margin-bottom: .5em;
13 font-family: arial, helvetica, sans-serif }
14 </style>
15 </head>
16 <body>
17  <div style = “width: 20%”> Here is some
18 text that goes in a box which is
19 set to stretch across twenty percent
20 of the width of the screen.</div>
21
22 <div style = “width: 80%; text-align: center”>
23 Here is some CENTERED text that goes in a box
24 which is set to stretch across eighty percent of
25 the width of the screen.</div>
26
27 <div style = “width: 20%; height: 150px; overflow: scroll”>
28 This box is only twenty percent of
29 the width and has a fixed height.
30 What do we do if it overflows? Set the
31 overflow property to scroll!</div>
32 </body>
```

```
33 </html>
```

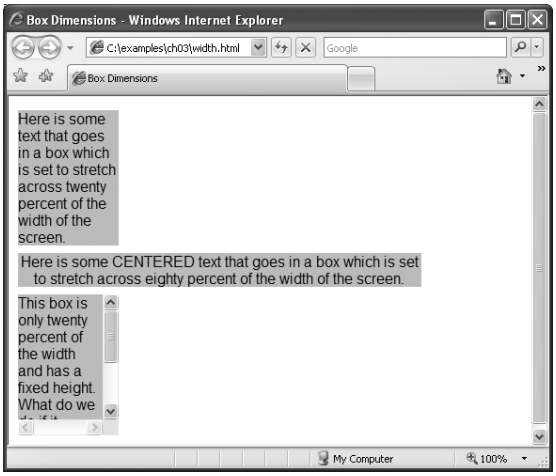


Figure 9: Element dimensions and text alignment.

The inline style in line 22 illustrates how to set the width of an element on screen; here, we indicate that the div element should occupy 20% of the screen width. Most elements are left-aligned by default; however, this alignment can be altered to position the element elsewhere. The height of an element can be set similarly, using the height property. The height and width values also can be specified as relative or absolute lengths. For example

```
width: 10em
```

sets the element's width to be equal to 10 times the font size. Line 27 sets text in the element to be center aligned; other values for the text-align property include left and right. One problem with setting both dimensions of an element is that the content inside the element can exceed the set boundaries, in which case the element is simply made large enough for all the content to fit. However, in line 32, we set the overflow property to scroll, a setting that adds scrollbars if the text overflows the boundaries.

◑ Box Model and Text Flow

All block-level XHTML elements have a virtual box drawn around them based on what is known as the box model. When the browser renders elements using the box model, the content of each element is surrounded by padding, a border and a margin (Fig. 10).

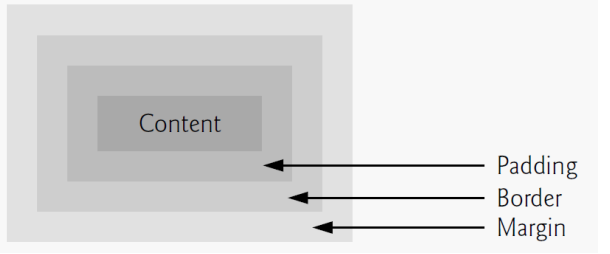


Figure 10. Box model for block-level elements.

CSS controls the border using three properties: border-width, border-color and border-style. We illustrate these three properties in Fig. 11.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5<!-- Fig. 3.11: borders.html -->
6 <!-- Borders of block-level elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Borders</title>
10 <style type = "text/css">
11 div { text-align: center;
12 width: 50%;
13 position: relative;
14 left: 25%;
15 border-width: 4px }
16 .medium { border-width: medium }
17 .thin { border-width: thin }
18 .solid { border-style: solid }
19 .double { border-style: double }
20 .groove { border-style: groove }
21 .inset { border-style: inset }
22 .outset { border-style: outset }
23 .dashed { border-style: dashed }
```

```
24 .red { border-color: red }
25 .blue { border-color: blue }
26 </style>
27 </head>
28 <body>
29 <div class = “solid”>Solid border</div><hr />
30 <div class = “double”>Double border</div><hr />
31 <div class = “groove”>Groove border</div><hr />
32 <div class = “inset”>Inset border</div><hr />
33 <div class = “dashed”>Dashed border</div><hr />
34 <div class = “thin red solid”>Thin Red Solid border</div><hr />
35 <div class = “medium blue outset”>Medium Blue Outset border</div
36 </body>
37 </html>
```

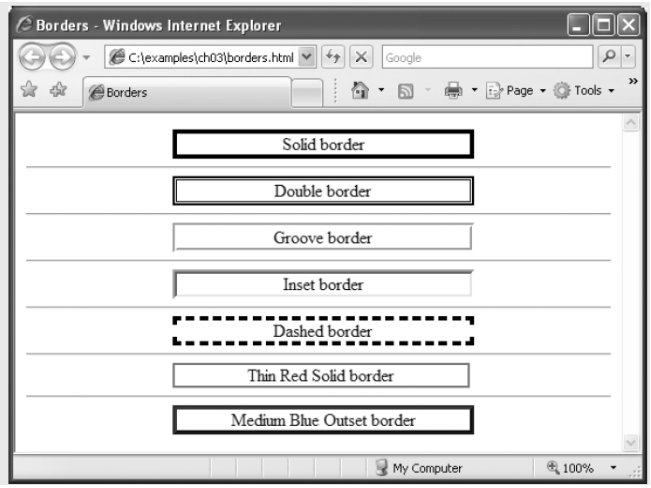


Figure 11. Borders of block-level elements.

Property border-width may be set to any valid CSS length (e.g., em, ex, px, etc.) or to the predefined value of thin, medium or thick. The border-color property sets the color. [Note: This property has different meanings for different style borders.] The border-style options are none, hidden, dotted, dashed, solid, double, groove, ridge, inset and outset. Borders groove and ridge have opposite effects, as do inset and outset. When border-style is set to none, no border is

rendered. Each border property may be set for an individual side of the box (e.g., border-topstyle or border-left-color). Note that we assign more than one class to an XHTML element by separating multiple class names with spaces, as shown in lines 36–37.

As we have seen with absolute positioning, it is possible to remove elements from the normal flow of text. Floating allows you to move an element to one side of the screen; other content in the document then flows around the floated element. Figure 12 demonstrates how floats and the box model can be used to control the layout of an entire page.

```
1 <?xml version = "1.0" encoding = "utf-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5<!-- Fig. 3.12: floating.html -->
6 <!-- Floating elements. -->
7 <html xmlns = "http://www.w3.org/1999/xhtml">
8 <head>
9 <title>Flowing Text Around Floating Elements</title>
10 <style type = "text/css">
11 div.heading { background-color: #bbddff;
12 text-align: center;
13 font-family: arial, helvetica, sans-serif;
14 padding: .2em }
15 p { text-align: justify;
16 font-family: verdana, geneva, sans-serif;
17 margin: .5em }
18 div.floated { background-color: #eeeeee;
19 font-size: 1.5em;
20 font-family: arial, helvetica, sans-serif;
21 padding: .2em;
22 margin-left: .5em;
23 margin-bottom: .5em;
24 float: right;
25 text-align: right;
26 width: 50% }
```

```
27 div.section { border: 1px solid #bbddff }
28 </style>
29 </head>
30 <body>
31 <div class = “heading”><img src = “deitel.png” alt = “Deitel” />
32 </div>
33 <div class = “section”>
34 <div class = “floated”>Corporate Training and Publishing</div>
35 <p>Deitel &amp;amp; Associates, Inc. is an internationally
36 recognized corporate training and publishing organization
37 specializing in programming languages, Internet/World
38 Wide Web technology and object technology education.
39 The company provides courses on Java, C++, Visual Basic, C#,
40 C, Internet and web programming, Object
41 Technology, and more.</p>
42 </div>
43 <div class = “section”>
44 <div class = “floated”>Leading-Edge Programming Textbooks</div>
45 <p>Through its publishing
46 partnership with Prentice Hall, Deitel &amp;amp; Associates,
47 Inc. publishes leading-edge programming textbooks,
48 professional books, interactive CD-ROM-based multimedia
49 Cyber Classrooms, satellite courses and DVD and web-based
50 video courses.</p>
51 </div>
52 </body>
53 </html>
```

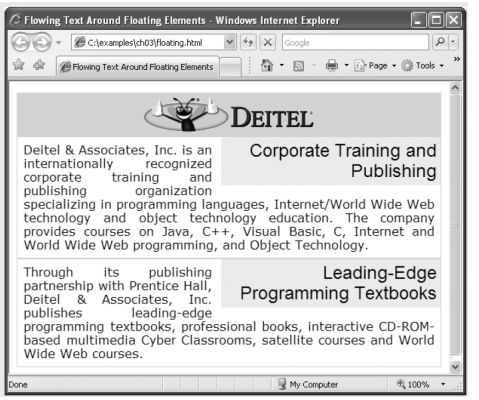


Figure 12. Floating elements.

Looking at the XHTML code, we can see that the general structure of this document consists of a header and two main sections. Each section contains a subheading and a paragraph of text. Block-level elements (such as divs) render with a line break before and after their content, so the header and two sections will render vertically one on top of another. In the absence of our styles, the subheading divs would also stack vertically on top of the text in the p tags. However, in line 24 we set the float property to right in the class floated, which is applied to the subheadings. This causes each subheading div to float to the right edge of its containing element, while the paragraph of text will flow around it.

Line 17 assigns a margin of .5em to all paragraph tags. The margin property sets the space between the outside of the border and all other content on the page. In line 21, we assign .2em of padding to the floated divs. The padding property determines the distance between the content inside an element and the inside of the element's border. Margins for individual sides of an element can be specified (lines 22–23) by using the properties margin-top, margin-right, margin-left and margin-bottom. Padding can be specified in the same way, using padding-top, padding-right, padding-left and padding-bottom. To see the effects of margins and padding, try putting the margin and padding properties inside comments and observing the difference. In line 27, we assign a border to the section boxes using a shorthand declaration of the border properties. CSS allows shorthand assignments of borders to allow you to define all three border properties in one line. The syntax for this shorthand is border: Our border is one pixel thick, solid, and the same color as the background-color property of the heading div (line 11). This allows the border to blend with the header and makes the page appear as one box with a line dividing its sections.

LEARNING ACTIVITY: IMPLEMENTING CSS FORMATTING

The cascading nature of CSS files is attributed to the fact that style information for a webpage can be defined in any of three different places, also known as style levels.

The preferred practice is to put style information in a separate file with a .css extension. Using formatting information contained within an external cascading style sheet is accomplished via the HTML link tag.

Read more details on:

https://www.theserverside.com/definition/cascading-style-sheet-CSS

What is the use of linking external style sheets in CSS?

Define your Direction: On a separate piece of paper, define the outlets of the different types of the terms according to function of cascading style sheets:

Terms	*Outlets*
Cell padding	
Embedded style sheet	
Simple mail transport protocol	
Grouping element	
Block-level elements	

ROLE MODEL



HÅKON WIUM LIE: *HE IS BEST KNOWN FOR DEVELOPING CASCADING STYLE SHEETS (CSS) WHILE WORKING WITH TIM BERNERS-LEE AND ROBERT CAILLIAU AT CERN IN 1994.*

BIOGRAPHY

Håkon Wium Lie (born 1965 in Halden) is a Norwegian web pioneer, a standards activist, a founding member of the Pirate Party of Norway, and the former Chief Technology Officer of Opera Software. He is also known as one of the financial founders of the Norwegian a far-right news, opinion and commentary website Resett.

He is best known for proposing the concept of Cascading Style Sheets (CSS) while working with Tim Berners-Lee and Robert Cailliau at CERN in 1994.

Education and career

Håkon Wium Lie attended Østfold University College, West Georgia College, and MIT Media Lab, receiving an MS in Visual Studies in 1991. On February 17, 2006, he successfully defended his PhD thesis at the University of Oslo.

His PhD thesis is background to the origins of CSS and a rationale to some of the design decisions behind it – particularly as to why some features were *not* included and why CSS avoids trying to become DSSSL.

While working with Tim Berners-Lee and Robert Cailliau at CERN in 1994, he proposed the concept of Cascading Style Sheets (CSS). As an employee at W3C, he developed CSS into a W3C Recommendation with Bert Bos. As a showcase and testbed, he integrated CSS into the Arena web browser. CSS is one of the fundamental

web standards, with profound impact on typography, aesthetics, and accessibility on the web.

Along with his work on the CSS specifications, Wium Lie has been an activist for standards in general. He challenged Microsoft to improve their support for standards in Internet Explorer and proposed the Acid2 test which was later developed and published by the Web Standards Project. He has argued against the use of formatting objects and CSS Regions on the web.

In 2006, Wium Lie started campaigning for browsers to support downloadable web fonts using common font formats. As of 2011, all major browser vendors have implemented web fonts this way. In 2007, Wium Lie started campaigning for the video element to make it easier to publish video on the web.

Wium Lie has also promoted the concept of printing from the web. His book on CSS, co-authored with Bert Bos, was produced from HTML and CSS files. These files were then converted to PDF by the Prince XML + CSS formatter. In 2005, he joined the board of YesLogic, the company that makes the Prince formatter.

Building on his experience with web printing, in 2011 Wium Lie proposed to extend CSS to support pagination on screens.

He has worked for, among others, the W3C, INRIA, CERN, MIT Media Lab, and Norwegian telecom research in Televerket.

- In December 1996, he became known as the CSS1 W3C Recommendation editor.
- In April 1999, he joined Opera Software.
- In 2014, he proposed to use the two unused Norwegian TLDs (.sj and .bv) to create privacy-enhanced zones.
- In 2015, he was the CTO of the Kon-Tiki2 Expedition and sailed from Lima to Easter Island on a balsawood raft

In 2018, Wium Lie was sued by Lovdata for publishing Norwegian court decisions on rettspraksis.no, a volunteer web site. In less than 24 hours, the web site was closed by the Oslo court, and Wium Lie was sentenced to pay the legal bills of Lovdata. Under Norwegian law, court decisions are exempted from copyright but Wium Lie was not allowed to appear in court. In the wake of this decision, members of the Norwegian Parliament has asked for changes in how Lovdata is organized.

Personal life

Wium Lie lives in Oslo, Norway. There, he has started web-based campaigns against high-rise buildings and advertising in the public space. Wium Lie also maintains a woodworking studio and runs an organic farm.

Awards and recognitions

In 1999, he was named to the MIT Technology Review TR100 as one of the top 100 innovators in the world under the age of 35.

In 2017, Wium Lie held a keynote at the WeAreDevelopers Conference 2017, talking about his contributions to the web today with the creation of CSS, and how it has evolved together with the web itself up to its current state.

KNOWLEDGE CHECK

1. **Explain the advantages of CCS.**
2. **Discuss on external style sheets.**
3. **Write a CSS rule that removes the underlines from all links inside list items (li) and shifts all list items left by 3 ems.**
4. **Write a CSS rule that places a background image halfway down the page, tiling it horizontally. The image should remain in place when the user scrolls up or down.**
5. **Write a CSS rule that changes the color of all elements containing attribute class = "greenMove" to green and shifts them down 25 pixels and right 15 pixels.**
6. **The CSS links properties are**
   a) :link, :visited, :hover, :active
   b) :link, :visit, :hover, :active
   c) :link, :visited, :over, :active
   d) :link, :visited, :hover, :active, :inactive
7. **Which of the following are the background properties in CSS?**

i) background-color ii) background-image iii) background-repeat

iv) background-position v) background

   a) i, ii, iii and iv only
   b) i, ii, iii and v only
   c) i, ii, iv and v only
   d) All i, ii, iii, iv and v

8. **The different ways to associate styles with a HTML document is/are**
   a) Embedded CSS with <style> element
   b) Inline CSS with style attribute.
   c) External CSS with <link> element.
   d) All of the above

9. Which of the following is correct CSS syntex for using font property?

a) .....

b) .....

c) .....

d) None of the above

10. In CSS tables, the possible values for the caption-side property can have the following values.

a) top, bottom, left or right

b) top, bottom, center, left or right

c) top or bottom

d) left or right

11. Which of the following selector selects the elements that are the default among a set of similar elements?

a) :default

b) :%

c) :disabled

d) none of the mentioned

12. Which of the following selector selects an element that has no children?

a) :empty

b) :nochild

c) :inheritance

d) :no-child

13. Which of the following selector selects the elements that are currently enabled?

a) :element

b) :empty

c) :enabled

d) none of the mentioned

14. Which of the following selector selects the element that is the first child of its parent that is of its type?

a) :first-of-type

b) :last-child

c) ::first-line

d) ::first-letter

15. Which of the following selector selects elements that do not match the selector s?

a) :!(s)

b) :nth-child(s)

c) :not(s)

d) none of the mentioned

16. Which of the following selector selects an element if it's the only child of its parent?

a) :root

b) :nth-oftype(n)

c) :only-child

d) none of the mentioned

17. Which of the following selector selects the element that is the target of a referring URI?

a) :target

b) :selection

c) ::selection

d) :URI

18. Which of the following selector applies styles to elements that are valid per HTML5 validations set either with the pattern or type attributes?

a) :valid

b) :required

c) :optional

d) :invalid

19. Which of the following selector selects the elements that are checked?

a) E ~ F

b) ::after

c) :checked

d) none of the mentioned

20. Why we are using box model and text flow model in cascading style sheet?

REFERENCES

1. Lie, Håkon Wium; Bos, Bert (1999). Cascading Style Sheets, designing for the Web. Addison Wesley. ISBN 0-201-59625-3. Retrieved 23 June 2010.
2. Lou Montulli; Brendan Eich; Scott Furman; Donna Converse; Troy Chevalier (22 August 1996). "JavaScript-Based Style Sheets". W3C. Retrieved 23 June 2010.
3. Cederholm, Dan; Ethan Marcotte (2009). Handcrafted CSS: More Bulletproof Web Design. New Riders. p. 114. ISBN 978-0-321-64338-4. Retrieved 19 June 2010.
4. Antti, Hiljá. "OOCSS, ACSS, BEM, SMACSS: what are they? What should I use?". clubmate.fi. Hiljá. Retrieved 2 June 2015.
5. Nitot, Tristan (18 March 2002). "Incorrect MIME Type for CSS Files". Mozilla Developer Center. Mozilla. Archived from the original on 2011-05-20. Retrieved 20 June 2010.
6. Nielsen, Henrik Frystyk (7 June 2002). "Libwww Hackers". World Wide Web Consortium. Retrieved 6 June 2010.
7. Bos, Bert; et al. (7 December 2010). "9.3 Positioning schemes". Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. W3C. Retrieved 16 February 2011.

CHAPTER 5

JAVASCRIPT



INTRODUCTION

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles.

JavaScript is most commonly used as a client side scripting language. This means that JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.

JavaScript can be used in other contexts than a Web browser. Netscape created server-side JavaScript as a CGI-language that can do roughly the same as Perl or ASP. There is no reason why JavaScript couldn't be used to write real, complex programs. However, this site exclusively deals with the use of JavaScript in web browsers.

SCENARIO: WHY TO LEARN JAVASCRIPT

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

Javascript is everywhere, it comes installed on every modern web browser and so to learn Javascript you really do not need any special environment setup. For example Chrome, Mozilla Firefox , Safari and every browser you know as of today, supports Javascript.

What skills will you need?

In order to work effectively in JavaScript, you must know:

- Features and uses of JavaScript
- Functions in JavaScript

FEATURES AND USES OF JAVASCRIPT

JavaScript is a scripting programming language that is used to make desktop widgets, applications, dynamic websites, PDF documents, developing game, etc. All web browsers support JavaScript.

JavaScript is a programming language that was once only used for interaction between client and user, control the browser but now it is also used to develop game and creating desktop applications.

This scripting language was developed by Netscape in 1995. Scripting language is a lighter version of programming language that is used to make the HTML page more interactive. JavaScript can only run inside a web browser.

If you are making a carrier in web designing, you must learn JavaScript as it goes hand in hand with XML and PHP. Quiz, polls, etc. that is present in the website to attract user and interact with them and engage them for a long time, all works on JavaScript.

Every person has used the java script. All browsers have support for JavaScript as many websites need it to run. A pop-up to update JavaScript appears whenever there is a need of one.

JavaScript is very different from java language though many people confuse the two being same. Programmers write the instructions in English rather than in numeric. It is simple to write codes in JavaScript as much of the coding is already available plus there is no need of any additional software and the code can be simply written on Notepad.

JavaScript language is used to make a webpage more dynamic and can be used again and again on different webpages of a website just by adding tags. This language has First Class Function functionality in it.

First Class Functions support function passing as argument to other functions, returns them as a value of other functions, and stores them as a data structure. Desktop widgets and PDF documents are also created using JavaScript.

JavaScript is easy to learn in comparison to other programming languages. JavaScript makes a webpage more interactive and engage users to the webpage. Most of the websites today use it and hence all the web browsers come equipped with JavaScript. It is considered better than HTML in web designing because HTML is static. Webpage reacts to a user click, mouse movement, etc. A pop-up can appear, drop down menu can be shown, sound can be played, etc. using JavaScript. By client-side programming it means that the code is sent to a computer and a web browser reads the code and runs it. JavaScript can be turned off in a Web browser by the user.

Advantages of JavaScript

The merits of using JavaScript are:

- Less server interaction: You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.
- Immediate feedback to the visitors: They don't have to wait for a page reload to see if they have forgotten to enter something.
- Increased interactivity: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.
- Richer interfaces: You can use JavaScript to include such items as dragand-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features:

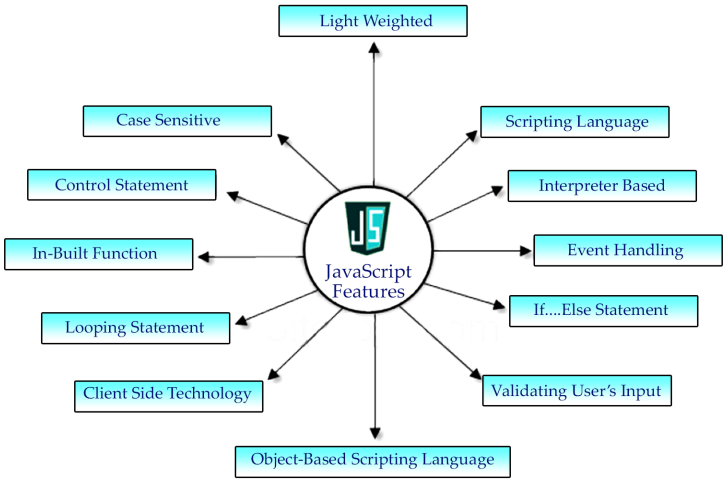
Programming language is a formal language, which comprises a set of instructions that produce various kinds of output.

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multithreading or multiprocessor capabilities.

Once again, JavaScript is a lightweight, interpreted **programming language** that allows you to build interactivity into otherwise static HTML pages.

◑ Features of JavaScript

JavaScript is a client side technology, it is mainly used for gives client side validation, but it have lot of features which are given below;



- JavaScript is an object-based scripting language.
- Giving the user more control over the browser.
- It Handling dates and time.
- It Detecting the user's browser and OS,
- It is light weighted.
- JavaScript is a scripting language and it is not java.
- JavaScript is interpreter based scripting language.
- JavaScript is case sensitive.
- JavaScript is object based language as it provides predefined objects.
- Every statement in javascript must be terminated with semicolon (;).
- Most of the javascript control statements syntax is same as syntax of control statements in C language.
- An important part of JavaScript is the ability to create new functions within scripts. Declare a function in JavaScript using **function** keyword.

◑ JavaScript Syntax

JavaScript can be implemented using JavaScript statements that are placed within the <script>... </script> HTML tags in a web page. You can place the <script> tags, containing your JavaScript, anywhere within you web page, but it is normally recommended that you should keep it within the <head> tags. The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>
 JavaScript code
</script>
```

The script tag takes two important attributes:

- Language: This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- Type: This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

So your JavaScript syntax will look as follows.

```
<script language="javascript" type="text/javascript">
 JavaScript code
</script>
```

First JavaScript Code

Let us take a sample example to print out "Hello World". We added an optional HTML comment that surrounds our JavaScript code. This is to save our code from a browser that does not support JavaScript. The comment ends with a "//-- >". Here "//" signifies a comment in JavaScript, so we add that to prevent a browser from reading the end of the HTML comment as a piece of JavaScript code. Next, we call a function document.write which writes a string into our HTML document.

This function can be used to write text, HTML, or both. Take a look at the following code.

```
<html>
<body>
<script language="javascript" type="text/javascript">
<!--
 document.write ("Hello World!")
//-->
</script>
</body>
</html>
```

This code will produce the following result:

Hello World!

JavaScript Data types

One of the most fundamental characteristics of a programming language is the set of data types it supports. These are the type of values that can be represented and manipulated in a programming language.

JavaScript allows you to work with three primitive data types:

- Numbers, e.g., 123, 120.50 etc.
- Strings of text, e.g. "This text string" etc.
- Boolean, e.g. true or false.



REMEMBER

JavaScript defines two trivial data types, null and undefined, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as object.

JavaScript Variables

Like many other programming languages, JavaScript has variables. Variables can be thought of as named containers. You can place data into these containers and then refer to the data simply by naming the container. Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the var keyword as follows.

```
<script type="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

You can also declare multiple variables with the same var keyword as follows:

```
<script type="text/javascript">
<!--
var money, name;
//-->
```

```
</script>
```

Storing a value in a variable is called variable initialization. You can do variable initialization at the time of variable creation or at a later point in time when you need that variable. For instance, you might create a variable named money and assign the value 2000.50 to it later. For another variable, you can assign a value at the time of initialization as follows.

```
<script type="text/javascript">
<!--
var name = "Ali";
var money;
money = 2000.50;
//-->
</script>
```

JavaScript is untyped language. This means that a JavaScript variable can hold a value of any data type. Unlike many other languages, you don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

JavaScript Variable Names

While naming your variables in JavaScript, keep the following rules in mind.

- You should not use any of the JavaScript reserved keywords as a variable name. For example, break or boolean variable names are not valid.
- JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, 123test is an invalid variable name but _123test is a valid one.
- JavaScript variable names are case-sensitive. For example, Name and name are two different variables.

JavaScript Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

◑ Operators

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript supports the following types of operators.

JavaScript has the following types of operators.

- Assignment operators
- Comparison operators
- Arithmetic operators
- Bitwise operators
- Logical operators
- String operators
- Conditional (ternary) operator
- Comma operator
- Relational operator

JavaScript has both binary and *unary* operators, and one special **ternary operator**, the conditional operator. A binary operator requires two operands, one before the operator and one after the operator:

operand1 operator operand2

For example, 3+4 or x*y.

Ternary operator is an operator that exists in some programming languages, which takes three operands rather than the typical one or two that most operators use.

A unary operator requires a single operand, either before or after the operator:

operator operand

or

operand operator

For example, x++ or ++x.

Assignment operators

An assignment operator assigns a value to its left operand based on the value of its right operand. The simple assignment operator is equal (=), which assigns the value of y to x.

There are also compound assignment operators that are shorthand for the operations listed in the following table:

Compound assignment operators		
Name	**Shorthand operator**	**Meaning**
Assignment	x = y	x = y
Addition assignment	x += y	x = x + y
Subtraction assignment	x -= y	x = x - y
Multiplication assignment	x *= y	x = x * y
Division assignment	x /= y	x = x / y
Remainder assignment	x %= y	x = x % y
Exponentiation assignment	x **= y	x = x ** y
Left shift assignment	x <<= y	x = x << y
Right shift assignment	x >>= y	x = x >> y
Unsigned right shift assignment	x >>>= y	x = x >>> y
Bitwise AND assignment	x &= y	x = x & y
Bitwise XOR assignment	x ^= y	x = x ^ y
Bitwise OR assignment	x |= y	x = x | y

Destructuring

The destructuring assignment syntax is a JavaScript expression that makes it possible to extract data from arrays or objects using a syntax that mirrors the construction of array and object literals.

```
var foo = [“one”, “two”, “three”];

// without destructuring
var one   = foo[0];
var two   = foo[1];
var three = foo[2];

// with destructuring
var [one, two, three] = foo;
```

Comparison Operators

A comparison operator compares its operands and returns a logical value based on whether the comparison is true. The operands can be numerical, **string**, logical, or object values. Strings are compared based on standard lexicographical ordering, using Unicode values. In most cases, if the two operands are not of the same type, JavaScript attempts to convert them to an appropriate type for the comparison. This behavior generally results in comparing the operands numerically. The sole exceptions to type conversion within comparisons involve the === and !== operators, which perform strict equality and inequality comparisons. These operators do not attempt to convert the operands to compatible types before checking equality. The following table describes the comparison operators in terms of this sample code:

String is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers.

```
var var1 = 3;

var var2 = 4;
```

Comparison operators		
Operator	**Description**	**Examples returning true**
Equal (==)	Returns true if the operands are equal.	3 == var1 "3" == var1 3 == '3'
Not equal(!=)	Returns true if the operands are not equal.	var1 != 4 var2 != "3"
Strict equal(===)	Returns true if the operands are equal and of the same type. See also Object. is and sameness in JS.	3 === var1
Strict not equal (!==)	Returns true if the operands are of the same type but not equal, or are of different type.	var1 !== "3" 3 !== '3'
Greater than(>)	Returns true if the left operand is greater than the right operand.	var2 > var1 "12" > 2
Greater than or equal (>=)	Returns true if the left operand is greater than or equal to the right operand.	var2 >= var1 var1 >= 3
Less than (<)	Returns true if the left operand is less than the right operand.	var1 < var2 "2" < 12
Less than or equal (<=)	Returns true if the left operand is less than or equal to the right operand.	var1 <= var2 var2 <= 5

◑ Arithmetic operators

An arithmetic operator takes numerical values (either literals or variables) as their operands and returns a single numerical value. The standard arithmetic operators are addition (+), subtraction (-), multiplication (*), and division (/). These operators work as they do in most other programming languages when used with floating point numbers (in particular, note that division by zero produces Infinity). For example:

```
1 / 2; // 0.5
1 / 2 == 1.0 / 2.0; // this is true
```

In addition to the standard arithmetic operations (+, -, * /), JavaScript provides the arithmetic operators listed in the following table:

Arithmetic operators		
Operator	**Description**	**Example**
Remainder (%)	Binary operator. Returns the integer remainder of dividing the two operands.	12 % 5 returns 2.
Increment (++)	Unary operator. Adds one to its operand. If used as a prefix operator (++x), returns the value of its operand after adding one; if used as a postfix operator (x++), returns the value of its operand before adding one.	If x is 3, then ++xsets x to 4 and returns 4, whereas x++ returns 3 and, only then, sets x to 4.
Decrement (--)	Unary operator. Subtracts one from its operand. The return value is analogous to that for the increment operator.	If x is 3, then --xsets x to 2 and returns 2, whereas x-- returns 3 and, only then, sets x to 2.
Unary negation (-)	Unary operator. Returns the negation of its operand.	If x is 3, then -xreturns -3.
Unary plus (+)	Unary operator. Attempts to convert the operand to a number, if it is not already.	+"3" returns 3. +true returns 1.
Exponentiation operator (**)	Calculates the base to the exponent power, that is, base^{exponent}	2 ** 3 returns 8. 10 ** -1 returns 0.1.

◑ Bitwise Operators

A bitwise operator treats their operands as a set of 32 bits (zeros and ones), rather than as decimal, hexadecimal, or octal numbers. For example, the decimal number nine has a binary representation of 1001. Bitwise operators perform their operations on such binary representations, but they return standard JavaScript numerical values.

The following table summarizes JavaScript's bitwise operators.

Bitwise operators		
Operator	**Usage**	**Description**
Bitwise AND	a & b	Returns a one in each bit position for which the corresponding bits of both operands are ones.

Bitwise OR	a | b	Returns a zero in each bit position for which the corresponding bits of both operands are zeros.
Bitwise XOR	a ^ b	Returns a zero in each bit position for which the corresponding bits are the same. [Returns a one in each bit position for which the corresponding bits are different.]
Bitwise NOT	~ a	Inverts the bits of its operand.
Left shift	a << b	Shifts a in binary representation b bits to the left, shifting in zeros from the right.
Sign-propagating right shift	a >> b	Shifts a in binary representation b bits to the right, discarding bits shifted off.
Zero-fill right shift	a >>> b	Shifts a in binary representation b bits to the right, discarding bits shifted off, and shifting in zeros from the left.

Bitwise Logical Operators

Conceptually, the bitwise logical operators work as follows:

- The operands are converted to thirty-two-bit integers and expressed by a series of bits (zeros and ones). Numbers with more than 32 bits get their most significant bits discarded. For example, the following integer with more than 32 bits will be converted to a 32 bit integer:

Before: 11100110111110100000000000000110000000000001

After: 10100000000000000110000000000001

- Each bit in the first operand is paired with the corresponding bit in the second operand: first bit to first bit, second bit to second bit, and so on.
- The operator is applied to each pair of bits, and the result is constructed bitwise.

For example, the binary representation of nine is 1001, and the binary representation of fifteen is 1111. So, when the bitwise operators are applied to these values, the results are as follows:

Bitwise operator examples

Expression	Result	Binary Description
15 & 9	9	1111 & 1001 = 1001
15 | 9	15	1111 | 1001 = 1111
15 ^ 9	6	1111 ^ 1001 = 0110

~15	-16	~00000000...00001111 = 11111111...11110000
~9	-10	~00000000...00001001 = 11111111...11110110

Bitwise Shift Operators

The bitwise shift operators take two operands: the first is a quantity to be shifted, and the second specifies the number of bit positions by which the first operand is to be shifted. The direction of the shift operation is controlled by the operator used.

Shift operators convert their operands to thirty-two-bit integers and return a result of the same type as the left operand.

The shift operators are listed in the following table.

Bitwise shift operators		
Operator	**Description**	**Example**
Left shift (<<)	This operator shifts the first operand the specified number of bits to the left. Excess bits shifted off to the left are discarded. Zero bits are shifted in from the right.	9<<2 yields 36, because 1001 shifted 2 bits to the left becomes 100100, which is 36.
Sign-propagating right shift(>>)	This operator shifts the first operand the specified number of bits to the right. Excess bits shifted off to the right are discarded. Copies of the leftmost bit are shifted in from the left.	9>>2 yields 2, because 1001 shifted 2 bits to the right becomes 10, which is 2. Likewise, -9>>2 yields -3, because the sign is preserved.
Zero-fill right shift(>>>)	This operator shifts the first operand the specified number of bits to the right. Excess bits shifted off to the right are discarded. Zero bits are shifted in from the left.	19>>>2 yields 4, because 10011 shifted 2 bits to the right becomes 100, which is 4. For non-negative numbers, zero-fill right shift and sign-propagating right shift yield the same result.

Logical Operators

Logical operators are typically used with Boolean (logical) values; when they are, they return a Boolean value. However, the && and || operators actually return the value of one of the specified operands, so if these operators are used with

non-Boolean values, they may return a non-Boolean value. The logical operators are described in the following table.

Logical Operators		
Operator	**Usage**	**Description**
Logical AND(&&)	expr1 && expr2	Returns expr1 if it can be converted to false; otherwise, returns expr2. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false.
Logical OR (||)	expr1 || expr2	Returns expr1 if it can be converted to true; otherwise, returns expr2. Thus, when used with Boolean values, || returns true if either operand is true; if both are false, returns false.
Logical NOT (!)	!expr	Returns false if its single operand can be converted to true; otherwise, returns true.

Examples of expressions that can be converted to false are those that evaluate to null, 0, NaN, the empty string (“”), or undefined.

The following code shows examples of the && (logical AND) operator.

```
var a1 =  true && true;      // t && t returns true
var a2 =  true && false;     // t && f returns false
var a3 = false && true;      // f && t returns false
var a4 = false && (3 == 4); // f && f returns false
var a5 = “Cat” && “Dog”;     // t && t returns Dog
var a6 = false && “Cat”;     // f && t returns false
var a7 = “Cat” && false;     // t && f returns false
```

The following code shows examples of the || (logical OR) operator.

```
var o1 =  true || true;      // t || t returns true
var o2 = false || true;      // f || t returns true
var o3 =  true || false;     // t || f returns true
var o4 = false || (3 == 4); // f || f returns false
var o5 = “Cat” || “Dog”;     // t || t returns Cat
var o6 = false || “Cat”;     // f || t returns Cat
var o7 = “Cat” || false;     // t || f returns Cat
```

The following code shows examples of the ! (logical NOT) operator.

```
var n1 = !true;  // !t returns false
var n2 = !false; // !f returns true
var n3 = !"Cat"; // !t returns false
```

String Operators

In addition to the comparison operators, which can be used on string values, the concatenation operator (+) concatenates two string values together, returning another string that is the union of the two operand strings.

For example,

```
console.log("my " + "string"); // console logs the string "my string".
```

The shorthand assignment operator += can also be used to concatenate strings. For example,

```
var mystring = "alpha";
mystring += "bet"; // evaluates to "alphabet" and assigns this value to mystring.
```

Conditional (ternary) Operator

The conditional operator is the only JavaScript operator that takes three operands. The operator can have one of two values based on a condition. The syntax is:

```
condition ? val1 : val2
```

If condition is true, the operator has the value of val1. Otherwise it has the value of val2. You can use the conditional operator anywhere you would use a standard operator.

For example,

```
var status = (age >= 18) ? “adult” : “minor”;
```

This statement assigns the value “adult” to the variable status if age is eighteen or more. Otherwise, it assigns the value “minor” to status.

Comma Operator

The comma operator (,) simply evaluates both of its operands and returns the value of the last operand. This operator is primarily used inside a for loop, to allow multiple variables to be updated each time through the loop.

For example, if a is a 2-dimensional array with 10 elements on a side, the following code uses the comma operator to update two variables at once. The code prints the values of the diagonal elements in the array:

```
for (var i = 0, j = 9; i <= j; i++, j--)
  console.log(“a[“ + i + “][“ + j + “]= “ + a[i][j]);
```

Relational Operators

A relational operator compares its operands and returns a Boolean value based on whether the comparison is true.

in

The in operator returns true if the specified property is in the specified object. The syntax is:

```
propNameOrNumber in objectName
```

where propNameOrNumber is a string or numeric expression representing a property name or array index, and objectName is the name of an object.

The following examples show some uses of the in operator.

```
// Arrays
```

```
var trees = ["redwood", "bay", "cedar", "oak", "maple"];
0 in trees;          // returns true
3 in trees;          // returns true
6 in trees;          // returns false
"bay" in trees;      // returns false (you must specify the index number,
                     // not the value at that index)
"length" in trees; // returns true (length is an Array property)

// built-in objects
"PI" in Math;              // returns true
var myString = new String("coral");
"length" in myString;  // returns true

// Custom objects
var mycar = { make: "Honda", model: "Accord", year: 1998 };
"make" in mycar;  // returns true
"model" in mycar; // returns true
```

instanceof

The instanceof operator returns true if the specified object is of the specified object type. The syntax is:

```
objectName instanceof objectType
```

where objectName is the name of the object to compare to objectType, and objectType is an object type, such as Date or Array.

Use instanceof when you need to confirm the type of an object at runtime. For example, when catching exceptions, you can branch to different **exception-handling** code depending on the type of exception thrown.

Exception handling is the process of responding to the occurrence, during computation, of exceptions – anomalous or exceptional conditions requiring special processing – often disrupting the normal flow of program execution.

For example, the following code uses instanceof to determine whether theDay is a Dateobject. Because theDay is a Date object, the statements in the if statement execute.

```
var theDay = new Date(1995, 12, 17);
if (theDay instanceof Date) {
  // statements to execute
}
```

Operator Precedence

The *precedence* of operators determines the order they are applied when evaluating an expression. You can override operator precedence by using parentheses.

The following table describes the precedence of operators, from highest to lowest.

Operator precedence	
Operator type	**Individual operators**
member	. []
call / create instance	() new
negation/increment	! ~ - + ++ -- typeof void delete
multiply/divide	* / %
addition/subtraction	+ -
bitwise shift	<< >> >>>
relational	< <= > >= in instanceof
equality	== != === !==
bitwise-and	&
bitwise-xor	^
bitwise-or	|
logical-and	&&
logical-or	||
conditional	?:
assignment	= += -= *= /= %= <<= >>= >>>= &= ^= |=
comma	,

LEARNING ACTIVITY: JAVASCRIPT - PLACEMENT IN HTML FILE

There is a flexibility given to include JavaScript code anywhere in an HTML document. However the most preferred ways to include JavaScript in an HTML file are as follows –

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

Read more details on:

https://www.tutorialspoint.com/javascript/javascript_placement.htm

Elaborate the use of logical operators and relational operators in javascript programming.

Define your Direction: On a separate piece of paper, explain the importance of the different types of the terms according to the features and uses of javascript:

Terms	***Importance***
Web browsers	
Scripting language	
Data structure	
Client side technology	
Boolean values	

FUNCTIONS IN JAVASCRIPT

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. It helps programmers in writing modular codes. Functions allow a programmer to divide a big program into a number of small and manageable functions. Like any other advanced programming language, JavaScript also supports all the features necessary to write modular code using functions. We were using these functions again and again, but they had been written in core JavaScript only once.

We need to define it. The most common way to define a function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Syntax

The basic syntax is shown here.

```
<script type="text/javascript">
<!--
function functionname(parameter-list)
{
 statements
}
//-->
</script>
```

Example

Try the following example. It defines a function called sayHello that takes no parameters:

```
<script type="text/javascript">
<!--
function sayHello()
{
 alert("Hello there");
}
//-->
</script>
```

◑ Calling a Function

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following code.

```
<html>
<head>
<script type="text/javascript">
function sayHello()
{
 document.write ("Hello there!");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="sayHello()" value="Say Hello">
</form>
<p>Use different text in write method and then try...</p>
</body>
</html>
```

Output:

Click the following button to call the function



◑ Function Parameters

We have seen functions without parameters. But there is a facility to pass different parameters while calling a function. These passed parameters can be captured inside the function and any manipulation can be done over those parameters. A function can take multiple parameters separated by comma.

Example

The following example. We have modified our sayHello function here. Now it takes two parameters.

```
<html>
<head>
<script type="text/javascript">
function sayHello(name, age)
{
 document.write (name + " is " + age + " years old.");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output:

Click the following button to call the function



Use different parameters inside the function and then try...

◑ The Return Statement

A JavaScript function can have an optional return statement. This is required if you want to return a value from a function. This statement should be the last statement in a function. For example, you can pass two numbers in a function and then

you can expect the function to return their multiplication in your calling program.

Example

The following example. It defines a function that takes two parameters and concatenates them before returning the resultant in the calling program.

```
<html>
<head>
<script type="text/javascript">
function concatenate(first, last)
{
 var full;
 full = first + last;
 return full;
}
function secondFunction()
{
 var result;
 result = concatenate('Zara', 'Ali');
 document.write (result );
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output:

Click the following button to call the function



Use different parameters inside the function and then try...

◑ Nested Functions

Loop is a sequence of instructions that is continually repeated until a certain condition is reached.

Prior to JavaScript 1.2, function definition was allowed only in top level global code, but JavaScript 1.2 allows function definitions to be nested within other functions as well. Still there is a restriction that function definitions may not appear within **loops** or conditionals. These restrictions on function definitions apply only to function declarations with the function statement.

Example

The following example to learn how to implement nested functions.

```
<head>
<script type="text/javascript">
<!--
function hypotenuse(a, b) {
 function square(x) { return x*x; }

 return Math.sqrt(square(a) + square(b));
}
function secondFunction(){
 var result;
 result = hypotenuse(1,2);
 document.write ( result );
}
//-->
</script>
</head>
```

```
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output:

Click the following button to call the function



Use different parameters inside the function and then try...

◑ Function () Constructor

The function statement is not the only way to define a new function; you can define your function dynamically using Function() constructor along with the new operator.

Syntax

Following is the syntax to create a function using Function() constructor along with the new operator.

```
<script type="text/javascript">
<!--
var variablename = new Function(Arg1, Arg2..., "Function Body");
//-->
</script>
```

The Function() constructor expects any number of string arguments. The last argument is the body of the function – it can contain arbitrary JavaScript statements, separated from each other by semicolons.



REMEMBER

The Function() constructor is not passed any argument that specifies a name for the function it creates. The unnamed functions created with the Function() constructor are called anonymous functions.

Example

The following example.

```
<html>
<head>
<script type="text/javascript">
<!--
var func = new Function("x", "y", "return x*y;");
function secondFunction(){
 var result;
 result = func(10,20);
document.write ( result );
}
//-->
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output:

Click the following button to call the function

Call Function

Use different parameters inside the function and then try...

LEARNING ACTIVITY: JAVASCRIPT GLOBAL VARIABLE

A JavaScript global variable is declared outside the function or declared with window object. It can be accessed from any function.

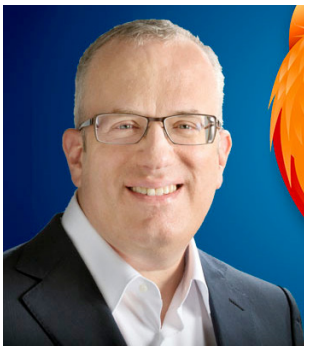
Read more details on:

https://www.javatpoint.com/javascript-global-variable

Write a simple javascript program using nested functions.

Define your Direction: On a separate piece of paper, explain the use of the different types of the terms according to the useful functions in JavaScript:

Terms	***Use***
Modular Codes	
Parameters	
Return Statement	
Function Declarations	
Data types	



ROLE MODEL

BRENDAN EICH: CREATOR OF THE JAVASCRIPT PROGRAMMING LANGUAGE

Brendan Eich is an American technologist and creator of the JavaScript programming language. He co-founded the Mozilla project,the Mozilla Foundation and the Mozilla Corporation, and served as the Mozilla Corporation's chief technical officer and briefly its chief executive officer.

Early Life

Brendan Eich received his bachelor's degree in mathematics and computer science at Santa Clara University.He received his master's degree in 1985 from the University of Illinois at Urbana-Champaign.

Eich started his career at Silicon Graphics, working for seven years on operating system and network code. He then worked for three years at MicroUnity Systems Engineering writing microkernel and DSP code, and doing the very first MIPS R4000 port of GCC.

Netscape and JavaScript

He started work at Netscape Communications Corporation in April 1995. Having originally joined intending to put Scheme "in the browser",Eich was instead commissioned to create a new language that resembled Java, JavaScript for the Netscape Navigator Web browser. The first version was completed in ten days in order to accommodate the Navigator 2.0 Beta release schedule,and was called Mocha, which was later renamed LiveScript in September 1995 and later JavaScript in the same month.Eich continued to oversee the development of SpiderMonkey, the specific implementation of JavaScript in Navigator, until 2011.

Mozilla

In early 1998, Eich co-founded the Mozilla project, with

a website at mozilla.org, that was meant to manage open-source contributions to the Netscape source code. He served as Mozilla's chief architect.AOL bought Netscape in 1999. After AOL shut down the Netscape browser unit in July 2003, Eich helped spin out the Mozilla Foundation.

In August 2005, after serving as Lead Technologist and as a member of the Board of Directors of the Mozilla Foundation, Eich became CTO of the newly founded Mozilla Corporation, meant to be the Mozilla Foundation's for-profit arm.

CEO appointment and resignation

On March 24, 2014, Eich was promoted to CEO of Mozilla Corporation.His appointment sparked controversy over a $1,000 political donation Eich had made in 2008 to the successful campaign for California Proposition 8, which sought to establish that, "Only marriage between a man and a woman is valid or recognized in California."This was criticized by gay rights activists on Twitter.In the ensuing public debate, OKCupid and two gay application developers called for a boycott of the company.Others at the Mozilla Corporation spoke out on their blogs in his favor.Board members wanted him to stay in the company with a different role.

On April 3, 2014, Eich stepped down as CEO and resigned from working at Mozilla. In his personal blog, Eich posted that "under the present circumstances, I cannot be an effective leader."

Following Eich's resignation, the National Organization for Marriage called for its own boycott of Mozilla, due to "gay activists who have forced him out of the company he has helped lead for years".

KNOWLEDGE CHECK

1. Illustrate the features of JavaScript.

2. Write the various data types which are used in JavaScript.

3. How to calling a function in JavaScript programming?

4. Define the function parameters.

5. What are nested functions?

6. Predict the output of the following JavaScript code.

```
<script type="text/javascript">

a = 8 + "8";

document.write(a);

</script>
```

a) 16

b) Complilation Error

c) 88

d) Run Time Error

7. Predict the output of the following JavaScript code.

```
<script type="text/javascript">

var a="GeeksforGeeks";

var x=a.lastIndexOf("G");

document.write(x);

</script>
```

a) 8

b) 0

c) 9

d) Error

8. Which of the following is not a reserved word in JavaScript?

a) interface

b) throws

c) program

d) short

9. Predict the output of the following JavaScript code.

```
<script type="text/javascript" language="javascript">

var a = "GeeksforGeeks";
var result = a.substring(4, 5);
document.write(result);

</script>
```

a) sf

b) ks

c) s

d) k

10. Predict the output of the following JavaScript code.

```
<script type="text/javascript" language="javascript">

var x=5;
var y=6;
var res=eval("x*y");
document.write(res);

</script>
```

a) "30"

b) 30

c) 5*6

d) "5*6"

11. Javascript is _____ language.

a) Programming

b) Applicatio

c) None of These

d) Scripting

12. JavaScript is _____ Side Scripting Language.

a) Server

b) ISP

c) None of These

d) Browser

13. JavaScript is designed for following purpose -

a) To Style HTML Pages

b) To add interactivity to HTML Pages.

c) To Perform Server Side Scripting Opertion

d) To Execute Query Related to DB on Server

14.....javaScript is also called client-side JavaScript.

a) Microsoft

b) Navigator

c) LiveWire

d) Native

15. What are variables used for in JavaScript Programs?

a) Storing numbers, dates, or other values

b) Varying randomly

c) Causing high-school algebra flashbacks

d) None of the above

16. What is the HTML tag under which one can write the JavaScript code?

a) <javascript>

b) <scripted>

c) <script>

d) <js>

17. Choose the correct JavaScript syntax to change the content of the following HTML code.

GeeksforGeeks

a) document.getElement("geek").innerHTML="I am a Geek";

b) document.getElementById("geek").innerHTML="I am a Geek";

c) document.getId("geek")="I am a Geek";

d) document.getElementById("geek").innerHTML=I am a Geek;

18. Which of the following is the correct syntax to display "GeeksforGeeks" in an alert box using JavaScript?

a) alertbox("GeeksforGeeks");

b) msg("GeeksforGeeks");

c) msgbox("GeeksforGeeks");

d) alert("GeeksforGeeks");

19. The external JavaScript file must contain <script> tag. True or False?

a) True

b) False

20. How many types of operators used in JavaScript prograraming?

REFERENCES

1. Adobe Systems Inc. Flash home page. http://www.adobe.com/products/flash/, 2010. Accessed on June 30, 2010.
2. Bhangal, Sham; Jankowski, Tomasz (2003). Foundation Web Design: Essential HTML, JavaScript, CSS, PhotoShop, Fireworks, and Flash. APress L. P.
3. Burns, Joe; Growney, Andree S. (2001). JavaScript Goodies. Pearson Education.
4. Dmitry Baranovskiy. Raphaël—JavaScript library. http://raphaeljs.com, 2010. Accessed on June 21, 2010.
5. Douglas Crockford. JavaScript: The good parts. O'Reilly Media, 2008.
6. Douglas Crockford. Javascript: The world's most misunderstood programming language. http://javascript.crockford.com/javascript.html, 2001. Accessed May 24, 2008.
7. Douglas Crockford. JavaScript: The world's most misunderstood programming language has become the world's most popular programming language. http://javascript.crockford.com/popular.html, 2008. Accessed May 24, 2008.
8. Douglas Crockford. JSLint: The JavaScript verifier.http://www.jslint.com/, 2002. Accessed August 4, 2008.
9. Duffy, Scott (2003). How to do Everything with JavaScript. Osborne.
10. Flanagan, David (2006). JavaScript: The Definitive Guide (5th ed.). O'Reilly & Associates.
11. Goodman, Danny; Markel, Scott (2003). JavaScript and DHTML Cookbook. O'Reilly & Associates.
12. Harris, Andy (2001). JavaScript Programming for the Absolute Beginner. Premier Press.
13. Haverbeke, Marijn (2011). Eloquent JavaScript. No Starch Press.
14. Kent Beck. Test-Driven Development: By Example. Addison-Wesley Professional, 2002.
15. Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform resource identifier (URI): Generic syntax. http://tools.ietf.org/html/rfc3986, January 2005. Accessed on July 7, 2010.

CHAPTER 6

PROGRAMING LANGUAGES



INTRODUCTION

A programming language is defined as "an artificial language designed to express computations that can be performed by a machine, particularly a computer" according to Wikipedia. While that may be an accurate description, it is not the only definition of a programming language. There are thousands upon thousands of programming languages in our ever advancing world and there are more being created every day. Programming languages range from the very basic binary language or "machine code" all the way up to the highest level languages such as Ruby, Python, and J with every shade in between. So why are there so many languages? Well there are many answers to that question. For one Computer Science is still an ever advancing field and people often find better ways to do things than those before them. Then there is the fact that some people prefer certain languages and some languages are better suited for the task at hand. However most modern programming languages do contain common elements and are Turing Complete.

SCENARIO: PROGRAMMING LANGUAGE GENERATIONS

Programming language generations are a way to classify computer programming languages according to their overall distance from the actual machine code that is generated and, in some ways, their ease of use by a programmer. There are three generally accepted programming language generations and a few whose definitions are not clearly agreed on. A first-generation computer programming language is straight machine code with no abstraction. Second-generation languages are abstracted machine code, such as assembly language, that are tied to a specific system architecture but are human readable and need to be compiled. Third-generation languages are the most common form, including standard languages such as C and Java®, while languages that are fourth-generation or up have varied definitions.

What skills will you need?

In order to work effectively with programming language, you must know:

- Php
- Asp.net
- Java basics

PHP

PHP stands for PHP: Hypertext Preprocessor, with that PHP standing for Personal HomePage [Tools]. This type of acronym is known as a retronym. Originally, in 1994, the language was designed as a small set of binaries used to collect some basic site traffic data. In 1997, the parser was rewritten by two Israelis and the name was changed to the current acronym — it being determined that hypertext preprocessor was a decidedly more acceptable name in the business world.

It is an open-source language, used primarily for dynamic web content and server-side applications. PHP is often pointed to as the main competitor with the following:

- Microsoft's C# — Visual Basic.NET — ASP family
- Sun's Java — JSP
- Macromedia's ColdFusion
- CGI — Perl

PHP has many open-source libraries included with the core build, and many more are readily available. Extensions exist to help it interface with a number of systems, including IRC, a number of compression formats, and Windows API. Other extensions exist to let PHP generate file formats on-the-fly, such as a popular extension that allows it to create Adobe Flash® movies.

Since version 3, PHP has integrated object oriented features. Version 5 built substantially on this limited functionality, and the language now has robust object oriented capabilities, including interfaces, exceptions, destructions, and abstracts.

PHP reached wide-spread popularity with version 4, released in 2000. In 2004, version 5 debuted, and it is now considered one the top languages used for server-side scripting.

No doubt much of its popularity is due to its relative ease to learn, and its notorious looseness. Arrays and variables in PHP are able to hold any type of object, variables need not be declared, and the syntax is remarkably simple.

Unlike many languages, such as C# or Perl, which have primarily a following of more generalist programmers, many PHP programmers know no other language. This occasionally causes it to be dismissed as a lesser language, but its growing popularity and the many robust and efficient sites built using it as a structure seem to dispel this myth.

PHP has occasionally been criticized for what are viewed by some as security flaws, in comparison to languages such as ASP. A lack of easily understandable error messages, a sometimes overly robust configuration file, and an obviously

incomplete set of built-in functions are also pointed to as areas which could use marked improvement.

Development continues apace, however, and with each successive build, the language appears to address more and more of the concerns raised by its open-source community.

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

◑ Characteristics of PHP

The PHP language revolves around the central theme of practicality. PHP is about providing the programmer with the necessary tools to get the job done in a quick and efficient fashion. Five important characteristics make PHP's practical nature possible:

- Simplicity
- Efficiency
- Familiarity
- Security
- Flexibility

Simplicity

A PHP script can consist of 10,000 lines or one line: whatever you need to get the job done. There is no need to include libraries, special compilation directives, or anything of the sort. The PHP engine simply begins executing the code after the first **escape sequence** (). If the code is syntactically correct, it will be executed exactly as it is displayed.

Efficiency

Efficiency is an extremely important consideration for working in a multiuser environment such as the WWW. PHP 4.0 introduced resource allocation mechanisms

and more pronounced support for object-oriented programming, in addition to session management features. Reference counting has also been introduced in the latest version, eliminating unnecessary memory allocation.

Escape sequence is a sequence of characters that does not represent itself when used inside a character or string literal, but is translated into another character or a sequence of characters that may be difficult or impossible to represent directly.

Familiarity

Programmers from many backgrounds will find themselves already accustomed to the PHP language. Many of the language's constructs are borrowed from C and Perl, and in many cases PHP code is almost indistinguishable from that found in the typical C or Pascal program. This minimizes the learning curve considerably.

Security

PHP provides developers and administrators with a flexible and efficient set of security safeguards. These safeguards can be divided into two frames of reference: system level and application level.

System-Level Security Safeguards

PHP furnishes a number of security mechanisms that administrators can manipulate, providing for the maximum amount of freedom and security when PHP is properly configured. PHP can be run in what is known as safe mode, which can limit users' attempts to exploit the PHP implementation in many important ways. Limits can also be placed on maximum execution time and memory usage, which if not controlled can have adverse affects on server performance. Much as with a cgi-bin folder, administrators can also place restrictions on the locations in which users can view and execute PHP scripts and use PHP scripts to view guarded server information, such as the passwd file.

Application-Level Security Safeguards

Several trusted data encryption options are supported in PHP's predefined function set. PHP is also compatible with many third-party applications, allowing for easy-integration with secure ecommerce technologies. Another

advantage is that the PHP source code is not viewable through the browser because the script is completely parsed before it is sent back to the requesting user. This benefit of PHP's server-side architecture prevents the loss of creative scripts to users at least knowledgeable enough to execute a 'View Source'. Because PHP is an embedded language, it is extremely flexible towards meeting the needs of the developer. Although PHP is generally touted as being used in conjunction solely with HTML, it can also be integrated alongside languages like JavaScript, WML, XML, and many others. Additionally, as with most other mainstream languages, wisely planned PHP applications can be easily expanded as needed. Browser dependency is not an issue because PHP scripts are compiled entirely on the server side before being sent to the user. In fact, PHP scripts can be sent to just about any kind of device containing a browser, including cell phones, **personal digital assistant (PDA)** devices, pagers, laptops, not to mention the traditional PC. People who want to develop shell-based applications can also execute PHP from the command line. Since PHP contains no server-specific code, users are not limited to a specific and perhaps unfamiliar Web server. Apache, Microsoft IIs, Netscape Enterprise Server, Stronghold, and Zeus are all fair game for PHP's server integration. Because of the various platforms that these servers operate on, PHP is largely platform independent, available for such platforms as UNIX, Solaris, FreeBSD, and Windows 95/98/NT. Finally, PHP offers access to external components, such as Enterprise Java Beans and Win32 COM objects. These newly added features put PHP in the big league, truly enabling developers to scale PHP projects upward and outward as need be.

Personal Digital Assistant (PDA) is a computer that fits in the palm of your hand to help collect such information as contacts, appointments, files, and programs.

Flexibility

The open source development strategy has gained considerable notoriety in the software industry. The prospect of releasing source code to the masses has resulted in undeniably positive outcomes for many projects, perhaps most notably Linux, although the success of the Apache project has certainly been a major contributor

in proving the validity of the open source ideal. The same holds true for the developmental history of PHP, as users worldwide have been a huge factor in the advancement of the PHP project.

"Hello World" Script in PHP

To get a feel for PHP, first start with simple PHP scripts. Since "Hello, World!" is an essential example, first we will create a friendly little "Hello, World!" script.

As mentioned earlier, PHP is embedded in HTML. That means that in amongst your normal HTML (or XHTML if you're cutting-edge) you'll have PHP statements like this

```
<html>

   <head>
      <title>Hello World</title>
   </head>

   <body>
      <?php echo "Hello, World!";?>
   </body>

</html>
```

It will produce following result

Hello, World!

If you examine the HTML output of the above example, you'll notice that the PHP code is not present in the file sent from the server to your Web browser. All of the PHP present in the Web page is processed and stripped from the page; the only thing returned to the client from the Web server is pure HTML output.

All PHP code must be included inside one of the three special markup tags ATE are recognized by the PHP Parser.

```
<?php PHP code goes here ?>

<?    PHP code goes here ?>

<script language="php"> PHP code goes here </script>
```

A most common tag is the <?php...?> and we will also use the same tag in our tutorial.

◑ Variables in PHP

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variables in PHP are identifiers prefixed with a dollar sign ($). For example:

```
 $name
$Age
$_debugging
$MAXIMUM_IMPACT
```

A variable may hold a value of any type. There is no compile- or runtime type checking on variables. You can replace a variable's value with another of a different type:

```
$what = "Fred";
$what = 35;
$what = array('Fred', '35', 'Wilma');
```

There is no explicit syntax for declaring variables in PHP. The first time the value of a variable is set, the variable is created. In other words, setting a variable functions as a declaration. For example, this is a valid complete PHP program:

```
 $day = 60 * 60 * 24;
echo "There are $day seconds in a day.\n";
```

There are 86400 seconds in a day.

A variable whose value has not been set behaves like the NULL value:

```
if ($uninitialized_variable === NULL) {
  echo "Yes!";
}
```

Yes



REMEMBER

Variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other.

Variable Variables

You can reference the value of a variable whose name is stored in another variable. For example:

```
$foo = 'bar';
$$foo = 'baz';
```

After the second statement executes, the variable $bar has the value "baz".

Variable References

In PHP, references are how you create variable aliases. To make $black an alias for the variable $white, use:

```
$black =& $white;
```

The old value of $black is lost. Instead, $black is now another name for the value that is stored in $white:

```
$big_long_variable_name = "PHP";
$short =& $big_long_variable_name;
$big_long_variable_name .= " rocks!";
print "\$short is $short\n";
print "Long is $big_long_variable_name\n";
$short is PHP rocks!
Long is PHP rocks!
$short = "Programming $short";
print "\$short is $short\n";
print "Long is $big_long_variable_name\n";
$short is Programming PHP rocks!
Long is Programming PHP rocks!
```

After the assignment, the two variables are alternate names for the same value. Unsetting a variable that is aliased does not affect other names for that variable's value, though:

```
$white = "snow";
$black =& $white;
unset($white);
print $black;
snow
Functions can return values by reference
function &ret_ref() {      // note the &
    $var = "PHP";
    return $var;
```

```
}

$v =& ret_ref();            // note the &
```

Variable Scope

The scope of a variable, which is controlled by the location of the variable's declaration, determines those parts of the program that can access it. There are four types of variable scope in PHP: local, global, static, and function parameters.

Local Scope

A variable declared in a function is local to that function. That is, it is visible only to code in that function (including nested function definitions); it is not accessible outside the function. In addition, by default, variables defined outside a function (called global variables) are not accessible inside the function. For example, here's a function that updates a local variable instead of a global variable:

```
 function update_counter ( ) {
   $counter++;
}
$counter = 10;
update_counter( );
echo $counter;
10
```

The $counter inside the function is local to that function, because we haven't said otherwise. The function increments its private $counter, whose value is thrown away when the subroutine ends. The global $counter remains set at 10.

Only functions can provide local scope. Unlike in other languages, in PHP you can't create a variable whose scope is a loop, conditional branch, or other type of block.

Global Scope

Variables declared outside a function are global. That is, they can be accessed from any part of the program. However, by default, they are not available inside functions. To allow a function to access a global variable, you can use the global keyword inside the function to declare the variable within the function. Here's how we can rewrite the update_counter( ) function to allow it to access the global $counter variable:

```
 function update_counter ( ) {
     global $counter;
     $counter++;
}
$counter = 10;
update_counter( );
echo $counter;
11
```

A more cumbersome way to update the global variable is to use PHP's $GLOBALS array instead of accessing the variable directly:

```
 function update_counter ( ) {
     $GLOBALS[counter]++;
}
$counter = 10;
update_counter( );
echo $counter;
11
```

Static Variables

A static variable retains its value between calls to a function but is visible only within that function. You declare a variable static with the static keyword. For example:

```
 function update_counter ( ) {
   static $counter = 0;
   $counter++;
   echo "Static counter is now $counter\n";
}
$counter = 10;
update_counter( );
update_counter( );
echo "Global counter is $counter\n";
Static counter is now 1
Static counter is now 2
```

Global counter is 10

Function Parameters

A function definition can have named parameters:

```
function greet ($name) {
  echo "Hello, $name\n";
}

greet("Janet");
```

Hello, Janet

Function parameters are local, meaning that they are available only inside their functions. In this case, $name is inaccessible from outside greet( ).

◑ PHP File Handling

File handling is an important part of any web application. You often need to open and process a file for different tasks.

PHP supports file handling which is used to read, write and append data in file.

- File Opening
- File Reading
- File Writing
- File Closing

File Opening in PHP

fopen() function is used to open a file in PHP. Its required two arguments, first the file name and then file opening mode.

The file may be opened in one of the following modes:	
r	Read only. Starts at the beginning of the file
r+	Read/Write. Starts at the beginning of the file
w	Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist
w+	Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist
a	Append. Opens and writes to the end of the file or creates a new file if it doesn't exist

a+	Read/Append. Preserves file content by writing to the end of the file
x	Write only. Creates a new file. Returns FALSE and an error if file already exists
x+	Read/Write. Creates a new file. Returns FALSE and an error if file already exists

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
?>
```

If an attempt to open a file fails then fopen returns a false value, otherwise it returns a file pointer which is used for further reading or writing of that file.

File Reading in PHP

Once a file is opened using fopen() function then it can be read by a function called fread(). This function requires two arguments. The file pointer and length in bytes of the file must be expressed.

The files size can be calculated using the filesize() function which takes the file name as its argument and returns the size of the file in bytes.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
```

```
  exit();
}

$fileSize = filesize( $fileName );
$fileData = fread( $fp, $fileSize );
?>
```

Reading a File Line by Line

The fgets() function is used to read a single line from a file and after a call to this function the file pointer has moved to the next line.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"r");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}

while(!feof($fp))
{
  echo fgets($fp). "<br>";
}
?>
```

File Writing in PHP

Using the PHP fwrite() function, a new file can be written or text can be appended to an existing file. This function requires two arguments, first specifying a file pointer and the string of data that is to be written. Optionally a third integer argument can be used to specify the length of the data to write. If the third argument is used, file writing would will stop after the specified length has been reached.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
```

```
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}
fwrite( $fp, "This is a sample text to write\n" );
?>
```

Closing a File in PHP

In PHP it is not system critical to close all your files after using them because, file will auto close after PHP code finishes execution.

You can close file, using fclose() function.

Example:

```
<?php
$fileName = "/doc/myFile.txt";
$fp = fopen($fileName,"w");
if( $fp == false )
{
  echo ( "Error in opening file" );
  exit();
}

//some code to be executed

fclose( $fp );
?>
```

The fclose() function requires a file pointer as its argument and then returns true when the closure succeeds or false if it fails.

LEARNING ACTIVITY: PROS AND CONS OF PHP

As with all programming languages, there are a number of pros and cons to the Hypertext Preprocessor (PHP) scripting language that affect how suitable it is for various applications. PHP advantages include that it's a free, open-source programming language, relatively easy to learn and flexible. PHP also supports a range of types of database, making it a good choice for applications that need to communicate with databases. Cons of PHP include that it's ineffective at producing desktop applications, relatively slow compared to some other languages and has poor support for handling errors.

Read more details on:

https://www.wisegeek.com/what-are-the-pros-and-cons-of-php.htm

How to closing a file in PHP using fclose() function?

Define your Direction: On a separate piece of paper, explain the position of the different types of the terms according to the php program:

Terms	*Position*
Server-Side Applications	
Open-source community	
Data encryption	
Identifiers	
Web application	

ASP.NET

ASP.Net is a web development platform provided by Microsoft. It is used for creating web-based applications. ASP.Net was first released in the year 2002.

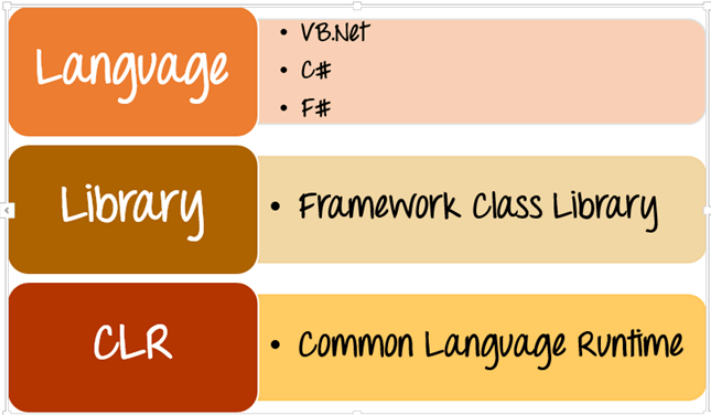
The first version of ASP.Net deployed was 1.0. The most recent version of ASP. Net is version 4.6. ASP.Net is designed to work with the HTTP protocol. This is the standard protocol used across all web applications.

ASP.Net applications can also be written in a variety of .Net languages. These include C#, VB.Net, and J#.

The full form of ASP is Active Server Pages, and .NET is Network Enabled Technologies.

◑ ASP.NET Architecture and its Components

ASP.Net is a framework which is used to develop a Web-based application. The basic architecture of the ASP.Net framework is as shown below.



The architecture of the.Net framework is based on the following key components

- **Language** – A variety of languages exists for .net framework. They are VB.net and C#. These can be used to develop web applications.
- **Library** - The .NET Framework includes a set of standard class libraries. The most common library used for web applications in .net is the Web library. The web library has all the necessary components used to develop. Net web-based applications.

- **Common Language Runtime** - The Common Language Infrastructure or CLI is a platform. .Net programs are executed on this platform. The CLR is used for performing key activities. Activities include Exception handling and Garbage collection.

Below are some of the key characteristics of the ASP.Net framework

- **Code Behind Mode** – This is the concept of separation of design and code. By making this separation, it becomes easier to maintain the ASP.Net application. The general file type of an ASP.Net file is aspx. Assume we have a web page called MyPage.aspx. There will be another file called MyPage.aspx.cs which would denote the code part of the page. So Visual Studio creates separate files for each web page, one for the design part and the other for the code.
- **State Management** – ASP.Net has the facility to control state management. HTTP is known as a stateless protocol. Let's take an example of a shopping cart application. Now, when a user decides what he wants to buy from the site, he will press the submit button.



REMEMBER

HTTP is a stateless protocol. When the user goes to the purchase page, HTTP will not store the information on the cart items. Additional coding needs to be done to ensure that the cart items can be carried forward to the purchase page.

The application needs to remember the items the user choose for the purchase. This is known as remembering the state of an application at a current point in time.

Such an implementation can become complex at times. But ASP.Net can do state management on your behalf. So ASP.Net can remember the cart items and pass it over to the purchase page.

- **Caching** – ASP.Net can implement the concept of Caching. This improve's the performance of the application. By caching those pages which are often requested by the user can be stored in a temporary location. These pages can be retrieved faster and better responses can be sent to the user. So caching can significantly improve the performance of an application.

ASP.Net is a development language used for constructing web-based applications. ASP.Net is designed to work with the standard HTTP protocol.

◑ ASP.NET Page Life Cycle

When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application. In other words, you can write your own code to override the default code.

The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives, are part of this control tree. You can see the control tree by adding trace= "true" to the page directive. We will cover page directives and tracing under 'directives' and 'event handling'.

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.

Following are the different stages of an ASP.NET page:

- **Page request** - When ASP.NET gets a page request, it decides whether to parse and compile the page, or there would be a cached version of the page; accordingly the response is sent.
- **Starting of page life cycle** - At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
- **Page initialization** - At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and the themes are applied. For a new request, postback data is loaded and the control properties are restored to the view-state values.
- **Page load** - At this stage, control properties are set using the view state and control state values.

- **Validation** - Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
- **Postback event handling** - If the request is a postback (old request), the related event handler is invoked.
- **Page rendering** - At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of page.
- **Unload** - The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done.

◑ ASP.NET Page Life Cycle Events

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.

Following are the page life cycle events:

- **PreInit** - PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.
- **Init** - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
- **InitComplete** - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- **LoadViewState** - LoadViewState event allows loading view state information into the controls.
- **LoadPostData** - During this phase, the contents of all the input fields are defined with the <form> tag are processed.
- **PreLoad** - PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.
- **Load** - The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.

- **LoadComplete** - The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler
- **PreRender** - The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
- **PreRenderComplete** - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
- **SaveStateComplete** - State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.
- **UnLoad** - The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as **database** connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

Database is an organized collection of data, generally stored and accessed electronically from a computer system.

ASP.NET - First Example

An ASP.NET page is made up of a number of server controls along with HTML controls, text, and images. Sensitive data from the page and the states of different controls on the page are stored in hidden fields that form the context of that page request.

ASP.NET runtime controls the association between a page instance and its state. An ASP.NET page is an object of the Page or inherited from it. All the controls on the pages are also objects of the related control class

inherited from a parent Control class. When a page is run, an instance of the object page is created along with all its content controls.

An ASP.NET page is also a server side file saved with the .aspx extension. It is modular in nature and can be divided into the following core sections:

- Page Directives
- Code Section
- Page Layout

Page Directives

The page directives set up the environment for the page to run. The @Page directive defines page-specific attributes used by ASP.NET page parser and compiler. Page directives specify how the page should be processed, and which assumptions need to be taken about the page.

It allows importing namespaces, loading assemblies, and registering new controls with custom tag names and namespace prefixes.

Code Section

The code section provides the handlers for the page and control events along with other functions required. We mentioned that, ASP.NET follows an object model. Now, these objects raise events when some events take place on the user interface, like a user clicks a button or moves the cursor. The kind of response these events need to reciprocate is coded in the event handler functions. The event handlers are nothing but functions bound to the controls.

The code section or the code behind file provides all these event handler routines, and other functions used by the developer. The page code could be precompiled and deployed in the form of a binary assembly.

Page Layout

The page layout provides the interface of the page. It contains the server controls, text, inline JavaScript, and HTML tags.

The following code snippet provides a sample ASP.NET page explaining Page directives, code section and page layout written in C#:

```
<!-- directives -->
<% @Page Language="C#" %>
<!-- code section -->
<script runat="server">
```

```
    private void convertoupper(object sender, EventArgs e)
    {
        string str = mytext.Value;
        changed_text.InnerHtml = str.ToUpper();
    }
</script>

<!-- Layout -->
<html>
    <head>
        <title> Change to Upper Case </title>
    </head>

    <body>
        <h3> Conversion to Upper Case </h3>

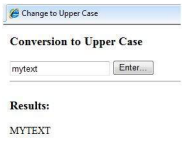
        <form runat="server">
            <input runat="server" id="mytext" type="text" />
                <input runat="server" id="button1" type="submit" value="Enter..."
OnServerClick="convertoupper"/>

            <hr />
            <h3> Results: </h3>
            <span runat="server" id="changed_text" />
        </form>

    </body>

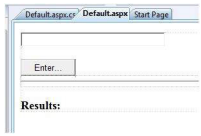
</html>
```

Copy this file to the web server root directory. Generally it is c:\iNETput\ wwwroot. Open the file from the browser to execute it and it generates following result:



Using Visual Studio IDE

Let us develop the same example using Visual Studio IDE. Instead of typing the code, you can just drag the controls into the design view:



The content file is automatically developed. All you need to add is the Button1_Click routine, which is as follows:

```
protected void Button1_Click(object sender, EventArgs e)
{
    string buf = TextBox1.Text;
    changed_text.InnerHtml = buf.ToUpper();
}
```

The content file code is as given:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="firstexample._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

    <head runat="server">
        <title>
            Untitled Page
        </title>
    </head>
```

```
<body>

    <form id="form1" runat="server">
        <div>

            <asp:TextBox ID="TextBox1" runat="server" style="width:224px">
            </asp:TextBox>

            <br />
            <br />

        <asp:Button ID="Button1" runat="server" Text="Enter..." style="width:85px"
onclick="Button1_Click" />
            <hr />

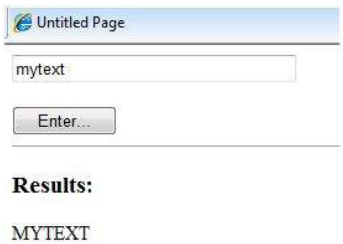
            <h3> Results: </h3>
            <span runat="server" id="changed_text" />

        </div>
    </form>

</body>

</html>
```

Execute the example by right clicking on the design view and choosing 'View in Browser' from the popup menu. This generates the following result:



◑ Personalization

Web sites are designed for repeated visits from the users. Personalization allows a site to remember the user identity and other information details, and it presents an individualistic environment to each user.

ASP.NET provides services for personalizing a web site to suit a particular client's taste and preference.

Understanding Profiles

ASP.NET personalization service is based on user profile. User profile defines the kind of information about the user that the site needs. For example, name, age, address, date of birth, and phone number.

This information is defined in the web.config file of the application and ASP.NET runtime reads and uses it. This job is done by the personalization providers.

The user profiles obtained from user data is stored in a default database created by ASP.NET. You can create your own database for storing profiles. The profile data definition is stored in the configuration file web.config.

Example. Let us create a sample site, where we want our application to remember user details like name, address, date of birth etc. Add the profile details in the web.config file within the <system.web> element.

```
<configuration>
<system.web>

<profile>
   <properties>
      <add name="Name" type ="String"/>
      <add name="Birthday" type ="System.DateTime"/>

      <group name="Address">
         <add name="Street"/>
         <add name="City"/>
         <add name="State"/>
         <add name="Zipcode"/>
      </group>

   </properties>
```

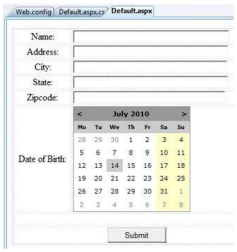
```
</profile>

</system.web>
</configuration>
```

When the profile is defined in the web.config file, the profile could be used through the Profile property found in the current **HttpContext** and also available via page.

HttpContext is an object that wraps all http related information into one place.

Add the text boxes to take the user input as defined in the profile and add a button for submitting the data:



Update Page_load to display profile information:

```
using System;
using System.Data;
using System.Configuration;

using System.Web;
using System.Web.Security;


using System.Web.UI;

using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
    {
```

```
        if (!this.IsPostBack)
        {
            ProfileCommon pc=this.Profile.GetProfile(Profile.UserName);

            if (pc != null)
            {
                this.txtname.Text = pc.Name;
                this.txtaddr.Text = pc.Address.Street;
                this.txtcity.Text = pc.Address.City;
                this.txtstate.Text = pc.Address.State;
                this.txtzip.Text = pc.Address.Zipcode;
                this.Calendar1.SelectedDate = pc.Birthday;
            }
        }
    }
}
```

Write the following handler for the Submit button, for saving the user data into the profile:

```
protected void btnsubmit_Click(object sender, EventArgs e)
{
    ProfileCommon pc=this.Profile.GetProfile(Profile.UserName);

    if (pc != null)
    {
        pc.Name = this.txtname.Text;
        pc.Address.Street = this.txtaddr.Text;
        pc.Address.City = this.txtcity.Text;
        pc.Address.State = this.txtstate.Text;
        pc.Address.Zipcode = this.txtzip.Text;
        pc.Birthday = this.Calendar1.SelectedDate;

        pc.Save();
    }
```

}

When the page is executed for the first time, the user needs to enter the information. However, next time the user details would be automatically loaded.

Attributes for the <add> Element

Apart from the name and type attributes that we have used, there are other attributes to the <add> element. Following table illustrates some of these attributes:

Attributes	Description
name	The name of the property.
type	By default the type is string but it allows any fully qualified class name as data type.
serializeAs	The format to use when serializing this value.
readOnly	A read only profile value cannot be changed, by default this property is false.
defaultValue	A default value that is used if the profile does not exist or does not have information.
allowAnonymous	A Boolean value indicating whether this property can be used with the anonymous profiles.
Provider	The profiles provider that should be used to manage just this property.

Anonymous Personalization

Anonymous personalization allows the user to personalize the site before identifying themselves. For example, Amazon.com allows the user to add items in the shopping cart before they log in. To enable this feature, the web.config file could be configured as:

<anonymousIdentification enabled="true" cookieName=".ASPXANONYMOUSUSER"

cookieTimeout="120000" cookiePath="/" cookieRequiresSSL="false"

cookieSlidingExpiration="true" cookieprotection="Encryption"

coolieless="UseDeviceProfile"/>

LEARNING ACTIVITY: ASP.NET - EVENT HANDLING

An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.

Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.

Read more details on:

https://www.tutorialspoint.com/asp.net/asp.net_event_handling.htm

How do you personalize user identity and other information details in asp.net?

Define your Direction: On a separate piece of paper, explain the importance of the different types of the terms according to the asp.net:

Terms	*Importance*
Web applications.	
Exception handling	
Page Request	
Binary Assembly	
Configuration File	

JAVA BASICS

Java is a general-purpose computer-programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA) meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is one of the most popular programming languages in use particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

The original and reference implementation Java compilers, virtual machines, and class libraries were originally released by Sun under proprietary licenses. As of May 2007, in compliance with the specifications of the Java Community Process, Sun relicensed most of its Java technologies under the GNU General Public License. Others have also developed alternative implementations of these Sun technologies, such as the GNU Compiler for Java (bytecode compiler), GNU Classpath (standard libraries), and IcedTea-Web (browser plugin for applets).

The latest version is Java 10, released on March 20, 2018, which follows Java 9 after only six months in line with the new release schedule. Java 8 is still supported but there will be no more security updates for Java 9. Versions earlier than Java 8 are supported by companies on a commercial basis; e.g. by Oracle back to Java 6 as of October 2017 (while they still "highly recommend that you uninstall" pre-Java 8 from at least Windows computers).

Java was initially launched as Java 1.0 but soon after its initial release, Java 1.1 was launched. Java 1.1 redefined event handling, new library elements were added.

In Java 1.2 Swing and Collection framework was added and suspend(), resume() and stop() methods were deprecated from Thread class.

No major changes were made into Java 1.3 but the next release that was Java 1.4 contained several important changes. Keyword assert, chained exceptions and channel based **I/O System** was introduced.

Java 1.5 was called J2SE 5, it added following major new features:

I/O System is the communication between an information processing system, such as a computer, and the outside world, possibly a human or another information processing system.

- Generics
- Annotations
- Autoboxing and autounboxing
- Enumerations
- For-each Loop
- Varargs
- Static Import
- Formatted I/O
- Concurrency utilities

Major release was Java SE 7 which included many new changes, like :

- Now String can be used to control Switch statement.
- Multi Catch Exception
- try-with-resource statement
- Binary Integer Literals
- Underscore in numeric literals, etc.

And the latest addition to the lot is, Java SE 8, it was released on March 18, 2014. Some of the major new features introduced in JAVA 8 are,

- Lambda Expressions
- New Collection Package java.util.stream to provide Stream API.
- Enhanced Security
- Nashorn Javascript Engine included
- Parallel Array Sorting
- The JDBC-ODBC Bridge has been removed etc.

Application of Java

Java is widely used in every corner of world and of human life. Java is not only used in software's but is also widely used in designing hardware controlling software components. There are more than 930 million

JRE downloads each year and 3 billion mobile phones run java.

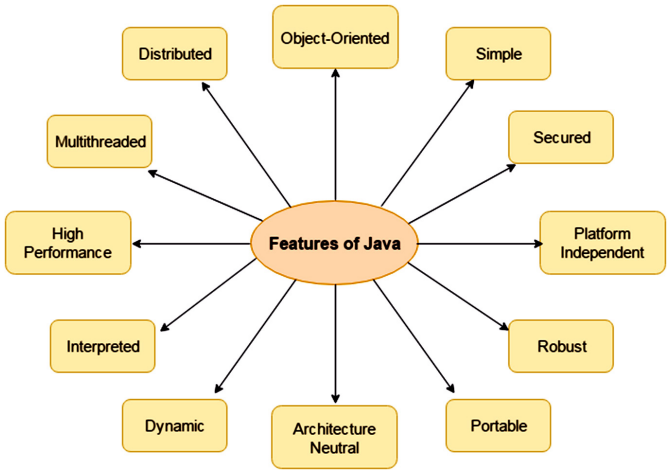
Following are some other usage of Java:

- Developing Desktop Applications
- Web Applications like Linkedin.com, Snapdeal.com etc
- Mobile Operating System like Android
- Embedded Systems
- Robotics and games etc.

◑ Features of Java

The main objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some awesome features which play important role in the popularity of this language. The features of Java are also known as:

- Simple
- Object-Oriented
- Portable
- Platform independent
- Secured
- Robust
- Architecture neutral
- Interpreted
- High Performance
- Multithreaded
- Distributed
- Dynamic



Platform Independent

Java is platform independent because it is different from other languages like C, C++ etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

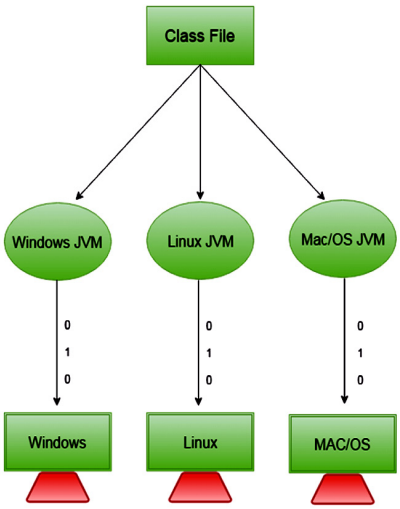
There are two types of platforms software-based and hardware-based. Java provides software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

- Runtime Environment
- API(Application Programming Interface)

Write once, run anywhere (WORA) was a slogan created by Sun Microsystems to illustrate the cross-platform benefits of the Java language.

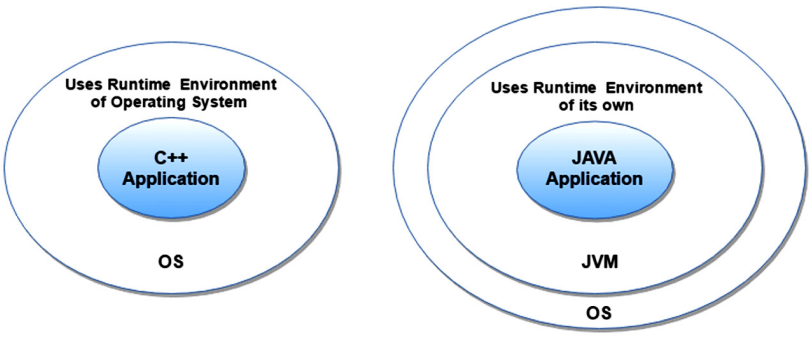
Java code can be run on multiple platforms e.g. Windows, Linux, Sun Solaris, Mac/OS etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms i.e. **Write Once and Run Anywhere (WORA)**.



Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- No explicit pointer
- Java Programs run inside virtual machine sandbox



Classloader: Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to dynamically load Java classes into the Java Virtual Machine. It adds security by separating the package for the classes of the local file system from those that are imported from network sources.

Bytecode Verifier: It checks the code fragments for illegal code that can violate access right to objects.

Security Manager: It determines what resources a class can access such as reading and writing to the local disk.

These security are provided by java language. Some security can also be provided by application developer through SSL, JAAS, Cryptography etc.

Robust

Robust simply means strong. Java is robust because:

- It uses strong memory management.
- There are lack of pointers that avoids security problems.
- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There is exception handling and type checking mechanism in java. All these points makes java robust.

◑ Simple Program of Java

We can write a simple hello java program easily after installing the JDK.

To create a simple java program, you need to create a class that contains main method.

Requirement for Hello Java Example

For executing any java program, you need to

- install the JDK if you don't have installed it, download the JDK and install it.
- set path of the jdk/bin directory. http://www.javatpoint.com/how-to-set-path-in-java
- create the java program
- compile and run the java program

Creating hello java example

Let's create the hello java program:

```
class Simple{
    public static void main(String args[]){
     System.out.println("Hello Java");
```

```
        }
    }
```

Java Security

The Java platform was designed with a strong emphasis on security. At its core, the Java language itself is type-safe and provides automatic garbage collection, enhancing the robustness of application code. A secure class loading and verification mechanism ensures that only legitimate Java code is executed.

The initial version of the Java platform created a safe environment for running potentially untrusted code, such as Java applets downloaded from a public network. As the platform has grown and widened its range of deployment, the Java security architecture has correspondingly evolved to support an increasing set of services. Today the architecture includes a large set of application programming interfaces (APIs), tools, and implementations of commonly-used security algorithms, mechanisms, and protocols. This provides the developer a comprehensive security framework for writing applications, and also provides the user or administrator a set of tools to securely manage applications.

The Java security APIs span a wide range of areas. Cryptographic and **public key infrastructure (PKI)** interfaces provide the underlying basis for developing secure applications. Interfaces for performing authentication and access control enable applications to guard against unauthorized access to protected resources.

Public key infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.

The APIs allow for multiple interoperable implementations of algorithms and other security services. Services are implemented in providers, which are plugged into the Java platform via a standard interface that makes it easy for applications to obtain security services without having to know anything about their implementations. This allows developers to focus on how to integrate security into their applications, rather than on how to actually implement complex security mechanisms.

The Java platform includes a number of providers that implement a core set of security services. It also allows for additional custom providers to be installed. This enables developers to extend the platform with new security mechanisms.

Basic Security Architecture

The Java platform defines a set of APIs spanning major security areas, including cryptography, public key infrastructure, authentication, secure communication, and access control. These APIs allow developers to easily integrate security into their application code. They were designed around the following principles:

Implementation Independence

Applications do not need to implement security themselves. Rather, they can request security services from the Java platform. Security services are implemented in providers (see below), which are plugged into the Java platform via a standard interface. An application may rely on multiple independent providers for security functionality.

Implementation Interoperability

Providers are interoperable across applications. Specifically, an application is not bound to a specific provider, and a provider is not bound to a specific application.

Algorithm Extensibility

The Java platform includes a number of built-in providers that implement a basic set of security services that are widely used today. However, some applications may rely on emerging standards not yet implemented, or on proprietary services. The Java platform supports the installation of custom providers that implement such services.

Security Providers

The java.security.Provider class encapsulates the notion of a security provider in the Java platform. It specifies the provider's name and lists the security services it implements. Multiple providers may be configured at the same time, and are listed in order of preference. When a security service is requested, the highest priority provider that implements that service is selected.

Applications rely on the relevant getInstance method to obtain a security service from an underlying provider. For example, message digest creation represents one type of service available from providers. An application invokes the getInstance

method in the java.security.MessageDigest class to obtain an implementation of a specific message digest algorithm, such as SHA-256.

MessageDigest md = MessageDigest.getInstance("SHA-256");

The program may optionally request an implementation from a specific provider, by indicating the provider name, as in the following:

MessageDigest md =

MessageDigest.getInstance("SHA-256", "ProviderC");

Figures 1 and 2 illustrate these options for requesting an SHA-256 message digest implementation. Both figures show three providers that implement message digest algorithms. The providers are ordered by preference from left to right (1-3). In Figure 1, an application requests an SHA-256 algorithm implementation without specifying a provider name. The providers are searched in preference order and the implementation from the first provider supplying that particular algorithm, ProviderB, is returned. In Figure 2, the application requests the SHA-256 algorithm implementation from a specific provider, ProviderC. This time the implementation from that provider is returned, even though a provider with a higher preference order, ProviderB, also supplies an SHA-256 implementation.

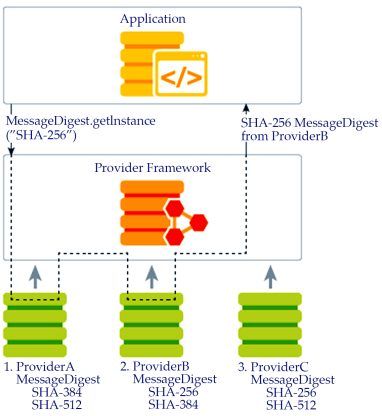


Figure 1: Provider searching.



REMEMBER

The various security areas (cryptography, authentication, etc.) each include descriptions of the relevant services supplied by the default providers. A table in Appendix C summarizes all of the default providers.

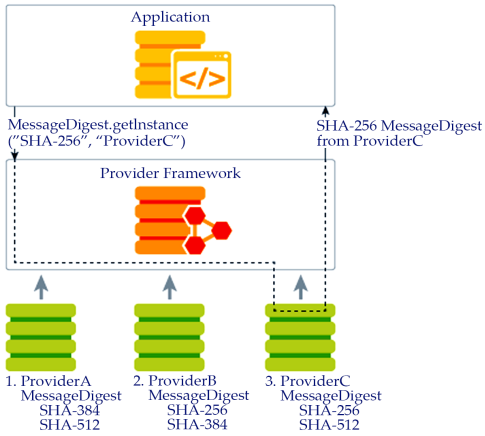


Figure 2: Specific provider requested.

The Java platform implementation from Oracle includes a number of pre-configured default providers that implement a basic set of security services that can be used by applications. Note that other vendor implementations of the Java platform may include different sets of providers that encapsulate vendor-specific sets of security services. When this paper mentions built-in default providers, it is referencing those available in Oracle's implementation.

File Locations

Certain aspects of Java security mentioned in this paper, including the configuration of providers, may be customized by setting security properties. You may set security properties statically in the security properties file, which by default is the java. security file in the lib/security directory of the directory where the Java Runtime Environment (JRE) is installed. Security properties may also be set dynamically by calling appropriate methods of the Security class (in the java.security package).

The tools and commands mentioned in this paper are all in the ~jre/bin directory, where ~jre stands for the directory in which the JRE is installed.

LEARNING ACTIVITY: JAVA PROGRAMMING BASICS

Java is one of the most popular and widely used programming language and platform. A platform is an environment that helps to develop and run programs written in any programming language.

Java is fast, reliable and secure. From desktop to web applications, scientific supercomputers to gaming consoles, cell phones to the Internet, Java is used in every nook and corner.

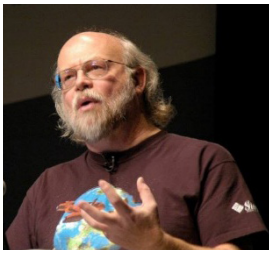
Read more details on:

https://www.geeksforgeeks.org/java-programming-basics/

How to java.security.Provider class encapsulates the notion of a security provider in the Java platform?

Define your Direction: On a separate piece of paper, explain the effect of the different types of the terms according to java programming:

Terms	*Effect*
Application Developers	
Platform Independent	
Software Environment	
Java Runtime Environment	
Application programming interfaces (APIs)	



ROLE MODEL

JAMES GOSLING: *A CANADIAN COMPUTER SCIENTIST, BEST KNOWN AS THE FOUNDER AND LEAD DESIGNER BEHIND THE JAVA PROGRAMMING LANGUAGE.*

BIOGRAPHY

James A. Gosling, O.C., Ph.D. (born May 19, 1955 near Calgary, Alberta, Canada) is a famous software developer, best known as the father of the Java programming language.

In 1977, James Gosling received a B.Sc in Computer Science from the University of Calgary. In 1983, he earned a Ph.D in Computer Science from Carnegie Mellon University, and his doctoral thesis was titled "The Algebraic Manipulation of Constraints". While working towards his doctorate, he wrote a version of emacs (gosmacs), and before joining Sun Microsystems he built a multi-processor version of Unix[1] while at Carnegie Mellon University, as well as several compilers and mail systems.

Since 1984, Gosling has been with Sun Microsystems, and is generally known best as the founder of the Java programming language.

He is generally credited as the inventor of the Java programming language in 1991. He did the original design of Java and implemented its original compiler and virtual machine. For this achievement he was elected to the United States National Academy of Engineering. He has also made major contributions to several other software systems, such as NeWS and Gosling Emacs. He also cowrote the "bundle" program, a utility thoroughly detailed in Brian Kernighan and Rob Pike's book The Unix Programming Environment

KNOWLEDGE CHECK

1. Write the characteristics of PHP.

2. How many types of variables use in PHP?

3. Discuss on PHP file handling system.

4. What is the importance of ASP.NET page life cycle?

5. Write a simple program of java.

6. PHP files have a default file extension of.

a. .html

b. .xml

c. .php

d. .ph

7. A PHP script should start with ___ and end with ___:

a. < php >

b. < ? php ?>

c. < ? ? >

d. < ?php ? >

8. Which of the following type specifier is invalid in print f ( ) functions

a. % a

b. % b

c. % c

d. % d

9. Which of the following function can assign the output to a string variable

a. echo ( )

b. print ( )

c. print f ( )

d. s print f ( )

10. Which of the following data type is not seal or datetype supported by PHP

a. Array

b. String

c. Float Boolean

11. Which file contains settings for all .NET application types, such as Windows, Console, ClassLibrary, and Web applications?

a. Web.config

b. Machine.config

c. Global.asax

d. All of the above

12. Which programming model should you implement if you want to separate your server-side code from your client-side layout code in a Web page?

a. Single-file model

b. Code-behind model

c. Inline model

d. Client-server model

13. You want to make a configuration setting change that will affect only the current Web application. Which file will you change?

a. Global.asax

b. Web.config in the root of the Web application

c. Machine.config

d. All of the above

14. Which of the following is not an ASP.NET page event?

a. Init

b. Load

c. Import

d. None of the above.

15. To implement a specified .NET Framework interface which directive is used?

a. @Register

b. @Control

c. @Reference

d. @Implements

16. Which one of these lists contains only Java programming language keywords?

a. class, if, void, long, Int, continue

b. goto, instanceof, native, finally, default, throws

c. try, virtual, throw, final, volatile, transient

d. strictfp, constant, super, implements, do

17. Which is a reserved word in the Java programming language?

a. native

b. subclasses

c. reference

d. array

18. Which is a valid keyword in java?

a. interface

b. string

c. Float

d. unsigned

19. Which is the valid declarations within an interface definition?

a. public double methoda();

b. public final double methoda();

c. static void methoda(double d1);

d. protected void methoda(double d1);

20. Demonstrate and also explain with a suitable example of ASP.NET architecture and its components.

REFERENCES

1. Arnold, Ken, James Gosling and David Holmes (2005). The Java Programming Language, 3rd edition. NJ: Prentice Hall.
2. Balagurusamy, E. (2007). Programming with JAVA: A Primer, 3rd edition. New Delhi: Tata McGraw-Hill.
3. Cadenhead, Rogers (2000). Teach Yourself Java in 24 Hours. New Delhi: SAMS Techmedia.
4. Deitel, Harvey and Paul Deitel (2003). JAVA How to Program, 5th edition. NJ: Prentice Hall.
5. Doug Lea, Concurrent Programming in Java: Design Principles and Patterns, Addison-Wesley, 2006.
6. Elliotte Rusty Harold, Java I/O, O'Reilly, 2002.
7. Elliotte Rusty Harold, Java Network Programming, Prentice-Hall, 2004.
8. Introduction to Programming Using Java v7 by David J. Eck. Hobart and William Smith Colleges. August 2014. CC-BY-NC-SA-3.0. I have used several chapters in this work.
9. Think Python 2e by Allen Downey. Green Tea Press. June 2014. Think Python 2e is a Free Book. It is available under the Creative Commons Attribution-NonCommercial 3.0 Unported License, which means that you are free to copy, distribute, and modify it, as long as you attribute the work and don't use it for commercial purposes. I have used Appendix A and the book overall as a model in this work.
10. Vincent J. Hardy, Java 2D API Graphics, Sun Microsystems Press, 2000.

INDEX

A

Application programming interfaces (APIs) 265
Arithmetic operator 204
Assignment operator 202, 209

B

background color 82
Berkeley Software Distributions (BSDs) 107
bgcolor 82
blue value(BB) 83
browsers 73, 83, 84
Bytecode 259, 262

C

Cascading Style Sheet (CSS) 139
Click Through Rate 12
Client browser 56
Client-side scripting 19
Client side scripting language 194
Common gateway interface (CGI) 20
Communication protocol 104
Computer program 55
Computer programming languages 104
Computer Science 230, 270
Content management systems (CMS) 21

D

Descendant selector 142
Dynamic Web pages 17

E

Efficiency 232
element backgrounds 173
e-mail link 89
Embedded code 139
embedded style sheet 160, 162
Error message 145
eXtensible HyperText Markup Language (XHTML) 3
Extensible Markup Language (XML) 4, 105, 119
external style sheet 166, 168

F

Facebook Application platform 106
File handling 240
File Transfer Protocol (FTP) 4
Functional programming 194

G

Google Maps 100
green value(GG) 83

H

horizontal length 174
Horizontal lines 54
HTML document 55, 71, 72, 77, 91
Hyperlinks 48
HyperText Markup Language (HTML) 48
Hypertext Preprocessor 231
Hypertext Transfer Protocol (HTTP) 105, 119

I

inline styles 158, 162, 164
Integrated object oriented 231
Internal links 87

J

Java language libraries 103
Java Runtime Environment(JRE) 263
JavaScript 145, 150, 191
JavaScript Object Notation (JSON) 105, 119
Java virtual machine (JVM) 259

L

Language dependent API 99
Language independent APIs 99

M

Microsoft Windows platforms 109

N

Nonbreaking space entity 56

O

Object-oriented languages 102, 103, 132
Open Social platform 106
Open-source community 232
Operating system 98, 104, 107, 108, 117, 118

P

Personal digital assistant (PDA) 234
Photorealistic 55
Platform Independent 262
Programming language 194, 195, 196, 199, 214, 222, 228, 230, 244, 259, 261, 269, 270, 273
Prototype 100
Pseudo-classe 150, 151, 157
Public key infrastructure (PKI) 265

R

Representational State Transfer (REST) 105
Resource allocation 232
Robust 261, 264
Rule render 141, 142

S

Scripting language 19, 20
Search engine optimization 11
Security mechanisms 233, 265, 266
Server-side applications 231
Server-side scripting 20, 21, 36, 43
Session management 233
Simple Mail Transport Protocol (SMTP) 161
Simple Object Access Protocol (SOAP) 105, 120
Software framework 103
Software library 103
Static web page 17
String 233, 242, 271
Style sheets 145, 165
Supporting object-oriented 194

T

Ternary operator 201
Traditional print media 3

U

unordered list 84

W

Web developer 139
Web publishing 5
website 82, 88
Web site 166
Web technology 6
World Wide Web Consortium (W3C) 64
World Wide Web (WWW) 4, 24
Write once, run anywhere\ 259

Level: Beginner to Advanced
Subject: Computer and Information Science

3G Handy Guide: Building Web Sites

In today digital world, website is need for every organization of any field and web technology is playing vital role in business development as its technology and working procedures are also changing speedily. So, up-to-date designing and development techniques in website making its wider scope and the course content are also growing gradually. Web design is used as a general term to describe any of the various tasks involved in creating a web page. More specifically, it refers to jobs focused on building the front-end of a web page. The web contains numerous pages, presenting information using different technologies and associated together with hyperlinks. There are two basic aspects to any web page found on the Internet. The first is a presentation that the user interacts with, usually visually, while the second is a back-end that includes information for non-human browsers.

Web designing encompasses many different skills and disciplines in the production and maintenance of websites including web graphic design; interface design; authoring, standardized coding using proprietary software; user experience design. Web Designing manages the way toward building up a site. A website specialist use pictures, content, html5, Responsive website composition, CSS3, data engineering, hues, text styles and other intuitive components to deliver a site page. Website specialists use markup language for the most part HTML for making structure and CSS for presentation. The web has gone through many changes over the past few decades, but HTML has always been the fundamental language used to develop webpages. Interestingly, while websites have become more advanced and interactive, HTML has actually gotten simpler. If you compare the source of an HTML5 page with a similar page written in HTML 4.01 or XHTML 1.0, the HTML5 page would probably contain less code. This is because modern HTML relies on cascading style sheets or JavaScript to format nearly all the elements within a page.

This Handy Guide introduces students to basic web design using HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets). Throughout the book students are introduced to planning and designing effective web pages; implementing web pages by writing HTML and CSS code; enhancing web pages with the use of page layout techniques, text formatting, graphics, images, and multimedia; and producing a functional, multi-page website. In computer programming, an application programming interface (API) is a set of subroutine definitions, protocols, and tools for building software and applications.

ISBN 978-1-98464-157-1



BIBLIOTEX
Digital Library

www.3ge-learning.com
email: info@3ge-learning.com