Web Scripting

Second Edition

Emmanuel Kazanidis, M.Sc.
Florida Valencia Ortiz
Mai Ryza D. Amante, M.A.
Stephen C. De Sagun
Jaime D.L. Caro, Ph.D.

I.C.TOPIA
ICT Training for the Future

THIS BOOK BELONGS TO:

I.C.TOPIA

ICT Training for the Future

WEB SCRIPTING

SECOND EDITION

Emmanuel Kazanidis, M.Sc.
Florida Valencia Ortiz
Mai Ryza D. Amante, M.A.
Stephen C. De Sagun
Jaime D.L. Caro, Ph.D.

Miriam College Quezon City SY 2016-2017

techFactorsInc

Fifth printing of the second edition, 2016
ISBN 978-971-0550-37-1

Published by TechFactors Inc.
Printed in the Philippines

Authors Emmanuel Kazanidis, M.Sc. Florida V. Ortiz, Mai Ryza D. Amante, M.A., and Stephen C. De Sagun
Series Editor Jaime D.L. Caro, Ph.D.
Cover Design Jiyas P. Suministrado

Content and Editorial Alvin Ramirez, Frances Ibañez, M.A., Alexander Lim, MBA, and Joanne April Ortiz
Creatives Jiyas Suministrado, Gilbert Lavides, and Regina Zapata
Systems Kim Benebese, Caselyn Dionisio, Mark Abliter, and Ezekiel Pattaguan

Exclusively distributed by TechFactors Inc.
101 V. Luna Road Ext., Sikatuna Village
Diliman, Quezon City
1101 Philippines

Telephone number: (632) 929 6924
E-mail address: info@techfactorsinc.com
Website: www.techfactorsinc.com

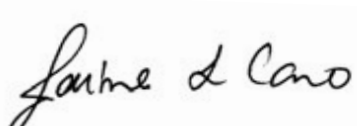
FOREWORD

The difference between high school students who are introduced to practical computing proficiencies and those who are not is their capacity to carry-over what they've learned about information technology into college and consequently apply them, resulting in more productivity inside and outside of the school environment. The TechFactors I.C.Topia courseware series gives high school students the means to make this happen with the use of the Department of Education (DepEd) curriculum.

The I.C.Topia courseware is a complete system for learning specific skill sets in relation to popular productivity applications used at home, the office, and in school. Such programs—both proprietary and open source—have been selected and highlighted in the series of books as integrated subjects in the lessons.

Each I.C.Topia book title is carefully chosen for its relevance to current trends of use. They combine applications and defined output processes specifically for their related aspects of use. Word processing, desktop publishing, and office applications are the basis for the office productivity courseware; web development and javascripting are used for the Internet learning set; programming and database are taken up in the software development kit; and lastly, digital photography, videography, and project management comprise the multimedia production book.

As a server- and Internet-based courseware series, I.C.Topia gives the benefit of on-the-go-learning that's relevant in today's interconnected world. It serves to develop practical know-how and adeptness in using networks during the skills-mastery process. I.C.Topia serves to produce productive graduates in a country that grows to be more technologically-dependent, connected, and conscious of projecting it's presence to the rest of the world.



Jaime D.L. Caro, Ph.D.

Series Editor

ABOUT THE AUTHORS

Emmanuel Kazanidis, M.Sc. graduated with Masters of Science degrees in both Astrophysics and Information Technology from the University of London in 2000 and 2001, respectively. He is also a Student Mentor at the ODL (Open and Distance Learning) Unit at the University of London, where he has developed online learning materials for the Unix Operating System and Java programming for students. Prior to this, he was a teacher of chemistry and physics at Genika Frontistiria Pirea in Greece. He is fluent in English and Spanish.

Florida Valencia Ortiz graduated cum laude from the University of the Philippines Diliman with a Bachelor of Arts degree in Philosophy. She graduated valedictorian from the Holy Rosary Parochial Institute, Orani, Bataan in 2000. In UP, she was the organization head of the UP Kabataang Pilosopo Tasyo (UP KAPITAS) and a member of the UP Taekwondo Club. Her hobbies include martial arts, dancing, digital photography, and video editing.

Mai Ryza D. Amante, M.A., earned her masters degree in Education major in Educational Technology from the University of the Philippines Diliman and also has a bachelor's degree in Development Communication (major in Science Communication) from UP Los Baños. She worked as a consultant at Elyon Academia Foundation Inc., providing assistance and advice in curriculum development and educational materials production. She was also Management Associate of MADECOR Group, where she was involved with research and training implementation in the field of education, and information and communications technology (ICT). She is now Communications and Publications Assistant at the Ramon Magsaysay Award Foundation (RMAF) and is a member of the Samahan ng mga Kurilikularista (UP Diliman).

Stephen C. de Sagun is an undergraduate student taking BS Computer Science at the University of the Philippines in Los Banos, Laguna. He is currently employed in 3Logix Philippines Inc. as a mobile device programmer and software developer for server applications.

ABOUT THE SERIES EDITOR

Jaime D.L. Caro, Ph.D. has more than 20 years of experience in education and research in the areas of Computer Science, Information Technology, and Mathematics. He received the degrees of Bachelor of Science major in Mathematics (cum laude) in 1986, Master of Science in Mathematics in 1994, and Doctor of Philosophy in Mathematics in 1996, all from the University of the Philippines Diliman. He spent a year as a post doctorate research fellow at the University of Oxford from 1997 to 1998. He is presently Assistant Vice President for Development of the University of the Philippines, Program Director of the UP Information Technology Training Center (UP ITTC), and a professor of Computer Science in UP Diliman. He is an honorary member of the Philippine Society of Information Technology Educators (PSITE), President of the Computing Society of the Philippines (CSP), and a member of the Technical Panel on Information Technology Education of the Commission on Higher Education (CHED). Dr. Caro is a recognized expert on Complexity Theory, Combinatorial Network Theory, Online Communities, and e-Learning.

TABLE OF CONTENTS

INTRODUCTION

In this course, the students will be taught to create their own webpages with Notepad, the basic text editor of the Windows® operating system. It will be used instead of WYSIWYG (What You See Is What You Get) editors to ensure that the students learn the basic structure and syntax of HTML. Matters of accessibility will be incorporated into each lesson as the principles of web design are thoroughly observed, such as the quality of text and images and the efficiency of layout and hyperlinks.

The second part of the book makes it easy to learn how to create client-side web applications using JavaScript. It provides a simplified and easy-to-follow learning material for aspiring webmasters or students having the interest in learning more advanced skills in web development.

LEARNING GOALS

At the end of the course, the student will be able to:

1. Create a well-formed website according to W3C standards.
2. Appreciate the use of HTML tags and CSS properties.
3. Evaluate other websites.
4. Create JavaScripts that would be useful in creating a functional and handy website.
5. Do programming using JavaScript.

DISCLAIMER

The webpages in this book were authored using Notepad 5.1 and rendered in Internet Explorer 6.0. It is not guaranteed that these webpages will display in exactly the same way in other browsers. Some HTML codes (e.g., the marquee tag) are acknowledged only by later versions of Internet Explorer. In addition, the screenshots of the browser windows were adjusted to highlight the elements relevant to the discussions.

I.C.TOPIA

Miriam College Quezon City SY 2016-2017

LESSON 01

The Webpage Untangled

At the end of this lesson, the student will be able to:

1. **Understand the basic concepts involved in webpage design.**
2. **Make his/her first webpage.**
3. **Write the basic patterns for HTML tags.**
4. **Insert a DOCTYPE declaration.**

It used to be that when someone says "web," we assocate it with spiders and weaving threads. Nowadays, we associate the web with a different kind of weaving. Instead of threads, we form a different kind of connection – a digital connection, from one computer to another, to various computers – and the web that emerged from these connections are as complicated and as complex as any spider's web. According to the Peabody Inst. (2003), the **Web** or **World Wide Web** is "[a] system using the Internet to access information stored on computers worldwide." It provides a means of communication for our fast-paced and highly technological society.

Webpage creation involves a local computer, where designers develop their webpage, and a server (a remote computer) which contains the website and serves up the webpages (which the designers created) for people to visit.

When you visit a website, the opening webpage is what you usually see on your screen every time you connect to the Internet. These webpages contain information that we search for or try to access whenever we surf the Internet. We do this by typing in a web address or a URL (Uniform Resource Locator) in the address bar of our browsers.

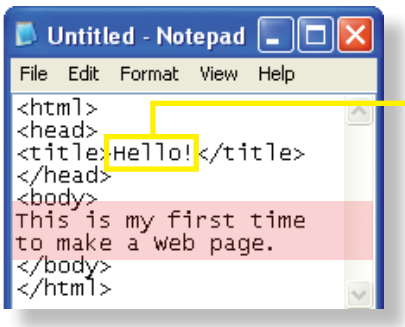
Your First Webpage

Webpages are created by designers using languages like Hypertext Markup Language (HTML) and eXtensible Hypertext Markup Language (XHTML). These languages are used to tell browsers how to display your webpage. Now, let's get started by creating your first HTML webpage.

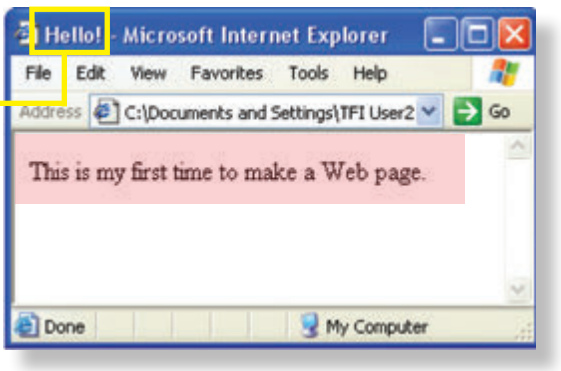
NOTE

As of this writing, the current version of HTML is HTML 5, but it is being superceded by XHTML. The difference between the two are minor but the main benefit of using XHTML is that it is becoming a standard in "non-computer" devices like cellphones, palm devices and scaled-down browsers. XHTML is also extensible, which is the fancy way of saying new tags can be added without a new document type declaration (DTD). More about DTDs will be given later.

Text Editor Display



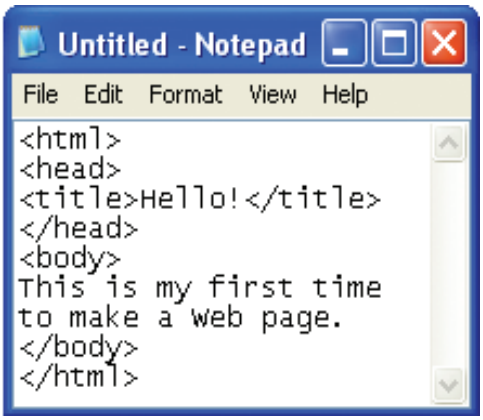
Web Browser Display



It is time to tell the world that you know how to make a webpage!

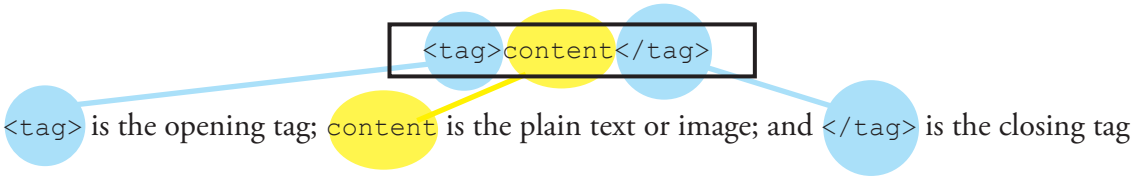
1. Open Notepad. It is commonly found under **Accessories** in the All Programs selection of the Windows Start menu.

2. Type what you see inside this text box of Notepad.



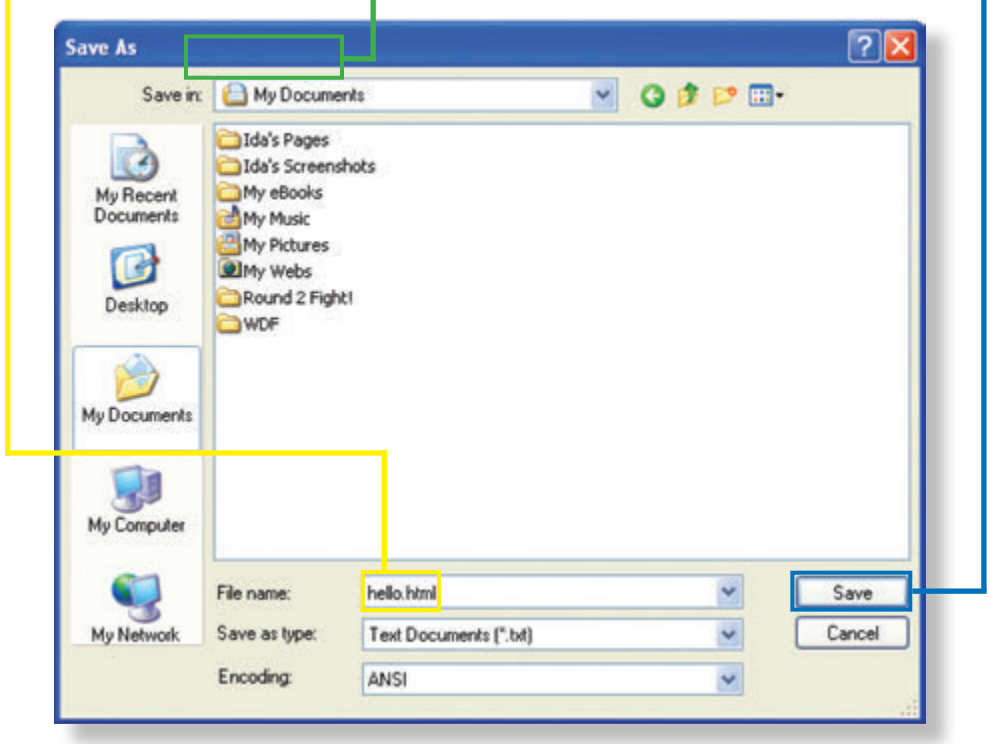
* The title bar also has the minimize, maximize/restore, and close buttons.

HTML makes use of tags. The tags in the previous HTML code are highlighted in the following example. It also shows the basic pattern used for most HTML tags:



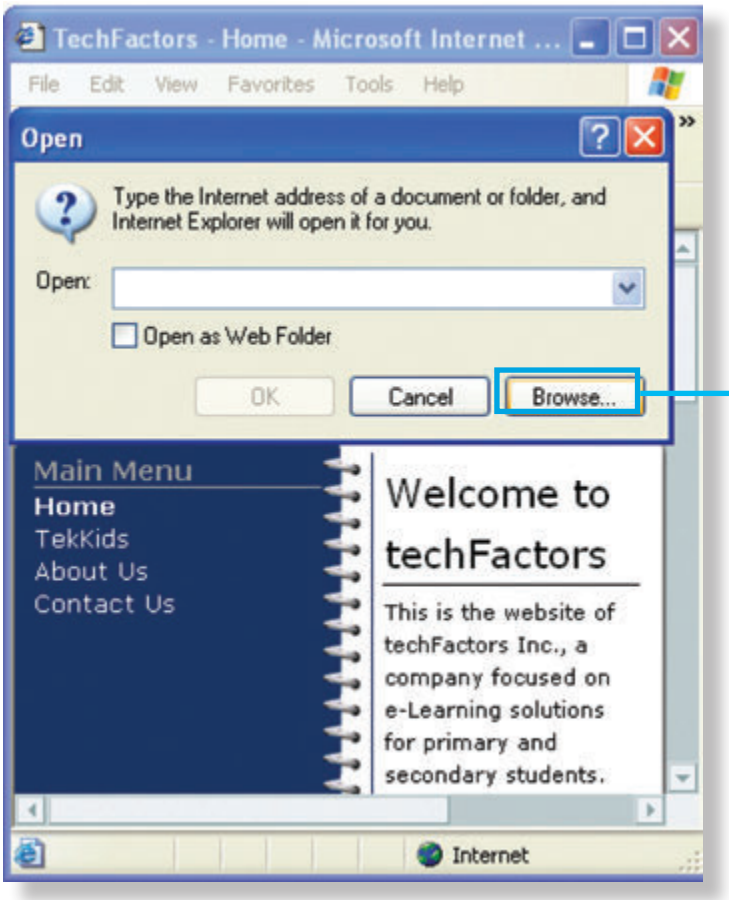
Note that for every opening tag there is an equivalent closing tag. The opening tags are different from the closing tags because the closing tags have a slash before the text. We will dissect what each tag in the text box stands for later in the book. For now, just type them in.

3. Save this file in your My Documents folder by going to Notepad's **File** menu, selecting **Save**, typing "hello.html", and clicking Save.



4. Open your browser (e.g., Internet Explorer).

5. Go to your browser's **File** menu, click **Open**, and click Browse .



6. Go to your **My Documents** folder, double-click hello.html, and click OK .

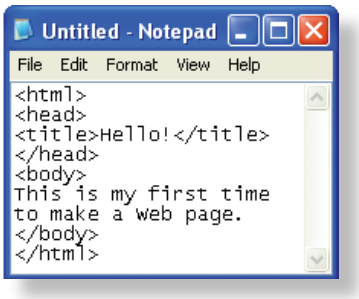
Your first webpage is finally on the browser!

NOTE

When you edit your HTML document and save the revisions in Notepad, the browser will only display the altered webpage if you reload the screen by clicking the Refresh button in the browser's toolbar.

HTML Tags

Each tag in the text box plays a role in the creation of your HTML.



HTML Tag	DESCRIPTION
<html>	Indicates that the file is composed of HTML commands
<head>	Usually contains the title tag
<title>	Encloses the written content what will appear in the title bar of the browser window
<body>	Includes all the other tags, elements and contents of your page

Notice how the tags are used. For example, <title>Hello!</title> is contained between <head> and </head>. The <title> tag is terminated first with </title>, before calling the closing tag </head>. This follows the First In Last Out (FILO) rule. When creating nested tags, the latest tag should be the first one closed before the tag enclosing it is closed.

Empty tags are stand-alone tags which have the following format:

```
<tag />
```

Empty tags can be thought of as shortcuts, since there is no data placed between the opening and the closing tags, although attributes can be placed here (more on this later). So in effect:

```
<tag></tag> is the same as <tag />
```

NOTE

HTML 5 is case-insensitive, therefore `<TAG />` and `<tag />` are interpreted the same. The convention when programming, however, is to use lowercase.

Web Tools

Notepad uses basic text formatting to make and edit text files. It may be the simplest tool for making webpages, but there are other web development tools available. WordPad, for example, can be used to do basic formatting, while Microsoft Word can be used to autocorrect entries. Microsoft FrontPage, which is a What You See Is What You Get (WYSIWYG) program, can also be used to lay out webpage components as these would appear if using a Graphical User Interface (GUI) so that you don't have to manually write the code, since images and icons are used. Using a WYSIWYG program is quick and convenient, but it could become troublesome since the program might add code that can be hard to debug.

CONNECT

Document Type Definition (DTD)

Every webpage must contain a DOCTYPE definition so the browsers know what type of document your webpage is. With HTML 4, the DOCTYPE definition should contain a DTD, which contains machine-readable grammar specifying the allowed and prohibited contents for the document. An example of such header is

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

In HTML 5, while no DTD is needed, the DOCTYPE definition is still necessary. This is because current web browsers now use special-purpose HTML parsers rather than general purpose DTD based parsers. With HTML 5, only the following should be included on the top of the page.

```
<!DOCTYPE html>
```

This Document Type Definition is for HTML 5

```
<!doctype html>

<html>
  <head>
    <title>Hello HTML</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. Define HTML, XHTML, and W3C.

   _____

2. What are the main elements of an HTML document? Describe each.

   _____

3. What is a text editor and how does it differ from a WYSIWYG editor?

   _____

B. Match the web-related words with the correct meanings/concepts.

Words	Meaning
_____ 1. Notepad	a. the governing body dedicated to improving the Web
_____ 2. Browser	b. mark-up codes
_____ 3. HTTP	c. advantageous against its counterpart because of its portability
_____ 4. HTML	d. instantly creates code based on components laid out by the user
_____ 5. WYSIWYG	e. program used to access and display pages and files on the Web
_____ 6. Tags	f. language for creating webpages
_____ 7. Server	g. a basic HTML editor
_____ 8. Empty Tag	h. protocol for networked information
_____ 9. W3C	i. where webpages are deployed
_____ 10. XHTML	j. tag that does not contain any data

C. Let's practice your HTML skills. How many of these tags can you recognize? Match the markup tags with the correct purpose/function.

Words	Meaning
_____ 1. <head></head>	a. defines a paragraph
_____ 2. <title></title>	b. defines the name of the HTML document
_____ 3. <body></body>	c. defines which version of (X)HTML a web document is actually using
_____ 4. DOCTYPE	d. defines the part of the web document that contains all the HTML elements
_____ 5.	e. contains general information about the HTML document

D. What kind of information can you find in the World Wide Web? List down at least ten types of information that you can access from the Web and give general examples of webpages where we can get such information. Two examples are provided below:

Type of Information	Examples
1. Personal information	Various blogs
2. Job listing	JobStreet.com
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	

W3C Standards

The future direction of the Web will most likely be spearheaded by the World Wide Web Consortium (W3C), an organization that is led by Tim Berners-Lee, the inventor of the Web. This organization is made up of more than 450 organizations, among which are Microsoft, Apple Inc., Sun Microsystems, as well as hardware and software manufacturers, academic institutions, and telecommunications companies. These members provide inputs and suggestions geared towards improving the Internet's functionality. Details about W3C can be found in its website http://www.w3c.org.

The tools for creating a webpage are the text editor and the web browser. The basic pattern for most HTML commands is <tag>content</tag>. The DOCTYPE declaration is necessary for every webpage. The main elements of an HTML document are: html, head, title, and body. W3C is the governing body that handles the Internet's functionality.

LESSON 02

Writing Out the Webpage

At the end of this lesson, the student will be able to:

1. **Use the heading and paragraph tags.**
2. **Create different kinds of lists and nest one upon another.**
3. **Input a preformatted text and a horizontal rule into the page.**

Making a webpage can be confusing because there are so many elements to consider! However, you can organize these elements. "Speaking" the HyperText Markup Language is one way of communicating the what, where, when, and how of presenting the webpage to the browser.

```
<html>
<head>
<title>Go Web!</title>
</head>
<body>
<h1>Go Web!</h1>
<h2>Go Web!</h2>
<h3>Go Web!</h3>
<h4>Go Web!</h4>
<h5>Go Web!</h5>
<h6>Go Web!</h6>
</body>
</html>
```

Format Tags

When you make webpages, you'll need to use headings in order to catch the attention of the Web users browsing your site. Headings are formatted big and bold because they need to "stand out" from the rest of the page's contents. They help the users figure out the different topics covered by your site. Even if the site you wish to develop is a personal or a hobby site, there will still be divisions for the different areas of your personality or hobby.

SEE MY HEAD(ING)?

Heading

Heading elements have opening (e.g., <h3>) and closing (e.g., </h3>) tags to enclose the topic or section title to be used. There are six levels of headings and all except the first level (i.e., <h1>) can be employed over and over throughout the HTML document. The first level heading should be your main or umbrella topic, and the other levels only your subtopics, so that the Web users will not be confused.

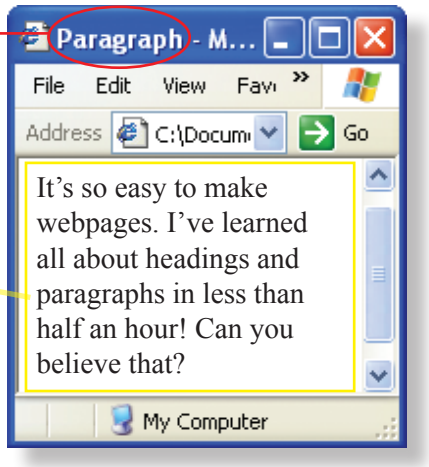
CONNECT

The `<h4>` heading usually displays in the default font size (which is 12pt) and in bold.

Paragraph

Headings may be used to introduce paragraphs. The **paragraph** element tags are simply `<p>` and `</p>`. As much as possible, paragraphs must be kept short. Try typing the following in your Notepad and see what happens:

```
<html>
<head>
<title>Paragraph</title>
</head>
<body>
<p>It's so easy to make
webpages. I've learned
all about headings and
paragraphs in less than
half an hour! Can you
believe that?</p>
</body>
</html>
```



If you want to break the paragraph into three, enclose each sentence in paragraph tags. Each sentence will now become its own paragraph.

Space

Spacing in HTML is overlooked by browsers. For example, browsers interpret an entry like "t h i s" (with two spaces between each letter) as only "t h i s" (with only one space between each letter), because multiple spaces are still considered as one. You can, however tell the browser to register spaces by typing " " for each space that you want. You can also use the " " command to connect two words that should always be seen together in the same line, rather than having those two words broken up by a line break if it reaches the limit of the paragraph length allowed in the browser window display. For example:

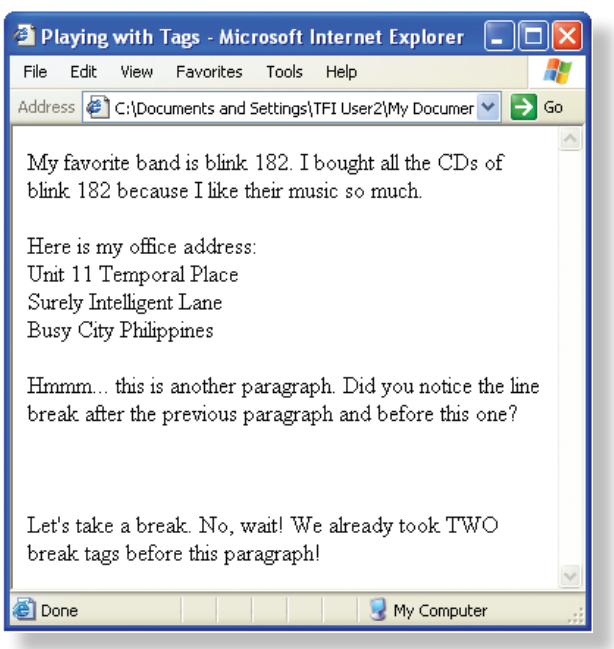
```
<p>My favorite band is blink 182. I bought all the CDs of blink 182
because I like their music so much.</p>
```

To make sure that "blink" and "182" are not split in the second sentence due to space constraints of the browser window, you need to type out "blink 182" instead of simply typing "blink 182". In this case, " " is a special character set that the browser translates as non-breaking space. For example:

Line Break

To introduce line breaks between texts, such as those in snail mail city addresses, insert the break empty tag `<br />`, like in the example that follows. Remember that there is always a line break between paragraphs so you do not have to put `<br />`, unless you want another Enter key effect. Look at the lower half of the screenshot.

```
<html>
<head>
<title>Playing with Tags</title>
</head>
<body>
<p>My favorite band is blink 182. I bought all the CDs of blink 182 because I like their music so much.</p>
<p>Here is my office address:<br/>
Unit 11 Temporal Place<br />
Surely Intelligent Lane<br />
Busy City Philippines</p>
<p>Hmmm... this is another paragraph. Did you notice the line break after the previous paragraph and before this one?</p>
<br /><br />
<p>Let's take a break. No, wait! We already took TWO break tags before this paragraph!</p>
</body>
</html>
```



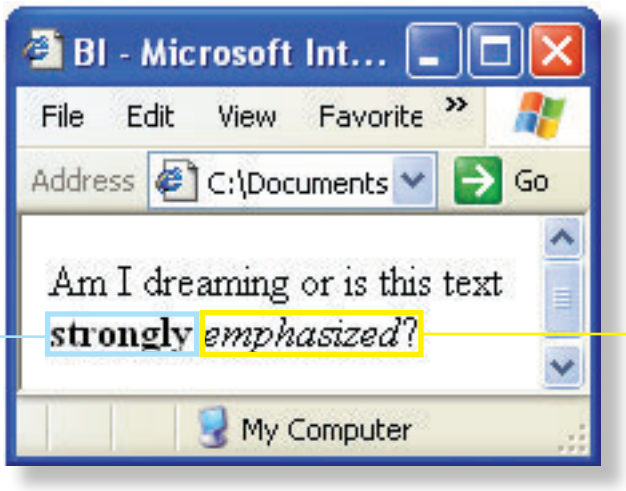
More Text Tags

You can also use tags for other text effects.

HTML Tag	Description
<b>	boldface
<i>	italics
	strongly; same as boldface
	Emphasized; same as italics

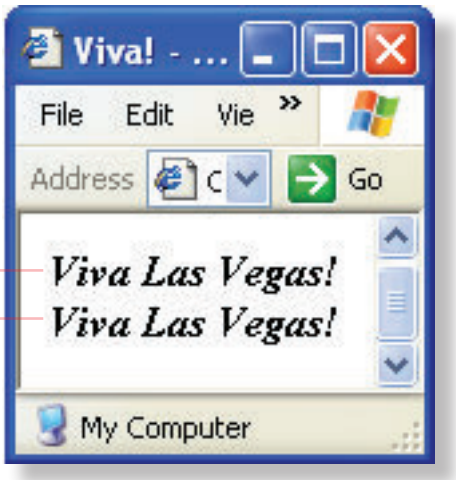
The following is an example using some of these tags:

`<strong>strongly</strong>` for boldface and `<em>emphasized</em>` for italics



Remember FILO? This applies to tags as well. Any combination of tags can be used, as long as you close the innermost tag before closing its enclosing tag, such as in the example below:

```
<strong><em>Viva Las Vegas!</em></strong>
```



Keep the **FILO** rule in mind – **F**irst **I**n **L**ast **O**ut. You may interchange the tags as long as the first that you open is the last that you close, like this:

```
<em><strong>Viva Las Vegas!</strong></em>
```

Tags for Lists

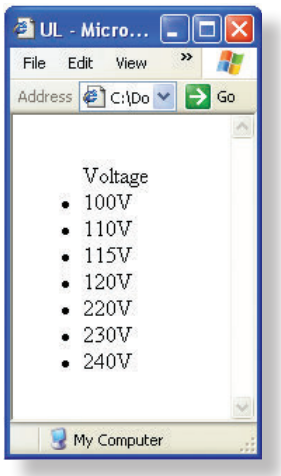
You can also display groups of texts as lists in HTML. Here are a few tags for lists.

HTML Tag	Description
	unordered or bulleted list
	ordered list
	line item

Unordered or Bulleted List

This kind of list is enclosed in `<ul> </ul>` tags, while each item is enclosed in `<li> </li>` tags.

```
<ul>Voltage
    <li>100V</li>
    <li>110V</li>
    <li>115V</li>
    <li>120V</li>
    <li>220V</li>
    <li>230V</li>
    <li>240V</li>
</ul>
```

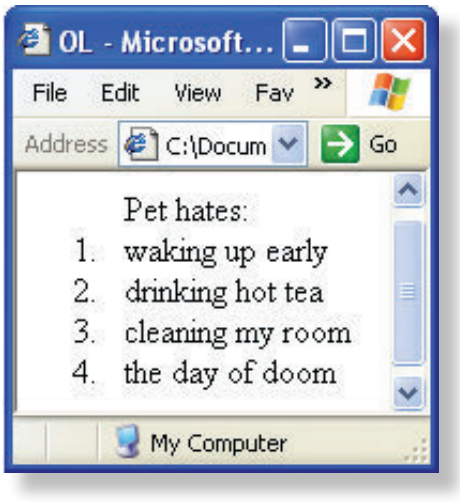


By default, unordered lists are displayed with solid round bullets. The bullet type can be changed by using the `type` attribute.

Ordered or Numbered List

This kind of list is enclosed in `<ol> </ol>` tags, again with each list item inside `<li> </li>` tags.

```
<ol>Pet hates:
    <li>waking up early</li>
    <li>drinking hot tea</li>
    <li>cleaning my room</li>
    <li>the day of doom</li>
</ol>
```



By default, ordered lists use Arabic numerals with a value "1" for the first list item. ,"2" for the second item, and so on. As with unordered lists, the appearance of these numerals can be changed by changing the type attributes. The options are as follows.

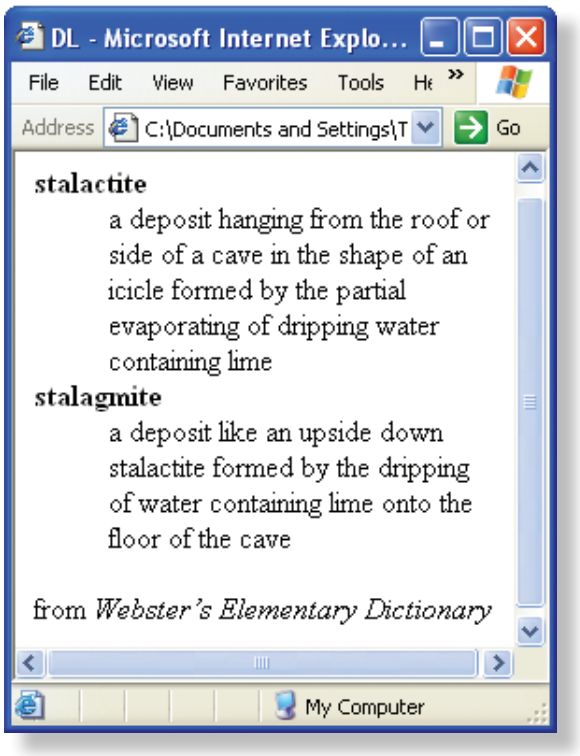
```
<ol type="1">  for Arabic numerals
<ol type="i"> or <ol type="I">  for Roman numerals
<ol type="a"> or <ol type="A">  for Alpha numerals
```

You can also change the starting point of an ordered list by adding the `start` attribute in the initial `<ol>` tag. For example, in Arabic numerals, if you want to start the list at 5, you should type "`<ol start="5">`".

Definition List

This kind of list is enclosed in `<dl> </dl>` tags. The term to be defined is within `<dt> </dt>` tags, while the term's definition is noted inside `<dd> </dd>` tags. Example:

```
<dl>

<dt><strong>stalactite</strong></dt>

<dd>a deposit hanging from the roof or side of a cave in the shape of an icicle formed by the partial evaporating of dripping water containing lime</dd>

<dt><strong>stalagmite</strong></dt>

<dd>a deposit like an upside down stalactite formed by the dripping of water containing lime onto the floor of the cave</dd>

<p>from <em>Webster's Elementary Dictionary</em></p>

</dl>
```



You can have as many descriptions for each term as you like. Inside the definition data `<dd>` tags you can also put paragraphs, line breaks, links, images, or even other lists.

NOTE

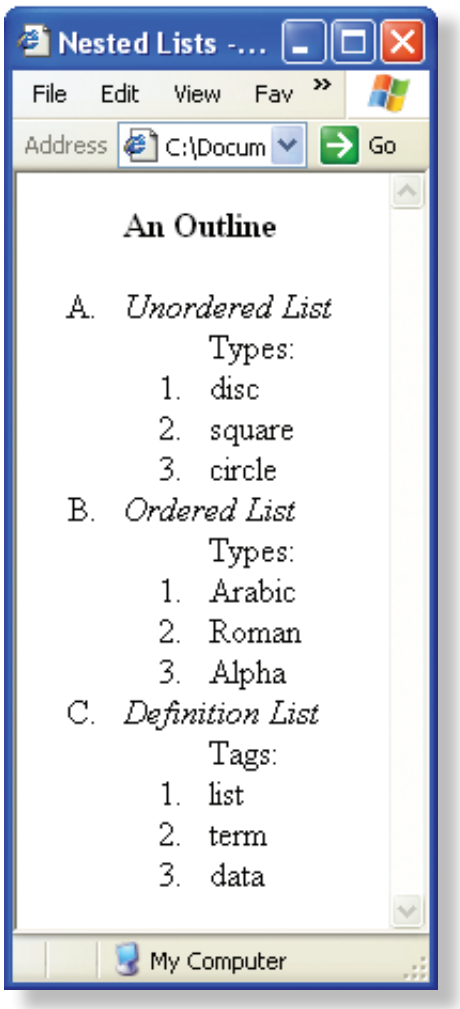
Definition lists are not lists of items.

Nested Lists

When your teacher asks you to do an outline, what s/he wants you to do, in effect, is to come up with a structured list, which you can create in a webpage. You can nest lists within lists to create a hierarchy of information, putting unordered or ordered lists inside the first list's item (<li>) tags.

To give you an idea of how this works, type in your Notepad:

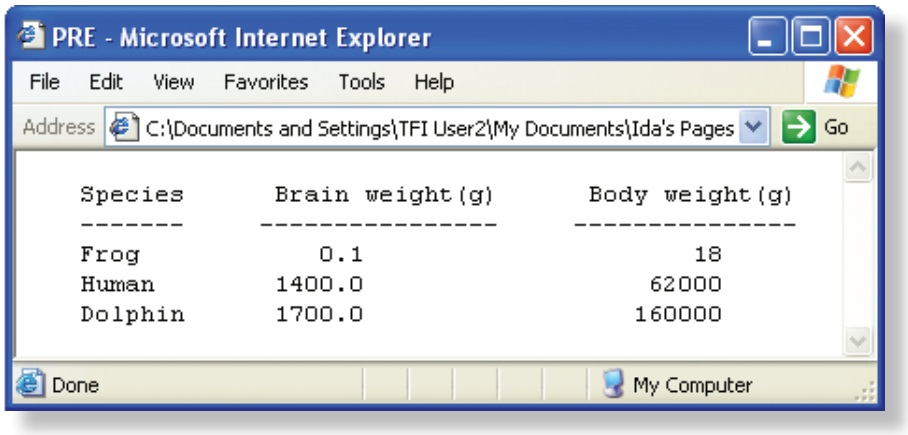
```
<html>
<head><title>Nested Lists</title></head>
<body>
<ol type="A"><b>An Outline</b><br /><br />
    <li><i>Unordered List</i>
        <ol>Types:
        <li>disc</li>
        <li>square</li>
        <li>circle</li>
        </ol>
    </li>
    <li><i>Ordered List</i>
        <ol>Types:
        <li>Arabic</li>
        <li>Roman</li>
        <li>Alpha</li>
        </ol>
    </li>
    <li><i>Definition List</i>
        <ol>Tags:
        <li>list</li>
        <li>term</li>
        <li>data</li>
        </ol>
    </li>
</ol>
</body>
</html>
```



Preformatted Text and Horizontal Rule

The **preformatted text** element (with the <pre> </pre> tags) tells the browser to display the text in fixed-width (i.e., monospace or typewriter) typeface and makes it appear *exactly* as it was typed in Notepad. This HTML tag also enables you to indent text. The following is an example typed in Notepad:

```
<pre>
   Species       Brain weight(g)       Body weight(g)
   -------     -----------------     ---------------
   Frog              0.1                     18
   Human          1400.0                  62000
   Dolphin        1700.0                 160000
</pre>
```



Horizontal Rule creates a horizontal line in an HTML page. It is normally used to separate contents. For example:

```
<p>text above the horizontal line</p>
<hr />
<p>text below the horizontal line</p>
```

HTML Links

Links or hyperlinks are pointers to the pages they reference and these are some of the most essential parts of the Web, since a large part of the Internet's success is due to its ability to link one webpage to other webpages.

There are two valid types of links in HTML, depending on the page your web browser is trying to access: external and internal. An **external link** is a link to a file or to a webpage that does not belong to the webpage's own site. An **internal link** is a link to a file or a webpage within the webpage's own site, or to any part of the webpage itself.

In addition, a **dead link** is a link to a webpage or file that does not exist. Clicking on this kind of link results in a webpage that displays such messages as "Error: File not found" or "The page could not be displayed."

Only one tag, the anchor tag, is used to describe links, and this tag has the <a> format.

External Link

The anchor tag has no meaning on its own. The anchor tag must include a **hypertext reference** or **href** attribute, which is a URL enclosed in quotes:

```
<a href="http://www.gov.ph">Republic of the Philippines</a>
```

In the browser, the absolute URL or the address itself (in this example, http://www.gov.ph) will not be displayed; instead, the anchor text (Republic of the Philippines) will be shown. Clicking on the anchor text link will transfer you to the URL itself, which in this case is the official government website of the Republic of the Philippines.

Internal Link

For internal links, you can use relative URLs, meaning that the browser will lead you to the URL of a link based on the current location of the HTML page or file inside the anchor opening tag. An example of this is:

```
<a href="project.html">this links to my project</a>
```

Both the webpage you are editing and the project.html file need to be located in the same directory or folder, so when someone clicks on the "this links to my project" hyperlink, the project. html file will be opened.

Internal links are very powerful because all of your webpages can be created on one computer and then tested to see if these are properly linked (i.e., no dead links). Once this testing is done, the file can then be uploaded to the web server or to another computer.

Links can be used to load not only .html files but also any file type you want, such as image files. To do this, include the name of the image file within the anchor opening tag:

```
<a href="pony.gif">In another window my pony is. See it you must.</a>
```

The display text in the browser will read: "*In another window my pony is. See it you must.*" When the user clicks on that link, it will then open the pony.gif file properly, unless the file is not located in the same directory as the webpage file itself.

You may want to organize your folders so that all image files are placed in the same directory and all .html files are kept in another. This kind of organization comes in handy when your website has many pages to support. To access your files properly, all you have to do is add extra text—the folder name and a forward slash before the filename—within the href attribute. For example, if pony.gif was placed in a sub-folder called Images, the HTML would be modified as follows:

```
<a href="Images/pony.gif">In another window my pony is. See it you must.</a>
```

Link Target: New Window

To make a link, open a page or a file in a new window, target="_new" must be added to the href attribute. For example:

```
<a href="pony.gif" target="_new">In another window my pony is.
                    See it you must.</a>
```

This example would launch the image in another browser window.

Linking to Other Parts of the Same Page

Navigating a very long webpage might become an exercise in patience as you scroll down it. To make things a little easier, you can place a link in a convenient place within the document that, when clicked, will take the screen to another section of the same document.

Start by naming the section, so that the link knows where to go once it is clicked. In the example below, assume that the text paragraph is at the very bottom of a very long text. The construction of a section name is shown in the next example.

```
<p>
      <a name="bottom" id="bottom">Yanomami Indians of the
Amazon</a>
</p>
<p>There are several ways to refer to the Yanomami Indians
of the Amazon Rainforest. Many people refer to them as the
Yanomami while others refer to them as Yanomama or Yanomano.
Either way, someone chooses to say their name, the word still
means wild, uncivilized people of the Amazon.
</p>
```

As a note, use both the name and id attributes within your section name for backward and forward compatibility to allow past, present, and future browser versions to understand your tags.

To access the paragraph from the beginning of the page, create a link there that will take you to the bottom of the page to read the requested information. This can be done as follows:

```
<a href="#bottom">More on the Yanomami Indians of the Amazon</a></p>
```

The browser will display "More on the Yanomami Indians of the Amazon," and clicking on the link will take you to the section. Note the hash (#) sign within the attributes; this means that the link is an anchor within the page, and not a link to another HTML file.

E-mail Address as Link

Links can refer to an e-mail address by using the following syntax structure:

```
<a href="mailto:minari08@hotmail.com">e-mail Minari</a>
```

When your viewer clicks on the link "e-mail Minari," the default e-mail program (such as Outlook Express) will open on the computer and the viewer will be able to type and send an e-mail to the specified address.

NOTE

If you want text to appear in the "Subject:" field of the e-mail program, just add "?subject=whattheemailisabout" after the e-mail address. Example:

```
href="mailto:minari08@hotmail.com?subject=whatever"> E-mail Minari</a>
```

Image as Link

A hyperlink does not always need to be a text. It can actually be anything that can be clicked on, so even an image can be a link. An example of HTML code used for using an image as a link is as follows:

Note that the code should be inserted inside the anchor tags.

Clicking on the image file pony.gif on the screen will transfer the viewer to an external website, as the URL specified in the link is an absolute.

NOTE

A hyperlink not only opens a file or a page, it may also open a download dialog box. In this instance, the link is called a download link.

These are the basic HTML elements: heading, paragraph, non-breaking space, line break, boldface, italics, lists, preformatted text, and horizontal rule. The HTML list types are: unordered or bulleted, ordered or numbered, definition, and nested lists.

Tag Name	Syntax	Example
Heading	<h1> - <h6>	HEADING
Paragraph		This is a one-sentence paragraph.
Non-Breaking Space		a not-so-very l o n g word
Line Break	 	means non-breaking space. Actually, it is only a special character, not a tag.
Boldface	or	Strong as bold!
Italics	or	Emphasize or italicize?
Unordered List		Ice cream flavors I looove: - Rocky Road (the best!) - Kahlua Brownie (yum!) - Choco Mallows (mmm)
Ordered List		
List Item		
Definition List	<dl>	<dl> <dt>koala</dt> <dd>a tailless Australian animal with thick fur and big hairy ears, sharp claws for climbing, and a pouch like the kangaroo's for carrying its young</dd> <dt>kookaburra</dt> <dd>an Australian king-fisher that has a call resembling loud laughter</dd> </dl>
Definition Term	<dt>	
Definition Data	<dd>	

Preformatted Text	<pre>	<pre>
		[1. English language – Dictionaries] I.G.&C. Merriam Company.
		PE1628.5.W37 1981 423 81-9458
		AACR2
		</pre>

Horizontal Rule	<hr />	text above the line <hr /> text below the line

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. When and how do you use the line break tag?

   _____

2. Illustrate the FILO principle.

   _____

3. Discuss the different kinds of lists.

   _____

4. What is the tag for creating paragraphs?

   _____

5. How many levels of headings does HTML support?

   _____

6. How is a non-breaking space represented in HTML?

   _____

7. To include an unordered list in a webpage, what HTML tag must be used?

   _____

8. To include an ordered list in a webpage, what HTML tag must be used?

   _____

9. To place an item in an unordered or ordered list, what HTML tag must be used?

   _____

10. What is a hyperlink?

    _____

11. What is the markup for opening a new window?

    _____

12. What is the tag for making an e-mail address link?

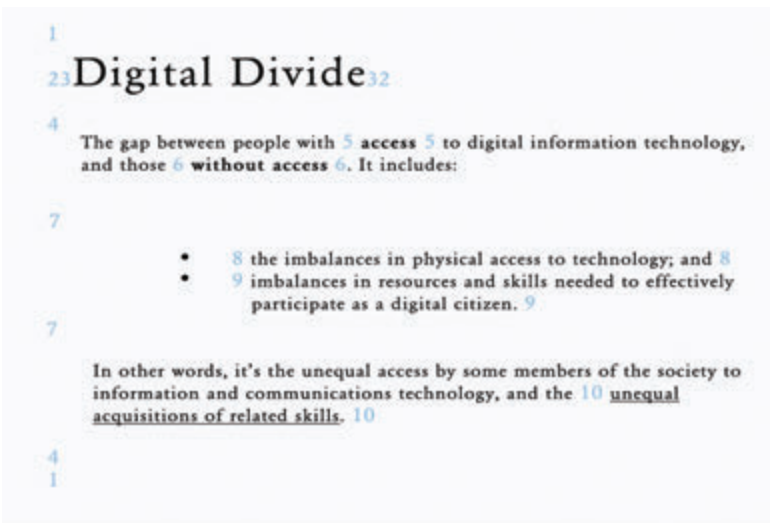
    _____

B. Match the HTML tags with the correct function. Write the letter of the correct answer on the blank.

1. <H1></H1> _____
2. _____
3.
 _____
4. _____
5. _____
6.   _____
7. _____
8. <H6></H6> _____
9. <dd></dd> _____
10. hypertext- x _____
11. <a> _____
12. # _____
13. download link x _____

a. defines the definition of a term in a definition list
b. inserts an item on a list
c. has the same effect as two <br /> tags
d. defines the largest header
e. has the same effect as tag
f. inserts a single line break
g. inserts a non-breaking space
h. has the same effect as tag
i. defines the smallest header
j. opens a download dialog box
k. text that leads to another text or information
l. when included in the href value, it
m. indicates that the link is an anchor
n. within the same page
o. anchor tag

C. The text below is an excerpt from Wikipedia which had been formatted using HTML tags. Can you guess what tags were used to render such format?



Digital Divide

The gap between people with access to digital information technology, and those without access. It includes:

- the imbalances in physical access to technology; and
- imbalances in resources and skills needed to effectively participate as a digital citizen.

In other words, it's the unequal access by some members of the society to information and communications technology, and the unequal acquisitions of related skills.

Write your answers below:

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____

LESSON 03

Planning Your Website

At the end of this lesson, the student will be able to:

1. **Identify the steps needed to plan for a website.**
2. **Determine the different sections appropriate for a website.**

Planning is an essential process in creating or doing almost everything—building a house, creating a business and so on. Planning saves a lot of time, money and effort. The worst mistake one could do when making a website is to go straight to a computer and start typing away to create the code. Chances are good that, when something new comes up, a lot of the code, or even all of it, may need to be re-written. To avoid this, here are some steps to make your website a success.

Website Planning

1. Creating a goal for your website

The purpose of the website should be clear to you. What do you want to achieve by creating the website? Is your goal to impress? To make a profit out of it? To amuse yourself? Write these goals down to keep you focused on what you want. Reading these written goals comes in handy in those times when you get confused and overwhelmed.

2. Identifying your target audience

More often than not, you will be creating a website for an audience, which is the group of people who are expected to visit your website, such as your friends, if it is a personal website, or potential customers if it is a commercial website. You need to think like your audience to identify their needs. Think of something that would make them want to visit your website again and stay longer. To help envision the possible audience of your site, ask yourself the following:

- audience characteristics – what will be the gender, the age range, the kind of work (or school) of the people whom you want to visit your website? Also, take into consideration their habits, cultural affiliation and interests. If the members of your audience are high school students, for example, it isn't likely they'll be interested in nursing homes for the aged.

- computer specifications – what will be the computer and Internet bandwidth speed of potential website audience? Graphics-heavy websites will take a long time to load on computers with slow processing and Internet bandwidth speeds.
- web experience – what is the computer proficiency of users? If your audience consists of elementary school kids who are just starting to learn how to use computers, making them navigate through hundreds of webpages would only frustrate them.

3. Determining content

After settling on your site's particular audience, think of incentives which will motivate them to come back and linger in your site. Here are some examples:

- For a website about clothing: to be up to date about current trends in fashion
- For a social site: to enable the users to meet cute and smart people

Take these things into account when deciding what functionalities and website sections will be used; more about this in the next section.

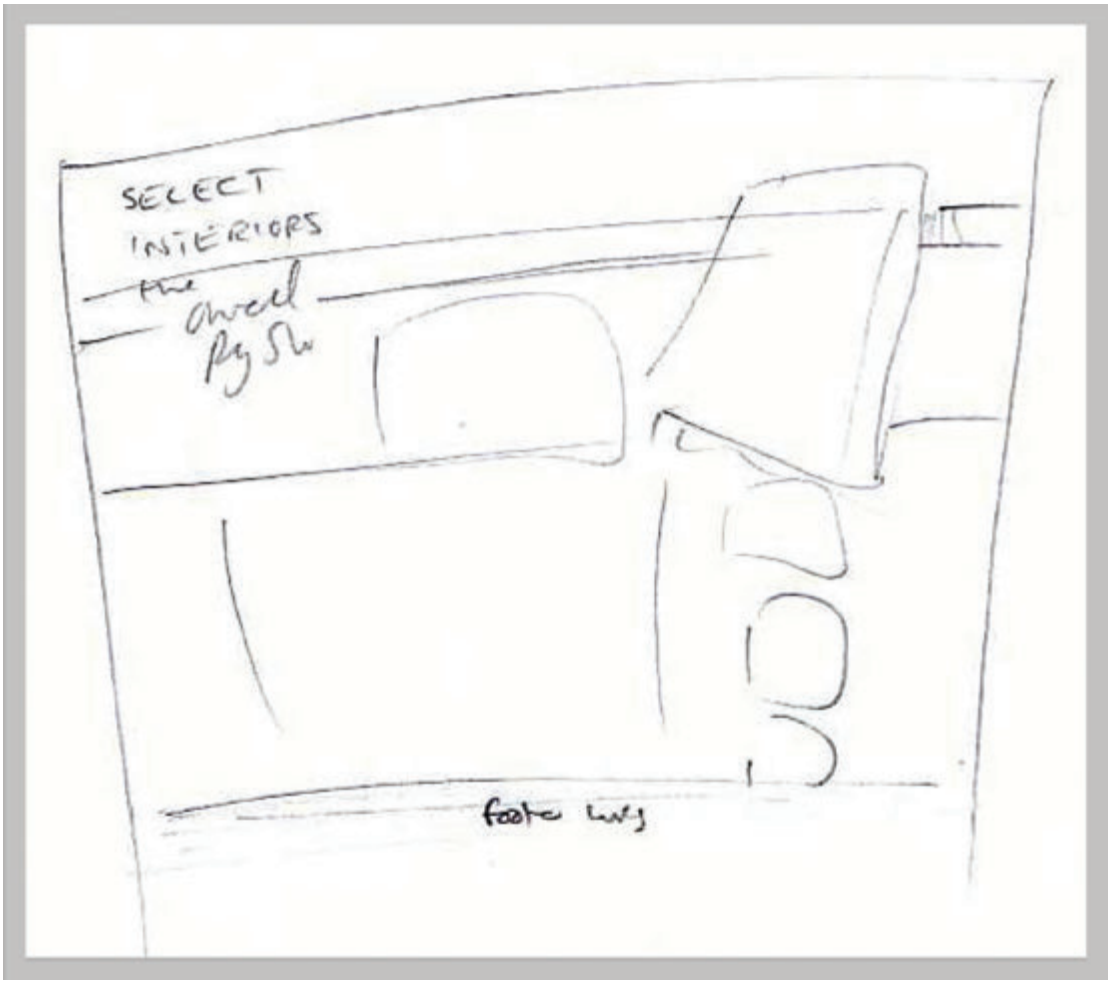
It is easy to get lost when choosing what content to include in your website, as there are many things you can do. Keeping your goals and target audience in mind will help you narrow down your choices significantly.

4. Visualizing the page flow

For this step, imagine that you are the user of the system. Think of how you would move from page to page, and think of how the system will react to every action. An example of page flow is:

5. Sketching the essential features of the look and feel of your website

This part requires a bit of imagination and design. Layout is how things will appear, and are arranged, in your page. The following sketch is an example of a pen and paper sketch derived from http://www.webdesignfromscratch.com/design-process/web-design-process.php.



The person who created this somewhat vague sketch apparently visualized the website name to go on the topmost part of the page and that the succeeding lines are to contain the page sections and the logo. The body or the content would be found in the middle of the page, and thumbnails of images would be found on the right side, while at the bottom of the page would be the footers.

Do not discount the use of pen and paper to visualize your page! When an idea comes to mind, chances are you will forget what it is when you don't jot it down right away, and doing so on pen and paper is a lot faster than opening up your computer. You can always recreate a cleaner version of your layout on your computer from your usually very rough pen and paper notes.

6. Planning site maintenance and update

It is best to design a website which is easy to maintain. This usually means automating certain features so that you don't need a lot of manual updates as time goes by. For example, if one of the features of your website is to show the present date, it would be absurd for you to be the one to update the date in the site every midnight. Instead, use or create a program to take care of this.

Future-proofing your website does not only entail automation, it also means being flexible for possible expansion. Recreating your site from ground up whenever a new idea comes in, for example, is sure to be a time waster. Your website must thus be easy to modify to take new ideas into account.

7. Creating your website

This is the part when you dirty your hands and start coding. HTML basics and other techniques in creating a website are discussed in the succeeding lessons.

8. Performing usability testing

Once the website is created, the next step is to have people test the site and to give their recommendations on how to make the site more user-friendly. Although a lot of people think this is a waste of time or is too expensive, large companies say otherwise. According to IBM, for every $1 spent for making your website easy to use, a return of $10 to $100 is made.

How would you conduct usability testing for your new website? Show five to ten people the finished product, and have them comment on the following areas:

- Do they see the whole point and purpose of the website?
- From the homepage, where would they click next? After that?
- Is it easy to navigate from page to page?

Website Sections

A website is a collection of webpages. When you start making your own site, the *home page* (i.e., your main page) must be saved as "index.html" or "default.html" regardless of the contents of the site. Browsers look for these files first whenever a web user types a URL without a specific filename.

Should you decide to make a striking entrance page, which is called a splash page, for your site, you can save this page as index.html, instead of your home page. The splash page usually consists of your site's logo or any image or illustration that best represents your site, and the links to your main pages. You can use different kinds of animation, slide shows—anything (as long as they don't take a long time to load) because the chief goal of the splash page is to entice web users and at the same time act as your website's title page.

Aside from the home page, there are other sections that your website can include. The following are examples of website sections of the techFactors site:

- Homepage
- About Us
- Learning Systems
- Technical Support

Website sections vary from website to website. There is really no hard or fast rule in grouping information to create website sections; it really all depends on how you would group topics into different sections. Here are other website sections.

Site Section

One of the website sections you can include is the "about the site" section, which essentially shows your reasons for making the site. Examples of these are:

- Explanations of what inspired you/required you/pushed you into producing the website.
- Story about the hard work you went through to create the website.
- State the significance of your site's title, logo, choice of background and images.

About Me/Contact Me Section

Your site can be about anything, and it could also have an "about me" section which gives information about yourself as the site's creator. A copyright section or a "contact me" section should be on every page of your site, just like this example:

```
<p>Copyright &copy; 2010
All Rights Reserved<br />
University of the Philippines<br />
<a href= "mailto:janice.aquinde@up.edu.
ph?SUBJECT=Clarifications">
Contact Me</a></p>
```

Site Map Section

Another important section that you might want to include in your website is the site map. This is a separate page consisting of all the links in your site—from your home page address up to every link to your picture in the photo gallery. This page doesn't need to have images because its main purpose is to direct the visitors to where they want to go. It is, after all, a map.

NOTE

Site maps are optional and are only required when your website is very large. Nevertheless, most corporate websites have removed their site maps altogether because of the dynamic nature of their contents.

Site Links Section

The site links section is not the same as the site map. Since your site is now on the Web, it is assumed that your site will not be an isolated work. Take advantage of what the Web has to offer by creating a separate page in your site that presents all the other sites you find interesting and/or which has material which could be related to your site. You can start by including your classmates' web addresses in your links page.

Guestbook Section

A guestbook is a webpage where your site's visitors can enter in their comments about your site or yourself. You can create the guestbook section yourself or avail of a free guestbook service. The good thing about creating the guestbook yourself is that you can make the design pattern of the guestbook consistent with the rest of your webpages.

On the other hand, if you avail of a free guestbook service, this will save you from creating a program to save comments received, but you have no control over data storage and retrieval. For instance, after you have registered for a free guestbook at http://www.bravenet.com/webtools/guestbook/index.php, the content administrators will notify you every time a visitor drops in and leaves a message in your guestbook. This way, you can reject or approve whatever the visitor said and even reply to that person. However, when a span of time lapses without anybody leaving a message in your guestbook, the host will close your guestbook and you will have no way of retrieving your visitors' entries.

One alternative is to create a guestbook page in your website and provide an e-mail link for the visitors who wish to have their messages posted on your guestbook page.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. What are the steps in planning for a website?

   _____

2. Give some of the characteristics of a potential audience you need.

   _____

3. At a minimum, what page is required for a website?

   _____

4. What section do audiences log in to and leave their remarks about the website?

   _____

5. What website section provides links to other websites?

   _____

6. What website section provides information about the website author?

   _____

7. What website section provides information about ideas on how the website is conceived?

   _____

8. What does usability mean?

   _____

9. Give some usability tips you have learned.?

   _____

10. What page provides a striking entrance?

    _____

B. Match the concepts with their correct function/description. Write the letter of the correct answer on the blank.

_____ 1. splash page
_____ 2. usability
_____ 3. index.html/default.html
_____ 4. site map
_____ 5. guestbook
_____ 6. about me/contact me
_____ 7. page flow
_____ 8. audience
_____ 9. planning
_____ 10. site link

a. visitors in the website
b. the sequence of pages as the user accesses the website
c. a measure of efficiency and elegance of a website
d. browsers will automatically bring you to this page when users enter only the domain name of the site
e. welcome or entrance page containing eye catching graphics
f. a process that saves the website owner time, money and effort
g. a website section that provides links to other websites
h. an outline of a website's content
i. visitors' online message log
j. a website section that provides information about the website author

C. Search the Internet for different websites, and list down five common website sections and the websites where you got these from. Compare notes with one of your classmates.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

LESSON 04

Experimenting with Fonts and Colors

At the end of this lesson, the student will be able to:

1. **Experiment with fonts.**
2. **Choose the background color that complements the text color and vice versa.**
3. **Input special characters and comments.**

One of the first things we notice is the exterior part of the things that we see—buildings, cars, especially people. It's pretty much the same with a webpage: people immediately notice the way you present it. This is why it is important to present your webpage in a manner that is pleasing to the eye. One of the ways you can do this is through the use Cascading Style Sheets.

Fashioning CSS

If HTML handles the contents of the webpage, then CSS is in charge of the presentation. HTML is about meaning, while CSS is about formatting—in other words, defining the characteristics of the design elements used in your HTML document.

The basic pattern for CSS commands is:

```
property: value
```

Where, for example, property is the feature of a paragraph you want to style, and value is the style itself, such as the height of one line in a paragraph. An example of CSS application inside HTML tags is this:

```
<span style="font-style: italic;">this
text is italicized</span>
```

In the example above, property is *font-style,* and the value is *italic*.

Types of CSS

There are three types of style sheets and they are arranged, or cascaded, according to superiority. This means that when the browsers read the HTML document of your webpage, they will follow the command of the most superior style sheet.

Here are the three types of style sheets.

1. In-line Style Sheet

   The first, most specific, and highest priority among the three different sheets is the in-line style sheet, which contains CSS commands embedded or placed inside HTML tags. Although this is the most superior of the three, this is also the least flexible.

2. Internal Style Sheet

   The second priority sheet is the internal style sheet, which resembles a summary of CSS commands located at the upper portion of the HTML document.

3. External Style Sheet

   The third and most highly recommended type of CSS sheet is the external style sheet, which lists all of the CSS commands in a separate document. This is a plain text file that can be written in any text editor and has the file extension name .css. Notepad is a good place to start when using the Windows operating system, although more specialized software applications called CSS editors also exist if you find text editors too basic for your needs.

External Style Sheet

See the color and font variation in the browser display? Try the following Notepad entries:

HTML Document
(save as "font.html")

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<link rel="stylesheet" type="text/css" href="external.css" />
<title>Stylissimo</title>
</head>
<body>
<h2>Self-Regeneration</h2>
<dl>
<dt><em>phoenix, phenix</em>     noun (Egyptian
mythology)</dt>
<dd>a bird of gorgeous plumage, sacred to the sun, reborn
from the ashes of the funeral pyre which it made for itself
when each life span of 500 or 600 years was over.</dd>
</dl>
<p>from <em>New Webster's Dictionary and Thesaurus</em></p>
</body>
</html>
```

External CSS File
(save as "external.css")

```
p {
  font-size: 80%;
}
h2 {
  font-family: Verdana;
}
em {
  color: red;
  font-weight: bolder;
}
```

To apply an external style sheet to your HTML document, use the `<link />` empty tag inside the `<head>` element, as in the above example. Through this, the browser can link the HTML document to your *external.css* file, read the CSS commands, and format the document according to the style information stored there.

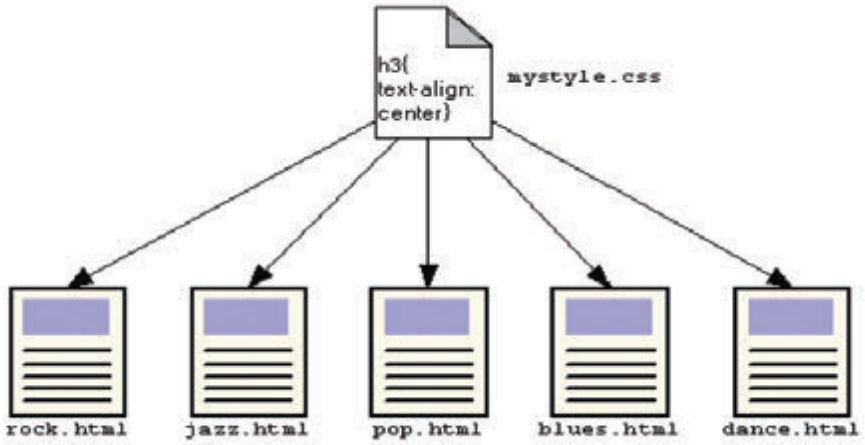
Let us look at what a page looks like without CSS, and compare how it differs from HTML with CSS. Your style sheet should only contain CSS style definitions. It should not contain any HTML tags.

HTML Document without CSS
(save as "stylissimo.html")

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Stylissimo</title>
</head>
<body>
<h2><font face="Verdana">Self-Regeneration</font></h2>
<dl>
   <dt><font color="red"><b><i>phoenix, phenix</i></b></font>
                noun (Egyptian mythology)</dt>
      <dd>a bird of gorgeous plumage, sacred to the sun, reborn
            from the ashes of the funeral pyre which it made
            for itself when each life span of 500 or 600 years
            was over.</dd>
</dl>
<p><font size="2em">from <font color="red"
size="2em"><b><i>New Webster's Dictionary and Thesaurus
</i></b></font></font></p>
</body>
</html>
```

This is the main idea behind style sheets: with an external style sheet you can change the look of an entire website by changing just one file. Indeed, an external style sheet is the best choice when the style is applied to *many* pages unless you want to retrace your markups, pinpoint which part of the textual or illustrated contents you want restyled, and do the changes one by one (which will take a lot of time).

Compare the HTML document above with the one that used an external style sheet to fashion each HTML element. The output is the same for the two webpages (the browser display at the start of this lesson). Which of them do you find easier to encode? You may not be able to answer this question yet, but when you have a lot of webpages to consider, an HTML document with CSS is easier to make and edit.



In the illustration, the mystyle.css style sheet center-aligns all the <h3> elements of your associated HTML files rock.html, jazz.html, pop.html, blues.html, and dance.html.

In the above example, it is the style sheet that offers consistency in the presentation of all the elements in your webpage, and not just the textual contents.

Deprecated Tags and Attributes

With CSS, font attributes tend to be obsolete, so the question that has to be asked is: What do we do with the "old" formatting tags and attributes (such as the *font* tag) that were in use until HTML 5.0? Do we drop them in favor of the new system?

The answer is, yes, because it was eventually decided that these formatting tags and attributes should be *deprecated* in favor of the style sheets. **Deprecated** means that, in a future version of XHTML, none of these tags will be accepted or supported by browsers.

Here's a list of some deprecated tags:

Tag	Description	Replacement
`<center>`	Centers text	`<p style="text-align:center">` (see text-align section)
`<u>`	Underlines text	text style sheets (see text-decoration section)
`<s>` or `<strike>`	Defines strikethrough text	text style sheets (see text-decoration section)
`<font>`	Identifies font characteristics	font style sheets (see font family section)

As a historical note, even before XHTML 1.0 was recommended by W3C on January 26, 2000. Even before that, style sheets were already used in previous HTML versions, as they had saved a lot of work for webmasters.

Style Sheet Syntax

The CSS syntax is made up of three parts: **selector**, **property**, and **value**. The whole structure looks like this:

```
selector { property: value }
```

The `selector` is the HTML tag you want to define while `property` corresponds to an attribute, with each property capable of taking on a `value`. The `property` and `value` are separated by a colon (`:`) and surrounded by curly brackets (`{}`). The structure "`property: value`" forms a **declaration**. For example:

```
p { font-family: arial }
```

If you want to specify more than one property, you must separate each property with a semicolon. For example:

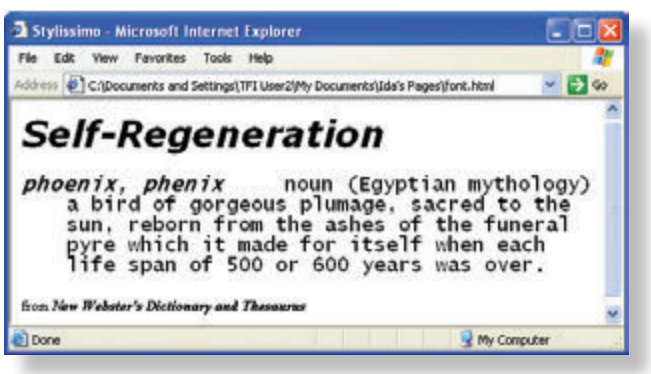
```
p { font-family: arial; font-style: italic }
```

To make the style definitions more readable, you can put each property on separate lines. It is also advisable to add a semicolon to the last entry for a tidier markup.

```
p {
 font-family: arial;
 font-style: italic;
}
```

The above specification means that your text will appear in italics in your browser using Arial typeface.

Font



To change the **typeface** of the characters in your page, you can apply the following `font` properties to the different selectors:

- `font-family` = the font that your text will be in. Be cautious of using fancy fonts, as not all web browsers can read all font types. On the other hand, know the common types such as Arial and Times New Roman—as these are the ones that the browsers remember to display as is. In specifying the `font-family` value, you may input more than one font type, as long as you separate them with commas. If the browser does not recognize the first one you specified, it will display the next one, and so on. If the browser does not recognize any of the font types, it will display your page's textual content in whatever default font type set in the user's browser. Also, remember to put font types with more than one name inside quotation marks, such as "`Monotype Corsiva`." Example:

```
h3 {
      font-family: "Lucida Console", Tahoma, arial;
}
```

- `font-size` = there are a number of ways to specify the font's size:

  1. Old 7 Size Font System – the values are:
     a. `xx-small`
     b. `x-small`
     c. `small`
     d. `medium`
     e. `large`
     f. `x-large`
     g. `xx-large`

     where `medium` is the default font size of the browser

  2. `larger` or `smaller` – relative to the Old 7 Size Font System
  3. `%` – percent relative to the default font size of the browser, e.g., `150%`
  4. `pt` – point size, the font-sizing system of Windows, e.g., `22pt`
  5. `em` – where the size of the font is multiplied by the value of the number, e.g., `3em` is three times the size of `1em`.

CONNECT

The standard size of fonts is measured by the `em` unit. It is based on the width of the capital letter "M." The default size of all webpage fonts is one (1) `em`. So if you say that the font is measured as `1.20em`, it translates as 120% of the default font size of the browser.

- `font-style` = possible values for this property are: `italic`, `normal`, and `oblique`, where `oblique` looks almost the same as `italic`.

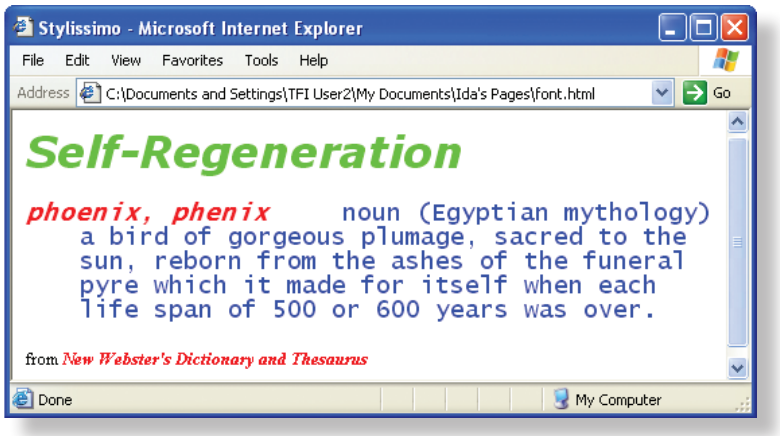
- `font-weight` = refers to how thick each letter will become. Its values are: `normal`, `lighter`, `bold`, and `bolder`. You may also use multiples of 100, where `100` is the lightest and `900` the boldest (`400` being the default).

While different ways of sizing fonts are used, it's best to stick to the font-sizing system that's easiest for you. To come up with the browser display illustrated at the beginning of this topic on `font` properties, change your *external.css* file to this:

```
p {
    font-size: 80%;
}
h2 {
    font-family: Verdana, "Dream of me", Garamond;
    font-size: 2em;
    font-style: oblique;
}
dl {
    font-family: "Lucida Console", Tahoma, arial;
    font-size: 110%;
}
em {
    font-weight: bolder;
}
```

Color

The `color` property refers to the color of the text. Its value can be any color, as long as it can be reproduced by the computer. Here's the catch though: the CSS rainbow is made up of **16,777,216** colors! Now how would you encode a single color?



CONNECT

Instead of seeing white light on the objects that receive the sun's rays that pass through some windowpanes, you can see a rainbow. This is because the glass in the windowpane acts as a prism, which splits the white light into its component colors. The primary colors of light are red, green, and blue, and the mixture of any of these three results in degrees of color. For example, if you mix red light and blue light you get violet light.

COLOR SPECTRUM

These are some of the ways to express `color` values:

- **By name.** There are 16 colors that can be encoded by name in CSS: `aqua`, `black`, `blue`, `fuchsia` (note the spelling), `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `purple`, `red`, `silver`, `teal`, `white`, and `yellow`. In addition, `transparent` can also be set as a color. For example:

```
p {
      color: black;
      font-size: 80%;
}
```

- **By the `rgb` value.** You type it in this format: `rgb(n,n,n)`
  where `n` is a number from **0** to 255.

Color	`rgb` value
Red	**r**gb(**255**,0,0)
Green	r**g**b(0,**255**,0)
Blue	rg**b**(0,0,**255**)
Black	rgb(0,0,0)
White	**rgb**(**255,255,255**)
and so on according to the different shades	

For example:

```
h2 {
      color: rgb(0,255,0);
      font-family: Verdana, "Dream of me", Garamond;
      font-size: 2em;
      font-style: oblique;
}
```

- **By the `rgb-hex` code.** `hex` means hexadecimal. It is composed of 16 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F (with 0 being the lowest and F being the highest). You can write the `rgb-hex` code in two ways:

Color	#RRGGBB	#RGB
Red	#ff0000	#f00
Green	#00ff00	#0f0
Blue	#0000ff	#00f
Black	#000000	#000
White	#ffffff	#fff
and so on according to the different shades		

For example:

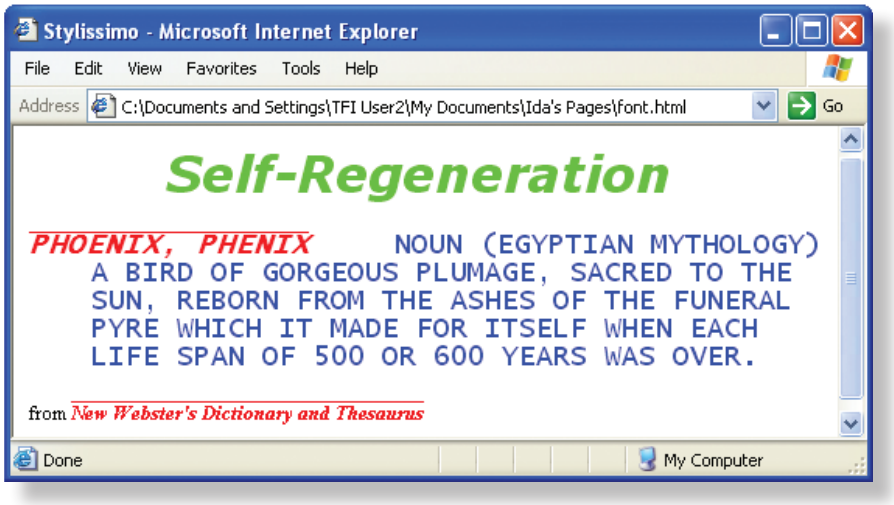
```
em {
      color: #f00;
      font-weight: bolder;
}
```

Do not forget the hash (#) sign for it denotes that the color value is in the `rgb-hex` format and not spelled out.

When using different ways to set color values, it would be best if you stick to the method of "color-mixing" that's easiest for you. To come up with the browser display shown before this discussion on `color` values, change your *external.css* file into this:

```
p {
   color: black;
   font-size: 80%;
}
h2 {
   color: rgb(0,255,0);
   font-family: Verdana, "Dream of me", Garamond;
   font-size: 2em;
   font-style: oblique;
}
dl {
   color: blue;
   font-family: "Lucida Console", Tahoma, arial;
   font-size: 110%;
}
em {
   color: #f00;
   font-weight: bolder;
}
```

Text



Together with `font` properties, you can control the appearance of the textual content through these `text` properties:

- `text-align` = whose values are: `left`, `right`, `center`, and `justify`.
- `text-decoration` = whose values are: `overline` (line above text), `line-through` (line through text), `underline` (use sparingly and if possible only on links), and `none` (usually to remove underlines from links).
- `text-transform` = whose values are: `capitalize` (to capitalize the first letter of every word), `uppercase` (to capitalize all the letters in a word), `lowercase` (to set capitals to lowercase for all the letters in a word), and `none`.

Look at the CSS code below and determine to which blocks the text properties are applied.

```
p {
  color: black;
  font-size: 80%;
}
h2 {
  text-align: center;
  color: rgb(0,255,0);
  font-family: Verdana, "Dream of me", Garamond;
  font-size: 2em;
  font-style: oblique;
}
dl {
  text-transform: uppercase;
  color: blue;
  font-family: "Lucida Console", Tahoma, arial;
```

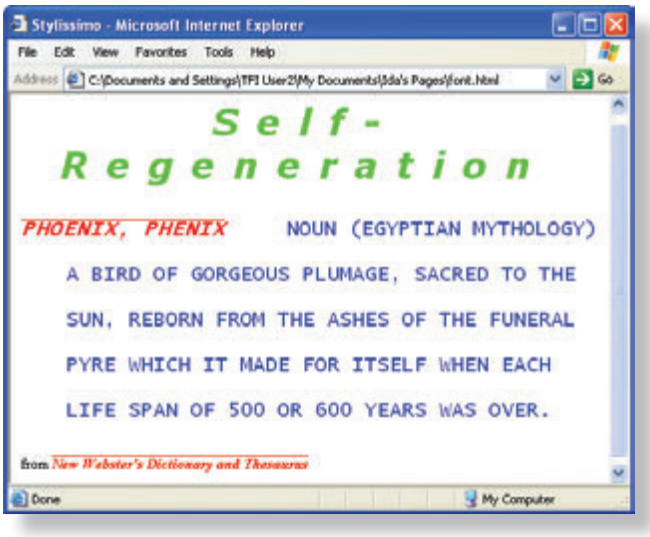
```
    font-size: 110%;
  }
  em {
    text-decoration: overline;
    color: #f00;
    font-weight: bolder;
  }
```

In addition, selectors may be styled with the use of the following properties:

- `letter-spacing` = refers to the space between letters, whose values are: `normal` or `_em` (indicating the length of space between letters, e.g., `5em`).
- `word-spacing` = refers to the space between words, whose values are: `normal` or `_em`.
- `line-height` = refers to the space between lines, whose values are: `normal`, `_%`, or `_em`. This adjusts the height of the line or lines of text (for example, in a paragraph).

To apply the above properties, change your *external.css* file into:

```
p {
      word-spacing: normal;
      color: black;
      font-size: 80%;
}
h2 {
      letter-spacing: 1em;
      text-align: center;
      color: rgb(0,255,0);
      font-family: Verdana, "Dream of me", Garamond;
      font-size: 2em;
      font-style: oblique;
}
dl {
      line-height: 230%;
      text-transform: uppercase;
      color: blue;
      font-family: "Lucida Console", Tahoma, arial;
      font-size: 110%;
}
em {
      text-decoration: overline;
      color: #f00;
      font-weight: bolder;
}
```



Special Characters

Most languages contain a number of special characters with diacritical marks that you may need to use, such as ñ, è, or ç. These are particularly important if you plan to display text in languages other than English.

Escape Sequences

There are also other characters with special meanings in HTML that cannot be used in a text just as they are. Such characters are, for instance, the left angle bracket (<) and the right angle bracket (>), which are used to indicate the beginning and the end of tags.

All escape sequences start with an ampersand (`&`) and end with a semicolon (;). They usually have two to six letters, and typically they resemble the characters they produce. For example, `&copy;` is the copyright escape sequence, which displays in the browser as ©.

The escape sequences of some of the most common special characters are:

Escape Sequence	Special Character	Description	Type
`<`	<	Less than	Symbols
`>`	>	Greater than	
`&copy;`	©	Copyright	
`&`	&	Ampersand	
`&quot;`	"	Double quotes (left hand and right hand)	

ñ	ñ	Lowercase n with tilde	Letters
è	è	Lowercase e with grave accent	
ç	ç	Lowercase c with cedilla	
â	â	Lowercase a with circumflex accent	
ü	ü	Lowercase u with umlaut	

From this table, you can see that the double quotes character has its own escape sequence, although it can be typed in directly from the keyboard.

NOTE

Escape sequences are case-sensitive. You cannot, for instance, use `&LT;` instead of `<`.

When there is no named value for a special character, you can use its numerical value instead. All characters have both a name and a numerical equivalent, such as: `&ccedil;` and `&#231;`, which will both display on the browser screen as "ç."

CONNECT

The standard numerical value of special characters is indicated in the ISO Latin 1 standard, which can be accessed at http://www.bbsinc.com/symbol.html.

Comment Tags

When you are writing an HTML document, it is very helpful to post reminders to yourself about what you are doing. You might need to refer to these comments, such as when you need to update a page you created long ago. You can also put down other useful information, like the name of the software and version you used to create a file, or even the date you last edited it.

NOTE

Comments will not appear in a web browser when the page is displayed. They will only be visible when the HTML code is viewed.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. What is an external style sheet?

_____

2. How do you encode the Copyright logo?

_____

3. Cite at least three font properties.

_____

4. What is the file name extension of an external style sheet?

_____

5. What font property is used to modify font color?

_____

6. What font property is used to modify font size?

_____

7. What font property is used to modify text alignment?

_____

8. What symbols are used to place comments in CSS documents?

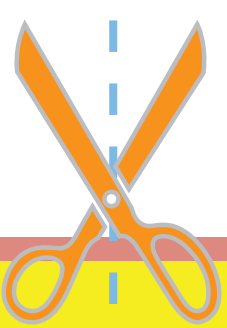
_____

9. Name at least 10 colors that can be encoded by name in CSS.

_____

10. What is the largest font size in the Old 7 Size Font System?

_____



B. Match the syntax/properties with their correct function. Write the letter of the correct answer on the blank.

_____	1. color	a.	aligns the text in an element
_____	2. text-align	b.	trademark symbol
_____	3. ™	c.	fixes or scrolls a background image with the rest of the page
_____	4. #ffffff	d.	CSS comment tag
_____	5. text-transform	e.	controls the letters in an element
_____	6. #000000	f.	white
_____	7. ©	g.	sets the color of a text
_____	8. <!—some text -->	h.	copyright symbol
_____	9. background-attachment	i.	black
_____	10. /*some text*/	j.	HTML comment tag

C. Match the CSS properties with their associated values. Write the letter of the correct answer on the blank.

_____	1. font-style	a.	normal, lighter, bold, bolder
_____	2. word-spacing	b.	Verdana, Trebuchet MS, Times New Roman
_____	3. text-decoration	c.	300%, 35pt, 5em
_____	4. font-weight	d.	capitalize, uppercase, lowercase, none
_____	5. line-height	e.	left, right, center, justify
_____	6. text-transform	f.	overline, strike-through, underline, none
_____	7. color	g.	italic, normal, oblique
_____	8. font-family	h.	normal, 2em
_____	9. font-size	i.	normal, 60%, 8em
_____	10. text-align	j.	red, rgb(200,80,165), #79f6ab

In HTML, you start the comment with <!-- and end the comment with -->. For example:

```
<!-- this is where you enter your comments -->
```

You must include the exclamation mark and the hyphens must be present on both sides.

In CSS documents, comments take the form:

```
/* comments in here please */
```

Style sheets define how HTML tags display in a browser. In this lesson, three types of CSS were presented: in-line, internal and external. In-line style sheet has CSS embedded inside the HTML tags. Internal has a summary of CSS tags located on top of the HTML page. External style sheet contains the CSS properties for all the HTML documents saved separately in another file with a .css file extension. The appearance of any text in a page may be manipulated through the different font properties, color values, and paragraph properties. Furthermore, there are other text entries that need to be encoded in a special way for them to accurately display in the page itself. These are called escape sequences or character entities. Lastly, you can enter comments in your HTML documents provided they are encoded as <!-- your comment -->.

LESSON 05

Working with the Box Model

At the end of this lesson, the student will be able to:

1. **Lay out pages using CSS properties.**
2. **Apply advanced HTML tags (span and div).**
3. **Adjust margins and padding.**

Any project that you plan to undertake might seem daunting at first sight. But more often than not, the tasks needed to complete the project can be made easier by using a system. For designing a website, the box model is a system that makes using HTML simple and easy.

Selected Style Sheet Topics

Recall CSS syntax being made up of three parts: selector, property, and value. The selector is the name of the HTML tag that you wish to style. There are benefits to grouping your *selectors*, or use *classes*, together in your internal or external style sheet documents. Not only does such grouping shorten the markups, it also makes your style definition look much more compact and organized. It is also through your selector classification that you can define different styles for the same HTML tag.

1. Grouping Selectors

Classifying selectors can speed things up. You may, for example, want some paragraphs of your HTML document to be aligned to the left and others aligned at the center. Formatting each paragraph manually, even with the use of the in-line style sheet, would take a lot of time. A quicker way to do this is as follows:

- Identify the classes inside the style sheet. The format for identifying or naming classes is .name. For example, you may decide to call the classes .one and .two as in the following example.

```
p.one { text-align: left; }
p.two { text-align: center; }
```

The first class will format a paragraph aligned to the left, and the second class formats paragraphs aligned at the center of your document.

- Group the selectors together if they share the same properties. Separate each individual selector by a comma. For example:

```
h1, h2, h3, h4 {
font-family: Arial;
font-style: italic;
}
```

This will make all h1, h2, h3, and h4 headings appear in italic Arial typeface.

2. Classifying Selectors

In HTML documents, content format can be controlled through the use of *class* attributes. An example of this is the following used inside the <p> opening tags:

```
<p class="one">This is left-aligned.</p>
<p class="two">This is centered.</p>
```

If we do this for all paragraphs, then we will have a way to change styles easily from the style sheet. For example, if you want the paragraphs that displayed text aligned to the left to be displayed the other way around (right aligned), all you have to do is change the CSS description from .one to .two:

```
p.one { text-align: right; }
p.two { text-align: center; }
```

Doing so will right-align all paragraphs having a `class="one"` in their tags.

NOTE

Only one class attribute can be specified per HTML tag.

Page Layout: Layout Table vs. CSS Positioning

HTML tables used to be the main tools in doing webpage layout. In truth, HTML tables were not designed as a formatting tool, but since HTML contents can be organized through borderless tables, HTML tables were used for laying out a webpage. One drawback of using tables for a layout is that the information in one particular cell does not necessarily imply a relationship with information in other cells, unlike in ordinary data tables.

CSS positioning, on the other hand, is not as intuitive as table layout and is difficult and time consuming to implement. CSS positioning, however, has long-run advantages which makes using it worth the trouble, thanks to the use of selectable, pre-defined *position* properties such as absolute, relative, static and fixed values; the *float* property, which has the values "left" and "right"; and the *clear* property, which has the values "left," "right," and "both." Choosing any element and specifying on which part of the webpage that element will appear using any property becomes possible.

NOTE

You may utilize the class selector for a more flexible identification system.

It is good to be proficient in both types, and the examples shown in the following pages are just two ways, among many, of encoding the same webpage. Observe the changes in the internal style sheet (from the original font.html in Lesson 4). The two HTML documents have the same output, which is the browser display at the start of this lesson.

Page Layout, Using Layout Table:

```
html><head><title>Layout Table</title>
<style type="text/css">
<!--
body {
background: url(phenix.jpg)
no-repeat bottom left;
color: #63c;
}
th {
color: #693;
font-family: verdana;
font-size: 2em;
```

```
text-align: center;
}
dl {
font-family: "Lucida Console";
text-align: right;
padding-left: 5em;
}p {
font-family: "Times New Roman";
font-size: 12pt;
text-align: right;
}
em {
color: #f63;
font-weight: bolder;
text-decoration: overline;
}
-->
</style></head>
<body><table>
<th>Self-Regeneration</th>
<tr><td> </td></tr>
<tr><td><dl>
<dt><em>phoenix, phenix</em><br />
noun (Egyptian mythology)</dt>
<dd> </dd>
<dd>a bird of gorgeous plumage, sacred to the sun,
reborn from the
ashes of the funeral pyre which it made for itself when
each life span of
500 or 600 years was over.</dd>
<br /><br />
<dd>something or someone seen as resembling this bird,
esp. with
respect to its power of self-regeneration.</dd>
</dl></td></tr>
<tr><td> </td></tr>
<tr><td><p>from <em>New Webster's Dictionary and
Thesaurus</em></p>
</td>
</tr>
</table></body>
</html>
```

Page Layout, CSS Positioning Method:

```
<html><head><title>Page Layout</title>
<style type="text/css">
<!--
body { background: url(phenix.jpg)
no-repeat bottom left;
color: #63c; }
h2 { color: #693;
font-family: verdana;
font-size: 2em;
text-align: center; }
dl { font-family: "Lucida Console";
text-align: right;
padding-left: 5em; }
em { color: #f63;
font-weight: bolder;
text-decoration: overline; }
.up { position: relative;
top: 0; }
.down { position: absolute;
bottom: 3em;
right: 0; }
-->
</style></head>
<body>
<h2 class="up">Self-Regeneration</h2>
<dl><dt><em>phoenix, phenix</em><br />
noun (Egyptian mythology)</dt>
<dd> </dd>
<dd>a bird of gorgeous plumage, sacred to the sun, reborn
from the
ashes of the funeral pyre which it made for itself when
each life span
of 500 or 600 years was over.</dd>
<br /><br />
<dd>something or someone seen as resembling this bird, esp.
with
respect to its power of self-regeneration.</dd>
</dl>
<p  class="down">from  <em>New  Webster's  Dictionary  and
Thesaurus</em>
</p>
</body>
</html>
```

CSS POSITIONING PROPERTIES

Description	Properties	Example
`absolute` – The element stays in one place. Even if you resize the browser window, it will remain in the part of the page you assigned it to.	`left`, `top`, `right`, and `bottom`	`.abs { position: absolute; bottom: 4em; right: 0; }`
`relative` – Its position depends on other elements. Even if you resize the browser window, it will not overlap into other elements (unlike the `absolute` value).	`left`, `top`, `right`, and `bottom`	`.rel { position: relative; top: 0; }`
`static` – The default position. The element would be in this position if you did not specify any value.	–	`hr { position: static; }`
`fixed` – The element stays in place even if you scroll up or down, left or right. Its position depends on the browser window and not on the page itself (unlike the `absolute` value).	`left`, `top`, `right`, and `bottom`	`pre { position: fixed; left: 0; }`
`float` – An element's position is either to the left or right of another element, with all the other contents enclosing it.	`left` and `right`	`.left { float: left; }`
`clear` – Ensures that there is no content around the element (unlike the `float` value).	`left`, `right`, and `both`	`.right { clear: right; }`

NOTE

Assign values in pixel or `em` units – they correspond to the part of the page (left, top, right, or bottom) the element is distanced from or to, and how much of a gap is maintained between the edge of the page and the element.

Span and Div Tags

Headings, paragraphs, lists, pre-formatted text, and tables are not the only sections a webpage can have. Other logical divisions can likewise be created in your HTML document by using the `span` and `div` tags.

The `span` tag defines the style of any in-line element and looks something like the following example:

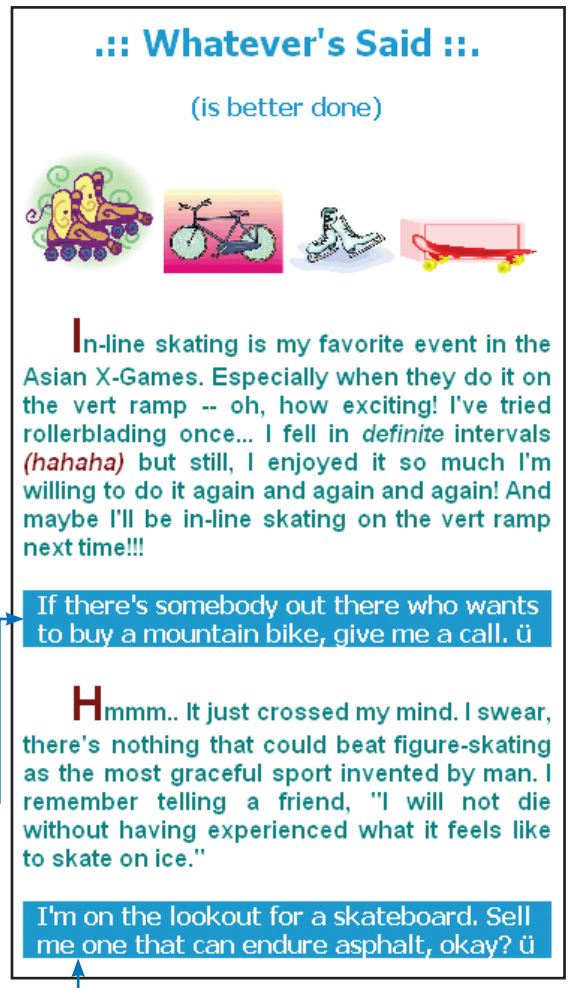
```
<span style="font-style: italic;">this text is italicized</span>
```

Let's say you want to change the font and color of the span tag, `this text is italicized`. Your problem is that it's only a part of a paragraph. If you don't know yet how to style texts within a sentence located inside the `<p>` element, your only option would be to use any or both of the boldface and italics tags. A more elegant solution is to use the `span` tag to style the font and color of any part of your paragraph any way you want, following the pattern above. Here's another example:

```
<html>
<head><title>In-line</title></head>
<body>
<img src="blades.jpg" width="90" />
<img src="bike.jpg" width="80" />
<img src="skates.jpg" width="70" />
<img src="board.jpg" width="100" />
<p style="color: teal; font-family: arial; font-size: small; font-weight: bold; text-align: justify; text-indent: 2em;">
  <span style="color: maroon; font-size: 2em;">I</span>n-line skating is my favorite event in the Asian X-Games. Especially when they do it on the vert ramp -- oh, how exciting! I've tried rollerblading once... I fell in <i>definite</i> intervals
  <span style="color: maroon; font-style: oblique;">(hahaha)</span>
  but still, I enjoyed it so much I'm willing to do it again and again and again! And maybe I'll be in-line skating on the vert ramp next time!!!</p>
</body>
</html>
```

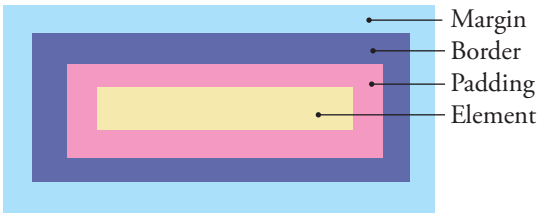
Where `span` is for **in-line** elements, `div` is for **block** elements so line breaks can be created. What you've learned about grouping and classifying selectors can be applied to either of these.

```
<html><head><title>In-line</title>
<style type="text/css"><!--
h2, h4 { color: #09c;
       font-family: tahoma;
       text-align: center; }
p { color: teal;
       font-family: arial;
       font-size: small;
       font-weight: bold;
       text-align: justify;
       text-indent: 2em; }
.box { background: #09c;
       color: white;
       font-family: tahoma;
       font-weight: bold; }
--></style></head>
<body>
<h2>.:: Whatever's Said ::.</h2>
<h4>(is better done)</h4>
<img src="blades.jpg" width="90" />
<img src="bike.jpg" width="80" />
<img src="skates.jpg" width="70" />
<img src="board.jpg" width="100" />
<p><span style="color: maroon; font-size:
2em;"> I</span>n-line skating is my favorite
event in the Asian X-Games. Especially when
they do it on the vert ramp -- oh, how
exciting! I've tried
rollerblading once... I fell in <i>definite
</i> intervals <span style="color: maroon;
font-style: oblique;">(hahaha)</span> but
still, I enjoyed it so much I'm willing to
do it again and again and again! And maybe
I'll be in-line skating on the vert ramp
next time!!!</p>
<div class="box">  If there's somebody
out there who wants<br />   to buy
a mountain bike, give me a call. &uuml;
</div>
<p><span style="color: maroon; font-size:
2em;"> H</span>mmm.. It just crossed my
mind. I swear, there's nothing that could
beat figure-skating as the most graceful
sport invented by man. I remember telling
a friend, "I will not die without having
experienced what it feels like to skate on
ice."</p>
<div class="box">  I'm on the lookout
for a skateboard. Sell<br />  me one
that can endure asphalt, okay? &uuml;
</div>
</body></html>
```



The Box Model

The appearance of any element may be enhanced by styling its borders, margins, and padding. These three properties comprise **the box model**. In the box model, the element is enclosed by the padding, which is then enclosed by the borders, and which, in turn, are enclosed by the margins.




```
.inline { background-color: #fcf;
      border: #f96 dotted medium;
      margin-top: 1em;
      margin-bottom: 1em;
      padding: 0.5em; }
... <span class="inline">
<img src="blades.jpg" width="90" />
  
<img src="bike.jpg" width="80" />
  
<img src="skates.jpg" width="70" />
  
<img src="board.jpg" width="100" />
</span>...
```

Margin and Padding

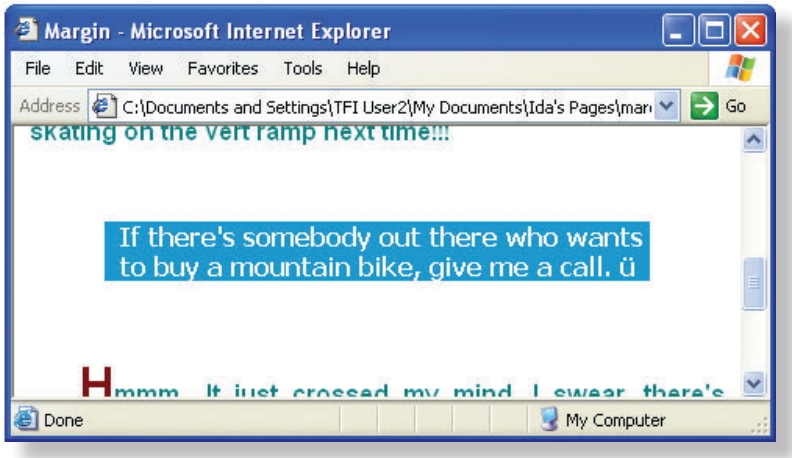
You can choose to combine the `border` properties with the `margin` and `padding` properties in designing your pages.

With the **`margin` property**, you can indicate the amount of space between the border of an element and the edge of your webpage, like the browser window display boundary through the `px`, `em`, or `%` units. You can also indicate the periphery of other elements in the same way, and if you wish to specify which sides you want to place margins in, you may give different values to the following `margin` properties: `margin-top`, `margin-bottom`, `margin-left`, and `margin-right`.

Add the `margin` property to the previous HTML document, like so:

```
.box { margin: 3em;
  background: #09c;
  color: white;
  font-family: tahoma;
  font-weight: bold; }
```

to come up with this output:



To create the opposite effect, just put a minus (-) sign before the unit (e.g., `margin-left: -3em;`).

CONNECT

First-line Indent

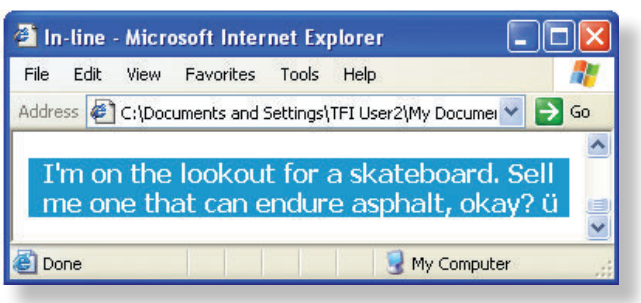
Sometimes you may want to indent the first line of each paragraph.The following style rule emulates the traditional way paragraphs are rendered in novels:

```
p { text-indent: 2em; margin-top: 0; margin-bottom: 0; }
```

It indents the first line of each paragraph by `2em` and suppresses the inter-paragraph spacing.

– from Dave Raggett's Introduction to CSS (http://www.w3.org/MarkUp/Guide/Style.html)

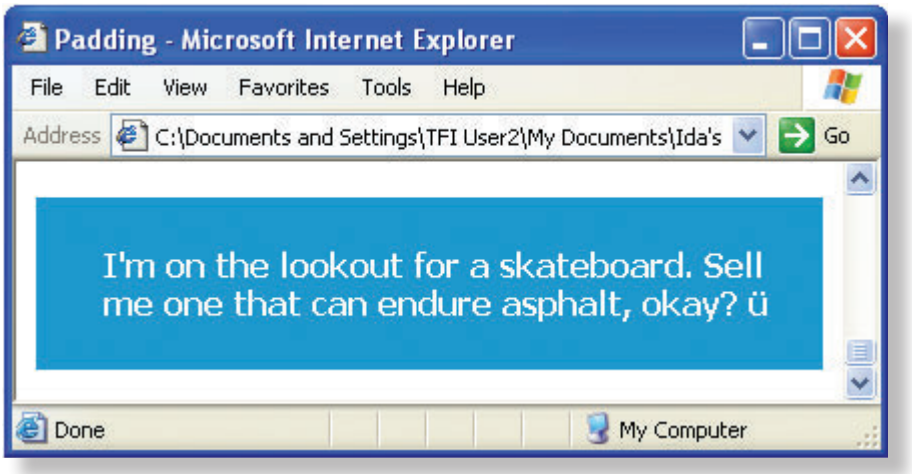
Using the **`padding` property**, you may indicate the amount of space between the border of the element and its contents through `px`, `em`, or `%` units. You may also give different values to the following `padding` sides: `padding-top`, `padding-bottom`, `padding-left`, and `padding-right`.



Add the `padding` property to the previous HTML document:

```
.box { padding: 1.5em;
  background: #09c;
  color: white;
  font-family: tahoma;
  font-weight: bold; }
```

to come up with this output:



NOTE

There is a great video tutorial found in YouTube, Box Model Web Page Layouts with CSS. Check it out: http://www.youtube.com/watch?v=2q2kmfgz9RY for more information.

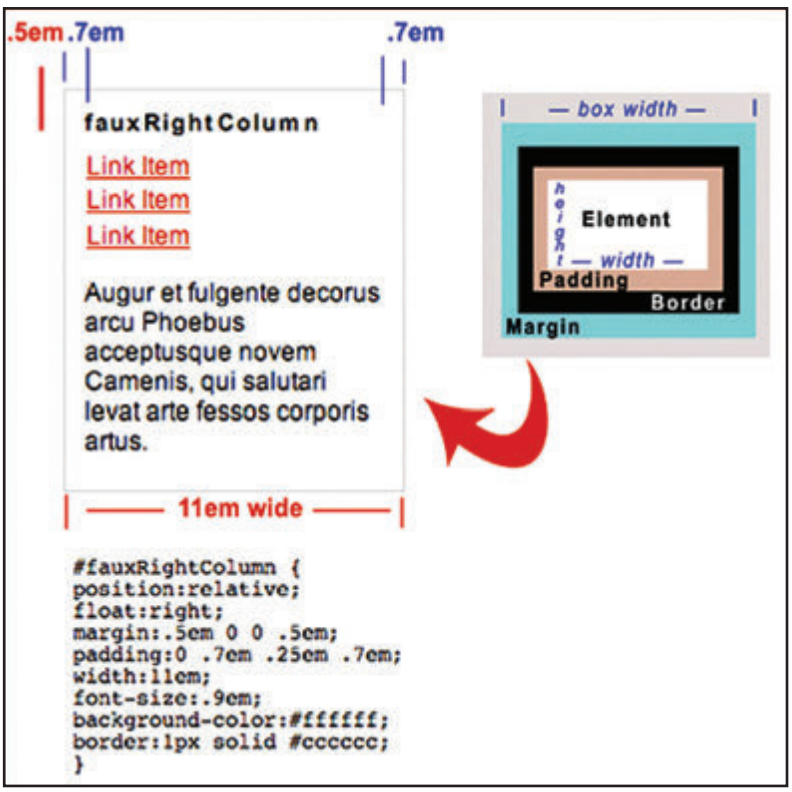
Page Layout Using Box Model

Most webpage layouts use the box model to arrange their website contents. The web template that we will now study is provided by the site http://csstinderbox.raykonline.com/. There are sites such as these which provide free CSS template. Do you see how the layout uses the box model?

Let us take a look at the box at the rightmost section of the page, *fauxRightColumn.* Checking style.css for the CSS code used for that section reveals the following code:

```
#fauxRightColumn {
position:relative;
float:right;
margin:.5em 0 0 .5em;
padding:0 .7em .25em .7em;
width:11em;
font-size:.9em;
background-color:#ffffff;
border:1px solid #cccccc;
}
```

Compare how the margin and padding correspond to the box model presented earlier.



* http://www.webhelpermagazine.com/2007/12/using-the-box-model-for-column-layouts/

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. Explain the box model concept.

_____
_____
_____
_____

2. What is the format for identifying or naming classes?

_____
_____
_____
_____

3. Enumerate the position, float, and clear properties and describe each.

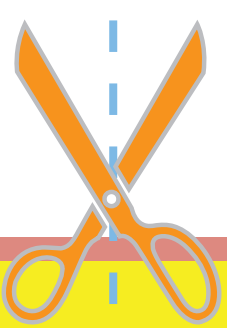
_____
_____
_____
_____

4. What HTML tag is used to define the style of any block element?

_____
_____
_____
_____

5. What HTML tag is used to define the style of any in-line element?

_____
_____
_____
_____



B. Test your CSS coding skills. Write the correct CSS syntax for each item below:

1. In-line CSS syntax for aligning a paragraph to the right

2. In-line CSS syntax for aligning an image called image.jpg to the right allowing text to flow around the image

3. In-line CSS syntax for placing an image called image.jpg in absolute position of 10 pixels to the left and 10 pixels to the top

4. Using in-line CSS, how do you place the image in No. 3 behind another element, like text?

5. Using in-line CSS, how do you place a red background color of size-30 pixels to the right, left, top, and bottom of an image called image.jpg?

6. Using internal or external CSS, how do you make all H1 headings display as underlined font?

You can save a lot of time if you group and classify selectors. Building pages is simpler using the positioning method of CSS. Other logical divisions in your HTML document can be created by using the `span` and `div` tags. The `margin` and `padding` properties can be applied in designing your pages using the box model.

LESSON 06

Experimenting with Multimedia

At the end of this lesson, the student will be able to:

1. **Manipulate text, images, borders, and background.**
2. **Alter the size of an image using the img tag attributes.**
3. **Embed animated GIFs.**

In this lesson, you will learn to make images and translate them into a format that you can upload digitally. So if you want to present information about butterflies, you just don't write the information without pictures attached. Visuals are important because they enhance information and make it much more interesting.

Multimedia

Multimedia is a combination of the different forms of media such as text, pictures, sounds, music, animation and video. Like other files or pages accessible through links, a multimedia file can also be just a mouse click away. Background music and film clips are examples of multimedia you can use for your pages. Although some multimedia files are small, it is common for other multimedia files (such as videos) to be huge in size. This needs to be taken into consideration when creating a webpage because long download times can frustrate users, especially if connections are slow.

Text and Images

There are three main image formats used in webpages: Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG), and Portable Network Graphics (PNG). You'll probably end up using a mix of GIF and JPEG images. A third format, PNG, combines the best of both.

- **JPEG** is a common format for images on the Web. It was designed as a standard for photographic images, allowing up to 16 million different colors in an image. To reduce its file size, JPEG is highly compressed for subtle changes in color within photos that the eye cannot perceive. The biggest limitation of JPEG images is that they cannot be animated or made transparent. The photos that you scan and upload to your computer are usually saved as JPEG files. The following is an example of a JPEG image:

JPEG

- **GIF** is one of the most common image types on the Web. This is a good format for illustrations, computer-generated images, cartoons, and images with large areas of solid colors. At the same time, GIF images can be transparent (images can appear to have non-rectangular boundaries) or animated (a short sequence of images within a single GIF file). The biggest limitation of GIF is that it supports only up to 256 different colors.

GIF

- **PNG** is basically a bitmapped image format for compressed color graphics and is now much used on the Internet after being a good option for graphic images like logos, text and photographs. It has the basic qualities of GIF while rendering graduated color images as effectively as JPEG, although it is not for professional graphics rendition. PNG files are comparatively light and meant to load easily over the Internet. Can you see any difference between the JPEG and PNG files that follow?

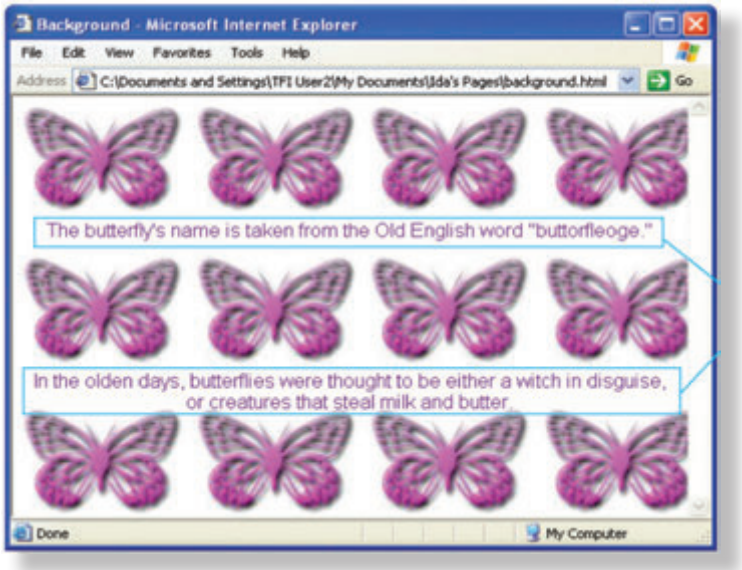
JPEG

PNG

Let's say you want to create a one page website on butterflies. It would be better to include pictures with the written information, since butterflies are quite visually varied. There are many ways to go about this, but for now we will be using our knowledge of internal style sheets, as this is most effective when a particular webpage has a unique style. This is also probably the easiest and fastest way to associate your page with a style sheet because you only have one document to consider.

If you expand your website to one which shows beautiful winged creatures, and not just butterflies, the options change, as your website will no longer be limited to a single page's design. Aim for creativity as well as for consistency throughout the website and all its webpages. For a guide to building effective websites, visit http://www.csus.edu/uccs/training/online/design/d_steps.htm* or read the Yale Style Manual Graphic Design 100 found at http://www.webstyleguide.com/page/index.html. For now, we will be using internal style sheets.

Image as Background



```
<!DOCTYPE html>
<html>
      <head>
            <title>Background</title>
            <style type="text/css">
            body {
                  background-image: url(flutterby.jpg);
                  color: purple;
                  font-family: arial;
                  text-align: center;
            }
            </style>
      </head>
      <body>
            <br /><br /><br /><br /><br />
            <p>The butterfly's name is taken from the Old
English word "buttorfleoge."</p><br /><br /><br /><br /><br/>
            <p>In the olden days, butterflies were thought to
be either a witch in disguise, or creatures that steal milk
and butter.</p>
      </body>
</html>
```

From the lesson about internal style sheets, the given CSS will set all the paragraphs of the document to being center-aligned, with the text font, Arial, and the text color, purple. The only unfamiliar property is *background-image.*

To avoid problems with old browsers that do not support style sheets, always put the content of the <style> tags within the HTML comment tags <!-- -->. Not doing so might result in the browsers displaying your style sheet markups to the viewer.

Background Values

The background of the page can be styled by using the following CSS properties:

Background-image. This is where you put the image source, which may be a relative or an absolute URL. The format is: url(image_filename.extension or subfolder/img.ext or http://www.site.com/image/img.ext).

Background-color. On the Web, most sites use white, black, and gray as background colors. In choosing the background color, make sure that this will not come into conflict with the text color—light backgrounds go perfect with dark texts and vice versa—so users will have no problem reading the page's content.

Background-position. If your image file is not big enough to occupy the whole page, you may indicate exactly where you wish to put it—top, bottom, center, left, right, or a combination of these (e.g., top left).

Background-repeat. You may tile your image file by using the value, repeat; repeat it at the topmost and bottommost part, using repeat-y (on the y-axis); repeat it side-by-side, using repeat-x (on the x-axis); not repeat it at all, using no-repeat.

Another way of encoding the background properties (and other properties as well) in the internal style sheet is through the shorthand method. Compare this:

```
<style type="text/css">
<!--
body {
background-color: #ffc;
background-image: url(tfi_logo_edited.gif);
background-position: bottom right;
background-repeat: no-repeat;
}
-->
```

with this:

```
</style>
<style type="text/css">
<!--
body {background: #ffc url(tfi_logo_edited.gif) bottom
right no-repeat; }
-->
</style>
```

The order in which the properties appear in the shorthand method does not matter. Notice how each property is translated. First, the word *background* is inserted, meaning that all the properties that follow are from the background family. Then the rest of the values are encoded directly.

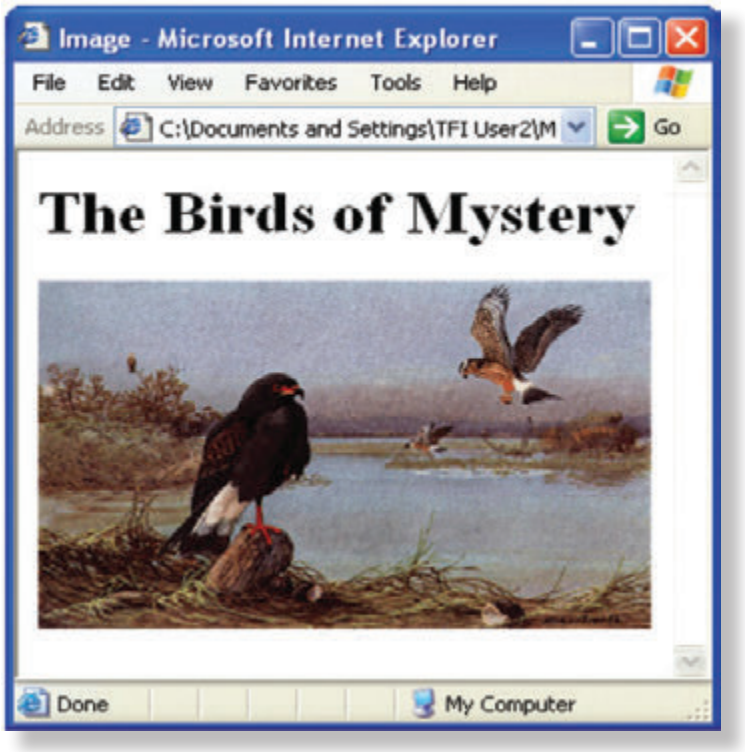
Both the longhand and the shorthand methods will display the following page.

Image as In-line Attachment

The *object* element needs to be added to display an image on a page without using it as the background. This element is used to add multimedia objects such as images, audio, videos, Java applets and Flash animations. For example:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Image<title>
</head

<body>
<h1>The Birds of Mystery</h1>
<img src="birds.gif" />
</body>
</html>
```

This code results in the following image:



In our code, the object tag has the following attributes:

- **type -** specifies the media type of the resource referenced by the *data* attribute
- **data** - the path or URL of the image you want to display on your page. The path points to the location where the image is stored. When encoded properly, this inserts an image called birds.gif in your document. The image will be displayed if it exists in the path or URL specified. By default, images are displayed in-line with the page's textual content.

NOTE

It is a good habit to keep your images together in one sub-directory that is separated from your HTML files. For example, if the folder's name is grafix, the tag should be modified this way:

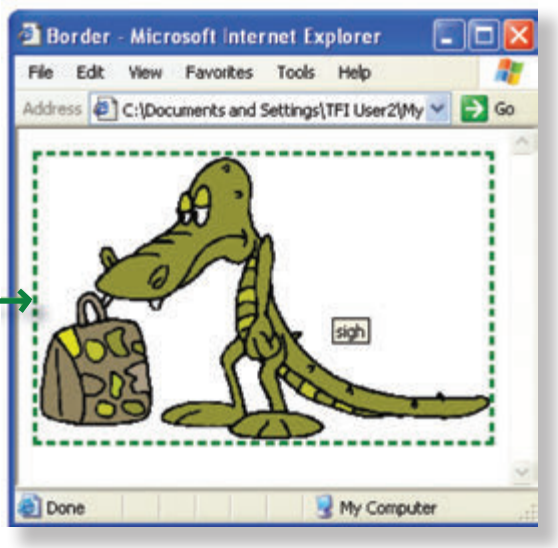
```
<object
     height="100%"
     width="100%"
     type="image/gif"
     data="grafix/birds.gif">
</object>
```

- **width and height** - the values of the width and height attributes specify the dimensions of the image. Although your images will still appear if you don't use these attributes, it is recommended that you include these so the browser will reserve this space when it loads your text, making the webpage load smoothly. Always check your pages first because altering the width and height attributes may distort the image.
- **alt -** this attribute will be used if the browser does not support the *object* tag.

Borders

There are many layouts that can be done with CSS. In addition to making background images, this can also be used to create borders in HTML.

```
<!DOCTYPE html>
<html>
   <head>
         <title>Border</title>
         <style type="text/css">
         <!--
         object {
         border-style: dashed;
         border-width: 3px;
         border-color: green;
         }

         -->
         </style>
   </head>
   <body>
         <object
               height="188"
         width="300"
               type="image/gif"
   data="croc.gif">
         </object>
   </body>
</html>
```



Some border properties are:

- **border-style** - you may choose among these values to control the appearance of the image's borders: dashed, dotted, double, solid, or none. You can even specify different values to the following border-style variations: border-top-style, border-bottom-style, border-left-style, and border-right-style.
- **border-width** - you may indicate the thickness or the width of the image's borders through these values: thin, medium, or thick; or through the desired width, which should be in pixels. You can also give different values to the following border-width variations: border-top-width, border-bottom-width, border-left-width, and border-right-width.
- **border-color** - you may assign any color or the value transparent to any of the following border-color variations: border-top-color, border-bottom-color, border-left-color, and border-right-color.

Audio

If you are going to use background sound or music in your webpage, think carefully about how well it goes with the content and whether or not it would be reasonable for the user to hear it every time he or she goes to view your page. Also, keep in mind that audio files may require plug-ins.

NOTE

Plug-ins are programs available on the Internet which augment the capabilities of certain software like the browser. By itself, your browser will not be able to play multimedia files, but it can through the use of a plug-in.

New browsers come installed with many plug-ins, but if an unfamiliar media type is encountered, a dialog box will usually pop up asking if you would like to download the plug-in.

Sound files can be large and load slowly, although the Musical Instrument Digital Interface (MIDI) format is an exception. MIDIs, which have the file extension .mid or .midi, are very small sound files, in contrast to the hugely popular MP3s. The downside of MIDI is that these cannot play sounds, only notes; in nonprofessional terms, MIDI cannot store songs, only tunes.

Other formats for audio files are:

Sound format	Extension	Description
Real Audio Format	.rm, .ram	This sound format was developed by Real Media for the Internet. It permits audio streaming with low bandwidths but the quality is reduced.
AU Format	.au	This sound format is supported by many software systems over a wide range of platforms.
Audio Interchange File Format	.aif, .aiff	This sound format was developed by Apple. The limitation of this file format is that it is not supported by all web browsers.
Sound Format	.snd	This sound format was developed by Apple. The limitation of this file format is that it is not supported by all web browsers.
Moving Pictures Experts Group	.mp3, .mpga	This sound format was originally developed for video but is now one of the most popular formats for music recordings. It combines good compression (small file size) and good quality.
Wave Format	.wav	This sound format was developed by IBM and Windows. It is supported by computers running on Windows and is fast gaining popularity and support in web browsers.

An object element can contain many *param* tags, which provide parameters for the object tag. The following is a example of its usage:

```
<object type="audio/x-wav" data="test.wav"
      width="200" height="20">
  <param name="src" value="data/test.wav">
  <param name="autostart" value="false">
  <param name="hidden" value="false">
  alt : <a href="test.wav">test.wav</a>
</object>
```

This code will load the test.wav file, as well as show a control panel on the page. In addition to object and param tags, notice that alt is used. Let us look at what each of the attributes does:

Param Tag

src - there are some browsers, like Internet Explorer, that need this attribute to understand where to look for the file.

autostart - if set to "false", the music will not start playing unless the user clicks the "Play" button in the control panel of the embedded media player. If the autoplay attribute is set to "true", however, the music will start playing once the page finishes loading.

hidden - if set to "false", the control panel will be visible. On the other hand, if the hidden attribute is set to "true", the control panel will not appear at all. Be careful when you do this, because if the user does not like the music and cannot stop it because there is no control panel, they might just close the browser and not return to your site! The control panel will also automatically take the dimensions specified by the browser (although the width and height attributes can resize this).

NOTE

The deprecated embed tag was used before the *object* tag was. The object tag is meant to replace the img and applet elements, as well as embed and bgsound; but since it is relatively new, not all browsers support it yet.

Videos

Video clips can be useful as online material because they can show us, among other things, situations in life beyond everyday experience, including those that are too dangerous for most of us to get into, such as landing on the Moon or bungee jumping. Videos can also be useful to display medical procedures or tutorials for software products.

Video files have the same main disadvantages as audio files. Although most new browsers come with video plug-ins pre-installed, they may also require other kinds of plug-ins once a video file is loaded. Another consideration is that video files are usually large in size, and take a long time to download.

Video format	Extension	Description
Audio Video Interleave	.avi	This video format was developed by Microsoft. It is supported by all computers running Windows.
Windows Media Format	.wmv	This video format was also developed by Microsoft. The disadvantage is that it cannot be played by non-Windows computers without an extra component installed.
Moving Pictures Experts Group	.mp3, .mpga	This video format is the most popular format on the Internet. It is cross-platform and is supported by most browsers.

Quicktime Format	.mov	This video format was developed by Apple. This format cannot be played on Windows machines unless you install an extra component.
Real Video Format	.rm, .ram	This video format was created by Real Media. It allows the streaming of video with low-bandwidth, but quality is compromised.
Shockwave (Flash)	.swf	This video format was created by Macromedia. This file format's player comes pre-installed in new versions of Internet Explorer.

Two situations of embedding videos in your HTML code will now be explored. One is embedding videos from YouTube, and the other one, embedding your own video.

Embedding YouTube Videos

This is the simpler of the two approaches, since you won't need to maintain the video and because the site already solves compatibility issues that come with embedding a video. To go around all the complicated codes, the tip is to just upload the video on YouTube, and then provide a link to that video on your site. The code snippet for that will look something like the following:

```
<object>
  <param name="movie"
    value="http://www.youtube.com/v/EBM854BTGL0&hl=en&fs=1&">
  </param>
  <param name="allowFullScreen" value="true">
  </param>
  <embed src="http://www.youtube.com/v/EBM854BTGL0&hl=en&fs=1"
    type="application/x-shockwave-flash" width="425" height="344">
  </embed>
</object>
```

What this code does is that it sets *movie* as the parameter name, and the *value* being the You Tube link to your video. It also has provisions for *allowFullScreen.* Video files require the same <embed /> tag that we have seen a while ago for audio files.

Embedding Your Own Videos

Embedding your own video is no longer recommended, but its syntax is very much the same as the attributes for sound, with the only difference being the values assigned to the attributes. It is also important to set the height and width of the player to the dimension in which the video was saved in as this affects video quality.

The body selector has the following background properties: `background-color`, `background-image`, `background-position`, and `background-repeat`. The image selector has the following border properties: `border-style`, `border-width`, and `border-color`. The two main image formats used in webpages are: GIF and JPEG. The object tag is used to express multimedia such as images, audio and videos in your code.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. Name a few disadvantages of using an image as a background for the page. Explain considering PNG.

   _____

   _____

2. What is the main difference between GIF and JPEG?

   _____

   _____

3. Can changing the width and height attributes distort an image? Explain.

   _____

   _____

4. What HTML tag/s is/are used to insert a style sheet in HTML documents?

   _____

   _____

5. What CSS property is used when placing background pictures in HTML documents?

   _____

   _____

6. What attribute specifies the image's source?

   _____

   _____

7. What attribute specifies the image's width?

   _____

   _____

8. What attribute specifies the image's height?

   _____

   _____

9. How many colors do GIF images support?

   _____

   _____

10. What HTML tag is used to place audio and video in webpages?

    _____

    _____

B. Match the syntax/properties/concepts with their correct description. You can draw a line between the matching items.

Words	Meaning
_____ 1. data	a. ideal format for web photos
_____ 2. GIF	b. attribute that defines the source of the multimedia
_____ 3. JPEG	c. defines the type of application the data attribute refers to
_____ 4. alt	d. defines if a background image will be repeated
_____ 5. background-repeat	e. format that's best for images with large portions of solid colors
_____ 6. background-image	f. a file format for audio file
_____ 7. type	g. used to support old browsers that do not support the object tag
_____ 8. object tag	h. inserts an image into the background of a webpage
_____ 9. midi	i. signals if an audio or video should start even without the user interference
_____ 10. autostart	j. tag used to express multimedia in your HTML code

C. Save a copy of your favorite webpage. View its source by clicking "View" then "Source" on your browser. Find all the image files (limit to only ten files) that are used by the page and list down their names, and the <img /> attributes that were used. Write your output in the space below. A sample answer is provided for your reference.

File Name	Attributes and Values
1. banner.gif	`Src="http://www.webhostph.com/images" sidth="750" height="300" alt="banner photo"`

LESSON 07

Processing Online Forms

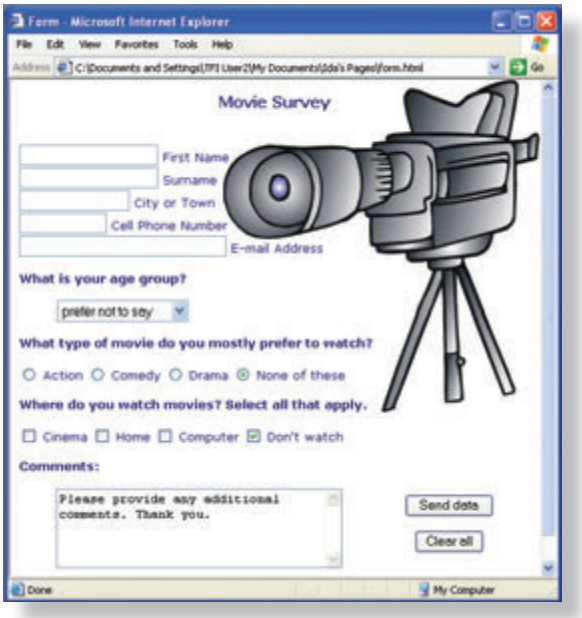
At the end of this lesson, the student will be able to:

1. **Examine forms, text area fields, select lists, and other input types.**
2. **Adapt the above to the different demands of the users of the page.**
3. **Process online forms.**

Sometimes, a decision has to be made by several people. During elections for class president, for example, we raise our hands or submit vote ballots for a certain student, which the teacher then counts. Similarly, using forms gives you the opportunity to collect information and survey public opinion on certain matters.

Online Forms

Online forms are webpages purposely designed for gathering information on the Internet. These HTML documents are sent back to the server once the user submits them. The main use of online forms is to gather feedback or opinion from the users for data processing, usually through a database.



We will build the Movie Survey form step-by-step as we go along the lesson.

NOTE

Using forms leads to effective communication between the user and the webmaster.

Content

Think about the information you will use, and the way you will present this information, when you develop a form. The form should be both visually appealing and functional, which means that you may need to take care when choosing which key points to include in a form.

Start building an online form by placing the `<form> </form>` tags inside the HTML body tags. You can start organizing the form by placing the `input` empty tag inside the `<form>` set of tags and have its attribute `type` specify the kind of input you are going to have. Another attribute, `name`, distinguishes one input field from another.

Input Fields

Let's say you want to create a form that asks people to submit their preferences on movies. To capture this information, input fields are needed within your HTML document.

Text

Text fields would be used when the information that you want to capture are people's names and addresses, for example.

CONNECT

Above all else, it is your job as the webmaster to assure the visitors that all information they enter in your online form will be confidential—in other words, that the personal contact details that they give will not make them vulnerable to such things as spamming (which is sending unsolicited e-mail to computer users linked to the Internet), identity theft, and credit card fraud. This needs to be stated within the form itself.

Use **text fields** when you want users to type letters or numbers in a form. The `type` attribute for these fields needs to be set to the value "`text`":

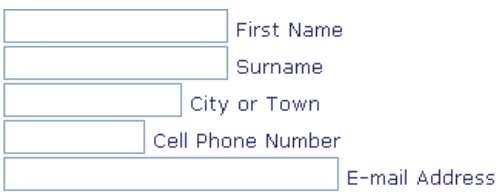
```
<form>
<input type="text" name="firstname" /> First Name<br />
<input type="text" name="lastname" /> Surname
</form>
```

NOTE

The web server, which is a computer that stores webpages, will not know from which text field the input came from without the **`name`** attribute.

To control the visible size of the text field, we use the `size` attribute. If the `size` attribute is not given, the default value as set by the browser is "`20`"—which stands for `20em`. Another important attribute is `maxlength`. This sets the maximum number of characters that can be typed into the field. This attribute is helpful where there are database length restrictions, or if we know the exact or maximum number of characters the user must provide. In the example below, `maxlength="11"` means that we are asking the user to specify a mobile number that must be up to 11 digits long.

```
<form>
<input type="text" name="firstname" /> First name<br />
<input type="text" name="lastname" /> Surname<br />
<input type="text" name="location" size="15" /> City or Town<br />
<input type="text" name="phone" size="11" maxlength="11" /> Cell Phone
Number<br />
<input type="text" name="email" size="32" /> E-mail Address
</form>
```



Radio Buttons

Radio buttons are used when you want the user to select only one option from a list. The term comes from the shape of the buttons in a car radio, where you can choose only one button in order to listen to a particular station. The `type` attribute needs to be set to the value "`radio`". Meanwhile, do not forget the `name` attribute because if you do, the user can only use *one* of the many sets of radio buttons you may have in your page. The `value` attribute, which can be used to set a default value for the input text field, is also important. A radio button without a `value` becomes meaningless when submitted to the server because the server won't know what the button stands for.

```
<p><b> What type of movie do you mostly prefer to watch?</b></p>
<input type="radio" name="movietype" value="action" /> Action
<input type="radio" name="movietype" value="comedy" /> Comedy
<input type="radio" name="movietype" value="drama" /> Drama
<input type="radio" name="movietype" value="none" /> None of these
```

NOTE

To group a set of related radio buttons together, their `name` attribute should be *one and the same*.

On the browser, radio buttons look like small circles. Here, the `checked` attribute can be used to set the default selection if we want to indicate a pre-chosen selection. Thus if the above HTML code is set to display `None of these` as the default selection, the modified code and browser display will look like:

```
<input type="radio" name="movietype" value="none" checked /> None of these
```

Check Boxes

Check boxes are similar to radio buttons, but are used when you want to let the user select more than one option among a number of choices. The `type` attribute would be set to the value "`checkbox`".

```
<p><b>Where do you watch movies? Select all that apply.</b></p>
<input type="checkbox" name="cinema" value="cinema" /> Cinema
<input type="checkbox" name="home" value="home" /> Home
<input type="checkbox" name="computer" value="computer" /> Computer
<input type="checkbox" name="donotwatch" value="donotwatch" /> Don't watch
```

In the browser, check boxes appear as little squares. Adding the `checked` attribute to one of these buttons will mark that button with a check when the page loads. For example, if we choose the option "Don't watch" to be checked, the HTML code for that button would appear as follows:

```
<input type="checkbox" name="donotwatch" value="donotwatch" checked />
Don't watch
```

Text Areas

Text areas are text fields that have more than one line for longer text input. These are usually used to record comments or feedback. The tag for this is `textarea`, as in this example:

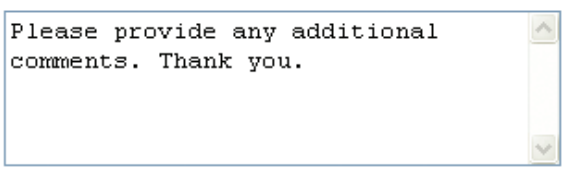
```
<textarea name="comments"></textarea>
```

You can specify the dimensions of the textbox with the `rows` and `cols` attributes, which stand for the number of rows and columns, respectively.

```
<textarea name="comments" rows="5" cols="35"></textarea>
```

There is also an option to include default text to appear in the field when the page loads.

```
<textarea rows="5" cols="35">Please provide
any additional comments. Thank you.
</textarea>
```



Select Fields

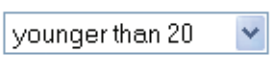
A **select field** provides drop-down menus that the user can use to choose information from a given list. Select fields are usually designed to look like a compact list of items, and they also take up very little space on the screen. You can create a select field with the `<select>` tag.

```
<p><b>What is your age group?</b></p>
<select name="age"> ... </select>
```

The select tag must have a `name` attribute to identify the checkbox, and this comes in handy when you have multiple checkboxes.

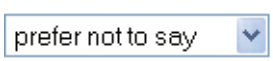
The `<option> </option>` tags define the text that is displayed in the menu.

```
<p><b>What is your age group?</b></p>
<select name="age">
<option value="teenager">younger than 20</option>
<option value="youngperson">between 21 and 35</option>
<option value="middleaged">older than 36</option>
<option value="noanswer">prefer not to say</option>
</select>
```



We can use a selected value to show which option will be displayed when the page loads. If this is not provided, the first item is selected by default, as seen above.

```
<option selected>prefer not to say</option>
```



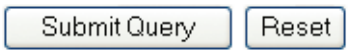
Submit Query and Reset Buttons

The 

button is essential, as it allows users to submit the information they typed in. The 

button, on the other hand, restores the form to the default state the viewers saw when they first loaded the page. The `input` tag is used here with the `type` attribute set to `submit` or `reset` for each button.

```
<input type="submit" />
<input type="reset" />
```



You can also change the name of the buttons to anything you want with the `value` attribute.

```
<input type="submit" value="Send data" />
<input type="reset" value="Clear all" />
```

Processing Forms

To process the form, tags must have the `action` and `method` attributes, which tell the webmaster what the user entered in the form. The `action` attribute specifies the name of the field and should be unique within the form. The `method` attribute provides content associated with the `action` attribute. With these requirements taken into account, the `form` opening tag would look like this:

```
<form action="URL for a program in the server" method="get">
                                or
<form action="URL for a program in the server" method="post">
```

The `get` method acts like a hyperlink that sends all the form values into the URL. Let's say you want to use this URL from the Philippine Daily Inquirer:

http://news.inq7.net/breaking/index.php?index=1&story_id=45320

The values after ?index=1&story_id=45320 are actually form inputs where the method is set to `get`. If the method is set to `get`, the target page must have an action that will deal with the information contained there. (Unfortunately, you cannot do that with ordinary HTML.)

When the method is set to `post`, the information is sent to the server and then goes to

the linked URL. The easiest way to use the `post` method is to put an e-mail link in the action value, as in:

```
action="mailto:minari08@hotmail.com"
```

NOTE

When the method is `post`, data are sent to the server. When the method is `get`, they are sent to the URL.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Form</title>
<style type="text/css">
<!--
h3 {
   text-align: center;
}
body {
   color: navy;
   background-image: url(camera.jpg);
   background-position: top right;
   background-repeat: no-repeat;
   font-family: Verdana, arial;
   font-size: 10pt;
}
p {
   line-height: 0.5em;
}

.next {
   position: relative;
   margin-left: 3em;
}
.beside {
   float: left;
   margin-left: 1em;
}
.last {
   float: right;
text-align: center;
margin-right: 1em;
}
-->
</style>
</head>
```

```
<body>
<h3>Movie Survey</h3><br />
<form action="mailto:minari08@hotmail.com" method="post">
<input type="text" name="firstname" /> First Name<br />
<input type="text" name="lastname" /> Surname<br />
<input type="text" name="location" size="15" /> City or Town<br />
<input type="text" name="phone" size="11" maxlength="11" /> Cell Phone
Number<br />
<input type="text" name="email" size="32" /> E-mail Address
<p><b>What is your age group?</b></p>
<p class="next"><select name="age">
<option value="teenager">younger than 20</option>
<option value="youngperson">between 21 and 35</option>
<option value="middleaged">older than 36</option>
<option value="noanswer" selected>prefer not to say</option>
</select></p>
<p><b>What type of movie do you mostly prefer to watch?</b></p>
<input type="radio" name="movietype" value="action" /> Action
<input type="radio" name="movietype" value="comedy" /> Comedy
<input type="radio" name="movietype" value="drama" /> Drama
<input type="radio" name="movietype" value="none" checked /> None of these
<p><b>Where do you watch movies? Select all that apply.</b></p>
<input type="checkbox" name="cinema" value="cinema" /> Cinema
<input type="checkbox" name="home" value="home" /> Home
<input type="checkbox" name="computer" value="computer" /> Computer
<input type="checkbox" name="donotwatch" value="donotwatch" checked /> Don't
watch
<p><b>Comments:</b></p>
<p class="beside">
<textarea name="comments" rows="5" cols="35">Please provide any
additional comments. Thank you.</textarea></p>
<p class="last"><br />
<input type="submit" value="Send data" /><br /><br /><br />
<input type="reset" value="Clear all" />
</p>
</form>
</body>
</html>
```

The output for the HTML document is the Movie Survey form.

The main function of an HTML form is to gather feedback or input from the users for the purposes of processing data. Forms are organized using the `<input />` empty tag. Types of input fields are: text fields, radio buttons, check boxes, text areas, and select fields.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. What is the difference between the Submit Query and Reset buttons?

   _____

2. How do you create a drop-down menu in a webpage?

   _____

3. What are the similarities and differences between radio buttons and check boxes?

   _____

4. These are webpages designed to gather information from the site visitors.

   _____

5. This input field is used when asking for alphanumeric inputs.

   _____

6. This input field is used when the user is asked to select a single option from a list.

   _____

7. These input fields are used when users are allowed to select more than one option among several choices.

   _____

8. These are text fields that have more than one line for longer text input.

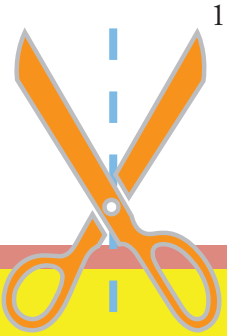
   _____

9. This input field provides a drop-down menu that the user can choose from.

   _____

10. This is an input type that produces a Submit Query button by default.

    _____



B. Match the concepts with their correct function/description. Write the letter of the correct answer on the blank.

_____ 1. online form
_____ 2. `type="text"`
_____ 3. web server
_____ 4. `type="radio"`
_____ 5. `type="checkbox"`
_____ 6. `textarea`
_____ 7. `select`
_____ 8. `type="submit"`
_____ 9. `type="reset"`
_____ 10. `action`

a. type of form that captures letters and numbers
b. computer that contains and puts webpages on the Web
c. resets the form to default values
d. a form attribute that indicates the script or program that would handle the form's contents
e. type of form that allows one selection from a number of choices
f. type of form that captures large amount of texts
g. submits the online form
h. type of form that provides a drop-down menu
i. provides a means for interaction between the web server and the user
j. type of form that allows multiple selections

C. Supply the required tags/code for each item below.

1. This is a form labeled "`First Name`" with the name "`fname`" and takes the first name of the user. The form should be size 20.

2. This is a radio button with three possible "fruit" selections: mango, banana, and apple.

3. This is a textarea labeled "`Message`" and named "`message`" with 40 columns and 10 rows. Include the following default message: "Type your message in this box."

4. This is a file upload facility with the `name="forUpload"`

5. This is a password field labeled "`Password`" with the name "`pwd`" and size 10. Set the maximum length of characters that can be entered to only 8.

LESSON 08

Making Dynamic Webpages

At the end of this lesson, the student will be able to:

1. **Add more life to their pages by following the markups provided in the lesson.**
2. **Explain the advantages of new technologies.**
3. **Apply JavaScript commands.**

You might get confused and overwhelmed by the sheer number of strange and wonderful things on the Internet that you can use in order to make your website more interesting. These interactive web tools allow you to make a unique web design that is totally different from any other, with your own imagination being the only limit.

Web Technologies

Besides GIFs, audio, and video, there are other technologies such as Flash and JavaScript, that can be used to improve your website.

Flash

Flash is a bandwidth-friendly, browser-independent vector graphic animation technology. Flash technology is currently the leading program for rich Internet content and applications. Compared to animated GIFs, Flash animation is smoother, loads faster because of smaller file size, and can also be interactive. There are whole sites which are written in Flash, rather than in HTML. To see a Flash animation, you need the necessary plug-in.

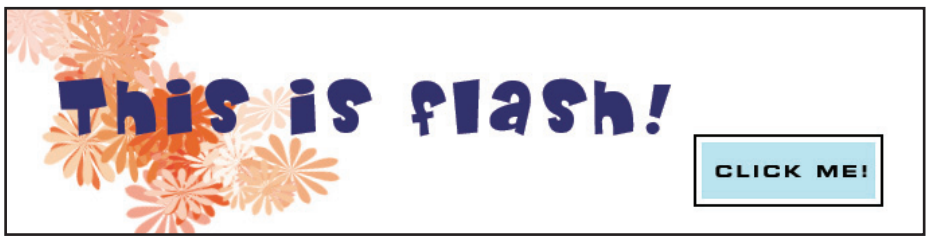
NOTE

While animated GIFs have been popular for some time, these are presently being replaced by Flash animation which has a lot more useful aspects, such as user interaction and faster loading times.

There are two types of Flash players. One version is that which needs ActiveX installed for Internet Explorer, and the other type is a Flash plug-in, which is what usually is needed by Mozilla Firefox. Also, some browsers will need you to manually install Flash. You can do this by going to the Adobe Flash site: http://get.adobe.com/flashplayer/thankyou/?installer=Flash_Player_10_for_Windows_-_Other_Browsers. After download and installation are completed, simply close your browser so the changes can take effect.

Here are individual frames of a sample application, the final output of which is an animation where flowers continue to change colors and fill up the frame while waiting for the web user to click the button. Like an audio file, this can be played over and over when properly encoded by the webmaster.

Frame 1



Frame 2

Frame 3

Frame 4

Frame 5

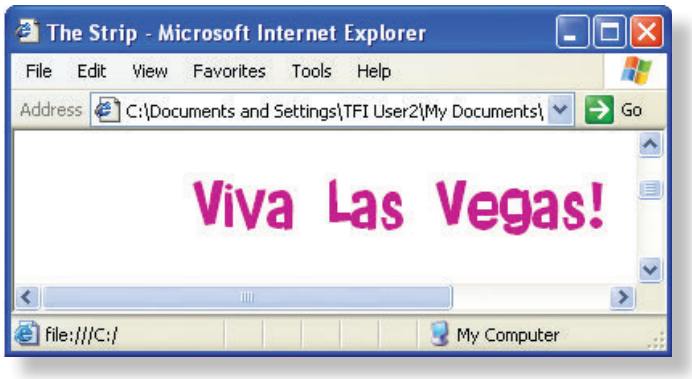
To take a look at some sample codes provided by Adobe, please visit their site at http://www.adobe.com/devnet/flash/samples/.

JavaScript

JavaScript is a scripting language designed for adding interactive features to HTML. The *scripts*, or JavaScript commands, are executed as soon as the browser opens an HTML document. Scripts are embedded directly in HTML files. Most browsers today support JavaScript, as it is one of the most popular languages for adding interactivity to HTML pages. The `<script>` tag is usually used to store JavaScript codes. Examine the HTML document that uses the `onMouseOut` and `onMouseOver` events below. This trick is also known as a **roll-over**.

```
<html>
    <head>
        <title>The Strip</title>
        <script type="text/javascript">
        if (document.images)
        {
            image1 = new Image;
            image2 = new Image;
            image1.data = "return.gif";
            image2.data = "returns.gif";
        }

        function chgImg(name, image)
        {
            if (document.images)
            {
                document[name].data = eval(image+".data");
            }
        }
        </script>
    </head>
    <body>
        <a href="www.thestrip.com" onMouseOver='chgImg("strip",
        "image2")'onMouseOut='chgImg("strip", "image1")'>
        <object type = "strip" type="image/gif" data="return.
        gif"></object>
    </body>

</html>
```

This is how the image link looks like before the cursor is pointed at it:



And this shows how the image link looks once the cursor lands on it:

Notice what happens when the mouse pointer is placed over the link. While the particular JavaScript code is outside the scope of this lesson, suffice to say that it is the code "chgImg" that handles how an image is displayed, depending on whether the mouse is hovered in or out of the text.

NOTE

Some newer browsers have security options that may "block" or prevent scripts like JavaScript from running. You may have to change the browser's security settings to accommodate such scripts.

In Internet Explorer, you can do this by clicking on Tools and selecting Internet Options. Click the Security tab, then click and drag the bar of the security level for the zone located at the lower left side of the dialog box to Low. Click OK to save the changes. In this setting, all scripts will be allowed to run. If the Security level bar does not appear, click the Default Level button.

Caution: Be warned that, in choosing the Low setting, your browser may allow harmful scripts to run too!

Web Tricks

Feel free to play with the codes in this lesson—tinker with them, input your own images, analyze the results—as you learn to make your webpages more dynamic.

Cycling Banner Ads

For these three banner ads to appear one after another in the same allotted space on the webpage, they have to be of the same size. The HTML and JavaScript documents for this trick are noted in the following table. 3000 milliseconds, or three seconds, is set as the period of time the user has to stand by before he or she sees the banner ads start the cycle. 5000 milliseconds is set as the period of time one banner ad is going to stay on the screen.

Webpage Notepad File
(saved as cycle.html)

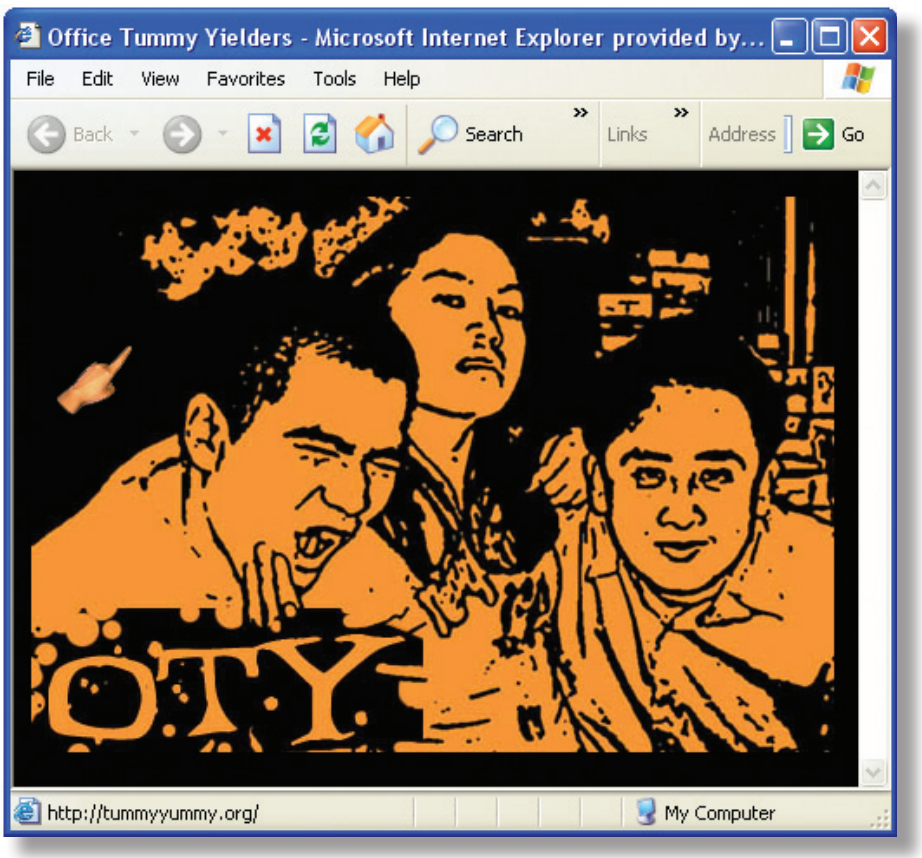
```
<html>
<head>
<title>cycling banner ads
</title>
<script type="text/javascript"
src="cycle.js">
</script>
</head>
<body>
<p><img src="Ida's Pages/
kingbikes.jpg" width="400"
height="100" border="0"
alt="Cycling Banner Ads"
name="rollimg" /></p>
</body>
</html>
```

JavaScript Notepad File
(saved as cycle.js)

```
var imgs = new Array(); var imgcnt
= 0; var thisimg = 0;
imgs[imgcnt++] = 'kingbikes.jpg';
imgs[imgcnt++] = 'ciena.jpg';
imgs[imgcnt++] = 'dollar.jpg';

function rotate() {
if (document.images) {
  thisimg++;
   if (thisimg >= imgcnt) thisimg
= 0;
        document.rollimg.src    =
imgs[thisimg];
  setTimeout("rotate();",5000);
}
}
setTimeout("rotate();",3000);
```

Status Bar Message

The status bar is the space on the bottommost part of the web browser. It's the blank field that shows the message "Done" when the website you accessed has finished loading. Also, when your mouse pointer goes over a **hotspot** (any clickable region in a webpage), the object's URL usually shows up in the status bar.



If your mouse pointer is not over any hotspot, the status bar displays nothing.

JavaScript command can let you type into the status bar. Simply follow the example (Please note that the code only works in Internet Explorer.):

```
<html><head><title>Status Bar Text</title></head>
<body>
...
<script type="text/javascript">
<!--
    var msg = "If I were you... holding the world right in my hands..."
    var delay = 100
    var startPos = 100
    var timerID = null

    var timerRunning = false
    var pos = 0

    StartScrolling()

function StartScrolling(){
    StopTheClock()
    for (var i = 0; i < startPos; i++) msg = " " + msg
    DoTheScroll()
}

function StopTheClock(){
    if(timerRunning)
        clearTimeout(timerID)
    timerRunning = false
}

function DoTheScroll(){
    if (pos < msg.length)
        self.status = msg.substring(pos, msg.length);
    else
        pos=-1;
    ++pos
    timerRunning = true
    timerID = self.setTimeout("DoTheScroll()", delay)
}
//-->
</script>
</body>
</html>
```

NOTE

The `script` tags can be put inside the `head` tags and/or the `body` tags. You can have as many JavaScript commands as you like within a webpage.

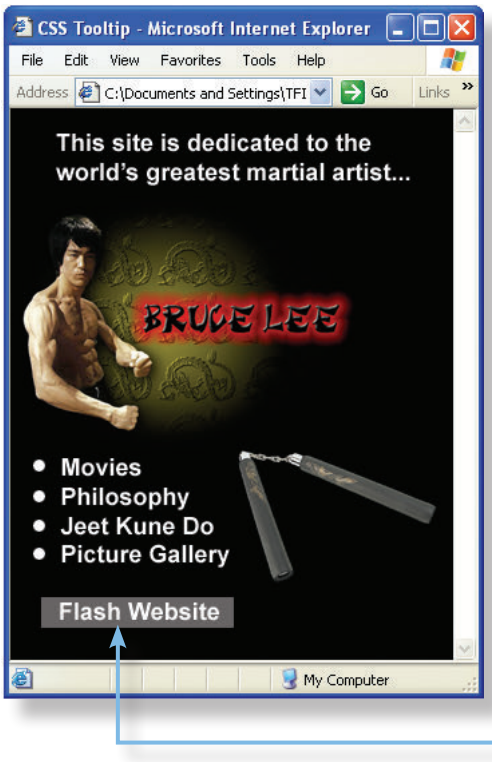
CSS Tooltip

The CSS Tooltip is another useful feature for web users. The tooltip features comes out when, for example, you point to a keyword in a webpage and information about it is immediately displayed beside your mouse pointer. To make CSS Tooltips, take note of the following markups and their accompanying screenshots.

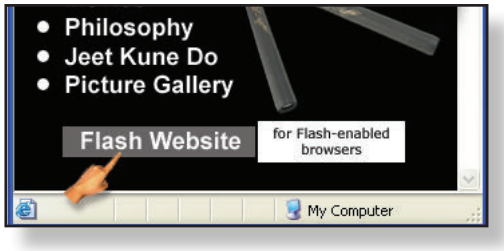
```
<html><head><title>CSS Tooltip</title>
<style type="text/css">
     <!--
     body { background: #000; color: white; font-family: "lucida sans";}
     h3 { margin-left: 1em; }
     img.1 { position: absolute; top: 5em; }
     img.2 { position: absolute; bottom: 2em; right: 2em; }
     ul { position: relative; top: 11em; line-height: 0.65em;
     margin-left: 1.8em; }
     a { color: white; font-weight: bolder; text-decoration: none; }
     -->
</style>

<!--start of CSS Tooltip code-->

<style>
     a.info {
          position: absolute; bottom: 1em;
          z-index: 24;
          background-color: #999; color: white;
          margin-left: 1.3em;
          padding-left: 0.55em; padding-right: 0.55em;
          padding-top: 0.09em; padding-bottom: 0.09em;
          text-decoration: none;
     }

     a.info:hover { z-index: 25; background-color: #ccc; }
     a.info span { display: none; }
     a.info:hover span {
          /*the span will display just on :hover state*/
          display: block;
          position: absolute;
          top: 0.25em; left: 10.5em; width: 11em;
          border: 1px solid black; padding: 0;
```

```
            background-color: white; color: #963;
            font-size: 0.75em; font-weight: light;
            text-align: center
        }
</style>
<!--end of CSS Tooltip code-->
```

Before you point your mouse to the Flash Website link:



As you point your mouse to the Flash Website link:



NOTE

There are lots of web tricks out there, and all you need to do to use a trick for yourself is to analyze the source codes of other sites to see how these tricks were done.

The two most popular multimedia technologies on the Web are Flash and JavaScript. Some of the features you can use in your webpages are: cycling banner ads, status bar messages, and CSS tooltips.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. Give instances when you can effectively use web multimedia technologies in your site.

   _____

2. What are plug-ins?

   _____

3. Manipulate the sample scripts and observe the changes in the outputs.

   _____

B. Match the concepts with their correct function/description. Write the letter of the correct answer on the blank.

____ 1. Flash	a. allows text to move across a webpage
____ 2. JavaScript	b. a very popular and powerful web scripting language
____ 3. scripts	c. scripting language for Flash
____ 4. onMouseOver	d. vector graphic animation technology
____ 5. onMouseOut	e. a set of computer instructions
____ 6. Actionscript	f. JavaScript event handler that defines what will happen when the mouse is placed over an element
____ 7. roll-over	g. JavaScript event handler that defines what will happen when the mouse is moved out of an element
____ 8. setTimeout	h. a JavaScript property used to call a function after a specified time, in milliseconds
____ 9. marquee	i. an aiding text written in CSS
____ 10. CSS tooltip	j. an effect that alternates images onMouseOver and onMouseOut

C. Supply the required information for each item below.

1. What is the basic syntax for embedding Flash animation (animation.swf) named "animation" into a webpage in high quality? The dimensions are 800 pixels wide and 200 pixels high.

2. JavaScript is a power-scripting language that lets webmasters add interactive and other cool applications on their webpages. Name five interactive/dynamic applications that can be created using JavaScript.
   1. _____
   2. _____
   3. _____
   4. _____
   5. _____

3. This is an inline JavaScript Code for placing the text "Go to Google" on the status bar when the mouse is placed over a link called "Search the Web" pointing to Google.com.

4. Do the same script as in No. 2 but this time, remove the text onMouseOut.

5. One JavaScript application that many web visitors would appreciate is the displaying of the current date on the page. Create the code for this in JavaScript.

LESSON 09

Other Web Matters

At the end of this lesson, the student will be able to:

1. **Upload files and submit the site to various search engines.**
2. **Maintain a web advertisement and a Guestbook.**
3. **Encode his/her name as author thru meta tags and know about IPR.**

Freedom of speech is one of the most important liberties that we enjoy today, and the Internet is a good place to practice this liberty. Because there is no single authority that can monitor the Web, people can express themselves to an audience that appreciates them. Maintaining a website and having an audience that views that website is an important interactive cycle that ensures that the Internet is a communication medium.

Meta Tags

"Meta" is a Greek term meaning "about" or "beyond." In HTML, we have tags that do not affect either the form or content of the webpage, although these tags have information about the webpage which the users can't access just by looking at the browser window display. These are called **`meta` tags**. They are empty tags located inside the `head` element and, like the comment tags, you won't see them anywhere on the webpage. Each `meta` tag must have a `content` attribute, which is usually preceded by the `name` attribute. There is a number of attribute values but these are the most common types: `author`, which is the webmaster's name; `description`, which is a brief explanation of what the webpage or website is about; `generator`, which is the text editor/s used by the author; and `keywords`, which are the important subjects on the webpage or website (Note that each metatag should be separated with a comma).

The information inside the `meta` tags, called **meta-data,** is used by search engines to identify what kind of information is within that webpage, so that, when someone searches for the same kind of information, a link to your webpage will be shown. This is the reason why you need to be specific and precise when stating these attribute values.

```
<head>
<meta name="author" content="Florida V. Ortiz" />
<meta name="description" content="I bid you welcome to my personal
site.  Godlike Flower.  Join me, in my Becoming..." />
<meta name="generator" content="Notepad" />
<meta name="keywords" content="godlike, ida, rida, florida, valencia,
ortiz, supremo, tasya, pilosopo tasyo, iska, iskolar ng bayan,
up student, philosophy major, filipino site, personal website,
articles, profile, pictures, online diary, ideal philosopher" />
<title>Florida's Home Page</title>
</head>
```

To view the markup commands of any page on the Web, click **View** on your browser's toolbar and select **Source or Page Source**. A window will then open the HTML document for that page, which is usually in Notepad.

Uploading Files

To instantly set up a website, you may avail yourself of a free web hosting service like Webs (http://www.webs.com), among others. You can also click the Easy Upload link from the Manage tab and start browsing for the files you need to upload.

Submitting Your Site for Crawling

Have you ever wondered how search engines come up with hundreds of addresses once you type something into the search field and click the 

, 

, or 

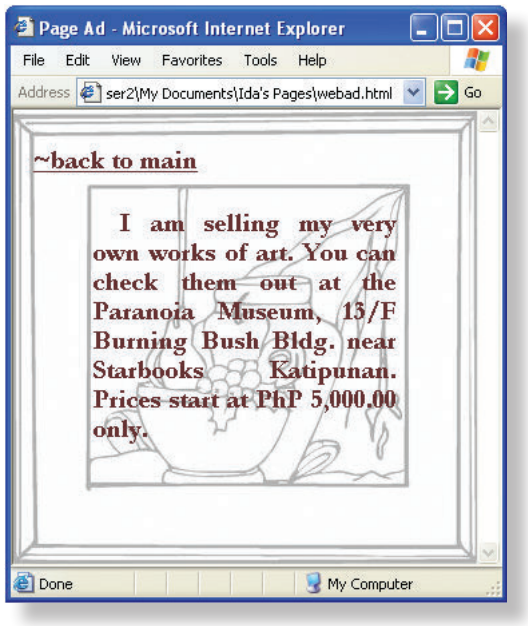
button? That's because search engines use spiders to track down the specific data that you're looking for. But these spiders are different from the eight-legged creatures we are all familiar with. These **web spiders**, or **web crawlers**, match your inquiry to any word in HTML documents all over the Web.

In order to be more visible to web spiders, you can use meta tags, or you can make them look at your website's contents by submitting your URL to any of the following: www.addme.com, www.ineedhits.com, www.freewebsubmission.com, www.yehey.com, and www.pugad.com. Just look for their submit your site free link and follow the instructions.

Web Advertisements

There are lots of things you can accomplish with your website, such as promote a good cause, plug an event, show your creative side, or sell a product. The last example is where web advertisement enters the picture.

A **web advertisement** may take the form of a banner or it may occupy a whole page. If you cut and pasted the ad from a website that distributes their banner ads for free, make sure that you properly link it to their site or input the URL as the value for the `longdesc` attribute. A web ad is not required for a website but if you want something more to add to your site, it is a rather good option. Take a look at this webpage ad:



According to www.en.wikipedia.org/wiki/Web_banner, the typical web banner ad is 468 x 60 pixels. Example:

AdSense

The first thing that usually comes to mind when the word "traffic" is mentioned is the flow of vehicles on a road. Internet traffic, however, is the flow of data to a certain website; and if you get enough traffic to your website, you could actually earn from it through Google AdSense.

The most popular search engine, Google.com, earns the majority of its revenue by allowing website owners and other companies to advertise on their search results pages. All of this is managed through a program called AdWords; and if you use AdSense to show the same ads on your site, you can share a portion of that revenue. This works well for the advertiser, as he gets more exposure, since his ads will be displayed in multiple websites (including yours); and the more that the advertiser's products or services are exhibited, the greater the chances that someone will visit his particular site.

Advertisers pay Google by the number of visits to their sites; the more the number of visits, the greater the amount they pay. And, in return, Google pays website owners who use AdSense to display AdWords advertiser ads a certain amount from the revenue that the advertisers pay Google.

How to Join AdSense

Given the ease with which money can be made from AdSense, many website owners have signed up for it—almost too many for Google to handle. In late 2008, Google toughened up the criteria for acceptance into AdSense, and these are that:

- Your domain must be your own top-level domain. A top-level domain is one wherein the domain name must be the likes of www.domain.com and not www.domain.com/mywebsite. Top-level domains are usually registered and paid-for domains. If your blog or website is hosted on a free web hosting site, then this doesn't count, unless the service is owned by Google, like Blogger.

- Your domain name must be active for at least 6 months. That is, there must be traffic and activity going on in your website in that time.

- The information provided in the AdSense registration must be the same as the domain name registered.

- Your website must contain a substantial amount of original content, which means that the majority of your website contents are unique. They can be your ideas, and not those which are merely copied from another site.

Once all these criteria are met, you are ready to apply. Go to http://adsense.google.com to fill up the necessary forms. Your website will be evaluated by Google, and if your application is approved, you can simply copy and paste the provided HTML code into the page where you would like the advertisements to appear.

Intellectual Property Rights (IPR)

Plagiarism, as defined by The Random House Dictionary for Writers and Readers (1990), is "the use without knowledge of the ideas or words or both of another writer, thereby passing them off as one's own."

When you conduct research, you must always cite your sources properly so that your teacher knows which parts you copied or reworded from others' works and which parts of your report are your original ideas. In some schools, the punishment for plagiarism is immediate expulsion. In many countries, encroaching upon other people's **intellectual property rights** is punishable by imprisonment and/or fine, depending on the circumstances. The legal term for this is "infringement of copyright laws." For more information on plagiarism, visit www.plagiarism.org.

Lots of information on the Internet are easily available, and while it can be tempting to put this information in your research paper without acknowledging your source, this is totally unethical. To learn how to cite sources, go to the library or to any website that discusses the different types of citation to credit your sources properly.

NOTE

In citing the sources, we usually use the American Psychological Association (APA) citation style. This is the citation guide used by standard scientific and technical publications.

The basic format of the APA citation is:

Author/s. (Date). *Title of Book/Article/Periodical/Volume* (Page Number/s). Place of Publication: Publisher.

For example, if we wanted to cite this book as a source, the APA citation would look like the following:

Kazanidis, Ortiz, Amante, De Sagun, Caro. (2012). Web Scripting. Manila, Philippines: TechFactors Inc.

Usability Tips

Wikipedia defines "usability" as "a term used to refer to the methods of measuring usability and the study of the principles behind an object's perceived efficiency or elegance." There are lots of usability tips, but here's a few that you might find useful:

1. **Cookie crumbs help in navigation.** Cookie crumbs help visitors specify in which part of the page or section they are in. The trail looks something like:

   Home > Web Design > Chapter 10 > Usability

   Usually, these bread crumbs are also links, so when you click on Web Scripting, for example, you will be brought to the list of all chapters.

2. **Arrange how elements appear in a page.** More often than not, people read a text from left to right. This means that more emphasis is given to elements that appear on the upper left side of the page, and less emphasis is given the further to the right and down you go.

3. **Provide enough contrast between the text and the background color.** It is best to use dark text against a light background.

4. **Put the "Submit" form button on the left, "Clear" on the right**. Most users are already used to the convention with forms that the Submit button is found on the bottom left of the page, while the Clear button is on the right of the Submit button.

5. **Retain form data.** This is particularly useful when users stray from the form for whatever reason if the form is only partially filled up. This way, they don't have to type everything in again when they return to the form.

6. **Make a custom "Not Found" Page.** When maintaining a website, it is inevitable that some errors will be encountered, like a broken link caused by a page being moved somewhere else, resulting in an error when the link is clicked. This is one example. In such a case, it would be very helpful if the error page could give the visitors an idea where the page may be found by presenting a search option, or perhaps give links to the most-visited pages in the website. Not Found pages are usually concise and straight to the point.

7. **Maintain a consistent look and feel throughout the webpages.** An elegant-looking website keeps the look and feel of the images and text all throughout the webpages. Refrain from creating wacky pages that are different from one another, unless this is otherwise the intention for the webpage.

8. **Minimize as much as possible the use of Flash.** There are still people who are not comfortable at having to download an application on the Web just to view some contents of a website.

9. **A Frequently Asked Questions (FAQs) page helps.** This is usual for e-commerce sites. Providing a FAQ page will not only benefit the user but also the people tasked to answer queries.

10. **Always give the user a way out.** Do not make dead end pages, or pages on your site where the user has nowhere else to go. Always give the user hyperlinks to other pages in your site which can potentially interest them. This is most helpful with e-commerce sites to keep the user from leaving. Otherwise, a frustrated user just might close the browser altogether and might not return.

Resources:

- http://www.keyrelevance.com/articles/usability-tips.htm
- http://www.2createawebsite.com/money/google-adsense.html
- http://en.wikipedia.org/wiki/Usability

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. What are search engines?

   _____

2. What is the basic format of the APA citation? Give an example.

   _____

3. Are meta tags reliable for web crawling? Why?

   _____

4. It is the Greek term meaning "about" or "beyond."

   _____

5. What is the typical size of a web banner?

   _____

6. What does IPR stand for and what is its significance?

   _____

7. What is AdSense? How will you potentially benefit from it as a website owner?

   _____

8. What does usability mean?

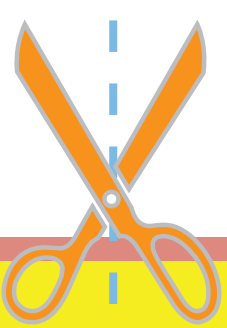
   _____

9. Give some usabillity tips you have learned.

   _____

10. Give a site that provides free website hosting.

    _____



B. Match the concepts with their correct function/description. Write the letter of the correct answer on the blank.

_____ 1. meta tag	a. theft of one's intellectual property
_____ 2. usability	b. a faster alternative to HTTP file upload
_____ 3. alt	c. an accessibility attribute of web images
_____ 4. search engine	d. a strategy used by some webmasters to "deceive" spiders to gain higher rankings in search engines
_____ 5. meta data	e. information that can be used to classify a document
_____ 6. web spider	f. a form of protection preventing people from owning other people's original works or ideas.
_____ 7. copyright	g. contains data crawled by search engines
_____ 8. plagiarism	h. searches documents using a certain keyword provided by the user
_____ 9. FTP	i. measure of efficiency and elegance by the user
_____ 10. cloaking	j. a computer script that crawls webpages on the Internet

C. Search engines are the primary means by which web visitors find websites. And since you've already used search engines, you should know that the first few results are usually the ones that get the searcher's attention. Hence, it is important for websites to improve rankings in major search engines. The use of meta tags is a good means to optimize a website for search engines. The key words placed in a meta tag tells a web spider what content can be found in a site or in the page. Hence, meta data have to be carefully chosen. If you are to provide ten key words for a website selling travel services in the Philippines, what would the most relevant keywords be?

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____

D. List down the top six websites listed in Google results when you search the keywords "web hosting Philippines"? Compare notes with one of your classmates and see whether you got the same results.

_____ _____
_____ _____
_____ _____

E. How do you cite this Web Scripting book if you are to include it as a reference in one of your manuscripts?

_____

The information inside the `meta` tags are used by search engines to identify what kind of information is in that webpage. A web advertisement may take the form of a banner or it may occupy a whole page. You can earn from web advertisements through AdSense. Be careful when putting in information that is not your own, and make sure that you cite the sources of your information, using the APA style, to refrain from plagiarism.

LESSON 10

Getting Started with JavaScript

At the end of this lesson, the student will be able to:

1. **Define what a script is.**
2. **Distinguish between client-side and server-side scripting.**
3. **Identify the tools needed to start using JavaScript.**

We have been talking about scripts in our earlier lessons. Before we get started with a specific scripting language called JavaScript, let's define again what a script is. A **script** is a set of computer instructions or commands that puts together or connects existing components to accomplish a new related task. Scripts are typically written in plain text form and are interpreted each time they are invoked.

What Is JavaScript?

JavaScript is a powerful client-side scripting language that is interpreted by the web browser. It enables web developers and designers to build more functional and interactive websites. Most major web browsers and those emerging on the Net support JavaScript. In fact, you would find that JavaScript is enabled by default in most of these browsers. And there are good reasons for this. For one, JavaScript, being a scripting language, allows webmasters to create cool applications on their pages with relatively easier coding work. It is also lightweight because unlike server-side scripts, it is directly interpreted by the browser without the need for compiling. This allows the webmasters to do all the cool stuff without significantly reducing loading speed. And last but not least, JavaScript is readily available at no cost to developers as opposed to platforms such as Flash.

At first glance, one would think that JavasScript is somewhat associated with Java. This is entirely inaccurate. JavasScript was created by Brendan Eich for Netscape, one of the pioneer web browsers. Originally it was called "Live Script" but then, for marketing reasons, it was changed to JavasScript since the Java programming language had become popular.

When you visit a JavaScript-enhanced website, you essentially download the HTML code to display the site and all scripts inside it. Most browsers have JavaScript enabled by default, so the scripts run automatically when the page loads or when you trigger a certain action. This kind of behavior brings into light some security issues regarding JavaScript which we will discuss shortly. But for now let us discuss the characteristics of JavaScript as a programming language.

Characteristics of JavaScript

Interpreted Language

JavaScript is what we call an interpreted language due to the fact that it requires a browser to run. Let's look at the difference between the Java programming language and JavaScript. Java is a programming language that needs its code to be compiled before it can be executed. On the other hand, JavaScript was created to work together with HTML and requires a web browser in order to run. Partly, this is the reason why the JavaScript source code can easily be seen when you view the source code of a JavaScript-enabled webpage.

Scripting Language

Scripting languages are treated like programming languages but they are normally created to perform simple repetitive tasks. They do not come with their own interfaces but they provide instructions to other programs on what to do. In the case of JavaScript for example, while it does not contain the program on how the whole webpage looks like, we can use it to change certain elements of the webpage through the browser.

Client-Side Scripting

There are basically two kinds of scripting for the Web—client-side and server-side. As the name implies, the key difference between these scripts is the location to which they are executed. A server-side script is executed on the actual server machine that hosts the webpage while a client-side script is executed on the very machine that you are using. JavaScript is a client-side scripting language while scripts written in ASP, PHP or JSP are called server-side scripting languages.

To highlight the difference between the two, let's say we create a script that gets the current time. When we use PHP to get the time, it will return the time on the location of the server machine that stores the actual webpage. So if the location of the host is in the United States of America, PHP will return the time in the USA. JavasScript however will return the time that is currently set on the computer that you are using at that moment.

Another difference between the two is the code. When you load a PHP script and view the source code on your browser, you can't see the actual PHP script but only HTML. The reason for this is that the server interprets the PHP script and just returns HTML. With JavasScript, you can view the source code together with the HTML elements of the webpage. The source code is present because it is the web browser on your local machine that interprets it.

Object-Oriented Programming

Another advantage of JavaScript is that it presents to you many things as objects and lets you easily manipulate them. OOP is a programming style that treats every element in a system as an object. In OOP, each object has its associated functions, called **methods**, which are the things that objects know how to do.

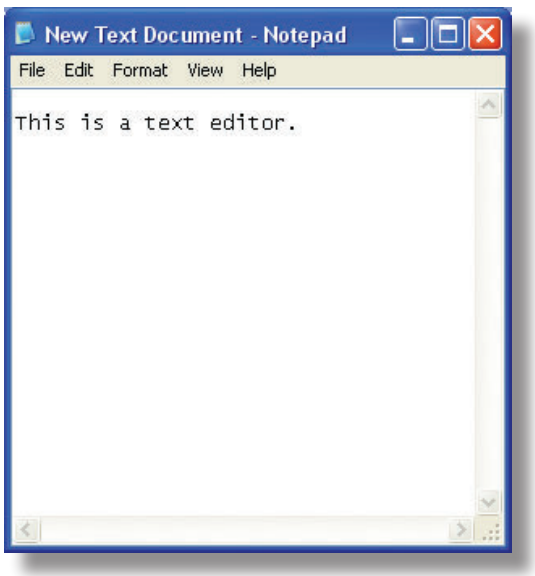
Let's look at a simple webpage in relation to JavasScript. The whole webpage is seen as an object called *document*. Now if we want to change the background color of the webpage, we change the object property *bgColor* to the color of your choice. Don't worry if you can't fully understand what *document* and what *bgColor* are. We'll go over them in detail in the succeeding lessons. For now, what's really important is for you to know that JavaScript is a very powerful object-based or **object-oriented programming (OOP)** scripting language (although JavaScript is not a full-blown OOP language). It lets you create or access objects and manipulate them according to your needs.

So, why should you use objects? Objects not only help you better understand how JavaScript works. They simplify your coding work because you can actually reuse objects many times on your page or even across pages; and they are easier to maintain.

What You Should Already Know

To start scripting, you will need some tools. Most likely, the computer you're using already has these tools:

- **Text Editor** – MS Notepad comes with your Windows operating system. There are also a lot of free text editors available for download in the Internet.



Most browsers can support JavaScript and have the ability to enable or disable it. In fact, most browsers have set JavaScript to "enabled" by default.

Now that you know the tools required to begin creating your JavaScript program, below are a few important things to know about JavaScript:

- JavaScript is normally embedded in your HTML code so you don't need modifications on the file extension *.htm or *.html.

- Although you would normally embed JavaScript into your HTML code, in some large scripts, you can save them in a separate file like **.js* and load them into your browser by linking.

The Key to Understanding JavaScript

To improve your learning experience of JavaScript, consider the following pointers:

- **Learn at an optimal pace and do a lot of practice.** Although JavaScript is relatively easier to learn than other scripting languages, it takes a lot of practice to actually get yourself familiar with the language. Learning any kind of programming or scripting language requires patience in going through the basics and then progress from there to implement more advanced concepts. In many cases, learners fail to learn because they try to accomplish too much too soon. By learning at an optimal pace, you can ensure that you'll master the essentials and you'll never find yourself stumbling in the depths of scripting.

- **Understand the concept of objects.** JavaScript is an object-based language although it is not classified as a full blown OOP. The ability to do anything in a programming language requires functions supported by the language. JavaScript has a few functions that are not part of objects but most of its functionalities are contained in its objects. To utilize JavaScript effectively is to understand what objects are and how you can manipulate them using methods, parameters and events.

  The following terms are inevitably used in JavaScript and in most other programming languages. Hence, it would be good to have a background on these concepts.

  - **Objects** are items that exist in the browser. The browser window, the page itself, the status bar, and the date and time stored in the browser are all objects.
  - **Methods** are actions to be performed on or by an object. Methods also represent functions that are designed for a specific purpose like doing math or parsing strings.

- **Properties** are an existing subsection of an object. It is also important to learn these basic concepts in programming:
- **Event** - something which happens or which is done such as loading a page, holding a mouse over a link, or clicking a link. You can actually write scripts and set them to execute when a specific event occurs.
- **Parameter** - data or other objects which describe the characteristics of an object.
- **Type** - type of variable such as integer, string, floating point, or Boolean.
- **Type conversion** - the act of converting a variable from one type to another.

A script is a set of computer instructions or commands that puts together or connects existing components to accomplish a new related task. JavaScript is a powerful scripting language used to create web functionalities. A text editor and a web browser are the tools needed in JavaScripting. The basic components of JavaScript language are objects, methods and properties. Knowing these components helps you to better understand the use of JavaScript.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Answer the following.

1. Enumerate the advantages of object-oriented programming.

_____

_____

_____

2. Differentiate client-side scripting from server-side scripting.

_____

_____

_____

3. What are some general uses of JavaScript?

_____

_____

_____

B. Identify the term described in each item.

_____1. a set of computer instructions or commands that puts together or connects existing components to accomplish a new related task

_____2. a kind of web scripting optimized to run on the machine that the client is using

_____3. an existing subsection of an object

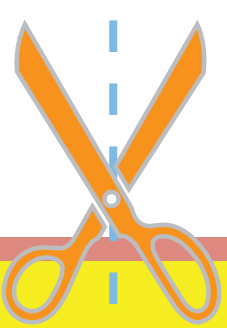
_____4. Brendan Eich developed JavaScript for this company

_____5. functions that are designed for a specific purpose like doing math or parsing strings

_____6. code returned by the server to your browser after executing the script

_____7. the act of converting a variable from one type to another

_____8. items that usually exist in the browser



_____9. a type of programming language that treats every element in a system as an object

_____10. a kind of scripting that is executed on the actual server machine that hosts the webpage

_____11. an occurrence in which something happens or is done such as loading a page, holding a mouse over a link, or clicking a link

_____12. this is another term for scripting languages that require a browser to run

_____13. a tool needed to be able to code JavaScript instructions

_____14. a concept in programming that includes integer, string, floating point, or Boolean

_____15. a tool that enables the user to interact with the scripting language

_____16. the extension name of a JavaScript file

_____17. an object where JavaScript's object is displayed

_____18. object property that sets the color the user wants

_____19. implemented to make sure JavaScript will not be used to inadvertently harm the innocent user

_____20. data or other objects used to describe the characteristics of an object

LESSON 11

JavaScript Syntax Rules and Variables

At the end of this lesson, the student will be able to:

1. **Describe objects, methods, properties and events.**
2. **Explain the basics of JavaScript syntax.**

Document Object Model

When JavaScript was introduced, there was a need for an interface to allow the script to access elements on the page. During its infancy, each browser had their own specification on how to access page elements. This became a problem for programmers since there was no set standard to follow and there were inconsistencies between webpages depending on Internet browser you are using. Eventually, the World Wide Web Consortium (W3C) was created so that there would be compatibility and agreement in the adoption of new standards.

The Document Object Model (DOM) is an interface that allows you to access and manipulate the contents of a document (in our lessons, a webpage) with a programming language. It is a structured object-oriented representation of the individual elements of a page with methods, properties and events for those objects. With DOM, programmers can easily create dynamic content for webpages. It's worthwhile to note that DOM is not exclusive for JavaScript—other programming languages such as Java, C++, C# also adhere to this standard.

Javascript Objects, Methods, Properties and Events

Let's begin our first JavaScript program with the classic "Hello World!" display. By now we should already know how to display "Hello World!" using HTML. But did you know that JavaScript can be used to display it, too? Check out the code snippet below:

```
1  <html>
2     <head>
3        <script type = "text/Javascript">
4              document.write("Hello World");
5        </script>
6     </head>
7     <body>
8        <br/>
```

```
9         This portion holds contents of the body tag.
10     <body>
11  <html>
```

Let us study the structure of the code snippet.

- Here we placed the JavaScript within the <head> tags of HTML. This means that JavaScript is executed even before the <body> tags.
- From lines 3-5 you will see the <script> tags that tell the browser that the lines of code within those tags are scripts. After closing the <script> tag, the browser will proceed with parsing normal HTML until it encounters another non-HTML script or instructions.
- Line 4 holds the actual Javascript code. The "`document`" **object** refers to the <body> tags of HTML while "`write`" is a **method** of the document object. This means that you want JavaScript to write the contents between the parenthesis and quotes to the HTML document.

Javascript does not always have to be placed within the head tags of an HTML file. Below are additional ways you can place your JavaScript code:

- JavaScript can be placed within HTML elements. These scripts are normally placed together with event triggers. We shall discuss more of this shortly.
- You can also create an external file with a *.js extension to contain JavaScript code that you can reuse throughout various HTML pages. With an external file, you can embed it on any HTML file by placing this code on your HTML.

```
<script style= "text/javascript" src="script.js"></script>
```

Since JavaScript is an OOP language, not only does it have objects and methods but objects also have properties. Check out the table below to see some of the common methods and properties for the document object.

Document Object Properties		Document Object Methods	
bgColor	Background Color.	write()	Writes a string of text to the document.
fgColor	Foreground / font color.	writeln()	Writes a string of text followed by a newline character to a document.

Document Object Properties		Document Object Methods	
Image	Set an image.	getElementById()	Returns a reference to the element whose ID is specified.
Location	Get or set the URL path.	getElementsByName()	Writes a string of text followed by a newline character to a document.

Let's see how we can use the Document Object Properties with the code below:

```
1 <html>
2    <head>
3    </head>
4    <body>
5       <Input Type = Button Value = "Red" Onclick = "document.bgColor =
        '#FF0000'">
6       <Input Type = Button Value = "Blue" Onclick = "document.bgColor =
        '#0000FF'">
7       <Input Type = Button Value = "Yellow" Onclick = "document.bgColor =
        '#FFFF00'">
8    <body>
9 <html>
```

In the code, we used 3 buttons to change the background color of the page whenever it is clicked. You can see this on lines 5-7. Take note of the Onclick event which is the trigger for our Javascript. Also notice that we were able to put the JavaScript together with the HTML between two double quotes.

One quirk for many programming languages is the use of the quotes. Since our script is nested within two double quotes, we can no longer use double quotes within the script, but instead we use single quotes. That is the reason why the hexadecimal values of the colors are within single quotes and not double quotes.

Events act as triggers to run scripts within HTML. Every element on the website has events that trigger JavaScript.

The ability of JavaScript to integrate object properties and methods with website events enables it to create rich dynamic websites. We shall discuss more of this in later lessons. However, as we go along, we will introduce a few object properties, functions and triggers in our examples.

A Simple Script

Let's put together what we have learned and combine methods, properties and events to create some of the most useful scripts for dynamic webpages.

One of the most popular scripts used are image roll-overs. The idea behind this is to change an image whenever the other user has his mouse over a certain page element. It is normally used to give emphasis to buttons and image links. In our example, we want our code to change the non-smiling emoticon to a smiling one when the user hovers his mouse over the image.

Look at the JavaScript code below and study how we can change one image to another, using the OnMouseOver and OnMouseOut events.

```
1  <html>
2     <head>
3     </head>
4     <body>
5        <img src = "Image1.jpg" NAME = "emoticon" OnMouseOver =
         "document.images.emoticon.src = 'Image2.png'" OnMouseOut =
         "document.images.emoticon.src = 'Image1.jpg'">
6     <body>
7  <html>
```

In order to create this effect, we gave our image a "`name`" that Javascript can reference. The reference to the image source can be seen in "`document.images.name.src`". In the code above, the images are only loaded when the user triggers the event. Normally, it's best to pre-load images before using them so the user has seamless experience with image roll-overs.

Syntax Guides

Now that you've learned some of the basics, it would be good for you to learn the rules in JavaScript syntax:

- In order for the HTML document to identify Javascript as the scripting language used, it is necessary to include `language="JavaScript"` in the opening <SCRIPT> tag.

- The syntax for manipulating an object by using a method or property is always *Object.Method()* or *Object.Property()*. The parentheses () are called the instance.

- **JavaScript is case-sensitive**. For example, if you called the function warning() in the previous example using wArning(), you'll encounter an error since the entry is incorrect. Another example is the bgColor property. If you don't type it like this, you'll get an undefined result.

- **JavaScript is processed from top to bottom**. The first one on the page is processed first while the script at the end is processed last. We can pre-load the script by placing it within the <head> tags. Elements of this script can be executed before any other script embedded in the HTML.

- Quotes are used to tell the browser what is to be displayed exactly as it is written in the script. Hence, don't place variables inside quotes; otherwise, the variable name will be displayed instead of the value assigned.

- Quotes may be single quote (') or double quote ("). Remember to use them consistently, especially when embedded within each other.

- You can add comments using "`//` " for a single line of comment or you may enclose your multiple-line comment between "`/*`" and "`*/`".

- Use the symbol **+** to concatenate a string.

- Statements may contain a semicolon at the end of them. However, this is not necessary in JavaScript unless there are multiple statements in one line. In this case, you must separate multiple statements in a single line by using a semicolon. But it is always good programming practice to use a semicolon to end a statement.

The last point is worth mentioning here because we are going to learn the basics of JavaScript variables in the next lesson.

JavaScript Variables

Let's start learning about JavaScript variables by first understanding what a variable is. Perhaps, the most familiar variable to highschool students is the variable in algebra class. In an algebra class, one encounters lots of variables. For example, in the Pythagorean Theorem, there's c2 = a2+b2. The letters a, b and c are called variables.

In both mathematics and computer science, a variable is a symbolic representation denoting a quantity or expression.

However, in mathematics, a variable often represents an "unknown" quantity that has the potential to change. In computer science, it represents a place where a quantity can be stored. It is a value that has the property of being associated with another value.

In JavaScript, **variables** are a container that contains a value, which can change as required.

Data Types

As mentioned earlier, JavaScript variables can contain values that can change as required. There are no types attached to a variable in JavaScript. This means, one need not declare the type of data a variable is supposed to contain. Here are some of the data types that a JavaScript variable may contain:

- **character** – any single character or symbol such as x, '@', 'y' etc.
- **string** – a series or combination of characters like "somevariable" or "64kgold"
- **integer** - number of the set Z = {..., -2, -1, 0, 1, 2, ...}
- **float (or double)** – any decimal number like 97.1 or -97.1
- **boolean** – truth values. This can either be TRUE or FALSE.
- **function** – set of instructions or arguments
- **object** – items that exist in the browser that one can manipulate
- **array** - a type of object that contains a set of values that are mapped into keys
- **undefined** – a variable that has not been given a value yet
- **null** – a variable that has been defined but has been assigned the value of null

It is important to note that types in JavaScript need not be declared and they may interchange as necessary. This means that if a variable is a string at one minute, it can be an integer the next.

Declaring Variables

Variables are declared with a *var* statement. It is always good practice to pre-define your variables using *var*. If you declare a variable inside a function, it can be accessed anywhere within the function.

The example below illustrates how to declare a variable:

```
// declaring one variable
var firstname;

// declaring several variables
var firstname, lastname;

// declaring one variable and assigning a
value
var firstname = "Jocelyn";

// declaring several variables and assigning a
      value
var firstname = "Juan", lastname = "Delacruz";
```

JavaScript is fairly lenient, and will create a variable anyway if you do not use the *var* keyword to declare a variable. However, this can lead to problems in your code. Hence, it is always good to use *var*.

It is also important to note that variables declared outside any function, or used without being declared with 'var' are global and can be used by the entire program.

Rules for Naming JavaScript Variables

When naming variables in JavaScript, it is important to remember that:

- Variables can contain any letter of the alphabet, digits 0-9, and the underscore character.

- There should be no spaces in between characters.
- It should not have punctuation characters (e.g., comma, full stop, etc.).
- The first character of a variable name cannot be a digit.
- Variable names are case-sensitive. Hence, *firstname* and *Firstname* are two different variables.

Let's try using your first JavaScript variable. Ready your Notepad:

```
<html>
<head><title>My JavaScript Variable</title>

<script style="text/javascript">
 var firstname = "Jocelyn";
</script>

</head>
<body>

<script style="text/javascript">
 document.write("hello there " + firstname)
</script>

</body>
</html>
```

In this example, we assigned a variable called firstname and assigned it the value "Jocelyn" inside our head tags. At the document body, we called the variable firstname and decided to output it to the browser together with a short string "hello there". Note that we did not place the variable inside the quotes so that the browser will not just output the variable name. Instead, we placed a + symbol to concatenate the whole string and to let the browser distinguish the variable.

The Document Object Model is an interface standard that defines how elements of a page can be accessed. Objects have properties, methods and events which can then be used to create dynamic webpages. Syntax is a set of rules that determines how a specific language will be written and interpreted.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. TRUE or FALSE: Write True if the statement is true; otherwise, write False. Encircle the words that make the statement incorrect and write the correct answer in the space provided.

_____1. Syntax refers to a set of rules that determines how a specific language will be written and interpreted.

_____2. The Document Object Model is an interface that allows you to access and manipulate the contents of a document with a programming language.

_____3. The method write(), writes a string of text followed by a newline character to a document.

_____4. JavaScript codes are placed in the head and body sections of an HTML page.

_____5. The syntax for manipulating an object by using a method or property is always Object.Method() or Object.Property().

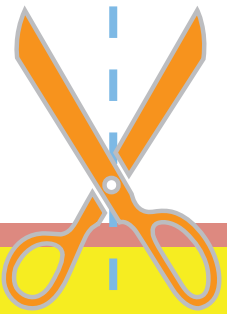
_____6. The . symbol is used to concatenate a string.

_____7. The + sign is used to tell the browser what is to be displayed exactly as it is written in the script.

_____8. Events act as triggers to run scripts within HTML.

_____9. JavaScript is case-sensitive.

_____10. It is not necessary to use semicolons in JavaScript even if there are multiple statements in one line..



B. Answer the following.

1. What are the ways for JavaScript to be inserted in an HTML file?

_____

_____

_____

_____

_____

2. Why is there a need for the WC3 with regard to the DOM interface?

_____

_____

_____

_____

_____

3. Explain the relationship between Objects, Methods, Properties and Events in Javascript.

_____

_____

_____

_____

_____

LESSON 12

JavaScript Operators

At the end of this lesson, the student will be able to:

1. **Define what a variable is.**
2. **Define what an operator is.**
3. **Distinguish among the different operators in JavaScript.**
4. **Identify the uses of the different types of operators.**
5. **Apply the different types of operators in JavaScript.**

This lesson takes off from the previous lesson which introduced the basics of the JavaScript language. Here you will gain a deeper understanding of the language as you learn about operators, which is a basic language construct.

JavaScript Variables

To recap, in JavaScript, variables are a container that stores information. The syntax for declaring variables is as follows:

```
var VariableName = value;
```

The keyword 'var' is not actually necessary but it's good practice to use it so we can easily see a variable in the code. Variable Names can contain any letter of the alphabet, digits 0-9 and the underscore character. There should be no spaces between characters and the first character of a variable name must not be a digit. And just like object methods and properties, Variable Names are case sensitive.

Remember our first Javascript code "`Hello World!`"? How can we display the same thing using variables?

```
1   <html>
2      <head>
3         <script type = "text/Javascript">
4            var greet1 = "Hello";
5            var greet2 = "World!";
6            document.write(greet1 + " " + greet2)
7         </script>
8      </head>
9      <body>
10     </body>
11  </html>
```

The code above will output "Hello World!". Lines 4 and 5 are variables that contain the words "Hello" and "World!" respectively. To put them together, we have to use a plus (+) sign. We also put a space between the two variables on line 6 or else the program will display "HelloWorld!". Here we used the + operator to concatenate several string variables. Let us now learn about the different operators we can use in Javascript.

What Is An Operator?

Normally, you'd use operators in programming to build an expression just as how you build a mathematical expression in your algebra class. In other words, you use an operator to evaluate (or to operate on) one or more values and then return a value resulting from whatever evaluation or operation it did on the value(s). An **expression**, in the simplest and least complicated term is "anything that has a value." Expressions are important building blocks of JavaScript because you use a lot of these in many of the applications you write in JavaScript.

One very easy way to understand the concept of operators in JavaScript is to relate it with the operators in your mathematics class. The basic operators in your math class include addition, subtraction, multiplication and division using the symbols +, -, *, and / respectively. These are easy to associate with JavaScript operators because the language also uses them and are categorized as math/arithmetic operators.

However, you'll find it really very interesting to know that there are other types of operators that you can use in JavaScript. If you are new to programming, this is probably something novel and therefore very interesting. For instance, the symbol + could mean another kind of operation, which is, concatenating or joining strings. For this operation, the symbol + does make a lot of sense since adding in mathematics is somewhat similar to joining strings. What's more interesting is that there are different types of operators for many other different uses.

Math/Arithmetic Operators

Arithmetic operators in JavaScript, as discussed earlier, are similar to the arithmetic operators in your mathematics class. They work in exactly the same manner. Hence this part of the lesson is simply a refresher.

Operator	Name	Example	Result
+	Addition	myVariable = 3 + 4	myVariable = 7
-	Subtraction	myVariable = 15 - 7	myVariable = 8
*	Multiplication	myVariable = 3*5	myVariable = 15
/	Division	myVariable = 20 / 4	myVariable = 5
%	Modulus	myVariable = 24 % 10	myVariable = 4
-	Negation	myVariable = -10	myVariable = -10
++	Increment	See example under While statement (Lesson 13 Controls and Loops)	
--	Decrement		

All operators seem very familiar except, perhaps, the modulus and the increment and decrement operations.

The increment operator ++ simply adds one to the value of your variable while the decrement subtracts one. These operators are normally used within a loop or control structure, which will be discussed in the next lesson.

Modulus is simply the remainder of a division operation. In our example above, we have a remainder 4 from 24/10. Let's try another example so you'll know how to use this operator. Type this code within your <body></body> tags in HTML:

```
<script style="text/
javascript">
 var num1 = 37%10;
 document.write(num1)
</script>

Result:
7
```

In this example, the remainder of "37 divided by 10" returns a result of 7. On your own, try the other operators during your free time so you can get used to using them.

Assignment Operators

Assignment operators, except for the "=" operator, which only assigns value, are used to both assign and perform math operators. It is some sort of shorthand for some less simple math operations.

Specifically, most of these operators first perform a math operation between two variables and then assign the result to one of the variables. This concept is quite difficult to digest without illustration. So here it is:

Operator	Name	Example
=	Assign var	x = 1, y = 2;
+=	Add and Assign	x+=y is the same as x=x+y
-=	Subtract and Assign	x-=y is the same as x=x-y
*=	Multiply and Assign	x*=y is the same as x=x*y
/=	Divide and Assign	x/=y is the same as x=x/y.
%=	Modulus and Assign	x%=y is the same as x=x%y

Similar to increment and decrement operators, you should use these operators within controls or loop structures. We'll use one of these operators later on in one of the examples in the next lesson.

Comparison Operators

Comparison operators are used when you want to compare two values. The table below lists down the JavaScript comparison operators and some examples:

Operator	Name	Example	Result
==	Is equal to	3==5	False
===	Is identical to (equal and of same type)	2===2	True
!=	Is not equal to	3!=3	False
!==	Is not identical to	4 !== 3	True
>	Greater than	4 > 3	True
>=	Greater than or equal to	4 >= 3	True
<	Less than	4 < 3	False
<=	Less than or equal to	4 <= 3	False

As shown in the table, the result of a comparison operator is a Boolean value, which is either True or False. The items that are most often compared are numbers. Here is how to write the example above in JavaScript. Place this code within the <body></body> of your HTML code:

```
<script style="text/javascript">
 var myComparison = 10<=8;
document.write(myComparison);
</script>

Result:
False
```

Logical/Boolean Operators

A **Logical operator** is similar to a Comparison operator in the sense that it also evaluates whether a certain statement is true or false. However, a Logical operator seems more complicated as it must make decision(s) based on one or more truth values (True or False).

Below are JavaScript Logical operators:

Operator	Name	Example	Result
&&	AND (both statements must be True)	True && True	True
||	OR (one must be True)	True || False	True
!	NOT (Flips truth value)	!True	False

Again, the best way to understand a difficult concept is to practice. So, open your text editors and let's start "thinking a bit more logical," so to speak.

```
<script style="text/javascript">
 var mylogic = 10<=8 || 10>=8
document.write(myComparison);
</script>

Result:
True
```

In the example, the browser returned a result of True. This is because we used OR, ||, which returns a True value if at least one of the arguments in the statement is True.

String Operators

These operators are used to perform simple operations on a string or a piece of text.

Operator	Name	Example	Result
=	Assign	`MyStringVar = "Johnny"`	Johnny
+	Concatenate	`Document.write("My name is "+ MyStringVar);`	My name is Johnny
+=	Concatenate and assign	`myString = "ab"+"cd"; myString += "ef";`	abcdef

These operators all look familiar to you because you have encountered them when we discussed the math and assignment operators. However, in those previous discussions, we performed operation in numbers. This time, let's try doing it with strings:

In this example, note that we concatenated and assigned the result to the variable, not just once, but twice.

```
<script style="text/javascript">
 var somestring = "Bat";
 somestring += "man";
 somestring += "AndRobin";
document.write(somestring);
</script>

Result:
BatmanAndRobin
```

A Simple Script

Most of our examples have predefined variables. However what makes JavaScript such a versatile scripting language for webpages is its ability to take user input and process them on the spot. Let's create a simple script that uses what we have learned in this lesson. Let's say we want to make a script that can add any two numbers.

```
1  <html>
2     <head>
3        <script type = "text/Javascript">
4           function add()
5           {
6              var A = Number (document.example.firstno.value);
7              var B = Number (document.example.secondno.value);
8              var C = A + B;
9              document.example.sumno.value = C;
10          }
11       </script>
12    </head>
13    <body>
14
15       <form name = "example">
16          <input type = type name = firstno value = "">
17          +
18          <input type = type name = secondno value = "">
19          =
20          <input type = type name = sumno value = "">
21          <br/><input type = button name = addno VALUE = "SUBMIT"
22          onClick = add()>
          </form>
23    </body>
24 </html>
```

Lines 15 to 22 show our form code. Lines 16 and 18 are the textboxes where the user types the values. In Line 20, we see the textbox where the sum of the values will be displayed. For all three, we have to specify a name for the textboxes so JavaScript will be able to recognize them.

On line 21, we have the code for our button. Notice that we use the onClick event with a custom- made function named add(). A **function** is a group of codes meant to do a specific purpose. Our custom function add() is meant to add the values of two textboxes and display them on the third one. Let's take a closer look at our function.

Lines 6 and 7 denote the creation of the variables A and B. It also assigns the value of the textboxes to the variables. Study how we write the value of the textbox so that JavaScript can recognize it. Document is the HTML page object while "example" is the name of the HTML form containing the textbox. The names of the textboxes are "firstno" and "secondno" while "value" is the property of the textbox. Put all of those together and we are able to let JavaScript recognize what is in the textbox.

Take careful note of the **Number()** function. In line 8, we see the variable declaration for C which is supposed to be the sum of the value of A and B. Without the Number() function, 1+1 will be equal to 11 because JavaScript does not know that the textbox value is a number. By

default, JavaScript treats the textbox value as a string. When we encase our values with the built in Number() function, JavaScript then knows that the values are to be treated as numbers and not as strings. So 1+1 will be equal to 2.

Finally in Line 9, we assign the variable C to the "sum" textbox so we can see the result of our equation. For example, if your user inputs a letter or any other character in the textboxes that isn't a number, then the value of C will be NaN which is short for Not a Number. Erroneous user inputs have always been a problem for web developers and JavaScript offers a solution for that. Look at the code below:

```
1   <html>
2   <head>
3   <script type = "text/Javascript">
4      function add()
5      {
6         var A = Number (document.example.firstno.value);
7         var B = Number (document.example.secondno.value);
8
9         if (!Number(A)||!Number(B))
10        {
11           alert ("One of your inputs isn't a number.");
12        }
13        else
14        {
15        var C = A + B;
16           document.example.sumno.value = C;
17        }
18     }
19
20  </script>
21  </head>
22  <body>
23     <form name = "example">
24        <input type = "text" name = "firstno" value = "">
25        +
26        <input type = "text" name = "secondno" value = "">
27        =
28        <input type = "text" name = "sumno" value = "">
29        <br/><input type = "button" name = "addno" value =
          "submit" onClick = add()>
30     </form>
31  </body>
32  </html>
```

SCORE

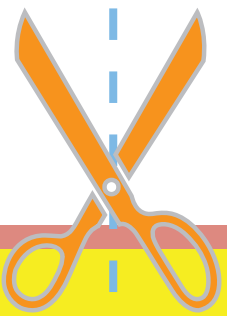
NAME: _____

SECTION: _____

DATE: _____

A. Match column A with column B. Write your answer on the space before each number.

Column A	Column B
_____ 1. &&	a. Flips truth value
_____ 2. %	b. x/=y is the same as x=x/y.
_____ 3. ++	c. Concatenate and assign
_____ 4. -=	d. One must be True
_____ 5. ||	e. Both statements must be True
_____ 6. ==	f. Is not equal to
_____ 7. *=	g. Increment
_____ 8. !	h. Decrement
_____ 9. !==	i. 2 = =7
_____ 10. +=	j. Modulus
_____ 11. -	k. Concatenate
_____ 12. /=	l. Multiply and Assign
_____ 13. +	m. Is not identical to
_____ 14. –	n. Negation
_____ 15. !=	o. a-=b is the same as a=a-b



B. Enumerate the 4 different types of JavaScript operators. Give a brief description about each type of operator.

1._____

2._____

3._____

4._____

In order to check if only numbers were submitted, we needed to create a conditional If-Else Statement as seen on Lines 9-17. There are actually two conditions being analyzed. The first condition is to check if at least one of the variables A and B are not numbers. If they are not, the program will issue an alert. The second condition will work if variables A and B are numbers. We will proceed with the addition operator and display the result on the "sumno" textbox.

Control statements are very useful in programming and we shall discuss more of that in the next lesson.

Operators are symbols that represent a specific action. There are different types of operators: arithmetic operators, assignment operators, comparison operators and string operators. These operators are used to incorporate text and numbers to complete an expression.

LESSON 13

Controls and Loops

At the end of this lesson, the student will be able to:

1. **Describe the different types of conditional statements.**
2. **Explain the functions of each type of conditional statement.**
3. **Create sample codes for each type of conditional statement.**

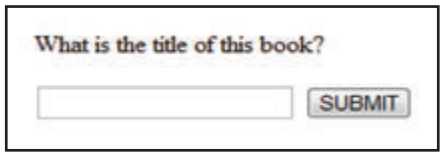
Still other basic operators are controls and loops which are conditional statements. These are sets of commands that perform different actions based on different conditions. In other words, they allow for the conditional execution of code fragments, creating the effect of a "smart" or "interactive" webpage because specific scripts can be written to respond to certain decisions.

In this lesson, we shall look at the *If statement* and its variants, *Switch statement*, *While loop*, and *For loop*. This should be enough for you to get more comfortable with scripting in JavaScript.

Controls

If Statement

The **If statement** is one of the most popular and most important conditional constructs in JavaScript and in many other programming languages. This conditional statement construct evaluates a condition to True or False. It then runs a specific code depending on the result of this evaluation. The example that follows is the basic syntax for an If statement:




```
1 <html>
2    <head>
3       <script type = "text/Javascript">
4          function check()
5          {
6             if (document.example.answer.value == "Web Scripting")
7             {
8             document.write("You answered correctly!");
```

```
9              }
10           }
11        </script>
12     </head>
13     <body>
14        <p>What is the title of this book?</p>
15        <form name = "example">
16           <input type = text name = "answer" value = "">
17           <input type = button name = "answerbutton" VALUE =
             "SUBMIT" onClick = check()>
18        </form>
19     </body>
20  </html>
```

The code structure for an If statement can be seen from lines 6 to 9. The condition that will be processed is between the parenthesis after the If statement. The resulting action is between the curly brackets.

In the example above, we created an HTML form that asks, "What is the title of this book?" If you type in "Web Scripting" (note that the answer is case-sensitive), then the page will display "You answered correctly!". But what if you got the answer wrong? With the code above, a false answer will not display anything. Confusing, isn't it? This is where the If Else statement comes in handy.

If Else Statement

The **If Else statement** is similar to the If statement, except that we are giving an alternative instruction in case the argument isn't True. Let's use the previous example and put in a statement if the answer is false.

```
1 <html>
2    <head>
3       <script type = "text/Javascript">
4          function check()
5          {
6             if (document.example.answer.value == "Web Scripting")
7             {
8             document.write("You answered correctly!");
9             }
10            else
11            {
```

```
12              document.write("Sorry. Try again.");
13              }
14           }
15        </script>
16     </head>
17     <body>
18        <p>What is the title of this book?</p>
19        <form name = "example">
20           <input type = text name = "answer" value = "">
21           <input type = button name = "answerbutton" VALUE =
              "SUBMIT" onClick = check()>
22        </form>
23     </body>
24  </html>
```

Notice in the example above that we've added an else statement from lines 10-13. This part tells the browser what to do when the wrong answer is keyed in. In this case, we specified that it has to output: "Sorry. Try again." The If-Else statement processes only two conditions: True or False.

If Else If Statement

But in the real world, you don't always evaluate just one condition. Sometimes, you would need to evaluate more than one or multiple conditions.

This is possible in JavaScript with the **If Else If statement**. The name refers to an If statement that depends on another If statement.

In plain layman's language, when you use the If Else If statement, you are simply telling the browser that if something is True, then do something, else, if the other thing is True, then do something else, and so on.

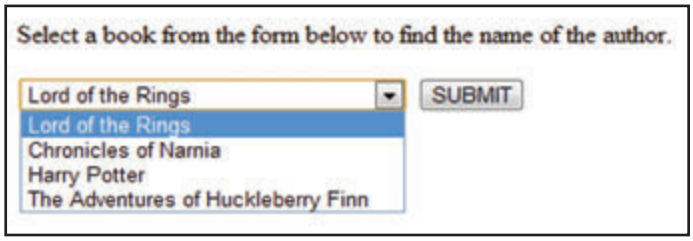
Let's modify our existing example to provide a statement when the user does not type anything but clicks on the submit button.

```
1  <html>
2     <head>
3        <script type = "text/Javascript">
4           function check()
5           {
6              if (document.example.answer.value == "Web
               Scripting")
7              {
8              document.write("You answered correctly!");
9              }
10             else if (document.example.answer.value == "")
11             {
12             document.write("You didn't answer the
               question.");
13             }
14             else
15             {
16             document.write("Sorry. Try again.");
17             }
18
19          }
20       </script>
21    </head>
22    <body>
23       <p>What is the title of this book?</p>
24       <form name = "example">
25          <input type = text name = "answer" value = "">
26          <input type = button name = "answerbutton" VALUE =
            "SUBMIT" onClick = check()>
27       </form>
28    </body>
29 </html>
```

We can see the Else If statement on lines 10-13. Simply put, it's like putting another If statement within the control. With the code above, if the user does not type anything on the textbox and clicks on Submit, the page will have the output: "You didn't answer the question."

Switch Statement

In the previous example, we used the *If Else If* statement to test for multiple conditions. This statement is quite easy to code even for beginners. But what if there are a lot of options? Using the If Else statements can become quite tedious. To address this issue, JavaScript offers an easier way to implement multiple conditions, that is, through the Switch statement.




```
<html>
   <head>
      <script type = "text/Javascript">
         function bookcheck()
         {
            var book = document.frmbook.cmbbook.value

            switch (book){
            case "1":
            alert("J.R.R Tolkien")
            break
            case "2":
            alert("C.S. Lewis")
            break
            case "3":
            alert("J.K Rowling")
            break
            case "4":
            alert("Mark Twain")
            break
            default:
            alert("undetermined")
            }
         }

      </script>
   </head>
   <body>
   <p>Select a book from the below to find the name of the author.</p>
   <form name = frmbook>
      <select name = cmbbook>
         <option value = 1>Lord of the Rings</option>
         <option value = 2>Chronicles of Narnia</option>
         <option value = 3>Harry Potter</option>
         <option value = 4>The Adventures of Huckleberry Finn
         </option>
      </select>
      <input type= button name = combo value= "SUBMIT"
      onClick=bookcheck()>
   </form>
```

```
39
40     </body>
41 </html>
```

In this example, we started by creating a custom function named bookcheck(). Within that function we declared a variable declaring a variable called "book" and using that variable to get the value of the selected item in the combo box. We then opened a Switch statement, passing in the variable we want to test which is "book". This is followed by a set of cases. The "break" construct after each case prevents the code from running into the next case. Prior to the closing bracket, we placed a default condition. This will be executed if none of the previous cases is true.

Loops

As you get more and more deeply hooked in JavaScript, at some point, you would want to execute a piece of code a number of times. For example, you might want to output "I will not sleep in algebra class again." a dozen times for some reason. You may code this manually. But that would be a very tedious process. What more if your teacher is in a really bad mood and wants you to write this a hundred times?

That would certainly take hours to code manually and you might even fall asleep in the process.

Luckily, JavaScript has a language construct that allows repetitive execution of codes, called loops. **Loops** are sets of instructions used to repeat the same piece of code until a specified condition is met or if the condition evaluates to True or False depending on how you want it.

JavaScript supports the While and For loops.

While Loop

The JavaScript **While Loop** executes code while a condition is true. Consider this example:

```
<script style="text/javascript">

var mypromise = 1;

while (mypromise <= 12) {
document.write(mypromise+". I will not sleep in algebra class
again.<br>");
mypromise += 1;
}
</script>
```

Result:
1. I will not sleep in algebra class again.
2. I will not sleep in algebra class again.
3. I will not sleep in algebra class again.
4. I will not sleep in algebra class again.
5. I will not sleep in algebra class again.
6. I will not sleep in algebra class again.
7. I will not sleep in algebra class again.
8. I will not sleep in algebra class again.
9. I will not sleep in algebra class again.
10. I will not sleep in algebra class again.
11. I will not sleep in algebra class again.
12. I will not sleep in algebra class again.

In the previous example, we set the initial value of the variable *mypromise* to 1. Notice that before the closing curly bracket, we used the increment operator on the variable. This adds 1 to the value of the variable. In the middle part of the code, we are instructing the browser that while the value of *mypromise* is less than or equal to 12, it will output our string. Hence, the code stopped upon executing 12 times.

Play with this code using the *add* and *assign* (+=) operator and see what happens.

For Loop

The previous example can be implemented in another way, that is, through the *For Loop*. Let's do this example:

```
<script style="text/javascript">

var mypromise;

for (mypromise = 1; mypromise <= 12; mypromise++) {
    document.write(mypromise+". I will not sleep in algebra
class again.<br>");
}
</script>
```

The code above returns the same result as in the previous example. Let's analyze the code to see how it's possible.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. TRUE or FALSE: Write Loops if the statement is true and Controls if the statement is false. Encircle the words that make the statement incorrect and write the correct answer in the space provided.

_____1. Controls and loops are sets of commands that perform different actions based on different conditions.

_____2. When the user applies the If Else statement, the user is simply giving the browser instructions that if something is True, then do something, else, if the other thing is True, then do something else, and so on.

_____3. Loops are sets of statements used to repeat the same piece of code until a specified condition is met or if the condition evaluates to True or False.

_____4. While Loop executes code while a condition is true.

_____5. Controls and loops allow for the conditional execution of code fragments, creating the effect of a "smart" or "interactive" webpage.

_____6. The If statement is a conditional statement construct that evaluates a condition to True or False.

_____7. Aside from the If Else If statement, the For Loop can be used to test instructions with multiple conditions.

_____8. The If Else If statement is a conditional statement construct that evaluates a condition to True or False.

_____9. For Loop statement executes the code while a condition is true.

_____10. JavaScript includes a language that constructs and allows repetitive execution of codes, called loops.

B. Write the output of the following instructions.

1.
```
<script style="text/javascript">

var greetings = 1;

while (greetings <= 8) {
document.write(greetings+". Hello World.<br>");
greetings += 3;
}
</script>
```

Output:

2.
```
<script style="text/javascript">

var greetings;

for (greetings = 1; greetings <= 8; greetings++) {
        document.write(greetings+". Hello World.<br>");
}
</script>
```

Output:

First, we declared a variable. Then, we opened a *For Loop* statement and placed our conditions inside the parentheses (). There are three blocks of code inside the parentheses. The first one sets the initial value of our variable. The middle block checks whether the value of our variable is less than or equal to 12. The third block of code increments the value of the variable by 1. You can also use += 1 in the third block instead of ++.

The If statement, Switch statement, While loop, and For loop are conditional statements used in JavaScript to create better functionality in a website. These are conditions that allow an instruction to be repeatedly executed.

LESSON 14

Functions and Events

At the end of this lesson, the student will be able to:

1. **Describe the use of functions and events in JavaScripting.**
2. **Explain how functions work in JavaScript.**
3. **Identify the different events that JavaScript can handle.**

This lesson introduces the use of functions and events. They are procedural statements which perform a specific task. You can write your own JavaScript using these two procedural statements.

Functions

You may or may have not noticed it but we have been dealing with functions since we have begun to discuss JavaScript. Do you remember document.write(), add() and check() from our previous lessons? All of those are functions—document.write() is a predefined function while add() and check() are user-defined functions. By now you should have some idea what functions are but let's delve deeper into the topic.

A **function** (also known as **method**) is a self-contained piece of code that performs a particular "function" when it is called.

JavaScript functions are defined using the key word "function" followed by its name and then open and close parentheses. This is known as an argument. An **argument** provides additional information needed by the function to process.

```
<html>
   <head>
      <script type = "text/Javascript">
         function add(no1, no2){
            var A = Number(no1);
            var B = Number(no2);
            var C = A + B;
            document.example.sumno.value = C;
         }
      </script>
   </head>
   <body>
```

```
    <form name = "example">

          <input type="text" name="firstno" value="">
          +
          <input type="text" name="secondno" value="">
          =
          <input type="text" name="sumno" value="">
          <br/><input type="button" name="addno"
          value="SUBMIT" onClick= "add(document.
          getElementsByName('firstno')[0].value,
          document.getElementsByName('secondno')[0].value)" >
    </form>
    </body>

</html>
```

Does the given code snippet look familiar? It is a modified version of the add() function from Lesson 12. Let's study what is different in this code. Notice that the add() function now accepts two arguments "no1" and "no2". When we write the function on the onClick event trigger, we pass these two arguments to the function.

Check out how we pass the argument—we use a built-in document object function called "getElementsByName". With this we just need to reference the name of the HTML object to call it. In the given example, the given name for the two textboxes are "firstno" and "secondno" respectively. However, when we use the function "getElementsByName", the result is an array. So for our purposes, we need to specify the position of the value in the array, and in this case the value is in the first position, hence we use [0]. If this is a bit confusing, don't worry, we'll tackle arrays in a later lesson.

So what's the difference between the add() function from Lesson 12 and the new one with arguments? The add(a,b) function is a lot more flexible than the regular add() function because the value is already being passed to the function rather than the function fetching a specific object's value. In fact, we can tweak the code a little bit so that it can be reused for other pages.

Although it sounds scary for beginners, it is not really that hard to write a function in JavaScript. Always remember functions are bits and pieces of code that are reusable.

Predefined Functions

JavaScript has an extensive library of built-in or predefined functions, which you can readily use. This is very handy because you can just call these functions without having the need to create them first. Hence, in short, JavaScript has created a lot of very common and useful functions and has offered them for instant access by JavaScript programmers.

Conversion and Comparison

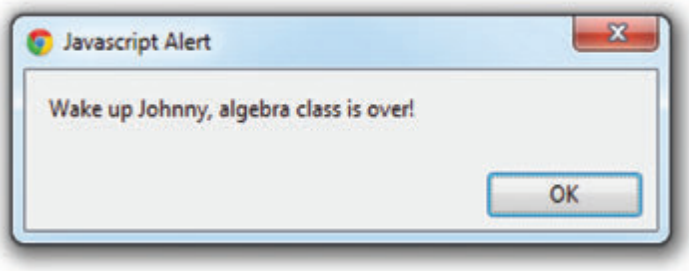
Name	Function	Example	Result
escape(string)	Encodes an ASCII string to ISO Latin-1 (ISO 8859-1), which is suitable for HTML processing	`escape("hello world!");`	hello%20 world%21
unescape(string)	Converts an ISO8859-1 character set to ASCII	`unescape("hello%20 world%21");`	hello world!
eval( )	Converts a string to integer or float value	`eval("10+5");`	15
isNaN(value)	Evaluates if a value is not a number. Returns false if the value is a number.	`isNaN("hello");`	true
typeof	Determines the type of a value	`typeof 10;`	number

Dialog Boxes

There are three JavaScript dialog boxes commonly used in webpages—the alert, confirm and prompt. Let's try each one:

- **Alert function** "**`alert( )`**"– displays an alert box with a message defined by the string message.

```
<script type="text/javascript">
   alert("Wake up Johnny, algebra class is over!");
</script>
```

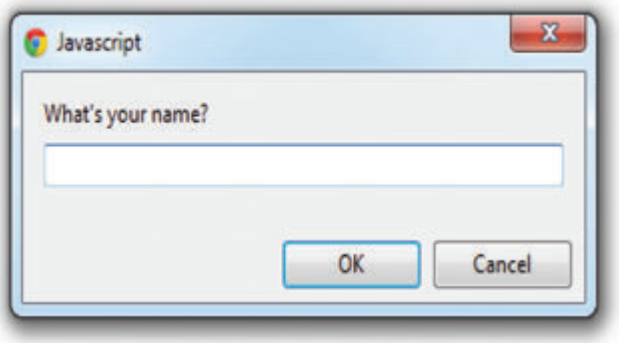


- **Confirm function** "**`confirm( )`**" – asks whether the user would continue or cancel the action. Here is an example:

```
if(confirm("Do you really want to wake up
Johnny?"))
{
document.write("OK, continuing to wake up
Johnny!");
}
else
{
document.write("Nah just let him sleep in.");
}
```

- **Prompt function** "**`prompt( )`**" – asks for an input from the user and passes it to the specified function. Here's an example:

```
<script type = "text/Javascript">
   var response = prompt("What's your name?");
   if (response != null)
   {
   document.write("Hi there" + response + "! Hope you like
   Javascript so far!");
   }
</script>
```



In this example, the user is prompted to input a value then choose OK or Cancel. If the user chooses Cancel, a NULL value is returned. If the user chooses OK, the string value entered in the field is returned.

Events

In most of our examples, we didn't give emphasis on when our functions or scripts are supposed to execute. For learning purposes, that is okay. However, in reality, you will need to determine when your scripts should run. You don't just run every script in your page anytime; it will confuse the browser, or worse, confuse your visitor.

Like we said earlier, a function does nothing unless it is called. This is where the event handler comes into play. You can have a dozen of the cool JavaScript functions on your page, but without event handlers to call them at the most appropriate time, they remain just a block of text that's not so cool.

Using event handlers, you can respond to an **event** (like when the page loads or unloads, when the user clicks on a link, or when the mouse hovers over an object) on your page. There are actually 18 event handlers that you can use to link your HTML elements to a piece of JavaScript. Learning how and when to use these event handlers will definitely improve the browsing experience on your webpages.

Event Handler	Event that it handles
onBlur	user has lost focus of an object like when clicking away a previously selected text field
onChanget	user has changed the object, then attempts to leave that field by clicking elsewhere
onClick	user clicked on the object
onDbClick	user clicked twice on the object
onFocus	user brought the focus to the object (i.e., clicked on it/tabbed to it)
onKeydown	a key was pressed over an element

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Identify the type of event handler for the following statements. Write your answer on the space provided.

_____1. The mouse/pointing device was released after being pressed down.

_____2. User has changed the object, then attempts to leave that field by clicking elsewhere.

_____3. A key was pressed over an element.

_____4. User has lost focus of an object like when clicking away a previously selected text field.

_____5. User submitted a form.

_____6. The cursor moved off the object.

_____7. The object has loaded.

_____8. User clicked twice on the object.

_____9. User selected some or all of the contents of the object. For example, the user selected some text within a text field.

_____10. A key was pressed over an element then released.

B. Encircle the letter of your answer.

1. This is a self-contained piece of code that performs a particular "function" when it is called.

a. event b. function c. argument d. event handler

2. It provides additional information needed by the function to process.

a. event b. function c. argument d. event handler

3. This lets you respond to an event on your page.

a. events b. function c. argument d. event handler

4. This dialog box displays an alert box with a message defined by the string message.

a. Alert function b. Prompt function c. Predefined function
d. Confirm function

5. This dialog box asks for an input from the user and passes it to the specified function.

a. Alert function b. Prompt function c. Predefined function
d. Confirm function

6. This dialog box asks whether the user would continue or cancel the action.

a. Alert function b. Prompt function c. Predefined function
d. Confirm function

7. This converts an ISO8859-1 character set to ASCII.

a. escape(string) b. eval( ) c. unescape(string) d. isNaN(value)

8. This is used to indicate that a value is not a number.

a. escape(string) b. eval( ) c. unescape(string) d. isNaN(value)

9. This function converts a string to integer or float value.

a. escape(string) b. eval( ) c. unescape(string) d. isNaN(value)

10. This encodes an ASCII string to ISO Latin-1 (ISO 8859-1), which is suitable for HTML processing.

a. escape(string) b. eval( ) c. unescape(string) d. isNaN(value)

Event Handler	Event that it handles
onKeyup	a key was released over an element
onKeypress	a key was pressed over an element then released
onLoad	the object has loaded
onMouseover	the cursor moved over the object (i.e., user hovers the mouse over the object)
onMousedown	the cursor moved over the object and the mouse/pointing device was pressed down
onMouseup	the mouse/pointing device was released after being pressed down
onMousemove	the cursor moved while hovering over an object
onMouseout	the cursor moved off the object
onReset	user has reset a form
onSelect	user selected some or all of the contents of the object. For example, the user selected some text within a text field.
onSubmit	user submitted a form
onUnload	user left the window (i.e., user closes the browser window)

A function is a self-contained piece of code that performs a particular "function". It works only when it is called in your script. JavaScript has a lot of built-in functions that can be readily used by JavaScript programmers. But event handlers are needed to call these functions at the right time. With event handlers, you can respond to an event when the user clicks on a link or when the mouse hovers over an object on your page.

LESSON 15

JavaScript Objects

At the end of this lesson, the student will be able to:

1. **Describe what a JavaScript Object is.**
2. **Explain the functions of the different types of objects in JavaScript.**
3. **Create objects using JavaScript.**

After learning the basics of the JavaScript language in the previous lessons, you can then advance to the next level in this lesson on objects. Specifically, you will learn how to create JavaScript objects.

What Is a JavaScript Object?

In the previous lessons, we have already seen how versatile JavaScript is in applying your desired functions for your page, but we have yet to explore JavaScript's full capabilities.

Like what we've emphasized in the first lesson, JavaScript is a powerful object-based language. It presents, or even lets you create many things in the form of objects, and lets you manipulate them to your needs. Hence, it can be said that a large proportion of JavaScript's capabilities lies in how it manipulates objects.

A **JavaScript Object** is the coding representation of objects that are used in your page or in your browser.

So why use objects in JavaScript? Here are some key reasons:

- **Increased code reuse** – reuse of objects makes your coding work easier. Especially in large scripts, you can create JavaScript objects that can be reused across pages.
- **Easier maintenance** – because you can use objects across pages, you don't have to rewrite your objects for every page in your website. This also means that when you need to make revisions on your code, you just need to revise one code and all changes will apply to every page in your site.
- **Increased extensibility** – you can increase the functionality of your new codes by simply taking existing objects and adding or revising a few codes or revising some scripts.

JavaScript Object Creation and Use

Before we actually start creating and using objects in JavaScript, you need to remember these important things:

- **Objects have properties. Properties** are the attributes of your JavaScript object. Pre-existing objects have default properties even while you set these for objects that you create. Here is an example of a property:

```
document.bgColor = "red";
```

You don't have to try this code. We'll do a lot of these in the succeeding lessons. The point of this example is that in this code, we are setting the property *bgColor* of your document (your page) which is white by default.

- **Objects also have methods.** Methods are the things that your object can do. Here is an example:

```
document.write("The method <b>\"write\"</b> writes this
text to the page");
```

In this example, we also used the object called document, which is your page. However, this time we used the method called *write()* to make our object write something or output something.

- **Event handlers** specify when your object is supposed to do something or when your objects are supposed to apply your preferred attributes. Here is an example:

```
<a href="javascript:void(0); onClick="document.bgColor =
'red';">change color</a>
```

This code will change the color of your page when the user clicks on the link.

At this point, it is important to know that JavaScript has three main types of objects by source:

- **Language Objects** – these are provided by the language and are not dependent on other objects. One example of this is the JavaScript Date object. This function is provided by the language or has been built-in in JavaScript.

- **Navigator Objects** – these are provided by the client browser. These objects are all sub-objects to the navigator object. One example is the Window object.

- **Objects created by the programmer** – these objects are defined by the programmer to perform the intended action.

In our previous examples, we are merely taking pre-existing objects, specifically the navigator object called *document*, which is your page, and setting its properties or methods. Basically, we use the following syntax for this:

```
//for specifying the object methods
//example: document.write(); or window.close();

theobject.themethod();


// for specifying the object properties
example: document.bgColor = "red";
theobject.theproperty
```

This is called the object property reference. Objects and their properties are normally referenced using a dot between each object or its property or method.

This syntax applies to both predefined and created objects. However, creating an object is totally different from calling a predefined object.

So how do we create our own object?

Object Creation

Creating an object in JavaScript is quite easy. Generally, objects may be created using the following syntax:

```
var someVariable = new Object()
```

When JavaScript sees the *New* operator, it creates a new generic object. Using this syntax, you can create objects like Array, Function, Date, Number, Boolean, and String. In fact, most objects may be created this way except for the Math object which cannot be instantiated (that is, instance of it may not be created).

Let's create some objects:

```
//This will create an array
var myFruitsArray = new Array("mango","guava","apple","avocado");

//This will create a string
var greeting = new String("Good morning ladies and
gentlemen!");

//This will create a date
var today = new Date();

//This will create a number
var myNum = new Number(5+5);
```

To check whether these objects have been successfully created, let's output their values:

```
document.write(myFruitsArray+"<br>");
document.write(greeting+"<br>");
document.write(today+"<br>");
document.write(myNum+"<br>");
```

Now that you've learned how to create and use your own objects, you can do a lot of new cool stuff on you page. However, before going into creating a lot of JavaScript objects, it is also very important to know that JavaScript has a lot of predefined objects—those that are provided by the language and by the browser.

Getting familiar with these objects will definitely ease up your programming work.

Main Object: The Navigator Object

JavaScript, being an object-based language, represents a lot of things as objects. This includes your web browser, which is represented by the Navigator object.

The JavaScript **Navigator object** is the object representation of the client Internet browser or web navigator program that is being used, whether it is an Internet Explorer, Netscape Navigator, Opera or Mozilla or any other client browser.

So how do we use this Navigator object? Let's try this example. Write what follows down with Notepad:

```
<script style="text/javascript">
<!--

var thisBrowser = navigator.appName;

document.write("You are using "+thisBrowser+"!");

//-->
</script>
```

This example, called the object *navigator*, uses the property *appName* to determine the name of the browser that we are using. Why is this important? One of the uses of this is for customizing or optimizing the page for certain browsers. This is handy because not all your visitors use the same kind of browser. Hence, optimizing your page for these types of browser will enhance the browsing experience of your visitors.

Note that there are other Navigator object properties which you may find useful for your page.

Navigator Object Properties

Property	Specifics
appCodeName	the name of the browser, such as "Mozilla"
appMinorVersion	the minor version number of the browser
appName	the name of the browser such as "Netscape Navigator" or "Microsoft Internet Explorer"
appVersion	the version of the browser. This may include information on compatibility and operating system name.
cookieEnabled	returns a true value, if cookie is enabled, and returns false, if not
cpuClass	the type of CPU
mimeTypes	an array of MIME type descriptive strings that are supported by the browser
onLine	returns a value of True if the browser is working online and False if working offline
platform	a description of the operating system platform like "Win32"
plugins	an array of plug-ins supported by and installed on the browser
systemLanguage	the language being used such as "en-us"

Property	Specifics
userLanguage	the language being used such as “en-us”
userAgent	describes the browser-associated user agent header like “Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322)”

The Navigator object also has some methods.

Navigator Object Methods

Method	Specifics
javaEnabled()	returns a Boolean value of “true” or “false” telling if the browser has JavaScript enabled
taintEnabled()	returns a Boolean value of “true” or “false” telling if the browser has tainting enabled. Tainting is basically a security protection mechanism for data.

These methods are a bit confusing because, as we observed in our previous examples, methods indicate an action. In the list above, the methods construct is more like that of a property. Nevertheless, they have the () with them, so they become distinct, just like some built-in JavaScript functions including alert(), prompt(), confirm(), date(), and close().

Let’s try an example:

```
<script style="text/javascript">
<!--

var thisBrowser = navigator.taintEnabled();

document.write(thisBrowser);

//-->
</script>
```

Navigator Objects

As mentioned earlier, the Navigator object is the highest level JavaScript object. Hence, there are other objects under this:

- **JavaScript MimeType Object** – this is a property of the Navigator object

Property	Specifics
description	the description of the MIME type
enabledPlugin	a plug-in object for the MIME type
type	the name describing the MIME type like for example "text/html"
suffixes	a list of file name extensions associated with the mime type. If there are multiple names, commas are used as separator.

- **JavaScript Plugin Object** – this is a property of the Navigator object

Property	Specifics
description	the description of the plug-in
filename	the filename of the plug-in
length	quantity of plug-ins associated MimeType objects
Name	name of the plug-in

- **JavaScript Window Object** – corresponds to the web browser window.

Window Object

This is the highest level JavaScript object and represents the web browser window.

Window Object Properties

Property	Specifics
closed	Boolean value that indicates whether the window is closed
defaultStatus	the default message that is loaded into the status bar when the window loads
length	the number of frames that the window contains
name	the window name
opener	the object that caused the window to open
status	pertains to the bar on the lower left side of the browser. This is used to display temporary messages.

One of the more interesting properties of the window object is the status. Have you seen a custom message in a status bar when you mouse over a link? Well, using this property, we can make our own custom message displayed on the status bar. Let's try the example that follows:

```
<a href="javascript:void(0);" onMouseOver="window.status='This
link will go nowhere';return true">some link</a>
```

This example displays a custom message at the status bar when the mouse hovers over the link.

Window Object Methods

Property	Specifics
`alert("message")`	displays message on the alert dialog box
`blur()`	takes the focus away from the window
`clearInterval(interval)`	clears the timeout interval
`clearInterval(timeID)`	removes a repetitive timeout that was set up using the setInterval function
`clearTimeout(timeID)`	removes a timeout that was set using the setTimeout function
`close()`	closes the current or the specified window
`confirm("message")`	displays message in the confirm dialog box
`focus()`	gives the focus to the window
`moveBy(x,y)`	moves the window by specified number of pixels in the X and Y direction
`moveTo(x,y)`	moves the window to a specific location in the browser
`open("URLname","Windowname", ["options"])`	opens and names a new window. This is basically how to create a pop-up window
`prompt("message","defaultmessage")`	displays a message in the prompt dialog box
`resizeBy(X,Y)`	adjusts the window size relative to the current value
`resizeTo(X,Y)`	adjusts the window size to set X and Y width and height values
`scroll(X,Y)`	makes the window scroll to the specified location
`scrollBy(X, Y)`	makes the window scroll by specified number of pixels from the current position

Property	Specifics
`setInterval(function(), milliseconds,[functargs])`	puts a timing interval for a function to be called
`setTimeout(function(), milliseconds,[functargs])`	used to call a function after the specified time in milliseconds

The window object methods are among the most interesting methods in JavaScript. Want to know why? Let's try an example:

```
<a href="javascript:void(0);" onClick="window.blur();">some link</a>
```

In the previous example, we used the event handler *onClick* so that when you click on the link, the function blur will be called. Notice that upon clicking on the link, your page disappears. It does not close the document but just takes it away from the focus.

Here's another cool thing that you can create from the document method *open()*. It is the pop-up window. Let's create our own pop-up window. To do this, first we need to create another page. Let's create one:

```
<html>
<head><title>My Pop Up</title>
<head>
<body>

This is my pop-up window

</body>
</html>
```

Save this HTML file as *pop.html* on the same folder as your existing page for ease of retrieval using the *open()* window method. To call this pop-up window, create the following code on your existing page:

```
<a href="javascript:void(0);"
onClick="open('pop.html','test','toolbar=no,menubar=no,width=200,
height=200,resizable=yes');">some link</a>
```

What will be opened is a small window without the other toolbars. You can actually turn on, or turn off, some of the toolbars and other window options. Consider this list of window options:

Pop-up Options	Description
alwaysRaised	if "yes," the created window is raised
directories	"yes" shows the directory buttons
height	the window height in pixels
left	the distance in pixels of the new window from the left side of the screen
location	"yes" displays the address line
menubar	"yes" displays the menu bar
outerWidth	the outer window width in pixels
outerHeight	the outer window height in pixels
resizable	"yes" allows the window to be resized
Status	"yes" displays the status bar
Scrollbars	"yes" allows the scroll bar to be displayed
Toolbar	"yes" displays the toolbar
Top	the distance in pixels of the new window from the top of the screen
Width	the window width in pixels
z-lock	"yes" puts the created window in the background

Window Object Events

Here is a comprehensive list of events that may be used to set when a method or function is to take place.

- onafterupdate
- onBeforeunload
- onBeforeupdate
- onBlur
- onClick
- ondblclick
- onError
- onerrorupdate
- onFocus
- onhelp
- onkeydown
- onkeypress
- onkeyup
- onLoad
- onmousedown
- onmousemove
- onmouseout
- onmouseover
- onmouseup
- onragstart
- onreadystatechange
- onresize
- onrowenter
- onrowexit
- onscroll
- onselectstart
- onUnload

JavaScript lets you create many things in the form of objects which can be manipulated according to your needs. There are three main objects by source, namely language objects, navigator objects, and objects created by the programmer. You can create objects like Array, Function, Date, Number, Boolean, and String. There are also lots of JavaScript predefined objects that are provided by the language and by the browser.

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Identify which Window Object Property is described. Write your answer on the blank below the description.

1. makes the window scroll to the specified location

   _____

2. removes a time out that was set using the setTimeout function

   _____

3. gives focus to the window

   _____

4. opens and names a new window

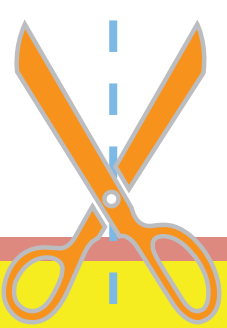
   _____

5. closes the current or the specified window

   _____

B. Identify which Pop-up option is being described.

1. allows the window to be resized _____
2. puts the created window in the background _____
3. diplays the address line _____
4. the outer window width in pixels _____
5. show directory buttons _____



C. Describe the properties of each of the following Navigator Objects.

1. JavaScript MimeType Object

Description _____

enabledPlugin _____

Type _____

Suffixes _____

2. JavaScript Plugin Object

Description _____

Filename _____

Length _____

Name _____

D. Enumerate the 3 three main types of JavaScript object by source. Give a brief explanation of each type of object.

1. _____
_____
_____
_____

2. _____
_____
_____
_____

3. _____
_____
_____
_____

LESSON 16

JavaScript Independent Objects

At the end of this lesson, the student will be able to:

1. **Identify the different independent objects in JavaScript.**
2. **Explain the functions of each independent object in JavaScript.**
3. **Apply the different independent objects in a sample program.**

This lesson deepens your understanding of JavaScript by introducing you to some of the most fundamental and useful objects in JavaScript.

Array Object

Arrays

Arrays are a fundamental part of most programming languages and scripting languages. They are basically an ordered stack of data with the same data type.

Using arrays, you can store multiple values under a single name. Instead of using a separate variable for each item, you can use one array to hold all of them.

So, for example, you have several fruit names that you want to store and display on the screen. You could store this in a simple variable like this:

```
myFruit0 = "mango";
myFruit1 = "guava"
myFruit2 = "apple"
```

This notation will work out fine. However, one disadvantage of using this approach is that you need to write out each variable name whenever you need to work with them.

Moreover, you can't loop through all your variables. That's where arrays come in handy. With arrays, you can put all your fruits in one "basket."

So, to improve the code above, we can do something like this:

```
var myFruit = new Array(3)
myFruit[0] = "mango"
myFruit[1]= "guava"
myFruit[2] = "apple"
```

Like most other programming languages, JavaScript creates arrays by first assigning an array object to a variable name. In this example, we assigned the array object to the variable *myFruit*. Then, we assigned values to each element contained in the array. Note that we started at an index number of zero in our fruits array. JavaScript starts arrays at zero (however, note that there are some languages that start at 1).

We can access an element in our array by calling the array name and indicating the index number of the element. Let's try this example:

```
<script style="text/
javascript">
<!--
var myFruit = new Array(3)
myFruit[0] = "mango"
myFruit[1]= "guava"
myFruit[2] = "apple"

document.write(myFruit[1]);

//-->
</script>

Result:
guava
```

The code above displays the value of the second element in the array, which is "guava".

However, it may seem that it would take you as much time or even more to write an array this way compared to the normal way of assigning values to variables.

Hence, here is a shorthand version of specifying an array:

```
<script style="text/javascript">
<!--
var myFruit = new
Array("mango","guava","apple");

//-->
</script>
```

To access an element within the array, we just need to use the same code we used earlier:

```
<script style="text/javascript">
<!--
var myFruit = new
Array("mango","guava","apple");
document.write(myFruit[1]);

//-->
</script>

Result:
guava
```

However, if we are just going to output an element from an array, there's not much utility. So, in the succeeding discussions, let's treat array as an object in JavaScript. By doing so, we can do a lot of other things to manipulate the array.

JavaScript Array Object Methods

Method	Specifics
chop()	used to truncate the last character of all strings elements of an array
concat()	used to join two or more arrays. It does not change the existing arrays and only returns a copy of the joined arrays.
grep(searchstring)	searches within an array and returns all elements that contain the specified search string
join(delimiter)	puts all elements in the array into a string, separating each element with the specified delimiter
pop()	pops the last string off the array
push(strings)	places the string elements at the end of the array

Method	Specifics
reverse()	reverses the order of the elements in an array
shift()	decreases array element size by one by shifting the first element off the array
sort()	sorts the array elements in dictionary order or using a compare function passed on to the method
splice()	replaces the elements in an array with specified values. Returns the elements that were replaced.
split(deliimiter)	splits a string using the specified delimiter and returns the delimited strings in an array
unshift()	places elements at the start of an array

Let's try some of these methods. First, let's try the *join(delimiter)*. Write this on your Notepad:

```
<script style="text/javascript">
<!--
var myFruit = new
Array("mango","guava","apple");
var myString = myFruit.join("/");
document.write(myString);

//-->
</script>

Result:
mango/guava/apple
```

In this example, we used the delimiter "/" to separate each element from the array in the output string.

Let's try another one, the *pop()* method.

```
<script style="text/javascript">
<!--
var myFruit = new
Array("mango","guava","apple");
var eaten = myFruit.pop();
document.write(eaten);
```

```
//-->
</script>

Result:
apple
```

In this example, the apple was the one "eaten" or popped off the array because it is at the last array element in the expression.

Another interesting method is the *split()*. Let's try an example. Let's say you have a string like this:

```
var myString= new String("mango::guava::apple");
```

Let's split this string into array elements by doing a *split()*:

```
var myString= new
String("mango::guava::apple");
myArray = myString.split("::");
document.write(myArray);

Result:
mango,guava,apple
```

The result is an array with which you can actually access each of the elements.

Date Object

JavaScript provides you with the ability to access the date and time of your user's local computer, which can be pretty cool in some cases. This is possible through the JavaScript date object.

First, let's learn how to get the date from your computer.

```
<script style="text/javascript">
<!--

var currentDate = new Date();
var day = currentDate.getDate();
var month = currentDate.getMonth();
var year = currentDate.getYear();
document.write(day + "/" + month + "/" +
year)

//-->
</script>
```

Look at the example above. The Date() construct that we used uses the current date and time on your machine to create an instance of the object Date.

In the example, the fun part is where we started getting the date values from our computer using the date methods. We have a number of methods for this object:

Method	Specifics
getDate()	returns the day of the month. The value is any number between 1 and 31.
getDay()	returns the day of the week as a value from 0 (Sunday) to 6 (Saturday)
getHours()	returns the hour as a value from 0 through 23
getMinutes()	returns the minutes as a value from 0 through 59
getMonth()	returns the month as a value from 0 through 11
getSeconds()	returns the seconds as a value from 0 through 59
getTime()	the number of milliseconds since January 1, 1970
getTimeZoneOffset()	time zone offset in hours which is the difference between GMT and local time
getYear()	returns the numeric four digit value of the year
parse()	the number of milliseconds after midnight January 1, 1970 till the given date expressed as a string
setDate(value)	sets the day of the month in the date object as a value from 1 to 31
setHours(value)	sets the hours in the date object with a value of 0 through 59
setMinutes(value)	sets the minutes in the date object with a value of 0 through 59
setMonth(value)	sets the month in the date object as a value of 0 through 11
setSeconds(value)	sets the seconds in the date object with a value of 0 through 59
setTime(value)	sets time on the basis of number of milliseconds since January 1, 1970

Method	Specifics
setYear(value)	sets the year in the date instance as a 4-digit numeric value
toGMTString()	converts date to GMT format
toLocaleString()	converts date to local time zone format
UTC()	based on a comma delimited string, the number of milliseconds after midnight January 1, 1970 GMT is returned

Math Object

Unlike other objects, **Math object** in JavaScript can be invoked without necessarily creating an instance of it. Hence, no need to declare new *Object()*.

The Math object's properties usually represent constant values like *pi* and Euler's constant. These constant values are the properties of the Math object. Here's a list:

Property	Specifics	Value
E	Euler's constant	2.718281828459045
LN2	natural log of the value 2	0.6931471805599453
LN10	natural log of the value 10	2.302585092994046
LOG2E	base 2 log of Euler's constant	1.4426950408889633
LOG10E	base 10 log of Euler's constant	0.4342944819032518
PI	number of radians in a 360-degree-circle	3.141592653589793
SQRT1_2	square root of one half	0.7071067811865476
SQRT2	square root of 2	1.4142135623730951

Let's try an example:

```
<script style="text/
javascript">
<!—

document.write(Math.PI);

//-->
</script>
```

In this example, we did not declare a new object because we already know that Math objects can be invoked without creating an instance of it. But of course, you can always assign the value of the object to a variable, like in the example that follows:

```
<script style="text/
javascript">
<!–

var myValue = MathPI;
document.write(myValueI);

//-->
</script>
```

The interesting part is the Math object methods. In the previous discussion, all we had were constant values. But in the succeeding discussion about Math object methods, we'll have the freedom to specify our values with which to do our operations. Below is a list of the methods that we can use with Math objects:

Method	Specifics	Example
abs(a)	returns the absolute value of a	Math.abs(-10) = 10
ceil(x)	rounds up the value of "a" to the next integer	Math.ceil(10.34) = 11
cos(a)	returns the cosine of "a" specified in radians	Math.cos(90*Math.PI/180) = 0
exp(a)	returns Euler's constant to the power of the passed argument	Math.exp(2) = 7.38905609893065
floor(a)	rounds the passed value down to the next lowest integer	Math.floor(10.89) = 10
log(a)	returns the natural log of the passed value	Math.log(2) = 0.6931471805599453
max(a,b)	returns the larger value of a or b	Math.max(7,10) = 10
min(a,b)	returns the lower value of a or b	Math.min(7,10) = 7
pow(a,b)	returns the value of a to the power b	Math.pow(2,4) = 16
random()	returns a random number between 0 and 1	Math.random() = 0.523441610759156
round(a)	rounds off the value of a to the nearest integer	Math.ceil(10.34) = 10
sin(a)	returns the sine of "a" specified in radians	Math.sin(90 * Math.PI/180) = 1
sqrt(a)	returns the square root of "a"	Math.sqrt(64) = 8
tan(a)	returns the tangent of a value in radians	Math.tan(0) = 0

Number Object

The number object's properties include the following:

Property	Specifics	Value
MAX_VALUE	largest value that may be used in JavaScript	1.7976931348623157e+308
MIN_VALUE	smallest value that may be used in JavaScript	5e-324

NaN	used to indicate that a value is not a number	Example:
		`if (Month < 1 || Month > 12) {`
		`  Month = Number.NaN`
		`}`

NEGATIVE_INFINITY	value returned if a negative overflow occurs	
POSITIVE_INFINITY	value returned if a positive value overflow occurs	

In order to use the number object, an instance of it must be created.

The Number object has no specific functions. However, there are associated built-in functions. Some of them are listed below:

Method	Specifics	Example
eval()	evaluates string expressions	`"5+7" = 12`
isNaN()	returns a Boolean indicating whether a certain value is a number	`isNaN(10) = false`
parseFloat()	used to convert a string to a decimal value	`parseFloat("10.3456") = 10.3456`
parseInt()	used to convert a string to an integer value	`parseInt("10.3456") = 10`
toString()	converts an object to a string	

String Object

As you may already know, a **string** is a series of characters. JavaScript provides a rich collection of useful methods or functions that we can use to manipulate strings. The following table provides a comprehensive list:

Method	Specifics
anchor(anchorName)	displays a string as an anchor
big()	displays the string in large format
blink()	makes the string blink
bold()	displays the string in bold
charAt(index)	returns a character from the specified location within a string
fontcolor(color)	specifies the color of the string when displayed
fontsize(size)	specifies the font size of the string when displayed
indexOf(pattern)	returns the index location of the first pattern encountered within a string
indexOf(pattern, index)	returns the index location of the first pattern encountered within a string. Starts search from the specified index.
italics()	displays the string in italics
lastIndexOf(pattern)	returns the index location of the last pattern encountered within a string
lastIndexOf(pattern, index)	returns the index location of the last pattern encountered within a string. Starts search from the specified index.
link(href)	displays string as a hypertext link
small()	displays the string in small format
split(separator)	splits a string into substrings using the specified separator character and returns the substring in array
strike()	displays the string in strikethrough format
sub()	displays the string as a subscript
substr(start, length)	returns a substring taken from a string beginning from the start index and continuing up to the specified length
substring(start, end)	returns a substring taken from a string beginning from the start index and continuing up to the specified end
sup()	displays the string as a superscript
toLowerCase()	converts the string to lowercase
toUpperCase()	converts the string to uppercase

Let's try some of these cool stuff, starting with indexOf():

```
<script style="text/javascript">
<!—
var myString = "This is cool";
var coolString = myString.
indexOf("i");
```

SCORE

NAME: _____

SECTION: _____

DATE: _____

A. Identify the term described in each item.

_____1. This is an object in JavaScript that can be invoked without necessarily creating an instance of it.

_____2. These are basically an ordered stack of data with the same data type.

_____3. This is a JavaScript array object method that splits a string using the specified delimiter and returns the delimited strings in an array.

_____4. This object has the ability to access the date and time of your user's local computer.

_____5. This sets the minutes in the date object with a value of 0 through 59.

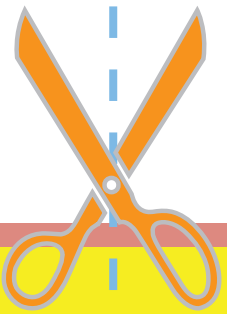
_____6. This string object returns a substring taken from a string beginning from the start index and continuing up to the specified end.

_____7. This is used to truncate the last character of all strings elements of an array.

_____8. This object has no specific functions, but has associated built-in functions.

_____9. This is a series of characters.

_____10. This method is used to return the last instance of the letter.



B. Match the items in column A with their descriptions in column B.

Column A	Column B
_____1. MAX_VALUE	a. time zone offset in hours which is the difference between GMT and local time
_____2. sqrt(a)	b. pops the last string off the array
_____3. italics()	c. largest value that may be used in JavaScript
_____4. getTimeZoneOffset()	d. value returned if a positive value overflow occurs
_____5. pop()	e. returns the square root of "a"
_____6. toLocaleString()	f. returns the sine of "a" specified in radians
_____7. sin(a)	g. used to indicate that a value is not a number
_____8. NaN	h. displays the string in italic
_____9. reverse()	i. reverses the order of the elements in an array
_____10. POSITIVE_INFINITY	j. converts date to local time zone format

```
document.
write(coolString);
//-->
</script>

Result:
2
```

In this example, we asked the script to search for the letter "i" in the given string and return its location within the string. Note that there are two instances of the letter "i" in the string. Our function is designed to return the first one it encounters. If you want to return the last instance of the letter, use the method *lastIndexOf()*.

Here's another cool example:

```
<script style="text/javascript">
<!—
var myString = "This is cool";
var coolString = myString.
substr("8,4");
document.write(coolString);
//-->
</script>

Result:
cool
```

In this example, the script goes over the string and searches for the start location. From this location, it captures the characters or the substrings up to the specified length, which, in the example, are 4 characters. Hence, the resulting substring is cool.

JavaScript's independent objects include the array, date, math, number and string objects. These objects have their own methods and properties that can be used to perform different functions or manipulate the objects.

L1

WWW - World Wide Web

Notepad - uses basic text formatting in order to make and edit text files

Browsers - used in viewing webpages

WYSIWYG - What You See Is What You Get

URL - Uniform Resource Locator, or the web address

HTML - HyperText Markup Language

Tags - HTML commands

HTTP - HyperText Transfer Protocol

XHTML - eXtensible HyperText Markup Language

Empty Tags - tags that do not have opening and closing tags

W3C - World Wide Web Consortium

L2

Headings - organize a document and distinguish different sections/topics

Nesting - placing one set of tags inside another set of tags

FILO - First In Last Out

Definition List - a list of terms with their corresponding definitions

Dead Link - a link that leads the user to a blank or error page

Download Link - a link that opens a download dialog box

External Link - a link either to a file or to a page that does not belong to the webpage's own site

Hyperlinks - clicked to access other files, webpages, or even a different part of the same webpage

Internal Link - a link to a file anywhere within the webpage's own site or to any part of the webpage

Links - pointers to the pages they reference, the most essential ingredients of the Web.

L3

Audience - is a group of people who are expected to visit your website

Usability - is a term used to denote the methods of measuring usability and the study of the principles behind an object's perceived efficiency or elegance

L4

Declaration - in webpage development, refers to the structure `property: value`

Deprecated - a deprecated element or attribute is one that has been outdated; it may become obsolete in the future, but browsers should continue to support it for backward compatibility

External Style Sheet - a separate document where all style sheet information are stored

Internal Style Sheet - stores style information in the `<head>` tag; no external file is required for the style sheet to work

In-line Style Sheet - stores information inside an HTML tag and, just like the internal style sheet, does not require an external file

Typeface - another name for font

Hexadecimal - a numbering system that is base-16 rather than our commonly used decimal numbering system of base-10

Escape Sequences - other characters with special meanings in HTML that cannot be used as they are in a text

Comments - written remarks in your HTML or CSS document which will not be displayed in the browser

L5

Data Tables - tables containing information that are organized into rows and columns

Layout Tables - tables containing data presented in such a way that information in a particular cell does not necessarily imply a relationship with information in other cells

Span - the tag defining the style of any in-line element

Div - the tag defining the style of any block element

In-line Element - an HTML element that does not cause the addition of preceding and following line feeds in the client browser (for example, or <i>); opposite of a block element (Zacker.com, 1997)

Block Element - an HTML element that is automatically preceded and followed by a line feed in the client browser (for example, <p> or <h1>); opposite of an in-line element (Zacker.com, 1997)

L6

Internal Style Sheet - resembles a summary of CSS commands located at the upper portion of the HTML document

Relative URL - gives the path from the current location of the page to the location of the destination page or resource (ArmmNet FAQ, 2002-2003)

Absolute URL - the complete path to a resource, independent of the location of the visited page (Lycos Glossary, 2003)

In-line Images - images inserted within a line of text

JPEG - from the *Joint Photographic Experts Group* that designed this format; allows up to 16 million different colors and is the best format used for photographs

GIF - *Graphics Interchange Format*; supports up to 256 different colors and is the best format used for illustrations

L7

Online Forms - electronic forms in webpages that allow users to enter information and send it back to the server

Spamming - sending unsolicited e-mail to Internet users

Web Servers - computers that store webpages

L8

Flash - a bandwidth-friendly and browser-independent vector graphic animation technology

Vector Graphics or **Geometric Modeling** - the use of geometrical primitives such as points, lines, curves, and polygons to represent images in computer graphics; it is used by contrast to the term raster graphics, which is the representation of images as a collection of pixels (dots) (Wikipedia, 2001-2005)

JavaScript - a scripting language designed for adding interactive features to HTML pages

Roll-over - the appearance of a link changes as you move the mouse over it; it consists of an image serving as a hypertext link (Raggett, 2000)

Hotspot - any clickable region in a webpage

CSS Tooltip - an aiding text that appears just when you roll on with the mouse (Psacake.com, 1998-2005)

L9

Meta Tags - tags that do not affect either the form or content of the webpage although they hold information about its contents

Meta-data - meta-information, or general information that can be useful for indexing and classifying your document

Search Engine - a program that searches documents for specified keywords and returns a list of the documents where the keywords were found (Webopedia, 2004)

Intellectual Property - a product of the intellect, such as an expressed idea or concept, that has commercial value (Plagiarism.org, 2003)

L10

Script - a set of computer instructions or commands that puts together or connects existing components to accomplish a new related task

Client-side scripting - execution of script is done at the client's computer

Server-side scripting - execution of script is done at the server

Methods - actions that objects know how to do

Object-Oriented Programming - a programming style that treats every element in a system as an object

L11

Syntax - a set of rules that determines how a specific language will be written and interpreted

Variables - are containers that contain a value, which can change as required

L12

Operator - something that you feed with one or more values to yield another value

Expression - anything that has a value

Modulus - the remainder of a division

Assignment Operators - used to both assign and perform math operators

Comparison Operators - used to compare two values

Logical Operator - evaluates whether a certain statement is true or false but must make decisions based on one or more truth values

String Operators - used to perform simple operations on a string or a piece of text

L13

If Statement - conditional statement construct that evaluates a condition to True or False. It then runs a specific code depending on the result of this evaluation.

If Else statement - similar to the If statement, except that an alternative instruction is given in case the argument isn't True

If Else If statement - refers to an If statement that depends on another If statement

Loops - sets of instructions used to repeat the same piece of code until a specified condition is met or if the condition evaluates to True or False depending on how you want it

While Loop - executes code while the condition is true

L14

Function or Method - a self-contained piece of code that performs a particular "function" when it is called

Argument - provides additional information needed by the function to process

Event - refers to an action on the page, like when the page loads or unloads, when the user clicks on a link, or when the mouse hovers over an object

L15

JavaScript Object - the coding representation of objects that are used in your page or in your browser

Properties - the attributes of a JavaScript object

Event Handler - specifies when an object is supposed to do something or when objects are supposed to apply preferred attributes

Navigator Object - the object representation of the client Internet browser or web navigator program that is being used

L16

Arrays - an ordered stack of data with the same data type

Date Object - provides the ability to access the date and time of a user's local computer

Math Object - an object which can be invoked without necessarily creating an instance of it

String - a series of characters

techFactorsInc

Making learning a great experience

In a fast-changing society, computers and the Internet make getting through the daily business of life a lot easier. In terms of communication, they are a must in this Age of Information. TechFactors recognizes the need for an integrated learning system where information technology education is delivered in a practical manner.

With the TechFactors DigiTITANS and I.C.Topia series for grade school and high school levels, students learn about computers and the Internet through actual laboratory interactive exercises, quickening comprehension and heightening their learning curve using the kind of proprietary and open source technology they are being taught in combination with what's required by the Department of Education (DepEd).

Both DigiTITANS and I.C.Topia allow learning in the context of available applications that are commonly used or preferred depending on the operating system used and what's installed. This dual system of teaching delivers practical competency in the use of hardware and software combinations. Overall, both courseware lines allow expectancies in demonstration and decision-making for student development.

ISBN 978-971-0550-37-1

9 789710 550371 >