I2201 cour

By Dr Iyad Zayour

Lebanese University- Faculty of Science

2018

Introduction

- History of internet from DARPA to WWW
    - Connection and connectionless communication
    - Voice and data network
-
- Definition of a communication protocol

- Definition of HTML,URL, HTTP, FTP, WWW,

- Web server (e.g. apache) and web browser
- Desktop versus web programming (richness)
- Web site
- Web designer (dreamweaver)
- Web browser
- Client side/ server side
- DNS

Designing a web site and going online :

- Starts at a HTML editor, text editor or web designer
- Test using a browser
-

HTTP: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.

IP

Hosting

W3C: consortium made up of member organizations which maintain full-time staff for the purpose of working together in the development of standards for the World Wide Web.

HTML

Example:

<HTML>

<HEAD>

<TITLE>Lebanese University</TITLE>

</HEAD>

<BODY>

This is what is displayed.

</BODY>

</HTML>

Colors

The BODY element of a web page is an important element in regards to the page's appearance. Here are the attributes of the BODY tag to control all the levels:

TEXT="#RRGGBB" to change the color of all the text on the page (full page text color.)

This element contains information about the page's background color, the background image, as well as the text and link colors

The TEXT attribute is used to control the color of all the normal text in the document. The default color for text is black. The TEXT attribute would be added as follows:

<BODY BGCOLOR="#FFFFFF" TEXT="#FF0000"></BODY>

In this example the document's page

color is white and the text would be red.

These attributes control the colors of the different link states:

1. LINK – initial appearance – default = Blue.

2. VLINK – visited link – default = Purple.

3. ALINK –active link being clicked–default= Yellow.

The Format for setting these attributes is:

```
<BODY BGCOLOR="#FFFFFF" TEXT="#FF0000" LINK="#0000FF"
      VLINK="#FF00FF"
     ALINK="FFFF00"></BODY>
```

The BODY element also gives you ability of setting an image as the document's background.

An example of a background image's HTML code is as follows:

```
<BODY BACKGROUND="hi.gif" BGCOLOR="#FFFFFF"></BODY>
```

Inside the BODY element, heading elements H1 through H6 are generally used for major divisions of the document. Headings are permitted to appear in any order, but you will obtain the best results when your documents are displayed in a browser if you follow these guidelines:

H1: should be used as the highest level of heading, H2 as the next highest, and so forth.

You should not skip heading levels: e.g., an H3 should not appear after an H1, unless there is an H2 between them.

<p>

Paragraphs allow you to add text to a document in such a way that it will automatically adjust the end of line to suite the window size of the browser in which it is being displayed. Each line of text will stretch the entire length of the window.

<BR> Break

- Line breaks allow you to decide where the text will break on a line or continue to the end of the window.
- A <BR> is an empty Element, meaning that it may contain attributes but it does not contain content.

The <BR> element does not have a closing tag

The <HR> element causes the browser to display a horizontal line (rule) in your document.

<HR> does not use a closing tag, </HR>.

Horizontal Rule, <HR> attributes:

Attribute	**Description**	**Default Value**
SIZE	Height of the rule in pixels	2 pixels

WIDTH	Width of the rule in pixels or percentage of screen width	100%
NOSHADE	Draw the rule with a flat look instead of a 3D look	Not set (3D look)
ALIGN	Aligns the line (Left, Center, Right)	Center
COLOR	Sets a color for the rule (IE 3.0 or later)	Not set

Character Formatting

- **<FONT SIZE="+2">** Two sizes bigger**</FONT>**
- The size attribute can be set as an absolute value from 1 to 7 or as a relative value using the "+" or "-" sign. Normal text size is 3 (from -2 to +4).
- **<B> Bold </B>**
- **<I>*Italic*</I>**
- **<U> Underline </U>**
- Color = "#RRGGBB" The COLOR attribute of the FONT element. E.g., **<FONT COLOR="#RRGGBB">this text has color</FONT>**

<PRE> Preformatted </PRE> Text enclosed by PRE tags is displayed in a mono-spaced font. Spaces and line breaks are supported without additional elements or special characters

<P><FONT SIZE="+1"> One Size Larger </FONT> - Normal –

<FONT SIZE="-1"> One Size Smaller </FONT><BR>

<B> Bold</B> - <I> italics</I> - <U> Underlined </U> -

<FONT COLOR="#FF0000"> Colored </FONT><BR>

<EM> Emphasized</EM> - <STRONG> Strong </STRONG> - <TT> Tele Type </TT><BR>

Alignment

Some elements have attributes for alignment (ALIGN) e.g. Headings, Paragraphs and Horizontal Rules.

The Three alignment values are : LEFT, RIGHT, CENTER.

<CENTER></CENTER> Will center elements.

<DIV ALIGN="value"></DIV> Represents a division in the document and can contain most other element type. The alignment attribute of the DIV element is well supported.

<TABLE></TABLE> Inside a TABLE, alignment can be set for each individual cell.

Headings

H1,H2 …

Lists

HTML supplies several list elements. Most list elements are composed of one or more <LI> (List Item) elements.

UL : Unordered List. Items in this list start with a list mark such as a bullet. Browsers will usually change the list mark in nested lists.

```
<UL>

<LI> List item …</LI>

<LI> List item …</LI>

</UL>
```

- List item …
- List item …

You have the choice of three bullet types: disc(default), circle, square.

These are controlled in Netscape Navigator by the "TYPE" attribute for the <UL> element.

```
<UL TYPE="square">

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

</UL>
```

- List item …
- List item …

- List item …

You have the choice of three bullet types: disc(default), circle, square.

These are controlled in Netscape Navigator by the “TYPE” attribute for the <UL> element.

OL: Ordered List. Items in this list are numbered automatically by the browser.

```
<OL>

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

</OL>
```

1. List item …
2. List item …
3. List item

You have the choice of setting the TYPE Attribute to one of five numbering styles.

▪ You have the choice of three bullet types: **disc(default), circle, square.**

▪ These are controlled in Netscape Navigator by the “TYPE” attribute for the <UL> element.

```
<UL TYPE=“square”>

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

</UL>
```

▪ List item …

▪ List item …

▪ List item …

TYPE	Numbering	Styles
1	**Arabic numbers**	**1,2,3, ……**
a	**Lower alpha**	**a, b, c, ……**
A	**Upper alpha**	**A, B, C, ……**
i	**Lower roman**	**i, ii, iii, ……**
I	**Upper roman**	**I, II, III, ……**

You can specify a starting number for an ordered list.

<OL TYPE ="i">

<LI> List item …</LI>

<LI> List item …</LI>

</OL>

<P> text ….</P>

<OL TYPE="i" START="3">

<LI> List item …</LI>

</OL>

Ordered List. Items in this list are numbered automatically by the browser.

Nested lists

You can nest lists by inserting a UL, OL, etc., inside a list item (LI).

EXample

<UL TYPE = "square">

<LI> List item …</LI>

<LI> List item …

<OL TYPE="i" START="3">

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

<LI> List item …</LI>

</OL>

</LI>

<LI> List item …</LI>

</UL>

Images

<IMG>This element defines a graphic image on the page.

Image File (SRC:source): This value will be a URL (location of the image) E.g. http://www.domain.com/dir/file.ext or /dir/file.txt.

Alternate Text (ALT): This is a text field that describes an image or acts as a label. It is displayed when they position the cursor over a graphic image.

Alignment (ALIGN): This allows you to align the image on your page.

Width (WIDTH): is the width of the image in pixels.

Height (HEIGHT): is the height of the image in pixels.

Border (BORDER): is for a border around the image, specified in pixels.

HSPACE: is for Horizontal Space on both sides of the image specified in pixels. A setting of 5 will put 5 pixels of invisible space on both sides of the image.

VSPACE: is for Vertical Space on top and bottom of the image specified in pixels. A setting of 5 will put 5 pixels of invisible space above and below the image.

Links

Click <A HREF="http://www.yahoo.com">here</A>to go to yahoo.
LINK - standard link - to a page the visitor hasn't been to yet. (standard color is blue - #0000FF).
VLINK - visited link - to a page the visitor has been to before. (standard color is purple - #800080).
ALINK - active link - the color of the link when the mouse is on it. (standard color is red - #FF0000).

Internal Links : Links can also be created inside large documents to simplify navigation.

<A NAME=“bookmark_name”></A>

<A HREF=“#bookmark_name”>Go To Book Mark</A>

Emails:

<A HREF=“mailto:kmf@yahoo.com”>Send meMore Information </A>

Tables

- The <TABLE></TABLE> element has four sub-elements:

1. Table Row<TR></TR>.
2. Table Header <TH></TH>.
3. Table Data <TD></TD>.
4. Caption <CAPTION></CAPTION>.

The table row elements usually contain table header elements or table data elements

```
<table border=“1”>
<tr>
<th> Column 1 header </th>
<th> Column 2 header </th>
</tr>
<tr>
<td> Row1, Col1 </td>
<td> Row1, Col2 </td>
</tr>
<tr>
<td> Row2, Col1 </td>
```

```
<td> Row2, Col2 </td>
</tr>
</table>
```

Tables Attributes

- **BGColor:** Some browsers support background colors in a table.
- **Width:** you can specify the table width as an absolute number of pixels or a percentage of the document width. You can set the width for the table cells as well.
- **Border:** You can choose a numerical value for the border width, which specifies the border in pixels.
- **CellSpacing:** Cell Spacing represents the space between cells and is specified in pixels.
- **CellPadding:** Cell Padding is the space between the cell border and the cell contents and is specified in pixels.
- **Align:** tables can have left, right, or center alignment.
- **Background:** Background Image, will be titled in IE3.0 and above.
- BorderColor, BorderColorDark.

- A table caption allows you to specify a line of text that will appear centered above or bellow the table.

```
<TABLE BORDER=1 CELLPADDING=2>

<CAPTION ALIGN="BOTTOM"> Label For My Table </CAPTION>
```

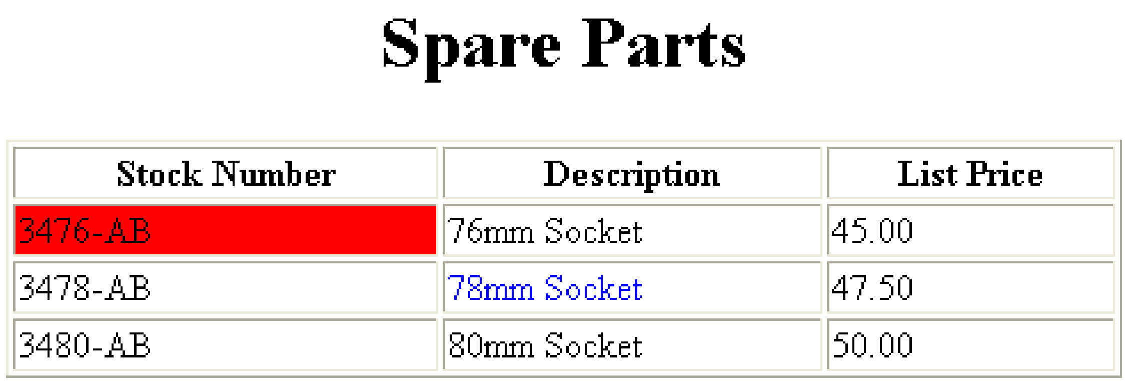
- The Caption element has one attribute ALIGN that can be either TOP (Above the table) or BOTTOM (below the table).
- Table Data cells are represented by the TD element. Cells can also be TH (Table Header) elements which results in the contents of the table header cells appearing centered and in bold text.

Table Data and Table Header Attributes

- **Colspan:** Specifies how many cell columns of the table this cell should span.

- **Rowspan*:*** Specifies how many cell rows of the table this cell should span.
- **Align*:*** cell data can have left, right, or center alignment.
- **Valign*:*** cell data can have top, middle, or bottom alignment.
- **Width*:*** you can specify the width as an absolute number of pixels or a percentage of the document width.
- **Height*:*** You can specify the height as an absolute number of pixels or a percentage of the document height.

```
<TABLE BORDER=1 width=50%>

<CAPTION><h1>Spare Parts <h1></Caption>

<TR><TH>Stock Number</TH><TH>Description</TH><TH>List Price</TH></TR>

<TR><TD bgcolor=red>3476-AB</TD><TD>76mm Socket</TD><TD>45.00</TD></TR>

<TR><TD >3478-AB</TD><TD><font color=blue>78mm
Socket</font></TD><TD>47.50</TD></TR>

<TR><TD>3480-AB</TD><TD>80mm Socket</TD><TD>50.00</TD></TR>

</TABLE>
```

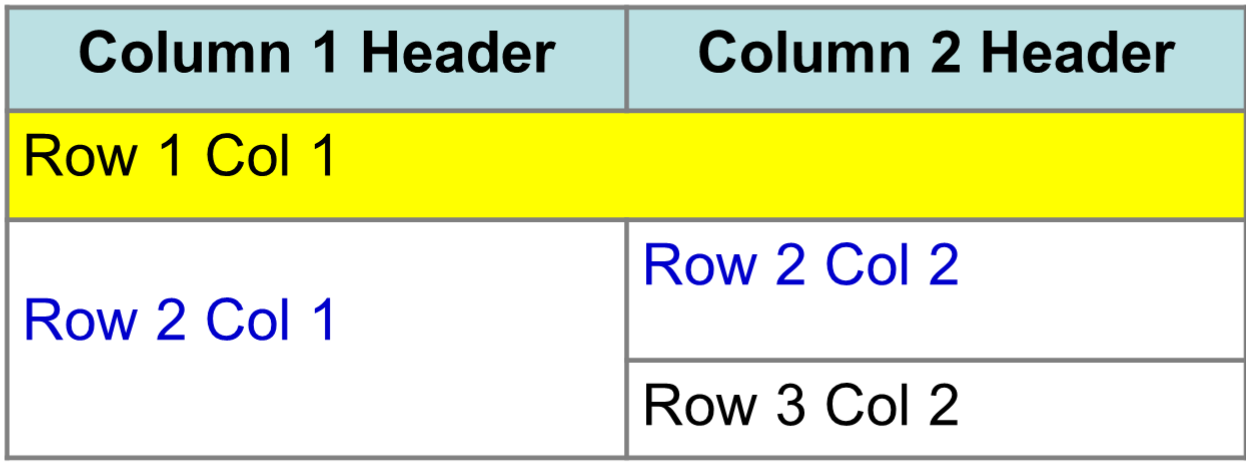


Spare Parts

Stock Number	Description	List Price
3476-AB	76mm Socket	45.00
3478-AB	78mm Socket	47.50
3480-AB	80mm Socket	50.00

<Table border=1 cellpadding =2>

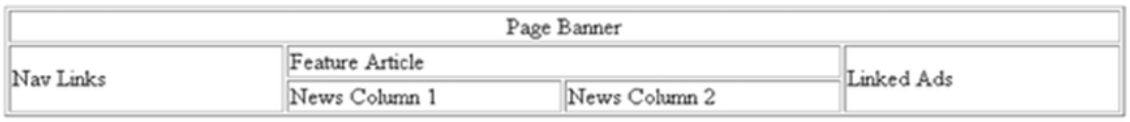
```
<tr><th> Column 1 Header</th><th> Column 2 Header</th></tr>

<tr><td colspan=2> Row 1 Col 1</td></tr>

<tr><td rowspan=2>Row 2 Col 1</td>

<td> Row 2 Col2</td></tr>

<tr><td> Row 3 Col2</td></tr>

</table>
```



Column 1 Header	Column 2 Header
Row 1 Col 1	
Row 2 Col 1	Row 2 Col 2
	Row 3 Col 2

What will be the output?

```
<TABLE BORDER width="750">

<TR><TD colspan="4" align="center">Page Banner</TD></TR>

<TR><TD rowspan="2" width="25%">Nav Links</TD><TD colspan="2">Feature
Article</TD><TD rowspan="2" width="25%">Linked Ads</TD></TR>

        <TR><TD width="25%">News Column 1 </TD><TD width="25%"><News Column 2
</TD></TR>

</TABLE>
```



	Page Banner		
Nav Links	Feature Article		Linked Ads
	News Column 1	News Column 2	

Forms

```
<FORM >

<P> First Name: <INPUT TYPE="TEXT" ></P>

<P><INPUT TYPE="SUBMIT"  VALUE="Send Info"></P>

</Form>
```

Form elements

- **Form elements have properties: Text boxes, Password boxes, Checkboxes, Select, Radio buttons, Submit, text area, File, Hidden and Image.**
- **The properties are specified in the TYPE Attribute of the HTML element <INPUT></INPUT>.**

Password

```
<form>
 Password: <input type="password" name="pwd">
</form>
```

Check box

```
<form>
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
<input type="checkbox" name="vehicle" checked>I have a car
</form>
```

Select

```
<select>
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="mercedes">Mercedes</option>
<option value="audi">Audi</option>
</select>


<select size =4>
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="opel">Opel</option>
<option value="audi">Audi</option>
```

```
</select>
```

Radio

```
<form>
<input type="radio" name="gender" value="male">Male<br>
<input type="radio" name="gender" value="female">Female
</form>
```

Button

```
<button type="button">Click Me!</button>
```

Also

<INPUT TYPE=“SUBMIT” NAME=“fsubmit1” VALUE=“Send Info”>

File

- **File Upload:** You can use a file upload to allow surfers to upload files to your web server.
- **<INPUT TYPE=“FILE”>**
- Browser will display
- File Upload has the following attributes:
- **SIZE:**is the size of the text box in characters.

<BODY bgcolor=lightblue>

<form>

<H3><font color=forestgreen>

Please attach your file here to for uploading to

My <font color =red>SERVER...<BR>

<INPUT TYPE="File" name="myFile" size="30">

<INPUT TYPE="Submit" value="SubmitFile">

</form>

</BODY>

Text area

- **<TEXTAREA></TEXTAREA>:** is an element that allows for free form text entry.

Browser will display

Textarea has the following attributes:

- **ROWS:** the number of rows to the textbox.
- **COLS:** the number of columns to the textbox.

```
<textarea rows="4" cols="50">
 At w3schools.com you will learn how to make a website. We offer free
tutorials in all web development technologies.
</textarea>
```

Image Submit Button

<INPUT TYPE="IMAGE" SRC="jordan.gif">

Image maps

- Designate certain areas of an image (called hotspots) as links
  - Element map
    - Attribute id
      - Identifies the image map
    - Element area
      - Defines hotspot
      - Attribute shape and coords
        - Specify the hotspot's shape and coordinates
      - Rectangular (shape = "rect")
      - Polygon (shape = "poly")
      - Circle (shape = "circle")

Ex:

```
<body>

<p>Click on the sun or on one of the planets to watch it closer:</p>

<img src="planets.gif" width="145" height="126" alt="Planets"
usemap="#planetmap">

< map name="planetmap">
<area shape="rect" coords="0,0,82,126" alt="Sun" href="sun.htm">
<area shape="circle" coords="90,58,3" alt="Mercury"href="mercu.htm">
<area shape="circle" coords="124,58,8" alt="Venus" href="venus.htm">
</map>

</body>
```

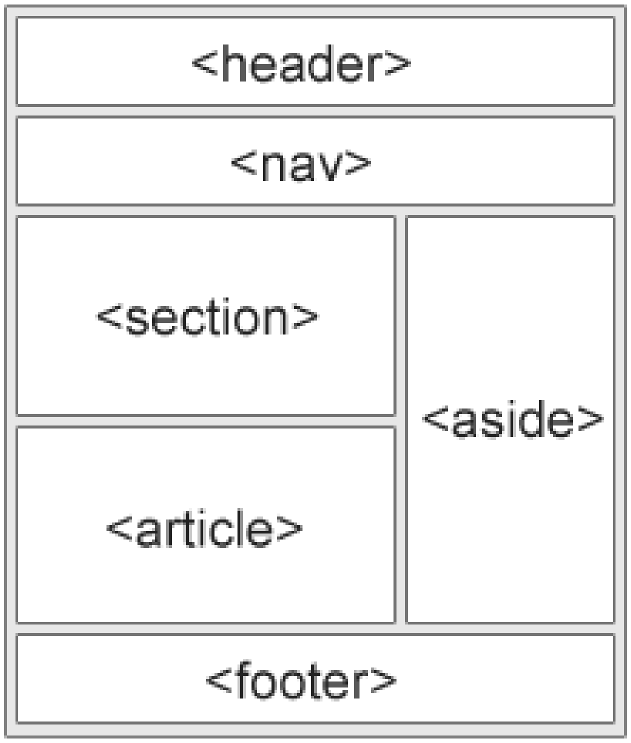
HTML 5

HTLM 5 added many elements

Semantic elements

New semantic elements like <header>, <footer>, <article>, and <section>.

This to help search engines identify the correct web page content.



<article>

Defines an article in the document. An article should make sense on its own, and it should be possible to read it independently from the rest of the web site

<aside>

Defines content aside from the page content. The <aside> element defines some content aside from the content it is placed in (like a sidebar). The aside content should be related to the surrounding content.

<footer>

Defines a footer for the document or a section. A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

<figure> and <figcaption>

The purpose of a figure caption is to add a visual explanation to an image.

In HTML5, an images and a caption can be grouped together in a <figure> element

```
<figure>
  <img src="pic_mountain.jpg" alt="The Pulpit
Rock" width="304" height="228">
  <figcaption> Fig1. - The Pulpit Rock, Norway.</figcaption>
</figure>
```

<section>

Defines a section in the document. A home page could normally be split into sections for introduction, content, and contact information.

<header>

Defines a header for the document or a section. The <header> element should be used as a container for introductory content.

New graphic elements

New graphic elements: <svg> and <canvas>.

Canvas

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

```
Using JavaScript
```

```
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
```

SVG

SVG stands for Scalable Vector Graphics. SVG is used to define graphics for the Web

The HTML <svg> element is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

Ex:

```
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" />
</svg>

</body>
</html>
```

New multimedia elements

New multimedia elements: <audio> and <video>.

Before HTML5 different audio and video required plug-in such as flash.

```
Video

<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>

Or use Autoplay
 X <video width="320" height="240" autoplay>

Audio

<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Cascading Style Sheets (CSS)

CSS is a language that describes the style of an HTML document.

CSS describes how HTML elements should be displayed.

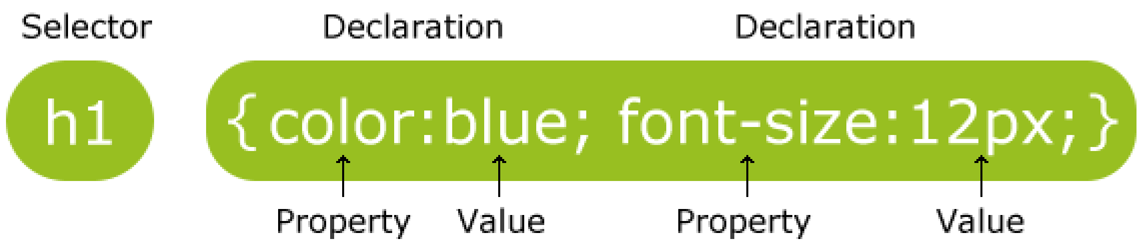
CSS saves a lot of work and create consistencies in formatting. It can control the layout of The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file

Multiple web pages all at once as external stylesheets are stored in CSS files.

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

```
p {
   color: red;
   text-align: center;
}
```

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

```
<head>

<style>

p {

text-align: center;

color: red;

}

</style>

</head>

<body>
```

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

```
<head>
<style>
#para1 {
text-align: center;
color: red;
}
</style>
</head>
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>
```

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

```
<head>
<style>
.center {
text-align: center;
color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
```

Inline styles

```
<STYLE CSS property: value>


<!--------In line styles ---------------------------------!>

<p style = "font-size: 20pt">This text has the
<em>font-size</em> style applied to it, making it 20pt.
</p>
```

Embedded styles

```
<style> .cl{color=green;}  </style>

<body> Click ya<A HREF="http://www.yahoo.com">here</A>
<h1> header 1</h1>
normal
<h1 class=cl> header 1 again </h1>
</body>
```

Ex:2

```
<style> .cl{color:green;}
```

```
.largeFont{font-size:30pt;} </style>

<body>
<h1> header 1</h1>
normal
<h1 class=cl> header 1 again </h1>
<p class=largeFont> class large font </p>
</body>



<head>
<title>Style Sheets</title>

<style type = "text/css">
h1      { font-family: arial, sans-serif }
p       { font-size: 14pt }
.special { color: blue }
</style>
</head>

<!-- this class attribute applies the .special style -->
<h1 class = "special">Deitel& Associates, Inc.</h1>

<p>Deitel&amp; Associates, Inc. is an internationally
```

courses and World Wide Web courses.</p>

Grouping selector

Use comma

```
h1, h2, p  {
   text-align:  center;
   color:  red;
}
```

Conflicting styles

The most specific applies

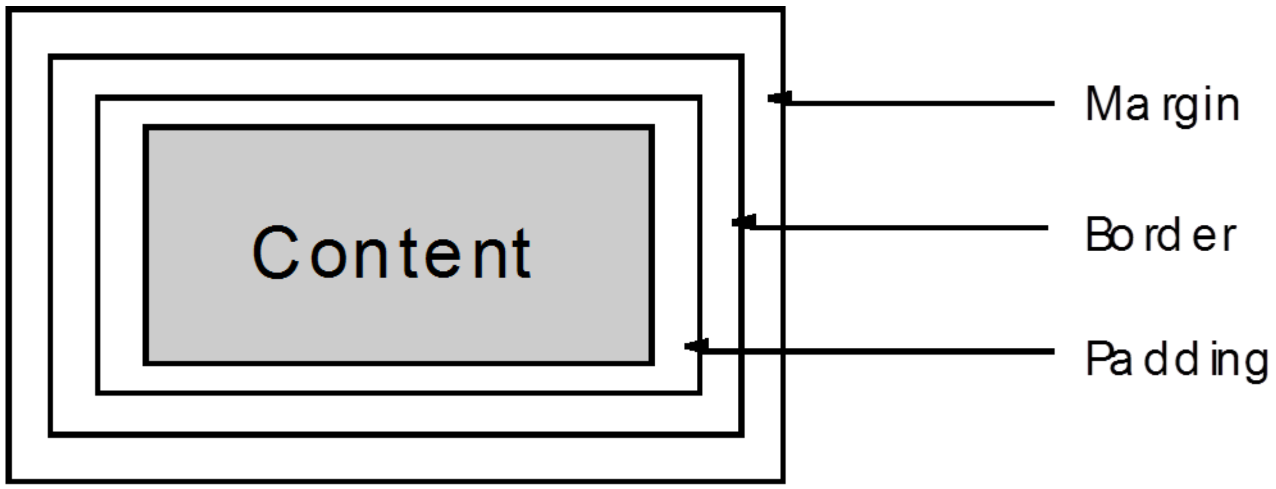
Linking external style sheets

```
<head>
<title>Linking External Style Sheets</title>
<link rel = "stylesheet" type = "text/css"
href = "styles.css" /></head>
```

More CSS attributes

```
body {background-color:#b0c4de;}

body {background-image:url("paper.gif");}

h1 {text-decoration:overline;}

 h2 {text-decoration:line-through;}

 h3 {text-decoration:underline;}

font-style:italic
font-style:oblique

letter-spacing

font-weight: normal, bold
```

Text Flow and the Box Model



Explanation of the different parts:

•Margin - Clears an area around the border. The margin does not have a background color, it is completely transparent

•Border - A border that goes around the padding and content. The border is inherited from the color property of the box

•Padding - Clears an area around the content. The padding is affected by the background color of the box

•Content - The content of the box, where text and images appear

width:250px; (for content)

padding:10px;

border:5px solid gray;

margin:10px;

ex:

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
div.ex
{
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
}
</style>
</head>

<body>

<imgsrc="w3css.gif" width="250" height="250">

<div class="ex">The picture above is 250px wide.
The total width of this element is also 250px.</div>

</body>
</html>
```

Border

```
p.one

 {
border-style:solid;
border-width:5px;
 }
p.two
 {
border-style:solid;
border-width:medium;
 }
```

JavaScript

- Has nothing to do with Java except for the syntax similarity
- Embedded within HTML page
- Executes on client
- Fast, no connection needed once loaded
- Interpreted (not compiled), No special tools required

client-side programming

programs are written in a separate programming (or scripting) language

e.g., JavaScript, VBScript

programs are embedded in the HTML of a Web page, with (HTML) tags to identi fy the program component

e.g., <script type="text/javascript"> … </script>

```
<html>
<head><title>My Page</title></head>
<body>
<script >

document.write('<B> This is my first JavaScript Page </B>');

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

the browser executes the program as it loads the page, integrating the dynamic output of the program with the static content of HTML

adding dynamic features to Web pages e.g.: validation of form data

limitations of client-side scripting

since script code is embedded in the page, it is viewable to the world

for security reasons, scripts are limited in what they can do e.g., can't access the client's hard drive since they are designed to run on any machine platform, scripts do not contain platform specific commands

Syntax

JavaScript code can be embedded in a Web page using <script> tags the output of JavaScript code is displayed as if directly entered in HTML

Ex1:

```
<head>
<title>JavaScript Page</title>
</head>
<body>
<script type="text/javascript">
   // silly code to demonstrate output
document.write("<p>Hello world!</p>");
document.write(" <p>How are <br/> " +
               " <i>you</i>?</p> ");
</script>
<p>Here is some static text as well.</p>
</body>
</html>
```

as in C++/Java, statements end with ;but a line break might also be interpreted as the end of a statement (depends upon browser)

JavaScript comments similar to C++/Java

//   starts a single line comment

/*…*/ enclose multi-li ne comments

Variables

JavaScript has only three primitive data types

String   : "foo" 'how do you do?'   "I said 'hi'."   ""

Number: 12   3.14159   1.5E6

assignments are as in C++/Java

```
message = "howdy";
```

pi = 3.14159;

variable names are sequences of letters, digits, and underscores that start with a letter or an underscore

variables names are case-sensitive

you don't have to declare variables, will be created the first time used, but it’s better if you use var statements

var message, pi=3.14159;

variables are loosely typed, can be assigned different types of values (Danger!)

Ex2:

Boolean : true false *Find info on Null, Undefined

```
<html>
<!–- ex 2-->

<head>
<title>Data Types and Variables</title>
</head>

<body>
<script type="text/javascript">
var x, y;
   x= 1024;
y=x;  x = "foobar";
document.write("<p>x = " + y + "</p>");
document.write("<p>x = " + x + "</p>");
</script>
</body>
</html>
```

Control statements

Same as in C++

Interactive Pages Using Prompt

crude user interaction can take place using prompt

1st argument: the prompt message that appears in the dialog box

2nd argument: a default value that will appear in the box (in case the user enters nothing)

the function returns the value entered by the user in the dialog box (a string)

if value is a number, must use parseFloat (or parseInt) to convert

forms will provide a better interface for interaction (later)

Ex 3:

```
<html>
<!-- COMP519  js05.html  08.10.10 -->

<head>
<title>Interactive page</title>
</head>

<body>
<script type="text/javascript">
  B  hhgjhbmjhgb userName = prompt("What is your name?", "");

userAge = prompt("Your age?", "");
varuserAge = parseFloat(userAge);

document.write("Hello " + userName + ".")
if (var userAge< 18) {
document.write("  Do your parents know " +
                    "you are online?");
   }
else {
document.write("  Welcome friend!");
   }
</script>
```

```
<p>The rest of the page...</p>
</body>
</html>
```

Event Driven programming

A common programming paradigm for visual languages where a statement or a function fires in respond to an event often performed by user

Example: onMouseover event

```
<html>
<head><title>My Page</title></head>
<body>
<a href="myfile.html" onMouseover="window.alert('Hello');">
My Page</A>
</p>
</body>
</html>
```

Example: onClick event

```
 <form>
<input type="button" Value="Press" onClick="window.alert('Hello');">
</form>
```

User-Defined Functions

- function definitions are similar to C++/Java, except:
  - no return type for the function (since variables are loosely typed)
  - no variable typing for parameters (since variables are loosely typed)
  - by-value parameter passing only (parameter gets copy of argument)

Function definitions (usually) go in the <head> section

<head> section is loaded first, so then the function is defined before code in the <body> is executed (and, therefore, the function can be used later in the body of the HTML document)

Ex 4:

```
<head>
Function isPrime(n)
// Assumes: n > 0
// Returns: true if n is prime, else false
```

```
{
if (n < 2) {
return false;
  }
else if (n == 2) {
return true;
  }
else {
for (var i = 2; i <= Math.sqrt(n); i++) {
if (n % i == 0) {
return false;
        }
      }
return true;
  }
}
</head<

<body>
<script type="text/javascript">
testNum = parseFloat(prompt("Enter a positive integer", "7"));

if (isPrime(testNum)) {
document.write(testNum + " <b>is</b> a prime number.");
    }
else {
document.write(testNum + " <b>is not</b> a prime number.");
    }
</script>
</body>
```

The HTML DOM (Document Object Model)

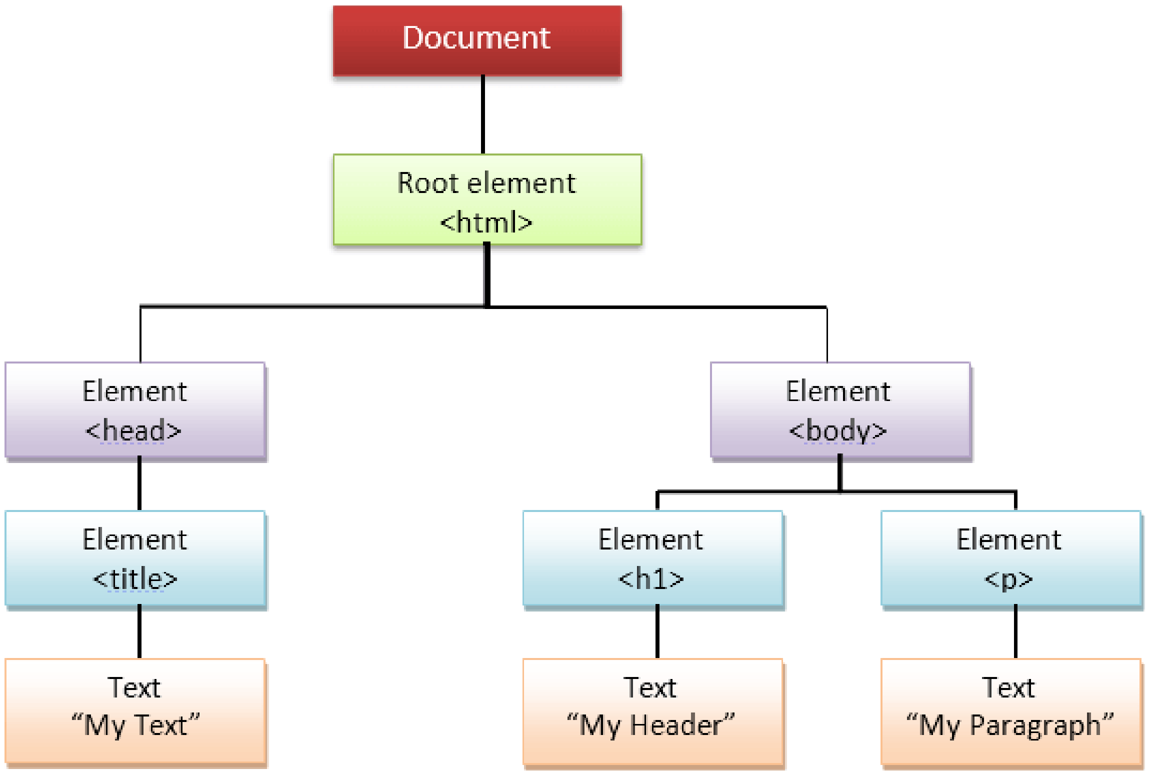
When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements .**

The **HTML DOM** model is constructed as a tree of **Objects**:



The HTML DOM Document Object

The *document* object represents your web page. When an HTML document is loaded into a web browser, it becomes a **document object**.

If you want to access any element in an HTML page, you always start with accessing the document object.

The document object is the root node of the HTML document and the "owner" of all other nodes:
(element nodes, text nodes, attribute nodes, and comment nodes).

The document object provides properties and methods to access all node objects, from within JavaScript.

Tip: The document is a part of the Window object and can be accessed as window.document.

Finding HTML Elements

Method	**Description**
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

Changing HTML Elements

Method	**Description**
element.*attribute* = *new value*	Change the attribute value of an HTML element
ex: element.innerHTML = new html content	Change the inner HTML of an element
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

element.style.*property* = *new style*	Change the style of an HTML element

Adding and Deleting Elements

Method	**Description**
document.createElement(*element*)	Create an HTML element
document.removeChild(*element*)	Remove an HTML element
document.appendChild(*element*)	Add an HTML element
document.replaceChild(*element*)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

Manipulating DOM elements

JavaScript Can Change HTML Content

This example uses the method getElementById to "find" an HTML element (with id="demo"), and changes the element content (**innerHTML**) to "Hello JavaScript":

```
<html><body><h1>My First JavaScript</h1>
<button type="button" onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time. </button>
<p id="demo"></p>
</body></html>
```

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

Example

document.getElementById("demo").style.fontSize = "25px";

JavaScript Can Validate Data

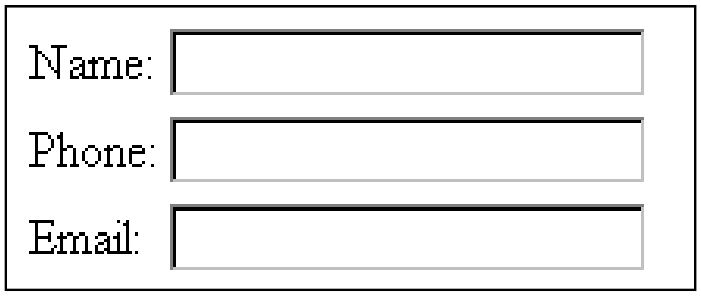
JavaScript is often used to validate input:

How to get hold of the HTML element that contains the data that need to be validated?

We saw the id method. Another method is by using the JavaScript DOM hierarchy **where form name can be navigated to form elements down to their properties using dots**

Forms and form elements must have unique names

Naming Form Elements in HTML



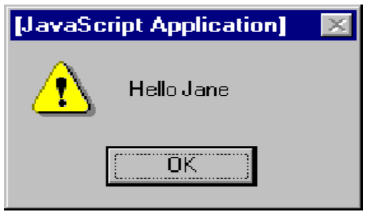
<form **name="addressform"**>
Name: <input **name="yourname"**><br />
Phone: <input name="phone"><br />
Email: <input name="email"><br />
</form>

Syntax to address form element: *document.**formname.elementname**.value*

Thus:
document.**addressform.yourname**.value
document.addressform.phone.value
document.addressform.email.value

ex: Personalising an alert box

<form name="**alertform**">
Enter your name:
<input type="text" name="**yourname**">
<input type="button" value= "Go" onClick="window.alert('Hello '
document.alertform.yourname.value);">
</form>



Hello Jane

Exercise:

```
<html><body>
<h1>JavaScript Can Validate Input</h1>
 <p>Please  input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>

Function myFunction() {
var x, text;
   // Get the value of the input field with id="numb"
   x = document.getElementById("numb").value;
   // If x is Not a Number or less than one or greater than 10
if (isNaN(x) || x < 1 || x > 10) {
text = "Input not valid";
   } else {
  text = "Input OK";
  }
document.getElementById("demo").innerHTML = text;
}
</script>
</body></html>
```

Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);

var element = document.getElementById("div1");
element.appendChild(para);
</script>
```

Example Explained

This code creates a new <p> element:

var para = document.createElement("p");

To add text to the <p> element, you must create a text node first. This code creates a text node:

var node = document.createTextNode("This is a new paragraph.");

Then you must append the text node to the <p> element:

para.appendChild(node);

Finally you must append the new element to an existing element.

This code finds an existing element:

var element = document.getElementById("div1");

This code appends the new element to the existing element:

element.appendChild(para);

Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element:

Example

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>

<script>
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
</script>
```

The HTMLCollection Object

The getElementsByTagName() method returns an HTMLCollection object.

An HTMLCollection object is an array-like list (collection) of HTML elements.

The following code selects all <p> elements in a document:

Example

var x = document.getElementsByTagName("p");

The elements in the collection can be accessed by an index number.

To access the second <p> element you can write:

y = x[1]; //Note: The index starts at 0.

The length property defines the number of elements in an HTMLCollection:

Example

var myCollection = document.getElementsByTagName("p"); // Create a collection of all <p> elements

// Display the length of the collection

document.getElementById("demo").innerHTML = myCollection.length;

Example

Change the background color of all <p> elements:

var myCollection = document.getElementsByTagName("p");

var i;

for (i = 0; i < myCollection.length; i++) {

  myCollection[i].style.backgroundColor = "red";

}

Example

Loop through every element in an HTMLCollection:

```
x = document.getElementsByTagName("*");
l = x.length;
for (i = 0; i < l; i++) {
  document.write(x[i].tagName + "<br>");
```

Example

```
var x = document.forms["frm1"];
var text = "";
var i;
for (i = 0; i <x.length; i++) {
    text += x.elements[i].value + "<br>";
}
document.getElementById("demo").innerHTML = text;
```

Example of how to display the href of a link

<html> <body>

<p>

<a href="/html/default.asp">HTML</a> <br>

<ahref="/css/default.asp">CSS</a> </p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

"The href of the first link is " + document.links[0].href;

</script>

</body> </html>

Example of adding an option to a list

Add a "Kiwi" option at the end of a drop-down list:

```
var x = document.getElementById("mySelect");
var option = document.createElement("option");
option.text = "Kiwi";
x.add(option);
```

External JavaScript

Scripts can also be placed in external files:

```
<body>
<h1>External JavaScript</h1>
<p id="demo">A Paragraph.</p>
<button type="button" onclick="myFunction()">Try it</button>
<p><strong>Note:</strong>myFunction is stored in an external file called "myScript.js".</p>
<script src="myScript.js"></script>
</body>
```

Additional HTML events

onfocus	The event occurs when an element gets focus
onfocusin	The event occurs when an element is about to get focus 2
onfocusout	The event occurs when an element is about to lose focus
oninput	The event occurs when an element gets user input
onselect	The event occurs after the user selects some text (for <input> and <textarea>)
onsubmit	The event occurs when a form is submitted
onchange	An HTML element has been changed

<select onchange="myFunction()">

onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element

onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

DOM manipulation examples

Display the title of the document (hint use document.title)

Find the number of forms in a document (hint use document.forms.length)
Find the name of the first form in a document

Return the number of images in a document

Return the id of the first image in a document

Adding event listener

In addition to using HTML tag like "onClick" to add event handling functions, JavaScript permits adding what is called "event listener" so to let a specific function runs when an event is fired.

When using the `addEventListener()` method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML markup.

Syntax

```
element.addEventListener(event, function);
```

The first parameter is the type of the event (like "`click`" or "`mousedown`" or any other HTML DOM Event.)

The second parameter is the function we want to call when the event occurs.

Example:

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert ("Hello World!");
}
```

JavaScript Strings

JavaScript strings are used for storing and manipulating text.

A JavaScript string simply stores a series of characters like "John Doe".

A string can be any text inside quotes. You can use single or double quotes:

Example

```
Var carname = "Volvo XC60";
var carname = 'Volvo XC60';
```

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';
```

String Length

The length of a string is found in the built in property **length**:

Example

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

Finding a String in a String

The **indexOf()** method returns the index of (the position of) the **first** occurrence of a specified text in a string:

Example

```
Var str = "Please locate where locate occurs!";
var pos = str.indexOf("locate");
```

The **lastIndexOf()** method returns the index of the **last** occurrence of a specified text in a string:

Example

```
varstr = "Please locate where 'locate' occurs!";
varpos = str.lastIndexOf("locate");
```

Both the indexOf(), and the lastIndexOf() methods return -1 if the text is not found.

JavaScript counts positions from zero.
0 is the first position in a string, 1 is the second, 2 is the third ...

Both methods accept a second parameter as the starting position for the search.

Searching for a String in a String

The **search()** method searches a string for a specified value and returns the position of the match:

Example

```
varstr = "Please locate where 'locate' occurs!";
varpos = str.search("locate");
```

The two methods, indexOf() and search(), are equal.

They accept the same arguments (parameters), and they return the same value.

The two methods are equal, but the search() method can take much more powerful search values like regular expressions.

Extracting String Parts

There are 2 methods for extracting a part of a string:

- slice(start, end)
- substr(start, length)

The slice() Method

slice() extracts a part of a string and returns the extracted part in a new string.

The method takes 2 parameters: the starting index (position), and the ending index (position).

This example slices out a portion of a string from position 7 to position 13:

Example

```
Var str = "Apple, Banana, Kiwi";
var res = str.slice(7,13);
```

The result of res will be:

```
Banana
```

If you omit the second parameter, the method will slice out the rest of the string:

Example

```
var res = str.slice(7);
```

or, counting from the end:

The substr() Method

substr() is similar to slice().

The difference is that the second parameter specifies the **length** of the extracted part.

Example

```
varstr = "Apple, Banana, Kiwi";
var res = str.substr(7,6);
```

The result of res will be:

```
Banana
```

If the first parameter is negative, the position counts from the end of the string.

If you omit the second parameter, substr() will slice out the rest of the string.

Replacing String Content

The **replace()** method replaces a specified value with another value in a string:

Example

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft","W3Schools");
```

The replace() method can also take a regular expression as the search value.

By default, the replace() function replaces only the first match. To replace all matches, use a regular expression with a g flag (for global match):

Example

```
str = "Please Microsoft visit Microsoft!";
var n = str.replace(/Microsoft/g,"W3Schools");
```

The replace() method does not change the string it is called on. It returns a new string.

Converting to Upper and Lower Case

A string is converted to upper case with **toUpperCase()**:

Example

```
var text1 = "Hello World!";       // String
var text2 = text1.toUpperCase();  // text2 is text1 converted to upper
```

A string is converted to lower case with **toLowerCase()**:

Example

```
var text1 = "Hello World!";       // String
var text2 = text1.toLowerCase();  // text2 is text1 converted to lower
```

The concat() Method

concat() joins two or more strings:

Example

```
var text1 = "Hello";
var text2 = "World";
text3 = text1.concat(" ",text2);
```

The **concat()** method can be used instead of the plus operator. These two lines do the same:

Example

```
var text = "Hello" + " " + "World!";
var text = "Hello".concat(" ","World!");
```

Extracting String Characters

There are 2 **safe** methods for extracting string characters:

- charAt(position)
- charCodeAt(position)

The charAt() Method

The **charAt()** method returns the character at a specified index (position) in a string:

Example

```
varstr = "HELLO WORLD";
str.charAt(0);            // returns H
```

The charCodeAt() Method

The **charCodeAt()** method returns the unicode of the character at a specified index in a string:

Example

```
varstr = "HELLO WORLD";

str.charCodeAt(0);         // returns 72
```

Converting a String to an Array

A string can be converted to an array with the **split()** method:

Example

```
var txt = "a,b,c,d,e";   // String
txt.split(",");          // Split on commas
txt.split(" ");          // Split on spaces
txt.split("|");          // Split on pipe
```

If the separator is omitted, the returned array will contain the whole string in index [0].

If the separator is "", the returned array will be an array of single characters:

Example

```
var txt = "Hello";       // String
var res= txt.split("");           // Split in characters
```

Regular expressions

A regular expression is a sequence of characters that forms a search pattern.

The search pattern can be used for text search and text replace operations.

Syntax: /pattern/modifiers;

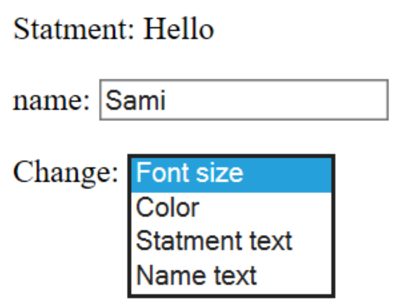
```
var str = "Visit W3SchooLs";
var n = str.search(/w3schools/i);

//return 3
```

```
var str = "Visit Microsoft!";
var res = str.replace(/microsoft/ig, "W3Schools");

//return Visit W3Schools!
```

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)

Example:

1. Given the HTML form:


Statment: Hello

name: Sami

Change:

If the user selects “Change Font size” the program should prompt him to enter the new font size and the font size for the Hello statement should change immediately accordingly.

If the user selects “Change Color” the program should prompt him to enter the new color name and the color for the Hello statement should change immediately accordingly.

If the user selects “Change Statement text” the program should prompt him to enter the new text and the text for the Hello statement should change immediately accordingly (i.e. Hello should be replaced by the new text).

If the user selects “Change Name text” the program should prompt him to enter the new text and the text for the Name input should change immediately accordingly (i.e. Sami should be replaced by the new text).

Finally, note that when the mouse pointer passes over the Statement Hello, its color should turn into red and when the pointer moves away the color should be returned to black.

Solution:

```
<html>

<head>

<script>

function a()

{ sval=document.f1.s.value;

 //alert(document.getElementById("stmt").innerHTML)

//alert ("ok " + sval);

if (sval== "clr" ) {

    vcolor=prompt("Enter color: ");

   document.getElementById("stmt").style.color=vcolor};

//alert ("ok ");

if (sval== "fnt" ) {  vsize=prompt("Enter size: ");

                    document.getElementById("stmt").style.fontsize=vsize;};

if (sval== "txt" ) {  vtxt=prompt("Enter txt: ");

 document.getElementById("stmt").innerHTML="Statment: " + vtxt};

 if (sval== "nmtxt" ) {  nmtxt=prompt("Enter name txt: ");

                     f1.nm.value=nmtxt;};

 }


</script>

</head>
```

```
<body>

<form name=f1>

<p id ="stmt"> Statment: Hello

<p> name: <input type= text name=nm  value=Sami >

<br>

<p>

Change:

<select name=s   onchange=a() >

 <option value=fnt >  Font  size</option>

 <option value=clr>  Color </option>

  <option value=txt>  Statment text </option>

   <option value=nmtxt>  Name text </option>

</select>

</form>

</body>
```

jQuery

Downloading jQuery

```
<script src="jquery-3.2.1.min.js"></script>
```

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

Both Google and Microsoft host jQuery.

```
<head>
```

```
 <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
</script>

<script>

$(document).ready(function(){

    $("p").click(function(){

        $(this).hide();

    });

});

</script>

</head>
```

jQuery Syntax

The jQuery syntax is tailor-made for selecting HTML elements and performing some action on the element(s).

Basic syntax is: $(selector).action()

A $ sign to define/access jQuery

A (selector) to "query (or find)" HTML elements

A jQuery action() to be performed on the element(s)

Examples:

$(this).hide() - hides the current element.

$("p").hide() - hides all <p> elements.

$(".test").hide() - hides all elements with class="test".

$("#test").hide() - hides the element with id="test".

The Document Ready Event

You might have noticed that all jQuery methods in our examples, are inside a document ready event:

```
$(document).ready(function(){

  // jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allow you to have your JavaScript code before the body of your document, in the head section.

Tip: The jQuery team has also created an even shorter method for the document ready event:

```
$(function(){

  // jQuery methods go here...

});
```

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

All selectors in jQuery start with the dollar sign and parentheses: $().

The element Selector

The jQuery element selector selects elements based on the element name.[2]

You can select all <p> elements on a page like this:

```
$("p")
```

Example

When a user clicks on a button, all <p> elements will be hidden:

```
   $("button").click(function(){

       $("p").hide();

   };
```

The #id Selector

The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

Example

When a user

```
 $("button").click(function(){

   $("#test").hide();

}); clicks on a button, the element with id="test" will be hidden:
```

The .class Selector

The jQuery class selector finds elements with a specific class.

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

Example

When a user clicks on a button, the elements with class="test" will be hidden:

```
$("button").click(function(){

   $(".test").hide();

};
```

jQuery Syntax for Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){
  // action goes here!!
});
```

jQuery Event Methods

$(document).ready()

The $(document).ready() method allows us to execute a function when the document is fully loaded. This event is already explained in the jQuery Syntax chapter.

click()

The click() method attaches an event handler function to an HTML element.

The function is executed when the user clicks on the HTML element.

The following example says: When a click event fires on a <p> element; hide the current <p> element:

Example

```
$("p").click(function(){
   $(this).hide();
});
```

dblclick()

The dblclick() method attaches an event handler function to an HTML element.

The function is executed when the user double-clicks on the HTML element:

Example

```
$("p").dblclick(function(){
   $(this).hide();
});
```

mouseenter()

The mouseenter() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer enters the HTML element:

Example

```
$("#p1").mouseenter(function(){
    alert("You entered p1!");

});
```

mouseleave()

The mouseleave() method attaches an event handler function to an HTML element.

The function is executed when the mouse pointer leaves the HTML element:

Example

```
$("#p1").mouseleave(function(){

    alert("Bye! You now leave p1!");

});
```

mousedown()

The mousedown() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

Example

```
$("#p1").mousedown(function(){

    alert("Mouse down over p1!");

});
```

mouseup()

The mouseup() method attaches an event handler function to an HTML element.

The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

Example

```
$("#p1").mouseup(function(){

    alert("Mouse up over p1!");

});
```

hover()

The hover() method takes two functions and is a combination of the mouseenter() and mouseleave() methods.

The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

Example

```
$("#p1").hover(function(){

   alert("You entered p1!");

},

function(){

   alert("Bye! You now leave p1!");

});
```

focus()

The focus() method attaches an event handler function to an HTML form field.

The function is executed when the form field gets focus:

Example

```
$("input").focus(function(){

   $(this).css("background-color", "#cccccc");

});
```

blur()

The blur() method attaches an event handler function to an HTML form field.

The function is executed when the form field loses focus:

Example

```
$("input").blur(function(){

   $(this).css("background-color", "#ffffff");});
```

Example:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

```
<html>  <head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

    $("#btn1").click(function(){

        alert("Text: " + $("#test").text());

    });

    $("#btn2").click(function(){

        alert("HTML: " + $("#test").html());

    });

});

</script>

</head>

<body>

<p id="test">This is some <b>bold</b> text in a paragraph.</p>

<button id="btn1">Show Text</button>

<button id="btn2">Show HTML</button>

</body> </html>
```

Set Values

```
$("#btn1").click(function(){
    $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
    $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
    $("#test3").val("Dolly Duck");
});
```