WEB DESIGN AND DEVELOPMENT
(HTML, CSS and JAVASCRIPT)

The Web is a short form of the World Wide Web (WWW). It is an information space where documents and other web resources identified by Uniform Resource Locators (URLs), interlinked by hypertext links, and accessible via the Internet are located. Web resources include websites and web applications. A website is a collection of related web pages, including multimedia content, typically identified with a common domain name, and published on at least one web server. Notable examples are wikipedia.org, google.com, and amazon.com. A Web application (Web app) is an application program or software that is stored on a remote server and delivered over the Internet through a browser interface. Web design and development is therefore the discipline concerned with designing and the development of web resources including websites and web software.

A website is a combination of different web pages linked together in a certain fashion according to the designer's will and organizational need served through the same domain name server. The developed website is prepared and maintained by a person, group or an organization. Website is typically written in technologies such as HTML, XHTML and DHTML and is hosted on single web server for its access through network. Each website is provided with a unique uniform resource locator (URL) and accessed through the hypertext transfer protocol (HTTP) responsible for making communication possible between client machine and the server by the application of different browsers. Websites provide hypermedia resources, a system allowing storage and access to text, graphics, audio and video in so called pages linked to each other in a way that integrates these different media elements. When a user clicks on a highlighted word or picture (icon), the browser converts the click to computer commands and brings the requested new information in the form of text, photograph, chart, song or movie clip to the user's computer.

Web design refers to both the aesthetic portion of the website and its usability. Web design considers the choice of colors, images, layout and the general look of the website in regard to its purpose. A web designer for example will consider a party website and choose vibrant colors such as yellow and orange rather than black and grey. Web Developers on the other hand, take a website design and actually make a functioning website from it. Web developers use HTML, CSS, JavaScript, PHP and other programming languages to bring to life the design files.

Web developers as those who turn the designs into a live website. Web developers uses web languages and software tools to develop the design and functionality of a website. Notice, that web developers are further split into two sub-categories; front-end developers, and back-end developers.

A front-end developer is the one who builds the interface, and provides the layout as the interaction between the back-end of the website and the user. Front-end developers use three main languages; Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS). With these languages, a developer is able to create a full-fledged website. From the main design layout, to inserting images, applying different types of typography and font families, implementing animations, the flow of different pages, form interfaces, and more.

Back-end developers are akin to web programmers and they are those who control the server data and requests. Usually a website requires back-end services if it contains dynamic data. This means, for example, users submitting a form with personal data (such as creating an account), or saving an article for your blog page. Generally, if the website requires data to be saved, and making it accessible at a later stage, it means that it would also need a database connection. Database connections are made possible by a direct connection from the server itself. Thus, a back-end developer then uses server languages such as PHP database queries by using languages such as SQL or MySQL. Backend developers in our case are programmers who create web applications.

We consider three technologies used in front end development for the web design and development. These are HTML, CSS and JavaScript.

HTML

HTML (Hyper Text Markup Language) is the language for building websites. HTML is not a programming language but a markup language. HTML was first created by Tim Berners-Lee, Robert Cailliau, and others starting in 1989. It stands for Hyper Text Markup Language. Hypertext means that the document contains links that allow the reader to jump to other places in the document or to another document altogether. The latest version is known as HTML5.
A Markup Language is a way that computers speak to each other to control how text is processed and presented. To do this HTML uses two things: tags and attributes.

Tags and attributes are the basis of HTML. Tags are used to mark up the start of an HTML element and they are usually enclosed in angle brackets. An example of a tag is: <h1>. Most tags must be opened <h1> and closed </h1> in order to function.

Attributes contain additional pieces of information. Attributes take the form of an opening tag and additional info is placed inside. An example of an attribute is:

<img src="mydog.jpg" alt="A photo of my dog.">

In this instance, the image source (src) and the alt text (alt) are attributes of the <img> tag.

Remember:

- The vast majority of tags must be opened (<tag>) and closed (</tag>) with the element information such as a title or text resting between the tags.
- When using multiple tags, the tags must be closed in the order in which they were opened. For example:

This is really important!

HTML Editors

To create a HTML document you need an editor. Examples of editors are Notepad, Sublime Text 3, Notepad++, Komodo Edit and Macromedia Dreamweaver.

Basic Structure of a HTML document

```
<!DOCTYPE html>
<html>
<head>
<title>The title of the Webpage</title>
</head>
<body>
Enter Webpage Contents Here
</body>
</html>
```

An explanation of the structure follows:

<! DOCTYPE...>

This tag defines the document type and HTML version. The <! DOCTYPE> declaration tag is used by the web browser to understand the version of the HTML used in the document. Current version of HTML is 5 and it makes use of the following declaration −

```
<! DOCTYPE html>
```

<html>

This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.

<head>

This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.

<title>

The <title> tag is used inside the <head> tag to mention the document title.

<body>

This tag represents the document's body which keeps other HTML tags like <h1>, , <p> etc.

NB: *There are many other tags depending on the task to be accomplished, and we will look at some of them in sections that follow.*

Example

```
<!DOCTYPE html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>This is a heading</h1>
<p>Hello World!</p>
</body>
</html>
```

Basic Tags

Heading Tag

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

```
<!DOCTYPE html>
<html>
  <head>
    <title>Heading Example</title>
  </head>
  <body>
    <h1>This is heading 1</h1>
    <h2>This is heading 2</h2>
    <h3>This is heading 3</h3>
    <h4>This is heading 4</h4>
    <h5>This is heading 5</h5>
    <h6>This is heading 6</h6>
  </body>
</html>
```

Paragraph Tag

The ** ** tag offers a way to structure your text into different paragraphs. Each paragraph of text should go in between an opening and a closing tag

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

Line Break Tag

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

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

```
  <head>
    <title>Line Break  Example</title>
  </head>
 <body>
    <p>Hello<br />
      This is HTML Assignment.<br />
      HTML enables us create websites<br />
      I love writing HTML content</p>
  </body>
</html>
```

Horizontal Lines

Horizontal lines are used to visually break-up sections of a document. The <hr> tag creates a line from the current position in the document to the right margin and breaks the line accordingly. For example, you may want to give a line between two paragraphs.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Horizontal Line Example</title>
  </head>
  <body>
    <p>This is paragraph one and should be on top</p>
    <hr />
    <p>This is paragraph two and should be at bottom</p>
  </body>
```

Non Breaking Space

In cases, where you do not want the client browser to break text, you should use a nonbreaking space entity   instead of a normal space.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Nonbreaking Spaces Example</title>
  </head>
  <body>
    <p>HTML is quite a wonderful  web development environment</p>
  </body>
</html>
```

Font

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <title>Font Formatting</title>
  </head>
  <body>
<font color="#993300">Showing Font Colors</font><br />
<font size="12">Font Size 12</font><br />
<font size="+6">Font that is six times higher than the present font size</font><br />
<font size="-3">Font that is three times than the present font size</font><br />
<font face="Times New Roman, Times, serif">Font face</font><br />
</body>
</html>

<!DOCTYPE html>
<html>
  <head>
    <title>Setting Basefont Color</title>
  </head>
  <body>
    <font face = "arial, verdana, sans-serif" size = "2" color = "#ff0000">
    <p>This is the page's default font.</p>
    <h2>Example of the &lt;basefont&gt; Element</h2>
        <p><font size = "+2" color = "darkgray">
        This is darkgray text with two sizes larger
      </font>
    </p>
    <p><font face = "courier" size = "-1" color = "#000000">
        It is a courier font, a size smaller and black in color.
      </font>
```

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

Font Formatting

```
<!DOCTYPE html>
<html>
  <head>
    <title>Font Formatting</title>
  </head>
 <body>
<b>Bolded Text</b><br />
<i>Italics Text</i><br />
<u>Underlined Text</u><br />
2<sup>nd</sup>nd<br />
h<sub>2</sub>0<br />
<strike>strikethrough text</strike><br />
<big>Text that is one size larger than that sorrounding it</big><br />
<small>Text that is one size smaller than that sorrounding it</small><br />
Below is a horizontal rule<br />
<hr />
</body>
</html>
```

Lists

There are three list types in HTML:

i. unordered list (bullets) — used to group a set of related items in no particular order
ii. ordered list (numbering) — used to group a set of related items in a specific order
iii. description list — used to display name/value pairs such as terms and definitions

Unordered List

An unordered list is a collection of related items that have no special order or sequence. This list is created by using HTML <ul> tag. Each item in the list is marked with a bullet.

You can use type attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc. Following are the possible options −

<ul type = "square">

<ul type = "disc">

<ul type = "circle">

Ordered List

If you are required to put your items in a numbered list instead of bulleted, then HTML ordered list will be used. This list is created by using <ol> tag. The numbering starts at one and is incremented by one for each successive ordered list element tagged with <li>.

You can use type attribute for <ol> tag to specify the type of numbering you like. By default, it is a number. Following are the possible options −

<ol type = "1"> - Default-Case Numerals.

<ol type = "I"> - Upper-Case Numerals.

<ol type = "i"> - Lower-Case Numerals.

<ol type = "A"> - Upper-Case Letters.

<ol type = "a"> - Lower-Case Letters.

You can use start attribute for <ol> tag to specify the starting point of numbering you need. Following are the possible options −

<ol type = "1" start = "4"> - Numerals starts with 4.

<ol type = "I" start = "4"> - Numerals starts with IV.

<ol type = "i" start = "4"> - Numerals starts with iv.

<ol type = "a" start = "4"> - Letters starts with d.

<ol type = "A" start = "4"> - Letters starts with D.

Definition List

HTML supports a list style which is called definition lists where entries are listed like in a dictionary or encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

<dl> − Defines the start of the list

<dt> − A term

<dd> − Term definition

</dl> − Defines the end of the list

Example

```
<html>
<head>
<title>Lists in HTML</title>
</head>
<body>
<ul type="square">
  <li>bread</li>
  <li>coffee beans</li>
  <li>milk</li>
  <li>butter</li>
</ul>
<ol type="i" start="5">
  <li>Gather ingredients</li>
  <li>Mix ingredients together</li>
  <li>Place ingredients in a baking dish</li>
  <li>Bake in oven for an hour</li>
  <li>Remove from oven</li>
  <li>Allow to stand for ten minutes</li>
  <li>Serve</li>
</ol>
<dl>
  <dt>ACSC 226</dt>
  <dd>Web Design and Development</dd>
```

```
  <dt>Name</dt>
  <dd>Value</dd>
  <dt>COSC 103</dt>
  <dd>Introduction to Computer Applications</dd>
</dl>
</body>
</html>
```

Hyperlinks

A webpage can contain various links that take you directly to other pages and even specific parts of a given page. These links are known as hyperlinks. Hyperlinks allow visitors to navigate between Web sites by clicking on words, phrases, and images. Thus you can create hyperlinks using text or images available on a webpage.

A link is specified using HTML tag <a>. This tag is called anchor tag and anything between the opening <a> tag and the closing tag becomes part of the link and a user can click that part to reach to the linked document.

```
<! DOCTYPE html>
<html>
    <head>
    <title>Hyperlink Example</title>
  </head>
 <body>
    <p>Click following link</p>
    <a href = "https://www.chuka.ac.ke" target = "_self"> Chuka University Website</a>
  </body>
</html>
```

The target attribute is used to specify the location where linked document is opened. Following are the possible options −

_blank (Opens the linked document in a new window or tab.)

_self (Opens the linked document in the same frame.)

_parent (Opens the linked document in the parent frame.)

_top (Opens the linked document in the full body of the window.)

_targetframe (Opens the linked document in a named targetframe.)

Student to research on how to link to different sections of the same page. Also research on email links, and image links.

Images

Images are also known as Graphics or Pictures. Images enhance visual appearance of the web pages by making them more interesting and colorful. The <img> tag is used to insert images in the HTML documents. It is an empty element and contains attributes only. The basic syntax of the <img> tag can be given with:

<img src="url" alt="some_text">

Each image must carry at least two attributes: the src attribute, and an alt attribute. The src attribute tells the browser where to find the image. Its value is the URL of the image file.

Whereas, the alt attribute provides an alternative text for the image, if it is unavailable or cannot be displayed for some reason. Its value should be a meaningful substitute for the image. The width and height attributes are used to specify the width and height of an image. The values of these attributes are interpreted in pixels by default.

<img src="logo.jpg" alt="chuka university logo" width="300" height="300">

Tables

HTML table allows you to arrange data into rows and columns. They are commonly used to display tabular data like product listings, customer's details, financial reports, and so on. A table can also be used to define the structure / layout of a HTML form.

You can create a table using the <table> element. Inside the <table> element, you can use the <tr> elements to create rows, and to create columns inside a row you can use the <td> elements. You can also define a cell as a header for a group of table cells using the <th> element. By default, table headings are bold and centered.

If you do not specify a border for the table, it will be displayed without borders. A border is set using the border property.

Spanning allow you to extend table rows and columns across multiple other rows and columns. Normally, a table cell cannot pass over into the space below or above another table cell. But, you can use the rowspan or colspan attributes to span multiple rows or columns in a table.

Cell padding specifies the space between the cell content and its borders. If you do not specify a padding, the table cells will be displayed without padding. To set the padding, use the cellpadding property.

Border spacing specifies the space between the cells. To set the border spacing for a table, use the cellspacing property.

You can specify a caption (or title) for your tables using the <caption> element. The <caption> element must be placed directly after the opening <table> tag. By default, caption appears at the top of the table.

```
<html>
<head>
<title>Tables in HTML</title>
</head>
<body>
<table border align="center" width="70%">
<tr>
<th colspan="2">Chuka University</th>
</tr>
<tr>
<th>Course Code</th>
<th>Course Name</th>
</tr>
<tr>
<td>ACSC 226</td>
<td>Web Design and Development</td>
</tr>
<tr>
<td>ACSC 327</td>
<td>Web Programming and Administration</td>
</tr>
<tr>
<td>COSC 280</td>
<td rowspan="2"> </td>
</tr>
<tr>
```

<td> </td>

</tr>

</table>

</body>

</html>

HTML Forms

HTML Forms are required to collect different kinds of user inputs, such as contact details like name, email address, phone numbers, or details like credit card information, etc.

Forms contain special elements called controls like inputbox, checkboxes, radio-buttons, submit buttons, etc. Users generally complete a form by modifying its controls e.g. entering text, selecting items, etc. and submitting this form to a web server for further processing.

The HTML <form> tag is used to create an HTML form and it has following syntax −

<form action = "Script URL" method = "GET|POST">

form elements like input, textarea etc.

</form>

Apart from common attributes, following is a list of the most frequently used form attributes −

Action - Backend script ready to process your passed data.

Method - Method to be used to upload data. The most frequently used are GET and POST methods.

Target - Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.

Enctype - You can use the enctype attribute to specify how the browser encodes the data before it sends it to the server. Possible values are

Application/x-www-form-urlencoded − This is the standard method most forms use in simple scenarios.

Mutlipart/form-data − This is used when you want to upload binary data in the form of files like image, word file etc.

HTML Form Controls

There are different types of form controls that you can use to collect data using HTML form:

- Text Input Controls
- Checkboxes Controls

- Radio Box Controls
- Select Box Controls
- File Select boxes
- Hidden Controls
- Clickable Buttons
- Submit and Reset Button

Text Input Controls

There are three types of text input used on forms −

- **Single-line text input controls** − This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML **<input>** tag.
- **Password input controls** − This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input> tag.
- **Multi-line text input controls** − This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML **<textarea>** tag.

Single-line text input controls

This control is used for items that require only one line of user input, such as search boxes or names. They are created using HTML <input> tag.

Password input controls

This is also a single-line text input but it masks the character as soon as a user enters it. They are also created using HTML <input>tag but type attribute is set to **password**.

Following is the list of attributes for <input> tag for creating text field.

type - Indicates the type of input control and for text input control it will be set to text.

name - Used to give a name to the control which is sent to the server to be recognized and get the value.

value - This can be used to provide an initial value inside the control.

size - Allows to specify the width of the text-input control in terms of characters.

maxlength - Allows to specify the maximum number of characters a user can enter into the text box.

Multiple-Line Text Input Controls

This is used when the user is required to give details that may be longer than a single sentence. Multi-line input controls are created using HTML <textarea> tag.

name- Used to give a name to the control which is sent to the server to be recognized and get the value.

rows- Indicates the number of rows of text area box.

cols- Indicates the number of columns of text area box

```
<html>
<head>
<title>Text Form Controls</title>
</head>
<body>
<form action=" " method="post">
UserName:
<input type="text" name="txtUserName" /><br />
Password:
<input type="password" name="txtPassword"/><br />
Description : <br />
    <textarea rows = "5" cols = "50" name = "description">
      Enter description here...
    </textarea>
</form>
</body>
</html>
```

Checkbox Control

Checkboxes are used when more than one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to checkbox.

Following is the list of attributes for <checkbox> tag.

type- Indicates the type of input control and for checkbox input control it will be set to checkbox..

name- Used to give a name to the control which is sent to the server to be recognized and get the value.

value- The value that will be used if the checkbox is selected.

checked- Set to checked if you want to select it by default.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Checkbox Control</title>
  </head>

  <body>
    <form method="post" action="">
      <input type = "checkbox" name = "chkmaths" value = "on"> Maths
      <input type = "checkbox" name = "chkphysics" value = "on"> Physics
    </form>
  </body>
</html>
```

Radio Button Control

Radio buttons are used when out of many options, just one option is required to be selected. They are also created using HTML <input> tag but type attribute is set to radio.

Following is the list of attributes for radio button.

type- Indicates the type of input control and for checkbox input control it will be set to radio.

name- Used to give a name to the control which is sent to the server to be recognized and get the value.

value- The value that will be used if the radio box is selected.

checked- Set to checked if you want to select it by default.

```
<!DOCTYPE html>
<html>
```

```
  <head>
    <title>Radio Box Control</title>
  </head>
  <body>
    <form method="post" action="">
      <input type = "radio" name = "subject" value = "radiomaths"> Maths
      <input type = "radio" name = "subject" value = "radiophysics"> Physics
    </form>
  </body>
</html>
```

Select Box Control

A select box, also called drop down box which provides option to list down various options in the form of drop down list, from where a user can select one or more options.

The following is the list of important attributes for the <select> tag:

name- Used to give a name to the control which is sent to the server to be recognized and get the value.

size- This can be used to present a scrolling list box.

multiple- If set to "multiple" then allows a user to select multiple items from the menu.

The following is the list of important attributes for the <option> tag:

value- The value that will be used if an option in the select box box is selected.

selected- Specifies that this option should be the initially selected value when the page loads.

label- An alternative way of labeling options

```
<!DOCTYPE html>
<html>
  <head>
    <title>Select Box Control</title>
```

```
  </head>

  <body>
    <form method="post" action="">
      <select name = "seldropdown">
        <option value = "Maths" selected>Maths</option>
        <option value = "Physics">Physics</option>
      </select>
    </form>
  </body>
</html>
```

File Upload Box

If you want to allow a user to upload a file to your web site, you will need to use a file upload box, also known as a file select box. This is also created using the <input> element but type attribute is set to file.

Following is the list of important attributes of file upload box −

name- Used to give a name to the control which is sent to the server to be recognized and get the value.

accept- Specifies the types of files that the server accepts.

```
<html>
  <head>
    <title>File Upload Box</title>
  </head>
  <body>
    <form method="post" action="">
      <input type = "file" name = "fileupload" accept = "image/*" />
    </form>
  </body>
```

```
</html>
```

Button Controls

There are various types of clickable buttons in HTML forms. These include:

submit- This creates a button that automatically submits a form.

reset- This creates a button that automatically resets form controls to their initial values.

button- This creates a button that is used to trigger a client-side script when the user clicks that button.

image- This creates a clickable button but we can use an image as background of the button

```
<html>
  <head>
    <title>Buttons in HTML Forms</title>
  </head>
  <body>
    <form method="post" action="">
      <input type = "submit" name = "btnsubmit" value = "Submit" />
      <input type = "reset" name = "btnreset"  value = "Reset" />
      <input type = "button" name = "btnok" value = "OK" />
      <input type = "image" name = "btnimagebutton" src = "logo.png" />
    </form>
  </body>
</html>
```

Hidden Form Controls

Hidden form controls are used to hide data inside the page which later on can be pushed to the server. This control hides inside the code and does not appear on the actual page. For example, following hidden form is being used to keep current page number. When a user will click next page then the value of hidden control will be sent to the web server and there it will decide which page will be displayed next based on the passed current page.

```
<html>
  <head>
    <title>File Upload Box</title>
```

```
  </head>
  <body>
    <form method="post" action="">
      <p>This is page 10</p>
      <input type = "hidden" name = "pagename" value = "10" />
      <input type = "submit" name = "btnsubmit" value = "Submit" />
      <input type = "reset" name = "btnreset"  value = "Reset" />
    </form>
  </body>
</html>.
```

CSS

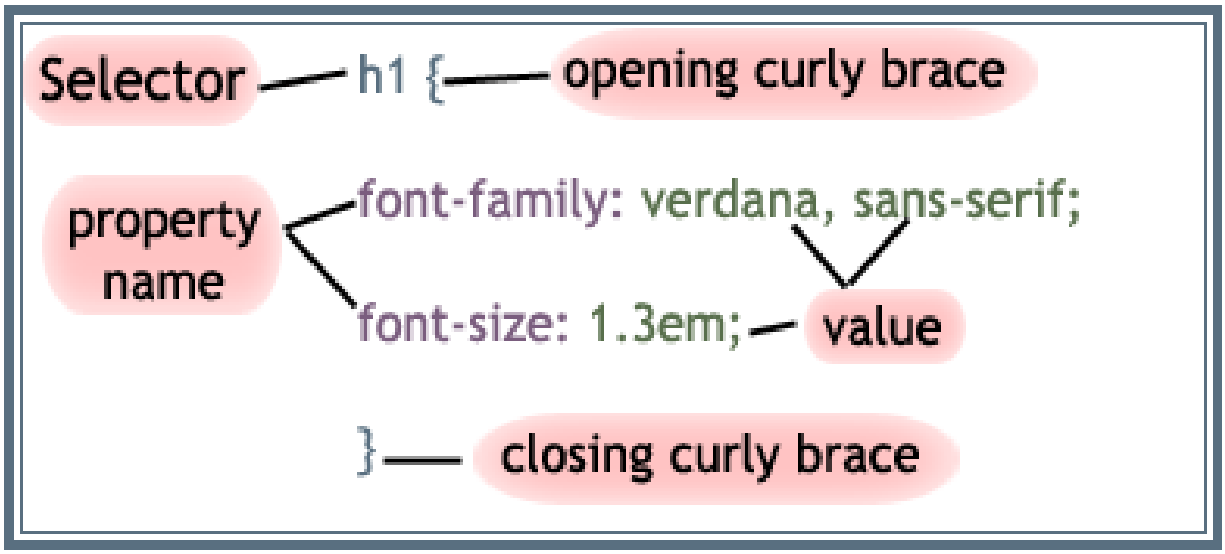
CSS is an abbreviation for Cascading Style Sheets. CSS works with HTML and other Markup Languages (such as XHTML and XML) to control the way the content is presented. Cascading Style Sheets enables to separate the appearance of a webpage from the content of a webpage. CSS is a recommendation of the World Wide Web Consortium (the W3C) for defining the presentation and look of a website. To achieve this, CSS makes use of styles which are presented in a style sheet.

Advantages of CSS in Web Design and Development

- Increases download speed- In a webpage which doesn't use CSS, the styling is achieved using formatting tags which in essence overloads the HTML content. HTML was intended for content not for the look. Use of CSS in a website reduces HTML 'clutter' hence compacting the files,
- Reduces time spend maintaining the website
- A more uniform website

A style sheet **consists of one or more rules** that describe how document elements should be displayed. Each rule in CSS has two parts: the selector and the declaration; and the declaration also has two parts, the property and the value. The selector defines any HTML element for which a change is deemed necessary e.g. body, p, table, th, hr etc. The declaration contains the property and value for the selector. The property is the attribute you wish to change and each property can take a value. The property and value are separated by a colon and surrounded by curly braces:

An example of a CSS Style Sheet rule and component parts is shown below:



Examples

body { background-color: black}

If the value of a property is more than one word, put quotes around that value:

body { font-family: "sans serif"; }

If you wish to specify more than one property, you must use a semi-colon to separate each property. This rule defines a paragraph that will have blue text that is centered.

p { text-align: center; color: blue }

You can group selectors. Separate each selector with a comma. The example below groups headers 1, 2, and 3 and makes them all yellow.

h1, h2, h3 { color: yellow}

Other Examples

Example 1

h1, h2, h3 {

font-weight : bold;

font-family : arial;

color : #000000;

background-color : #FFFFFF;

```
}
```

Example 2

```
body {
color:#000000;
background-color : #ffffff;
font-family : arial, verdana, sans-serif;
}
```

Example 3

```
h1 {
font-size : 18px;
}
p {
font-size : 12px;
}
```

Example 4

```
table {
background-color : #efefef;
border-style : solid;
border-width : 1px;
border-color : #999999;
}
```

Example 5

```
th {
background-color : #cccccc;
font-weight : bold;
padding : 5px;
}
```

Example 6

```
p {
   font-family: "Times New Roman", Times, serif;
}
```

The font-style property is mostly used to specify italic text

```
h1 {
   font-size: 40px;
}

h2 {
   font-size: 30px;
}

p {
   font-size: 14px;
}
```

Example 7

```
p.normal {
   font-weight: normal;
}

p.thick {
   font-weight: bold;
}
```

Example 8

```
body {
   color: blue;
}

h1 {
   color: green;
}
```

Example 9

```
h1 {
   text-align: center;
}

h2 {
```

```
    text-align: left;
}

h3 {
    text-align: right;
}
```

Example 10

```
h1 {
    text-decoration: overline;
}

h2 {
    text-decoration: line-through;
}

h3 {
    text-decoration: underline;
}

a {
    text-decoration: none;
}
```

Example 11

```
p.uppercase {
    text-transform: uppercase;
}

p.lowercase {
    text-transform: lowercase;
}

p.capitalize {
    text-transform: capitalize;
}
```

Attaching CSS Styles to Web Development

There are three ways that styles can be associated with an HTML document.

First, styles can be placed inline in a document. Second, a style sheet can be embedded in the head of an HTML document. The third way of associating web pages with style sheets is to place a link in the head of the HTML file to an external style sheet. With this link, when the browser

begins reading the page, it sees the style sheet link, and downloads the style sheet, then uses the style sheet to draw the page.

Inline CSS Styles

Style rules can be specified right in the HTML tag, with **style="style rules"**. For example:

** **

The style rules above look like this!

** **

Specifying inline styles is useful when you only want to apply those styles to a specific element, as is the case above. For example, if you wanted to emphasize a piece of text in your sentence, you could specify a local style of "color: red" to a element so that it looks like this

it looks like this

Internal (Embedded) CSS Style Sheets

Style sheets can be embedded into the <head> element of HTML documents. To do this, the style sheet itself is placed inside a <style> tag within the <head></head> section. The styles defined within the internal CSS are applicable to the content of a whole webpage in which it is defined.

Example of an internal CSS style:

<head>

<style type="text/css">

hr

{

color: brown

}

p

{

font-family: Arial, verdana, "sans-serif";

}

body

{

background-color: #ffffff

}

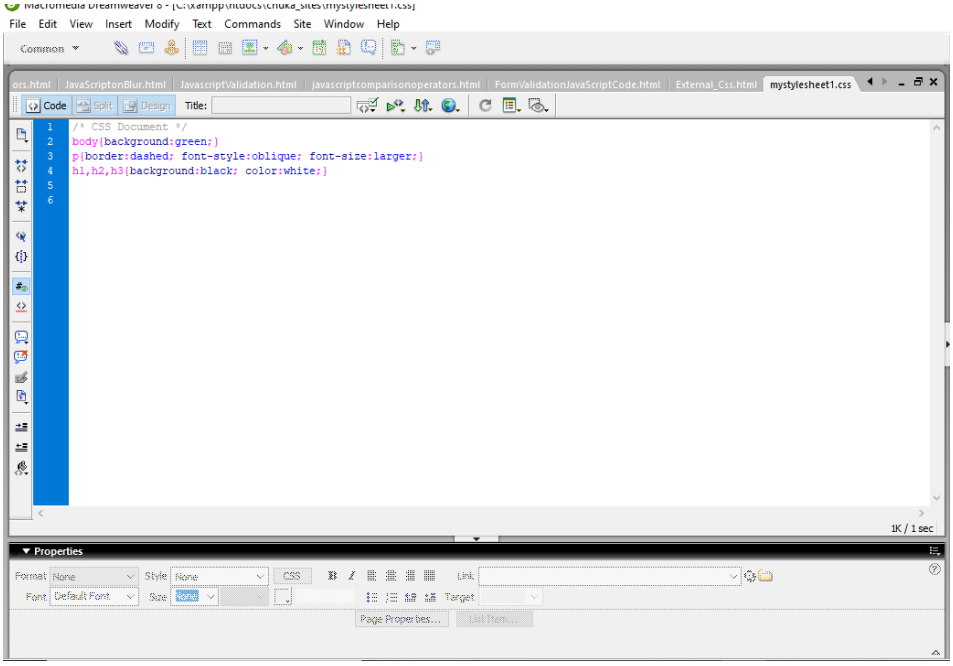
</style>

</head>

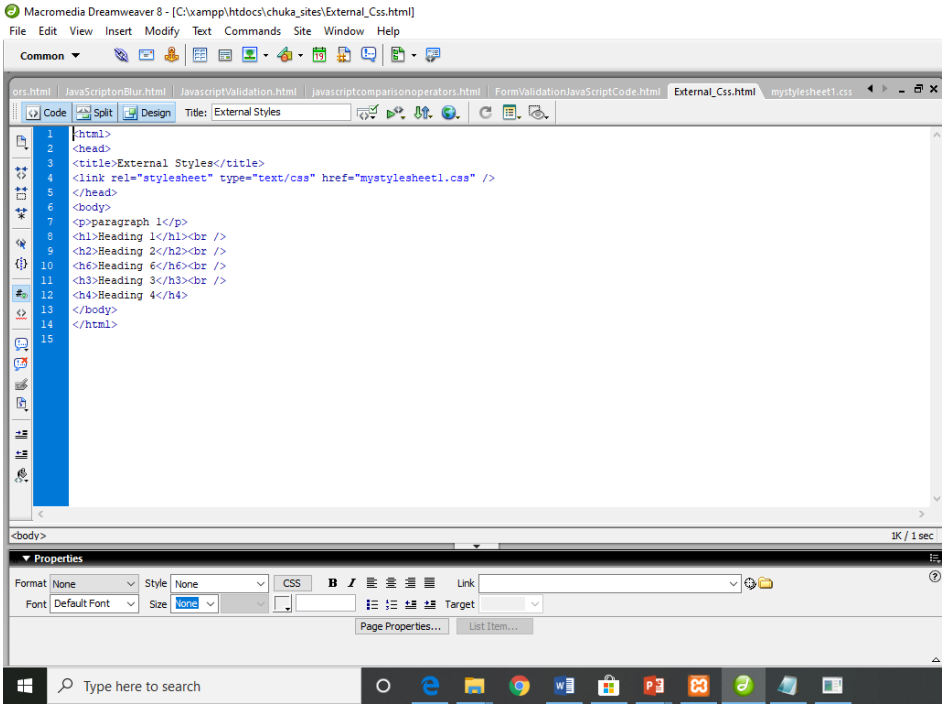
External CSS Style Sheets

An external style sheet is just a text file, containing **property: value** style rules. The only difference between the contents of an external style sheet and having your style rules in the document itself is that you do **not** include the <style> </style> tags, or any other HTML tags. To link a HTML document to a style sheet, you place a link to the style sheet in the head of the document. Linking to an external style sheet is ideal when applying styles to many pages. With an external style sheet, you can change the look of an entire web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

Step 1: Create a .css file by opening Macromedia Dreamweaver→New→Basic Page→CSS



Step2: Call the .css external file in your webpage as needed



Types of CSS selectors

The Type Selector

```
<html>
<head>
<title>Untitled Document</title>
<style type="text/css">
h4
{
color:#6600FF
}
</style>
</head>
<body>
<h4>Heading 4</h4>
</body>
</html>
```

Universal Selector

Rather than selecting elements of a specific type, the universal selector matches the name of any element type. This rule renders the content of every element in our document in green.

```
 <html>
<head>
```

```
<title>Untitled Document</title>
<style type="text/css">
*
{
color: green;
}
</style>
</head>
<body>
<p>Hi there</p>
<br />
<h2>I am H2</h2>
<br />
<h4>Heading 4</h4>
</body>
</html>
```

Descendants Selector

It is used to apply a style rule to an element only when it lies inside a particular element. As given in the following example, the style rule will apply to element only when it lies inside the <ul> tag.

```
<html>
<head>
<title>Untitled Document</title>
<style type="text/css">
ul em
{
color: maroon;
}
</style>
</head>
<body>
<ul>
<li>
<em>Oranges</em>
</li>
<li><em>Mangoes</em></li>
</ul>
<em>Hello, I am for emphasis</em>
```

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

Class Selector

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
<html>
<head>
<title>Class Selectors</title>
<style type="text/css">
.red
{
color: red;
}
</style>
</head>
<body>
<h1 class=“red">Heading 1 referencing class selector named black</h1>
<br />
<h1> Heading 1 NOT referencing any selector class</h1>
<p class=“red">Paragraph referencing the class selector</p>
</body>
</html>
```

This rule renders the content in red for every element with class attribute set to *red* in our document

This can be made a bit more specific.

```
<html>
<head>
<style type="text/css">
h1.blue {
color: blue;
}
</style>
</head>
<body>
<h1 class="blue">H1 inheriting the specific class CSS rule</h1>
<p class="blue">Paragraph referencing the specific class. No change visible</p>
</body>
</html>
```

This rule renders the content in blue for only <h1> elements with class attribute set to *blue*.

You can apply more than one class selectors to a given element

```
<html>
<head>
<title>Many Class Selectors</title>
<style type="text/css">
.size36
{
font-size:36px;
}
.decoration
{
text-decoration:line-through;
}
.magenta
{
color:magenta;
}
</style>
</head>
<body>
<p class="decoration size36 magenta">Paragraph referencing many css class selectors</p>
<p>paragraph without styles</p>
</body>
</html>
```

CSS IDs

A *style ID* is a custom set of formatting specifications that can be applied only to one element in a web page. You can use IDs across a set of pages but only once per time within each page. For example, suppose you have a title within the body of all your pages. Each page has only one title, but all the pages themselves include one instance of that title. Following is an example of a selector with an ID indicated, plus a property and a value:

```
p#title

{

font: 24pt Verdana, Geneva, Arial, sans-serif

}
```

Notice that this selector includes a hash mark, or pound sign (#), after p, followed by a descriptive ID name. When referencing a style ID in HTML code, simply specify the ID name in the id attribute of an element, like so:

```
<p id=”title”>Some Title Goes Here</p>
```

Everything between the opening and closing tags will appear in 24- point Verdana text—but only once on any given page. You will often see style IDs used to define specific parts of a page for layout purposes, such as a header area, footer area, main body area, and so on. These types of areas in a page will appear only once per page, so using an ID rather than a class is the appropriate choice. An ID attribute identifies one element on a page. An ID selector is a name preceded by a hash character (#).

Example 2:

#class { color: navy; border: 1px black solid; }

This html is written:

#class { color: navy; border: 1px black solid; }

Using CSS on different elements

Background

You can set the following background properties of an element:

The **background-color** property is used to set the background color of an element.

The **background-image** property is used to set the background image of an element.

The **background-repeat** property is used to control the repetition of an image in the background.

The **background-position** property is used to control the position of an image in the background.

The **background-attachment** property is used to control the scrolling of an image in the background.

The **background** property is used as a shorthand to specify a number of other background properties.

Examples

This text has a yellow background color.

<table style="background-image:url(/images/pattern1.gif);">

<tr><td>

This table has background image set.

</td></tr>

</table>

The following example demonstrates how to set the background image position 100 pixels away from the left side and 200 pixels down from the top.

<table style="background-image:url(/images/pattern1.gif);

background-position:100px 200px;">

<tr><td>

This table has background image positioned 100 pixels away from the left and 200 pixels from the top.

</td></tr>

</table>

Background attachment determines whether a background image is fixed or scrolls with the rest of the page. The following example demonstrates how to set the fixed background image.

<p style="background-image:url(/images/pattern1.gif);

background-attachment:fixed;">

This paragraph has fixed background image.

The following example demonstrates how to set the scrolling background image.

<p style="background-image:url(/images/pattern1.gif);

background-attachment:scroll;">

This parapgraph has scrolling background image.

You can use the *background* property to set all the background properties at once. For example:

This paragraph has fixed repeated background image.

Fonts

The **font-family** property is used to change the face of a font.

The **font-style** property is used to make a font italic or oblique.

The **font-variant** property is used to create a small-caps effect.

The **font-weight** property is used to increase or decrease how bold or light a font appears.

The **font-size** property is used to increase or decrease the size of a font.

The **font** property is used as shorthand to specify a number of other font properties.

Examples

This text is rendered in either georgia, garamond, or the default serif font depending on which font you have at your system.

The following example demonstrates how to set the font style of an element. Possible values are *normal, italic and oblique*.

The following example demonstrates how to set the font variant of an element. Possible values are *normal and small-caps*.

This text will be rendered as small caps

The following example demonstrates how to set the font weight of an element. The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900*.

This font is bold.

This font is bolder.

This font is 900 weight.

The following example demonstrates how to set the font size of an element. The font-size property is used to control the size of fonts. Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %*.

This font size is 20 pixels

This font size is small

This font size is large

You can use the *font* property to set all the font properties at once. For example:

Applying all the properties on the text at once.

Text

Text can also be manipulated using CSS properties. You can set the following text properties of an element:

The **color** property is used to set the color of a text.

The **direction** property is used to set the text direction.

The **letter-spacing** property is used to add or subtract space between the letters that make up a word.

The **word-spacing** property is used to add or subtract space between the words of a sentence.

The **text-indent** property is used to indent the text of a paragraph.

The **text-align** property is used to align the text of a document.

The **text-decoration** property is used to underline, overline, and strikethrough text.

The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.

The **white-space** property is used to control the flow and formatting of text.

The **text-shadow** property is used to set the text shadow around a text.

Examples

The following example demonstrates how to set the text color. Possible value could be any color name in any valid format.

This text will be written in red.

The following example demonstrates how to set the direction of a text. Possible values are *ltr or rtl*.

This text will be rendered from right to left

The following example demonstrates how to set the space between characters. Possible values are *normal or a number specifying space.*

This text is having space between letters.

The following example demonstrates how to set the space between words. Possible values are *normal or a number specifying space.*

This text is having space between words.

The following example demonstrates how to indent the first line of a paragraph. Possible values are *% or a number specifying indent space.*

This text will have first line indented by 1cm and this line will remain at its actual position this is done by CSS text-indent property.

The following example demonstrates how to align a text. Possible values are *left, right, center, justify.*

This will be right aligned.

This will be center aligned.

This will be left aligned.

The following example demonstrates how to decorate a text. Possible values are *none, underline, overline, line-through, blink.*

This will be underlined

This will be striked through.

This will have a over line.

This text will have blinking effect

The following example demonstrates how to set the cases for a text. Possible values are *none, capitalize, uppercase, lowercase.*

This will be capitalized

This will be in uppercase

This will be in lowercase

The following example demonstrates how to set the shadow around a text. This may not be supported by all the browsers.

If your browser supports the CSS text-shadow property, this text will have a blue shadow.

Images

Images play an important role in any webpage. Though it is not recommended to include a lot of images, but it is still important to use good images wherever required. CSS plays a good role to control image display. You can set the following image properties using CSS.

The **border** property is used to set the width of an image border.

The **height** property is used to set the height of an image.

The **width** property is used to set the width of an image.

The **-moz-opacity** property is used to set the opacity of an image.

The *border* property of an image is used to set the width of an image border. This property can have a value in length or in %.

A width of zero pixels means no border.

The *height* property of an image is used to set the height of an image. This property can have a value in length or in %. While giving value in %, it applies it in respect of the box in which an image is available.

<img style="border:1px solid red; height:100px;"

src="/images/css.gif" />

<img style="border:1px solid red; height:50%;"

src="/images/css.gif" />

The *width* property of an image is used to set the width of an image. This property can have a value in length or in %. While giving value in %, it applies it in respect of the box in which an image is available.

<img style="border:1px solid red; width:100px;"

src="/images/css.gif" />

<img style="border:1px solid red; width:100%;"

src="/images/css.gif" />

The *-moz-opacity* property of an image is used to set the opacity of an image. This property is used to create a transparent image in Mozilla. IE uses **filter:alpha(opacity=x)** to create transparent images.

In Mozilla (-moz-opacity:x), x can be a value from 0.0 - 1.0. A lower value makes the element more transparent (The same things goes for the CSS3-valid syntax opacity:x).

In IE (filter:alpha(opacity=x)), x can be a value from 0 - 100. A lower value makes the element more transparent.

<img style="border:1px solid red;-moz-opacity:0.4;filter:alpha(opacity=40);"

src="/images/css.gif" />

Links

There are different properties of a hyper link that can be set using CSS. For example, you can set the following properties of a hyperlink:

The **:link** signifies unvisited hyperlinks.

The **:visited** signifies visited hyperlinks.

The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.

The **:active** signifies an element on which the user is currently clicking.

Usually, all these properties are kept in the header part of the HTML document.

Remember a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective. Also, a:active MUST come after a:hover in the CSS definition as follows:

<style type="text/css">

a:link {color: #000000}

a:visited {color: #006600}

a:hover {color: #FFCC00}

a:active {color: #FF00CC}

</style>

<style type="text/css">

a:active {color: #FF00CC}

</style>

Click This Link

Tables

The **border-collapse** specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.

The **border-spacing** specifies the width that should appear between table cells.

The **caption-side** captions are presented in the <caption> element. By default, these are rendered above the table in the document. You use the *caption-side* property to control the placement of the table caption.

The **empty-cells** specifies whether the border should be shown if a cell is empty.

The **table-layout** allows browsers to speed up the layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.

This property can have two values *collapse* and *separate*. The following example uses both the values:

<style type="text/css">

table.one {border-collapse:collapse;}

table.two {border-collapse:separate;}

td.a {

border-style:dotted;

border-width:3px;

border-color:#000000;

padding: 10px;

}

td.b {border-style:solid;

border-width:3px;

```
border-color:#333333;
padding:10px;
}
</style>
<table class="one">
<caption>Collapse Border Example</caption>
<tr><td class="a"> Cell A Collapse Example</td></tr>
<tr><td class="b"> Cell B Collapse Example</td></tr>
</table>
<br />
<table class="two">
<caption>Separate Border Example</caption>
<tr><td class="a"> Cell A Separate Example</td></tr>
<tr><td class="b"> Cell B Separate Example</td></tr>
</table>
```

The border-spacing property specifies the distance that separates the adjacent cells' borders. It can take either one or two values; these should be units of length.

If you provide one value, it applies to both vertical and horizontal borders. Or you can specify two values, in which case, the first refers to the horizontal spacing and the second to the vertical spacing:

```
<style type="text/css">
/* If you provide one value */
table.example {border-spacing:10px;}
/* This is how you can provide two values */
table.example {border-spacing:10px; 15px;}
</style>
```

The caption-side property allows you to specify where the content of a <caption> element should be placed in relationship to the table. The table that follows lists the possible values.

This property can have one of the four values *top*, *bottom*, *left*, or *right*. The following example uses each value.

<style type="text/css">

caption.top {caption-side:top}

caption.bottom {caption-side:bottom}

caption.left {caption-side:left}

Borders

The *border* properties allow you to specify how the border of the box representing an element should look. There are three properties of a border you can change:

The **border-color** specifies the color of a border.

The **border-style** specifies whether a border should be solid, dashed line, double line, or one of the other possible values.

The **border-width** specifies the width of a border.

The border-color property allows you to change the color of the border surrounding an element. You can individually change the color of the bottom, left, top and right sides of an element's border using the properties:

border-bottom-color changes the color of bottom border.

border-top-color changes the color of top border.

border-left-color changes the color of left border.

border-right-color changes the color of right border.

<style type="text/css">

p.example1{

border:1px solid;

border-bottom-color:#009900; /* Green */

border-top-color:#FF0000; /* Red */

border-left-color:#330000; /* Black */

border-right-color:#0000CC; /* Blue */

}

p.example2{

border:1px solid;

border-color:#009900; /* Green */

}

</style>

This example is showing all borders in different colors.

This example is showing all borders in green color only.

The border-style property allows you to select one of the following styles of border:

none: No border. (Equivalent of border-width:0;)

solid: Border is a single solid line.

dotted: Border is a series of dots.

dashed: Border is a series of short lines.

double: Border is two solid lines.

groove: Border looks as though it is carved into the page.

ridge: Border looks the opposite of groove.

inset: Border makes the box look like it is embedded in the page.

outset: Border makes the box look like it is coming out of the canvas.

hidden: Same as none, except in terms of border-conflict resolution for table elements.

You can individually change the style of the bottom, left, top, and right borders of an element using the following properties:

border-bottom-style changes the style of bottom border.

border-top-style changes the style of top border.

border-left-style changes the style of left border.

border-right-style changes the style of right border.

This is a border with none width.

```
This is a solid border.
</p>
<p style="border-width:4px; border-style:dashed;">
This is a dahsed border.
</p>
<p style="border-width:4px; border-style:double;">
This is a double border.
</p>
<p style="border-width:4px; border-style:groove;">
This is a groove border.
</p>
<p style="border-width:4px; border-style:ridge">
This is aridge border.
</p>
<p style="border-width:4px; border-style:inset;">
This is a inset border.
</p>
<p style="border-width:4px; border-style:outset;">
This is a outset border.
</p>
<p style="border-width:4px; border-style:hidden;">
This is a hidden border.
</p>
<p style="border-width:4px;
border-top-style:solid;
border-bottom-style:dashed;
border-left-style:groove;
border-right-style:double;">
```

This is a a border with four different styles.

The border-width property allows you to set the width of an element borders. The value of this property could be either a length in px, pt, or cm, or it should be set to *thin, medium, or thick.*

You can individually change the width of the bottom, top, left, and right borders of an element using the following properties:

border-bottom-width changes the width of bottom border.

border-top-width changes the width of top border.

border-left-width changes the width of left border.

border-right-width changes the width of right border.

This is a solid border whose width is 4px.

This is a solid border whose width is 4pt.

This is a solid border whose width is thin.

This is a solid border whose width is medium;

This is a solid border whose width is thick.

<p style="border-bottom-width:4px;

border-top-width:10px;

border-left-width: 2px;

border-right-width:15px;

border-style:solid;">

This is a a border with four different width.

The border property allows you to specify color, style, and width of lines in one property:

The following example shows how to use all the three properties into a single property. This is the most frequently used property to set border around any element.

This example is showing shorthand property for border.

Margin

The *margin* property defines the space around an HTML element. It is possible to use negative values to overlap content. The values of the margin property are not inherited by the child elements. Remember that the adjacent vertical margins (top and bottom margins) will collapse into each other so that the distance between the blocks is not the sum of the margins, but only the greater of the two margins or the same size as one margin if both are equal.

We have the following properties to set an element margin.

· The **margin** specifies a shorthand property for setting the margin properties in one declaration.

· The **margin-bottom** specifies the bottom margin of an element.

· The **margin-top** specifies the top margin of an element.

· The **margin-left** specifies the left margin of an element.

· The **margin-right** specifies the right margin of an element.

JAVASCRIPT

JavaScript is a scripting language used for client side scripting to provide dynamism to a website. It is not related to Java. It was developed by Netscape 2.0 in 1995 and was initially named **LiveScript**. Later Netscape changed the name to JavaScript, possibly because of the excitement being generated by Java. JavaScript is a general-purpose language and the core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

There are many reasons why a Web Developer may benefit from by inclusion of JavaScript as a web development tool. These reasons include:

a) **Less server interaction: -** When users are giving data through a form, validation of data is necessary. For example when users are inputting their email address we would want to ensure that the email address is in the right format of a valid email address. Without JavaScript, this validation is done by the server. However through JavaScript you can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

b) **Immediate feedback to the visitors:** JavaScript enables one to define object level validation such that is a certain data is required, the user must fill in this data before moving forward. Users do not have to wait for a page reload to see if they have forgotten to enter something.

c) **Increased interactivity:** JavaScript introduces dynamism and interactivity to a web system. For example you can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

d) **Richer interfaces:** You can use JavaScript for creation of richer user interfaces. Through JavaScript a developer can include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

JavaScript can be placed in many parts of a webpage depending on the needed task. The areas of placement include:

i. JavaScript in the <head>...</head> section.

ii. JavaScript in the <body>...</body> section.

iii. JavaScript in both the <body>...</body> and the<head>...</head> sections.

iv. JavaScript as an external file and then called into the webpage by including a call in <head>...</head> section.

JavaScript in the Head Section

When the JavaScript is to run as a response to some event, such as when a user clicks somewhere, then you will place that script in the head section. The following is such an example of a JavaScript function which is executed when a button labelled Say Hello is clicked.

```
<html>
<head>
<script type="text/javascript">
function sayHello ()
{
alert("Hello World");
}
</script>
</head>
<body>
Click here for the result
<input type="button" onClick="sayHello( )" value="Say Hello" />
</body>
</html>
```

Javascript in the Body Section

If you need a script to run as the page loads so that the script generates content in the page, then the script goes in the <body> portion of the document. This can be done without any function defined using JavaScript. In the following example, a string Hello World is written on the webpage using the document. Write inbuilt function when a webpage loads on a browser.

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write ("Hello World")
</script>
<p>This is web page body </p>
</body>
</html>
```

Javascript in Both Head and Body Sections

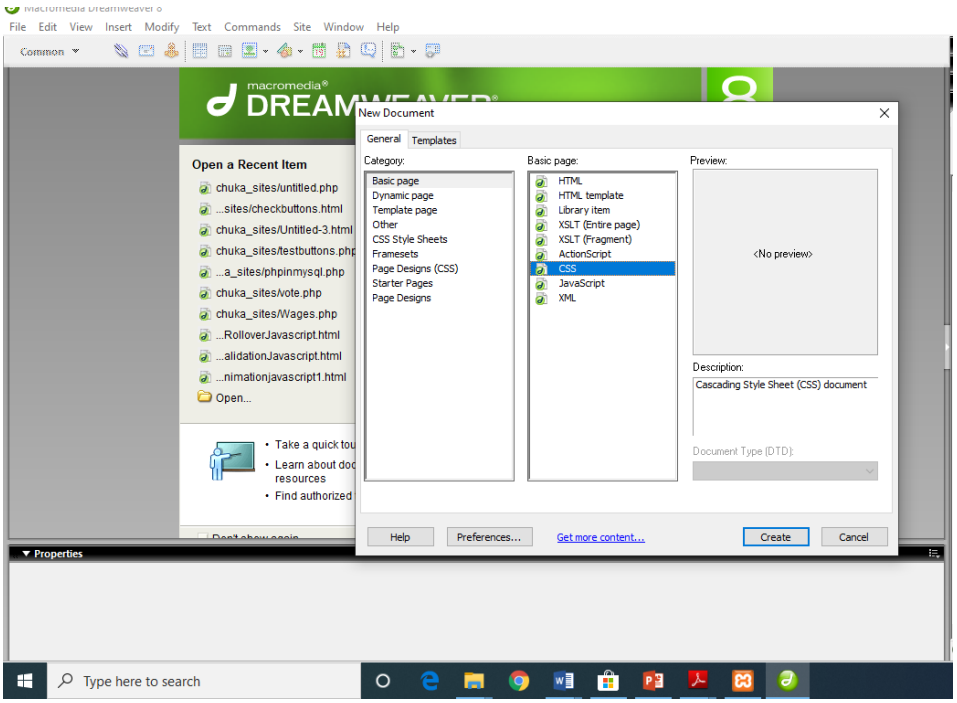
The following example demonstrates the use of JavaScript in both the head and body sections

```
<Html>
<head>
<script type="text/javascript">
function sayHello ( )
{
alert("Hello World")
}
</script>
</head>
```

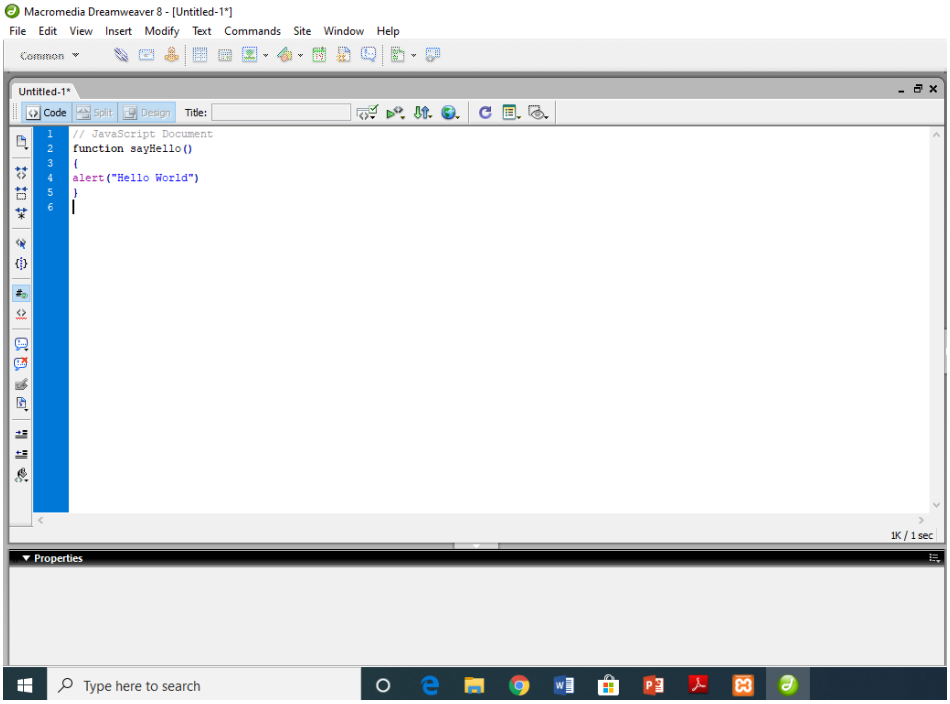
```
<body>
<script type="text/javascript">
document.write("Hello World")
</script>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

Javascript as an External File

For large projects where the developer (s) are reusing identical JavaScript code on multiple pages of a website, the JavaScript code could be placed on a separate file saved with a .js file extension, and for any webpage which needs the JavaScript code the .js file is called at the <head> </head> section. The .js file can be called by as many pages in the project as needful. To create the .js file, open Macromedia Dreamweaver, click File→New→Basic Page→JavaScript



Then enter your JavaScript code and save the file as filename.js



myJavascript.js

function sayHello()

{

alert("Hello World")

}

After creating the .js file, call it into the webpage by referencing it in the head section

Webpage.html

<Html>
<head>
<script type="text/javascript" src="myJavascript.js" >
</script>
</head>
<body>
.....
</body>
</Html>

Javascript Syntax

- **Semi colons are optional**- Simple statements in JavaScript are generally followed by a semicolon character, just as they are in C, C++, and Java. JavaScript, however, allows you to omit this semicolon if each of your statements are placed on a separate line.
- **JavaScript is a case-sensitive language**- This means that the language keywords, variables, function names, and any other identifiers must always be typed with a consistent capitalization of letters.
- **Comments**- JavaScript supports both C-style and C++-style comments. Thus:

  Single line comments:

  // Single line comment

  Or

  <!--. Single line comments //-->.

  Multiline comments

  /* multiline comments */

Javascript Variables

In JavaScript, before you use a variable, you must declare it. Variables are declared with the **var** keyword. The following example demonstrates this:

```
<body>
<script type="text/javascript">
var firstname="judy";
var lastname="cheruto";
document.write(firstname);
</script>
</body>
```

Multiple variables can be declared with the same var keyword as per the example that follows:

```
<body>
<script type="text/javascript">
var firstname, lastname;
firstname="judy";
lastname="cheruto";
document.write(firstname);
</script>
</body>
```

Javascript Data Types

JavaScript allows you to working with three primitive data types and these are:

- **Numbers**, e.g. 123, 120.50 etc.
- **Strings** of text, e.g. "hello world" etc.
- **Boolean**, e.g. true or false, Yes or No etc.

JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value. In addition to these primitive data types, JavaScript supports a composite data type known as **object**.

JavaScript is and **untyped** language which means that a JavaScript variable can hold a value of any data type and hence a developer does not have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.

Javascript Operators

JavaScript supports the following types of operators.

- Arithmetic Operators (+, -, *, /, %, ++, --)Comparison Operators

  == (Equal)

  != (Not Equal)

  > (Greater than)

  < (Less than)

  >= (Greater than or Equal to)

  <= (Less than or Equal to)

- Logical Operators

  && (Logical AND)

  || (Logical OR)

  ! (Logical NOT)

- Assignment Operators

  = (Simple Assignment)

  Assigns values from the right side operand to the left side operand

  Example: C = A + B will assign the value of A + B into C

  += (Add and Assignment)

It adds the right operand to the left operand and assigns the result to the left operand.

Example: C += A is equivalent to C = C + A

-= (Subtract and Assignment)

It subtracts the right operand from the left operand and assigns the result to the left operand.

Example: C -= A is equivalent to C = C - A

***= (Multiply and Assignment)**

It multiplies the right operand with the left operand and assigns the result to the left operand.

Example: C *= A is equivalent to C = C * A

/= (Divide and Assignment)

It divides the left operand with the right operand and assigns the result to the left operand.

Example: C /= A is equivalent to C = C / A

%= (Modules and Assignment)

It takes modulus using two operands and assigns the result to the left operand.

Example: C %= A is equivalent to C = C % A

Javascript Functions

A function is a group of reusable code which can be called anywhere in your program. This eliminates the need of writing the same code again and again. Functions helps programmers in writing modular codes and allows a programmer to divide a big program into a number of small and manageable functions.

Before we use a function, we need to define it. To define a function in JavaScript is by using the:

Function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces as in the format below:

<script type="text/javascript">

Function functionname (parameter-list)

{

Statements;

}

</script

To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the following JavaScript Function Examples.

Example 1

```
<Html>
<head>
<script type="text/javascript">
function greetings ( )
{
document.write ("Hello there!");
}
</script>
</head> <body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick=“greetungs ( )" value=“Greet someone">
</form>
</body>
</Html>
```

Example 2

```
<Html>
<head>
<title>Untitled Document</title>
<script type="text/javascript">
function openwindow ( )
{
window.open("http://www.chuka.ac.ke")
}
</script>
</head>
<body>
<form>
<input type="button" value="open window" onclick="openwindow ( )" />
</form>
</body>
</Html>
```

Example 3

```
<Html>
<head>
```

```
<script type="text/javascript">
function sayHello (name, age)
{
document.write (name + " is " + age + " years old.");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="sayHello('Zara', 7)" value="Say Hello">
</form>
</body>
</html>
```

Example 4

```
<Html>
<head>
<script type="text/javascript">
function concatenate (first, last)
{
var full;
full = first + last;
return full;
}
function secondFunction ( )
{
var result;
result = concatenate('Zara', 'Ali');
document.write (result );
}
</script>
</head>
<body>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
</body>
</html>
```

Javascript Pop up Boxes

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

Alert Pop up Box

An alert box is often used if you want to notify the user of something. When an alert box pops up, the user will have to click "OK" to proceed. The following is an example demonstrating a JavaScript Alert pop up box.

```
<Html>
<head>
<title>JavaScript Alert Pop Up Box</title>
</head>
<body>
<script type="text/javascript">
window.alert ("You have clicked the close button");
</script>
</body>
</Html>
```

OR

```
<Html>
<head>
<title>JavaScript Alert Pop up Box</title>
</head>
<body>
<script type="text/javascript">
alert (“You have clicked the close button");
</script>
</body>
</Html>
```

Confirm Pop up Box

JavaScript uses a confirm box if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. The following two examples demonstrate the use of confirm buttons in JavaScript.

Example 1

```
<Html>
<head>
<title>JavaScript Confirm Pop up Box</title>
</head>
<body>
<script type="text/javascript">
var response;
if(confirm("press a button!"))
{
```

```
response="you pressed OK!";
}
else
{
response="You pressed Cancel!";
}
document.write(response);
</script>
</body>
</Html>
```

Example 2

```
<Html>
<head>
<title>JavaScript Confirm Pop up Box</title>
</head>
<body>
<script type="text/javascript">
var name=confirm("press a button")
if (name==true)
{
document.write("you pressed ok")
}
else
{
document.write("you pressed cancel")
}
</script>
</body>
</Html>
```

Prompt Pop up Box

A JavaScript prompt box is often used if you want the user to input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. The following examples demonstrates the use of a JavaScript Pop up Box.

Example 1

```
<Html>
<Head>
<Title>JavaScript Prompt Pop up Box</title>
</head>
<body>
```

```
<script type="text/javascript">
var person = prompt ("Please enter your name", "Enter Name");
if (person == null || person == "")
{
var response = "User cancelled the prompt.";
}
else
{
var response = "Hello " + person + "! How are you today?";
}
document.write(response);
</script>
</body>
</Html>
```

Example 2

```
<Html>
<head>
<title>JavaScript Prompt Pop up Box</title>
</head>
<body>
<script type="text/javascript">
var name=prompt("please enter your name", " ")
if(name!=null && name!=" ")
{
document.write("Hello"+" "+name)
}
</script>
</body>
</Html>
```

Javascript Radio Buttons

```
<Html>
<head>
<title>JavaScript Radio Button</title>
</head>
<body>
<form>
which browser is your favorite<br />
<input type="radio" name="browser" onclick="check(this.value)"
value="Explorer" />Microsoft Internet Explorer<br />
<input type="radio" name="browser" onclick="check(this.value)"
value="Firefox" />Mozilla Firefox<br />
```

```
<input type="text" name="answer" />
</form>
</body>
</html>
```

Javascript Checkboxes

```
<Html>
<head>
<title>JavaScript Checkboxes</title>
<script type="text/javascript">
function check ( )
{
coffee=document.forms[0].coffee
answer=document.forms[0].answer
txt=" "
for(i=0;i<coffee.length;i++)
{
if(coffee[i].checked)
{
txt=txt+coffee[i].value+" "
}
}
answer.value="you ordered a coffee with"+txt
}
</script>>
</head>
<body>
<form>
How would you like your coffee?<br />
<input type="checkbox" name="coffee" value="cream" />with cream<br />
<input type="checkbox" name="coffee" value="sugar" />With sugar<br />
<input type="text" name="answer" size="30" />
<input type="button" name="test" onclick="check()" value="Order" />
</form>
</body>
</html>
```

Javascript Events and Event Handling

Events are things that happen. For example, the sun rising is an event. The sun setting is an event. You can choose to react to those events. For example, when the sun rises, you might get out of bed or you might not. When the sun sets, you might turn on a light or might go to bed.

In web design and development, events are the things that happen in a web page. For example, a user might move the mouse over a button, click a button, or submit a form. Like the example of the sun rising, you can choose to react to the event or you can ignore it. JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page. The following are examples of common events encountered in a web page:

i. Page Loading
ii. Button Click
iii. Key Press
iv. Closing a Window
v. Resizing a Window

Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

In general, there are four types of events:

- **Form events:** Includes things like selecting a check box or submitting the form itself. Reacting to form events is one of the most common things that JavaScript programmers do.
- **Mouse events:** This can be anything from a mouse click to mouse movements on the page. That’s right; you can actually track where the mouse is on a page and react to it.
- **Keyboard events:** Things that happen through the keyboard, like a key press, are considered keyboard events.
- **Page events:** Things like the page loading or unloading are considered page events.

Response to events is handled through event handlers. The following are examples of event handlers available in JavaScript:

- onBlur
- onMouseOver
- onChange
- onClick
- onFocus
- onLoad
- onSubmit
- onUnload
- onSelect

Below we look at some of these event handlers and examples in JavaScript.

onClick Event Type

This is the most frequently used event type which occurs when a user clicks the left button of his mouse. You can put your validation, warning etc., against this event type. Instances when an onClick event handler is called include when:

- An object in a button (regular, radio, reset and submit) is clicked,
- A link is pushed
- A checkbox is checked or
- An image map area is selected.

The following is an example of an onClick event JavaScript code:

```
<Html>
<head>
<script type="text/javascript">
function sayHello ( )
{
document.write ("Hello World") ;
}
</script>
</head>
<body>
<p> Click the following button and see result</p>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

onChange Event Type

A change event occurs when a select text or text area field loses focus and its value has been modified. The onChange event handler executes JavaScript code when a change event occurs. The onChange event handler is used to validate data after it's modified by a user. The following is an example of onChange in action:

```
<Html>
<title>example of onChange event handler</title> <head>
<script type="text/javascript">
function validchange (form)
{
var input;
input=document.frmChuka.txtFees.value;
```

```
alert("You have changed the Fees from  to 45000 " + input );
}
</script>
</head>
<body>
Try changing the value from 45000 to something else:<br>
<form name="frmChuka">
<input type="text" name="txtFees" value="45000" size=10
onChange="validchange(this.form)">
 </form>
</body>
</html>
```

onFocus Event Type

An onFocus event handler executes JavaScript code when input focus enters the field either by tabbing in or by clicking but not selecting input from the field. For windows, frames and framesets the event handler executes JavaScript code when the window gets focused. The following is an example of this event in JavaScript code:

```
<html>
<title>example of onFocus event handler</title>
<head>
</head>
<body>
 <h3>example of onFocus event handler</h3>
 click your mouse in the text box :<br/>
<form name="myform">
<input type="text" name="data" value="" size=10 onFocus='alert("you focused the
textbox!!")'> </form>
</body>
</html>
```

onLoad Event Type

An onLoad event occurs when a window or image finishes loading. For windows, this event handler is specified in the body attribute of the window. In an image, the event handler will execute handler text when the image is loaded.

```
<Html>
<title>Example of onLoad Event Handler</title>
<head>
<Script type="text/javascript">
function hello()
{
alert("Hello there...\nThis is an example of onLoad.");
```

```
}
</script>
</head>
<body onLoad="hello()"> <H3>Example of onLoad Event Handler</H3>
</body>
</html>
```

onSubmit Event Type

An onSubmit event handler calls JavaScript code when the form is submitted. For example after filling in a form and clicking the Submit button, the onSubmit event is called. Here is an example to demonstrate this:

```
<Html>
 <title> Example of onSubmit Event Handler </title>
<head> </head>
<body>
<Script type="text/javascript">
<h3>Example of onSubmit Event Handler </h3>
Type your name and press the button<br>
<form name="myform" onSubmit="alert('Thank you ' + myform.data.value +'!')">
<input type="text" name="data">
<input type="submit" name ="submit" value="Submit this form">
</form>
</Scipt>
</body>
</Html>
```

onUnLoad Event Type

An onUnload event handler calls JavaScript code when a document is exited. For example when a webpage is closed. The following is an example for this event handler:

```
<Html>
<head>
<Script type="text/JavaScript">
function goodbye()
{
alert("Thanks for Visiting!");
}
</script>

</head>

<body onUnLoad="goodbye()">

<h3>Example of onUnload event handler</h3>
```

Look what happens when you try to leave this page...

</body>

</Html>

onBlur Event Type

When a web component is a point of focus it is said to have focus and when it is no longer the component of focus, it is said to have lost focus. A blur event occurs when a select, text or text area on a form loses focus. The onBlur event handler executes JavaScript code when a blur event occurs. An example touching on a textfield on which users are to type their username. We may want to use JavaScript to ensure that the user never leaves the textfield component if the username has not been entered or the entered username is against the required format. When a user attempts to leave the textfield, the onBlur event handler calls the required () function to confirm that the username has a legal value

```
<Html>
<title>Example of onBlur Event Handler</title>
<head>
<Script type= “text/JavaScript">
function valid(form)
{
 var input=0;
 input=document.myform.data.value;
 if (input<0)
{
alert("Please input a value that is less than 0");
}
}
</Script>
</head>
<body>
<h3> Example of onBlur Event Handler</h3>
Try inputting a value less than zero:<br>
<form name="myform">
<input type="text" name="data" value="" size=10 onBlur="valid(this.form)">
</form>
</body>
</Html>
```

Other JavaScript Examples

Example 1: Image Rollover

```
<html>
<head>
<title>Rollover with a Mouse Events</title>
<script type="text/javascript">
if(document.images)
{
var image1 = new Image(); // Preload an image
image1.src="100_2440.jpg";
var image2 = new Image(); // Preload second image
image2.src = "100_2408.jpg";
}
</script>
</head>
<body>
<p>Move your mouse over the image to see the result</p>
<a href="#" onMouseOver="document.myImage.src=image2.src;"
onMouseOut="document.myImage.src=image1.src;">
<img name="myImage" src="100_2440.JPG" />
</a>
</body>
</html>
```

Example 2: Add Two Integers

```
<Html>
<head>
<title>Obtaining inputs via JavaScript</title>
</head>
<body>
Enter First Number : <br>
<input type="text" id="num1" name="num1"> <br>
Enter Second Number : <br>
<input type="text" id="num2" name="num2"> <br>
Result : <br>
<input type="text" id="result" name="result"><br>
<input type="button" name="clickbtn" value="Display Result" onclick="add_number()">
<script type="text/javascript">
function add_number()
{
```

```
var first_number = parseInt(document.getElementById("num1").value);
var second_number = parseInt(document.getElementById("num2").value);
var result = first_number + second_number;
document.getElementById("result").value = result;
}
</script>
</body>
</html>
```

Example 3: JavaScript Form Validation

```
<html>
<head>
<title>Form Validation</title>
<script type="text/javascript">
// Form validation code will come here
function validate()
{
if( document.myForm.Name.value == "" )
{
alert( "Please provide your name!" );
document.myForm.Name.focus() ;
return false;
}
if( document.myForm.EMail.value == "" )
{
alert( "Please provide your Email!" );
document.myForm.EMail.focus() ;
return false;
}
if( document.myForm.code.value == "" ||
isNaN( document.myForm.code.value ) ||
document.myForm.code.value.length != 3 )
{
alert( "Please provide a Country Code in the format ###." );
document.myForm.code.focus() ;
return false;
}
if( document.myForm.Country.value == "-1" )
{
alert( "Please provide your country!" );
return false;
}
return( true );
```

```
}
</script>
</head>
<body>
<form action=" " name="myForm"
onsubmit="return(validate());">
<table cellspacing="2" cellpadding="2" border="1" align="center">
<tr>
<td align="right">Name</td>
<td><input type="text" name="Name" /></td>
</tr>
<tr>
<td align="right">EMail</td>
<td><input type="text" name="EMail" /></td>
</tr>
<tr>
<td align="right">Country Code</td>
<td><input type="text" name="code" /></td>
</tr>
<tr>
<td align="right">Country</td>
<td>
<select name="Country">
<option value="-1" selected>[choose yours]</option>
<option value="1">KENYA</option>
<option value="2">UGANDA</option>
<option value="3">TANZANIA</option>
</select>
</td>
</tr>
<tr>
<td align="right"></td>
<td><input type="submit" value="Submit" /></td>
</tr>
</table>
</form>
</body>
</html>
```

95551

NB:

- For a robust grounding in web design and development and especially the three technologies considered in this class, the website https://www.w3schools.com/ is recommended.
- Now that you are able to use create websites in HTML, CSS and JavaScript; move on to Content Management Systems and Frameworks such as WordPress, Joomla, WiX etc.