Responsive Web Designing

seed®
beyond the obvious

Responsive Web Designing

seed
Official Curriculum

Index

Chapter 1 - Introduction to HTML

Introduction to HTML

HTML stands for **Hypertext Markup Language**. Let's take each of these words in sequence.

Hypertext is ordinary text that has been dressed up with extra features, such as formatting, images, multimedia, and links to other documents.

Markup is the process of taking ordinary text and adding extra symbols (for example, an editor's proofreading symbols are a type of markup). Each of the symbols used for markup in HTML is a command that tells a browser how to display the text. Markup can be very simple, or it can be very complicated. Either way, the underlying text being marked up is always present and viewable.

Language is actually a key point to remember about HTML. HTML is a computer language, related to computer programming languages (like BASIC, C and Pascal). HTML has its own syntax, slang, and rules for proper communication.

Markup Languages are special type of computer languages because they are solely concerned with classifying the parts of a document according to their function – in other words, indicating which part is the title of the document, which part is subheading, which part is the name of the author, and so on. It's not really correct to speak of "Programming HTML" because HTML isn't really a programming language. Instead, HTML is a markup language that has a different goal than that of creating a program.

HTML file is…

An HTML file is what a Web browser uses to generate a Web page. At its heart, the World Wide Web is nothing but a vast collection of HTML files residing on the hard drives of computers spread throughout the world, and a transport protocol for transferring these files from computer to computer. These HTML files, in turn, are simply text files that can be easily read and understood by human eyes.

Web browsers, such as Netscape Navigator or Microsoft Internet

Explorer, interpret HTML files in order to display Web pages. This is the main function of a Web browser. Whenever you use a browser to view a page on the World Wide Web, the browser has converted that Web page from an HTML file.

HTML and SGML

HTML is a subset of SGML (Standard Generalized Markup Language). SGML documents are more complex than HTML. SGML coding provides machine-level display format and function commands.

HTML resembles simplified SGML. The observation that SGML is to HTML as HTML is to plain text seems reasonable on the surface. SGML code constructs are not based as much in "plain English" as HTML. HTML's use of English language editing markup elements is a key reason for the popularity and success of the World Wide Web.

So, in other words, HTML is the language used to encode World Wide Web documents. It is a document-layout and hyperlink-specification language that defines the Syntax and Placement of special embedded directions that aren't displayed by a web browser, but tell it how to display the content of the document, including text, images, and other supported media. The language also tells you how to make a document interactive through special hypertext links, which connect your document with other documents on your local system, the World Wide Web, and other Internet resources such as FTP (File Transfer Protocol)

Types of Elements

Empty Tags

- Line Break and Horizontal lines

  `e.g.: <br> <hr> etc.`

Container Tags

- Begin Tag: Enclosed within angle brackets <B>
- End Tag: Enclosed within angle brackets with a forward slash </B>

  `e.g.: <h1> heading1 </h1>`

HTML elements perform a defined task (Make a text bold, insert a paragraph break, or format and number a list in a predetermined manner).

HTML uses two types of elements: Empty (or open) tags and Container tags. These tags differ because of what they represent. (Empty tags represent formatting constructs, such as line breaks and horizontal rules.

Container tags define a section of text (or of document itself) and specify the formatting the construction for all of the selected text. A Container tag has both a beginning and an ending tag. The ending tag is identical to the beginning tag, with the addition of a forward slash. Most containers can overlap and hold other containers or empty tags.

Text indicates the actual text that will be displayed on the HTML document. This is usually embedded between the Begin and End tags.

Every HTML Tag contains of

- A Tag name
- Sometimes followed by a tag attribute

  Tags attributes belong after the tag name each separated by a tab, space, or return characters. The order of attributes in single tag is not important. An attribute's value, is the one that follows an equal sign after the attribute name. If an attribute's value is a single word or number, you may simply add it after the equal sign. All other values should be enclosed in single or double quotation marks.

  Tag and attribute names are not case-sensitive.

  ```
  e.g.: <h1 align=”right”>heading1</h1>
  ```

Global Attributes

The global attributes below can be used on any HTML element

Attribute	Description
class	Specifies one or more classnames for an element (refers to a class in a style sheet)
draggable	Specifies whether an element is draggable or not
id	Specifies a unique id for an element

lang	Specifies the language of the element's content
style	Specifies an inline CSS style for an element
title	Specifies extra information about an element

Head Section Elements

Head Element

- Contains information about the document such as BASE element, TITLE element etc.,
- It defines the head section of a document
- Encompasses the head section within the <HEAD> Start tag and </HEAD> End tag
- It defines the function of the document
- HEAD information is not displayed as part of the document
- HEAD element must come right after the <HTML> tag

```
<html>
<head>
<base href= "http://www.xyz.com/public"> <title>
xyz's home page </title>
</head>
<body>
</body>
</html>
```

Head Section Elements:

```
<title>...</title>
```

Title for the document in the browser title bar and for bookmarks.

Body Element

Documents provide most of the content. It’s your job as an author to create HTML documents that are informative and interesting about the subjects of which you are knowledgeable. The largest part of these documents will be the body element, where you will put the text and images that make up the content. The text that you enclose in the body tag can be formatted and categorized in following different ways.

Formatting Text

- Logical Formatting Elements

  e.g.: `<cite>,<code>, <em>, <kbd>, <samp>, <strong>, <var>`

- Physical Formatting Elements

  e.g.: `<b>, <i>, <tt>`

Logical Formatting Elements

One of the ideas behind HTML is that documents should be laid out in a logical and structured manner. This gives the users of the documents as much flexibility as possible. With this in mind, the designers of HTML created a number of formatting elements that are labeled according to the purpose they serve rather than by their appearance. The advantage of this approach is that documents are not limited to a certain platform. Although they may look different on various platforms, the content and context will remain the same.

Physical Formatting Elements

Having said that HTML is intended to leave the appearance of the document up to the browser, you will now see how you can have limited control over what the reader sees. In addition to the logical formatting elements, it is possible to use physical formatting elements that will change the appearance of the text in the browser.

Body Section Elements can be formatted in more ways such as

Text Level Formatting

This means you are making up at least a single character, but often much more than that.

e.g.:

```
<b>, <big>, <i>, <u>,<s> or <strike>, <small>,
<sub>, <sup> <tt>, <abbr>,<address>, <code>,
<em>, <kbd>,
<samp>, <del>, <strong>, <var>…etc.
```

See the appendices for further details of the above tags.

The Paragraph or Block Level

This means you are formatting a specific logical chunk of the document. Block level formatting is usually applied to larger content than text level formatting tags. As such, the block level tags define the major sections of the document such as paragraph.

e.g.

```
<blockquote>, <basefont>, <center> <br>, <div>,
<hr>,  <h1> …..<h6>, <p>, <pre>, <span>…..etc.
```

See the appendices for further details of the above tags.

Character Entities

Web browsers do not read certain characters if they appear in an HTML document. To get around this limitation, codes have been assigned to certain characters.

These include accented letters, copyright symbols, or even the angle brackets used to enclose HTML elements. To use such characters in an HTML document they must be "escaped" using a special code. Some of the more commonly used characters are shown in Table:

“	&quote;	Quotation mark
&	&amp;	Ampersand
<	&lt;	less than
>	&gt;	Greater than
©	&copy;	Copyright symbol
®	&reg;	Registered trademark
	&nbsp;	Non breaking space

Linking Between Web Sites

To link your Web page to any other page on the World Wide Web, you use something called a *hypertext anchor.* To create a hypertext anchor, you use the <A> HTML tag.

```
<a>...</a>
```

Creates a hypertext anchor. Used for creating hypertext links and targets for hypertext links. The following table describes the attributes for this tag.

Attribute	Description
`href = url`	The location where you are taken if you click the link.
`name = string`	The name used for named anchors.

Here's an example of how you would link a Web page to the HotWired Web site:

```
<html>
<head>
<title> inter-site anchor </title>
</head>
<body>
<a href="http://www.google.com"> Google </a>
</body>
</html>
```

Notice that the `<a>` tag is a container tag. The text that it contains becomes a hypertext link. In this example, the text Google is displayed in a browser with an underline. This tells the viewer of the page that the text is a hypertext link. If you click the word Google, the home page of the Goggle Web site is loaded into your browser.

You supply the Web address as the value of the `href` attribute of the `<a>` tag. This address can be the URL of any page on any Web site in the world.

Need of List

To display list of information on the web page. HTML defines five kinds of lists:

Types of List are as follows:

- Unordered List
- Ordered List

Unordered List

Creates an unordered list with bulleted list items. The following table describes the attributes for this tag:

```
<ul>
   <li> Apple </li>
   <li> Banana </li>
   <li> Pine-apple </li>
</ul>
```

Ordered List

Creates an ordered list with numbered list items. The following table describes the attributes for this tag.

```
<ol>
   <li> Apple </li>
   <li> Banana </li>
   <li> Pine-apple </li>
</ol>
```

Images

```
<img>
```

Specifies an image to appear in the document. The following table describes the attributes for this tag.

Attribute	Description
`alt=string`	Alternative text to display when the browser has graphics turned off. On certain browsers, a balloon with this text appears over the image.
`height=number`	Height of the image in pixels.
`src=url`	The Internet address of the image.
`width=number`	Width of the image in pixels.

Formatting with Tables

Tables have both an obvious and a not-so-obvious use. First, the obvious one, tables can be used to create a table of information. Whenever you need to present information in rows and columns, use a table. For example, you can use a table to display a list of cities and their area codes.

In general, tables can be used to present in readable form what would otherwise be overwhelming doses of information. A less obvious but more powerful use of tables is for page formatting.

Creating a Simple Table of Information

Tables can be extremely complex. Many table tags are available, and each tag has many attributes. However, many of these attributes are either Microsoft- or Netscape-specific. However, a table also can be very simple. Here's an example of a very simple table.

To create larger tables, simply add more rows and columns with the `<tr>` and `</tr>` tags.

For example, suppose you want to display a table containing a list of cities and their area codes. You could do it like this:

```
<html>
<head><title> area codes </title></head>
<body>
<table border=1>
<caption>City Codes</caption>
<tr>
<td> Pune    </td>
<td> 020    </td>
</tr>
<tr>
<td> Mumbai </td>
<td> 022    </td>
</tr>
</table>
</body>
</html>
```

In this example, the `<tr>` tag is used to create three table rows, one row for each city. Each table row contains two `<td>` tags, which create the columns containing the city name and the city area code and `<caption>` tag is used to set caption or header for table.

Spanning Columns and Rows

`colspan` attribute is used to merge columns from the same row whereas `rowspan` attribute is used to merge columns from the different rows

```
<table border='1' width='600'>
   <caption> Student Information </caption>
   <tr>
     <th rowspan='2'> Student Name </th>
     <th colspan='3'> Languages Known </th>
   </tr>
   <tr>
     <th> HTML5 </th>
     <th> CSS3 </th>
     <th> jQuery </th>
   </tr>
   <tr>
   <th> Kedar Joshi </th>
   <th> Yes </th>
   <th> Yes </th>
   <th> Yes </th>
   </tr>
   <tr>
   <th> Mandar Kulkarni </th>
   <th> Yes </th>
   <th> No </th>
   <th> Yes </th>
   </tr>
</table>
```

Working with HTML Forms

To create a genuinely interactive Web site, you need to use HTML forms. HTML forms enable you to gather and respond to the information provided by the visitors to your Web site. Using forms, you can create check boxes, radio buttons, and text areas.

To create HTML forms, you use the <FORM> tag. This tag works as a container tag, enclosing other form elements and specifying, through its action, what should be done with the information gathered within the form.

```
<form>...</form>
```

Encloses a fill-out form. The following table describes the attributes for this tag.

Attribute	**Description**
action=url	The address of the server-side program that will process the form, or the e-mail address that specifies where to send the form contents.
enctype=form encoding	The content encoding to use for the form contents. For a file upload button to function properly, you should use multipart/form-data as the value of this attribute.
method=post | get	The method of sending the forms content.
target=name	Specifies to return the results of the form submission to the targeted window or frame.

Text Boxes

The most basic form element is a text box. You can create a text box with nothing more than the tag `<input>` with no attributes. However, to create a useful text box, you need to include the `name` attribute.

```
<input>
```

Creates a fill-in form input element. The following table describes the attributes for this tag.

Attribute	Description
`checked = checked`	Determines whether a check box is checked when the form first appears.
`maxlength = number`	Specifies the maximum number of characters a TEXT or PASSWORD input element will accept.
`name = name`	Specifies the name for an input element.
`size = number`	Specifies the displayed width of the input element.
`src = image`	Specifies the image URI for the image submit button.
`value = initial value`	Specifies the initial value of a form element.
`type = text | password | checkbox | radio | submit | reset| pile | hidden | image`	Specifies an input widget for a fill-in form element.

The list following describes the types. Types for the TYPE attribute of the <INPUT> command:

`text`	Text box (default).
`password`	Password box in which input is hidden.
`radio`	Multiple related check boxes.
`submit`	Submit form button.
`reset`	Button that clears the form.
`file`	File upload button.
`hidden`	Non-displayed form field.
`image`	Submit form image.

e.g.:

```
<input type=”text” name=”text1”>
```

Form Buttons

Three types of buttons are typically used in HTML forms. You've already been introduced to the first type of button - the submit button. When you click the submit button**,** the form is processed by whatever php file is referred to in the `action` attribute. The syntax of the submit button is as follows:

```
<input type=submit value="do it!">
```

The `value` attribute determines the text that appears on the button.

The image button has almost exactly the same effect as the submit button. This type of button, however, appears as an image instead of the typical gray rectangle. Following is an example of how to use an image button:

```
<input type=”image” src=”myimg.gif”>
```

An interesting aspect of the image button is that when clicked it not only submits the form data, it also submits the exact coordinates of the image which was clicked. For example, if you click dead center on an image button that has a width and height of five pixels, the x,y coordinates of 3,3 are passed with the rest of the form information.

The final type of button is the reset button.

```
<input type=”reset” value=”clear all”>
```

When a user of a form clicks the reset button, all the form fields are returned to their initial state. For example, text boxes with no default value will be cleared; those with default `value` attributes set will be reinitialized to the specified value.

As in the case of the submit button, you specify the text that appears on a reset button by supplying a `value` attribute, similarly you do it for the text box as in the next example:

```
<form action="/somedirectory/mypage.php"
method="post">
 <input name="textbox1"><br>
 <input name=”textbox2” value=”mydefault”><br>
 <input type=submit value=”submit me”><br>
 <input type=reset value=”clear me!”>
</form>
```

Password Boxes

Suppose you want the visitors to your Web site to register before they can use it. Using standard text boxes, you could create an HTML form that demands the user's name and password. However, you wouldn't want people to enter their passwords in view of others who might be looking over their shoulders. To protect the user's password, you need to use a password box. A password box works like a text box except that when information is entered, it's hidden. Here's an example:

```
password: <input type="password" maxlength=”20”>
```

It’s important to realize that text entered in a password box, though hidden, is not encrypted when it’s submitted. In theory, this means that someone could steal the text entered into a password box off the wires as it moves across the Internet to your Web Site.

When the preceding HTML file is displayed in a Web browser, you can enter text into the password box in the same way as you would with a normal text box. However, all of the text entered will be masked (typically with asterisks). You can use both the `size` and `maxlength` attributes with password boxes to control the screen width of the box and the maximum number of characters that can be entered into it.

Check Boxes

You also can use multiple check boxes with the same name to gather information. For example, suppose you want to find out how the visitors to your Web site discovered the site, and you want to allow for the possibility that some users discovered your site in more than one way. You could use the following HTML code:

```
How did you find out about this web site? <br>
magazine: <input name="discover" type=”checkbox”
value="magazine"> <br>
```

```
search engine : <input name="discover"
type=”checkbox” value="search"> <br> friend:
<input name="discover" type=”checkbox”
value="friend"> <br>
```

By default, check boxes initially appear without a check mark. You can override this default by including the *checked* attribute.

```
<input name="mycheckbox" type=”checkbox”
value="yes" checked=”checked”>
```

Radio Buttons

Using radio buttons, you can present a choice of multiple values. Unlike when using check boxes, however, the user can select only one radio button at a time.

For example, suppose you want to determine the gender of visitors to your Web site. You could use radio buttons to allow visitors to specify whether they are male or female, since no visitors can be both. Here’s an example of how you could do this:

```
Please indicate your sex: <br>
male: <input name="sex" type=”radio”
value="male"> <br>
female: <input name="sex" type=”radio”
value="female"> <br>
```

Notice that both radio buttons are given the same name. When the user of this Web page hits the submit button, the value of only one radio button will be sent. The value sent is specified in the `value` attribute of the radio button.

You can use the `checked` attribute to specify the radio button that will be selected when the Web page is first loaded. You can also have as many radio buttons as you like in a form. If you want to have multiple groups of radio buttons, simply give the radio buttons in each group different names.

Creating Drop-Down List Boxes

As an alternative to check boxes and radio buttons, you can create drop-down list boxes to display a menu of alternatives. A drop-down list box shows only one selection at a time. To view the other possible selections, you click the arrow attached to the box. One advantage of using a drop-down list box is that it occupies less space on your Web page than radio buttons or check boxes.

You create a drop-down list box using the `<select>` and `<option>` tags, like this

```
please select your city: <br>
<select name="city">
<option> Select <option>
<option> Pune <option>
<option> Mumbai <option>
</select>
```

Creating Text Areas

With the form elements described so far, the user can't enter more than one line of text. The `<textarea>` form tag creates a text area, where you can give the user more room for freedom of expression. Use this tag whenever you want someone to enter a paragraph of text.

```
<textarea>...</textarea>
```

Creates an input box that can accept multiple lines of text. Place default text between the start and end tags. A text area must appear within a form. The following table describes the attributes for this tag.

Attribute	Description
`cols=number`	The width of the text area in characters.
`rows=number`	The vertical height of the text area in characters.

Creating Hidden Fields

Using hidden fields, you can include information in your forms that will never be displayed onscreen. This information will be included, however, when the form is submitted.

Here's an example of a hidden field in a form:

```
<input name="secret" type=”hidden” value="you
cannot see me!">
```

When this HTML file is displayed in a Web browser, the only thing you see on the screen is the submit button. The hidden field, named secret, doesn't appear. However, when the form button is clicked, the value “You cannot see me!” is submitted as part of the contents of the form.

Creating File Upload Buttons

Suppose you want to create a Web site where people can advertise houses for sale. It would be nice if people could upload pictures of their homes. Or suppose you want to create a Web site devoted to short stories. It would nice if people could upload their stories. For example, in Microsoft Word format.

In theory, you can do this with the `type=file` attribute of the `<input>` tag. Using this attribute, you can create a file upload button on a form. When users click this button, they can select a file to upload from their local hard drive. Following is an example of how this can be done:

```
<form enctype=”multipart/form-data"
action="/somedirectory/mypage.php"
method=”post”>
please choose a picture to upload: <br>
<input name="picture" type=”file”> <br>
<input type=submit value="submit me!">
</form>
```

Example – Creating student registration form

```
<form action="/somedirectory/mypage.php"
method=”post”>
<table>
  <caption><u> Student Reistration Form
</u></caption>
  <tr>
    <td> First name: </td>
    <td> <input type='text' name='firstname'
size=’30’> </td>
  </tr>
  <tr>
    <td> Last name: </td>
    <td> <input type='text' name='lastname'
size='30'> </td>
```

```
  </tr>
  <tr>
   <td> Password: </td>
   <td> <input type='password' name='password'
size='30'> </td>
  </tr>
  <tr>
   <td> Select Gender: </td>
   <td>
   <input type='radio' name='sex' value='male'
checked='checked'> Male
   <input type='radio' name='sex'
value='female'> Female
     </td>
  </tr>
  <tr>
   <td> Select Hobbies: </td>
   <td>
   <input type='checkbox' name='hobbies'
value='Dancing'> Dancing       <input
type='checkbox' name='hobbies' value='Singing'>
Singing<br>
   <input type='checkbox' name='hobbies'
value='Trekking'> Trekking
   <input type='checkbox' name='hobbies'
value='Painting'> Painting
   </td>
  </tr>
  <tr>
   <td> Select City: </td>
   <td>
       <select name='city'>
           <option>Select</option>
           <option>Pune</option>
           <option>Mumbai</option>
           <option>Delhi</option>
       </select>
   </td>
  </tr>
  <tr>
   <td> Your comments </td>
   <td>
        <textarea rows='3' cols='25'>Write your
comments here</textarea>
   </td>
  </tr>
  <tr>
   <td><input type='submit' value='Submit'></td>
   <td><input type='reset' value='Cancel'></td>
  </tr></table>
</form>
```

Example – Displaying address using <fieldset>, <legend>, <address>

```
<fieldset>
  <legend> Address </legend>
  <address>
   SEED Infotech, <br>
     Income-tax lane, <br>
     Near null stop, <br>
     Kothrud, Pune - 411038
  </address>
</fieldset>
```

What is HTML5?

HTML5 will be the new standard for HTML. The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then. HTML5 is still a work in progress. However, the major browsers support many of the new HTML5 elements and APIs.

How Did HTML5 Get Started?

HTML5 is cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG). WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

- New features should be based on HTML, CSS, DOM, and JavaScript
- Reduce the need for external plug-ins (like Flash)
- Better error handling
- More markup to replace scripting
- HTML5 should be device independent
- The development process should be visible to the public

HTML5 Document

```
<!DOCTYPE html>
<html>
<head>
<title>Title of the document</title>
</head>

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

HTML5 - New Features

Some of the most interesting new features in HTML5:

- The <video> and <audio> elements for media playback
- Support for local storage
- New content-specific elements, like <article>, <footer>, <header>, <nav>, <section>
- New form controls, like calendar, date, time, email, url, search
  - **Browser Support for HTML5**
  - HTML5 is not yet an official standard, and no browsers have full HTML5 support. But all major browsers (Safari, Chrome, Firefox, Opera, and Internet Explorer) continue to add new HTML5 features to their latest versions.

  - **New Elements in HTML5**

- The internet, and the use of the internet, has changed a lot since HTML 4.01 became a standard in 1999. Today, several elements in HTML 4.01 are obsolete, never used, or not used the way they were intended. All those elements are removed or re-written in HTML5.
- To better handle today's internet use, HTML5 also includes new elements for drawing graphics, adding media content, better page structure, better form handling, and several APIs to drag/drop elements, find Geolocation, include web storage, application cache, web workers, etc.

New Media Elements

Tag	Description
<audio>	Defines sound content
<video>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed>	Defines a container for an external application or interactive content (a plug-in)
<track>	Defines text tracks for <video> and <audio>

New Form Elements

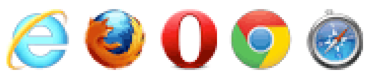
Tag	Description
<datalist>	Specifies a list of pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

Video on the Web

Until now, there has not been a standard for showing a video/movie on a web page. Today, most videos are shown through a plug-in (like flash). However, different browsers may have different plug-ins.

HTML5 defines a new element which specifies a standard way to embed a video/movie on a web page: the <video> element.

Browser Support



Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <video> element.

Note: Internet Explorer 8 and earlier versions do not support the <video> element.

Example

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
</video>
```

The control attribute adds video controls, like play, pause, and volume. It is also a good idea to always include width and height attributes. If height and width are set, the space required for the video is reserved when the page is loaded. However, without these attributes, the browser does not know the size of the video, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the video loads).

You should also insert text content between the <video> and </video> tags for browsers that do not support the <video> element. The <video> element allows multiple <source> elements. <source> elements can link to different video files. The browser will use the first recognized format.

Audio on the Web

Until now, there has not been a standard for playing audio files on a web page. Today, most audio files are played through a plug-in (like flash). However, different browsers may have different plug-ins.

HTML5 defines a new element which specifies a standard way to embed an audio file on a web page: the <audio> element.

Browser Support

Internet Explorer 9+, Firefox, Opera, Chrome, and Safari support the <audio> element.

Note: Internet Explorer 8 and earlier versions, do not support the <audio> element.

HTML5 New Input Types

HTML5 has several new input types for forms. These new features allow better input control and validation. This chapter covers the new input types:

- `color`
- `date`
- `datetime`
- `datetime-local`
- `email`
- `month`
- `number`
- `range`
- `search`
- `tel`
- `time`
- `url`
- `week`

Note: Not all major browsers support all the new input types. However, you can already start using them; if they are not supported, they will behave as regular text fields.

Examples:

Input Type: color

The color type is used for input fields that should contain a color.

Example

```
Select your favorite color: <input type="color"
name="favcolor">
```

Input Type: date

The date type allows the user to select a date.

Example

```
Birthday: <input type="date" name="bday">
Birthday: <input type="date" name="bday">
```

Web storage in HTML 5

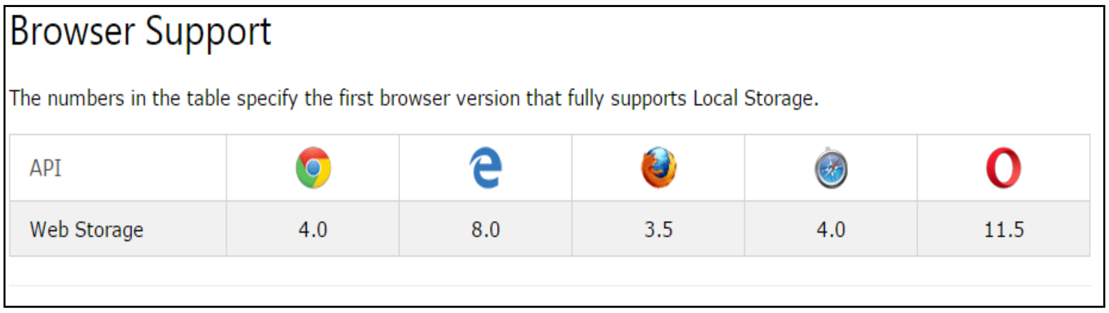
To store data at client side usually cookies are used.

- Cookies are included with every HTTP request.
- Affect the performance of the web application.
- The data is typically sent to server in an unencrypted form.
- Cookies have storage limitation of 4 KB of data .
- It is incapable of storing large data beyond its storage capacity.
- But it is not a suitable option for stroing large sized data on the client side.

Solution :

- Store the data required by web applications locally within the user's browser.
- HTML 5, offers storage features such as
  - localStorage
  - sessionStorage
- Storage features
  - These storage options are more secure
  - They can store large amounts of data locally
  - They size of the data does not affect website performance(load time).

- Data is persistent & always available for access whenever required until deleted explicitly.
- Browser support

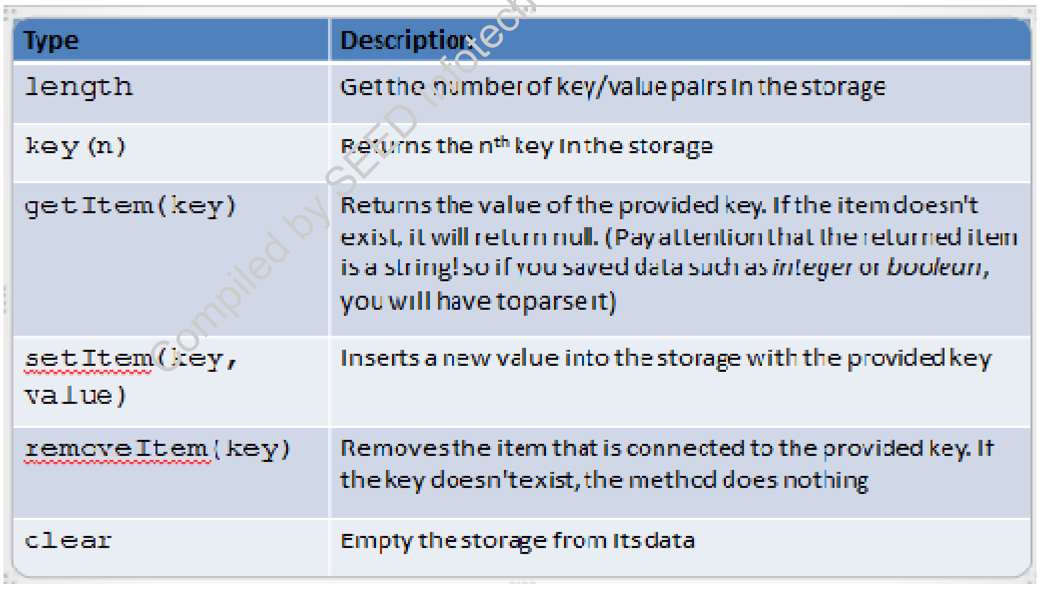


Browser Support

The numbers in the table specify the first browser version that fully supports Local Storage.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

Methods and Properties used in Web Storage



Type	Description
`length`	Get the number of key/value pairs in the storage
`key(n)`	Returns the nth key in the storage
`getItem(key)`	Returns the value of the provided key. If the item doesn't exist, it will return null. (Pay attention that the returned item is a string! so if you saved data such as *integer* or *boolean*, you will have to parse it)
`setItem(key, value)`	Inserts a new value into the storage with the provided key
`removeItem(key)`	Removes the item that is connected to the provided key. If the key doesn't exist, the method does nothing
`clear`	Empty the storage from its data

```
LocalStorage Demo
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (localStorage.clickcount) {
            localStorage.clickcount =
Number(localStorage.clickcount)+1;
        } else {
            localStorage.clickcount = 1;
        }
document.getElementById("result").innerHTML =
"You have clicked the button " +
localStorage.clickcount + " time(s).";
    } else {

document.getElementById("result").innerHTML =
"Sorry, your browser does not support web
storage...";
    }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()"
type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter
increase.</p>
<p>Close the browser tab (or window), and try
again, and the counter will continue to count
(is not reset).</p>
</body>
</html>
```

Web Storage – sessionStorage

- It stores data for one session and deletes it after closing the browser window
- Similar to localStorage but it stores the data only for one session
    - **i.e. the data remains until the user closes that window or tab.**

Session Storage demo

```
<!DOCTYPE html>
<html>
<head>
<script>
```

```
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (sessionStorage.clickcount) {
            sessionStorage.clickcount =
Number(sessionStorage.clickcount)+1;
        } else {
            sessionStorage.clickcount = 1;
        }

document.getElementById("result").innerHTML =
"You have clicked the button " +
sessionStorage.clickcount + " time(s) in this
session.";
    } else {

document.getElementById("result").innerHTML =
"Sorry, your browser does not support web
storage...";
    }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()"
type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter
increase.</p>
<p>Close the browser tab (or window), and try
again, and the counter is reset.</p>
</body>
</html>
```

Chapter 2- Introduction to CSS

Introduction to CSS

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- External Style Sheets can save a lot of work
- External Style Sheets are stored in CSS files

HTML Styles (CSS)

How to Use Styles

When a browser reads a style sheet, it will format the document according to it. There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet (Embedded)
- Inline styles

External Style Sheet

An external style sheet is ideal when the style is applied 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:

```
<head>
<link rel="stylesheet" type="text/css"
href="mystyle.css">
</head>
```

Internal Style Sheet (Embedded)

An internal style sheet can be used if one single document has a unique style. Internal styles are defined in the <head> section of an HTML page, by using the <style> tag, like this:

```
<head>
<style type="text/css">
body {background-color:yellow;}
p {color:blue;}
</style>
</head>
```

Inline Styles

An inline style can be used if a unique style is to be applied to one single occurrence of an element. To use inline styles, use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example below shows how to change the text color and the left margin of a paragraph:

```
<p style="colo
r:blue;margin-left:20px;">This is a
paragraph.</p>
```

HTML Style Tags

Tag	Description
`<style>`	Defines style information for a document
`<link>`	Defines the relationship between a document and an external resource

The id and class Selectors

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

The id Selector

The id selector is used to specify a style for a single, unique element. The id selector uses the id attribute of the HTML element, and is defined with a "#". The style rule below will be applied to the element with id="para1":

Example

```
<html>
<head>
<style type="text/css">
#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>
</html>
```

The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements. This allows you to set a particular style for many HTML elements with the same class. The class selector uses the HTML class attribute, and is defined with a "."

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

Example

```
<html>
<head>
<style type="text/css">
.center
{
text-align:center;
}
```

```
</style>
</head>

<body>
<h1 class="center">Center-aligned heading</h1>
<p class="center">Center-aligned paragraph.</p>
</body>
</html>
```

CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

- background-color:red;
- background-image:url('image.gif');
- background-repeat:no-repeat;
- background-repeat:repeat-x;
- background-repeat:repeat-y;
- background-position:top left

CSS Text

Text Color

```
color:blue;
color:#00ff00;
color:rgb(255,0,0);
```

Text Alignment

```
text-align:center;
text-align:right;
text-align:justify;
```

Text Decoration

```
text-decoration:overline;
text-decoration:line-through;
text-decoration:underline;
text-decoration:blink;
```

```
text-decoration:none;
```

Text Transformation

```
text-transform:uppercase;
text-transform:lowercase;
text-transform:capitalize;
```

Text Indentation

```
text-indent:50px;
```

CSS Font

Font Family

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

Font Style

```
font-style:normal;
font-style:italic;
font-style:oblique;
```

Font Size

```
font-size:40px;
font-size:30%;
font-size:14em;
font-size:14
```

CSS Links

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). Special for links are that they can be styled differently depending on what state they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

Example

```
a:link {color:#FF0000;}      /* unvisited link
*/
a:visited {color:#00FF00;}  /* visited link */
a:hover {color:#FF00FF;}  /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
```

CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

List

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

List properties for unordered list:

```
ul{list-style-type: circle;}
ul{list-style-type: square;}
ul{list-style-image: url('image.gif');}
ul{list-style-type: none;}
```

List properties for ordered list:

```
ol{list-style-type:upper-roman;}
ol{list-style-type: lower-alpha;}
ol{list-style-type: lower-roman;}
ol{list-style-type: upper-alpha;}
```

CSS Outlines

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out". However, it is different from the border property. The outline is not a part of the element's dimensions, therefore the element's width and height properties do not contain the width of the outline.

CSS Display and Visibility

Hiding an Element - display:none or visibility:hidden

Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:

1) visibility:hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

   Example

```
h1 {visibility:hidden;}
```

2) display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as the element is not there:

   Example

```
h1 {display:none;}
```

CSS Display - Block and Inline Elements

The following example displays list items as inline elements:

Example

```
li {display:inline;}
```

The following example displays span elements as block elements:

Example

```
span {display:block;}
```

CSS Float

What is CSS Float?

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it. Float is very often used for images, but it is also useful when working with layouts. If an image is floated to the right, a following text flows around it, to the left:

Example

```
img
{
float:right;}
```

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.

Example 1 - Creating a Transparent Image

The CSS3 property for transparency is **opacity**. First we will show you how to create a transparent image with CSS.

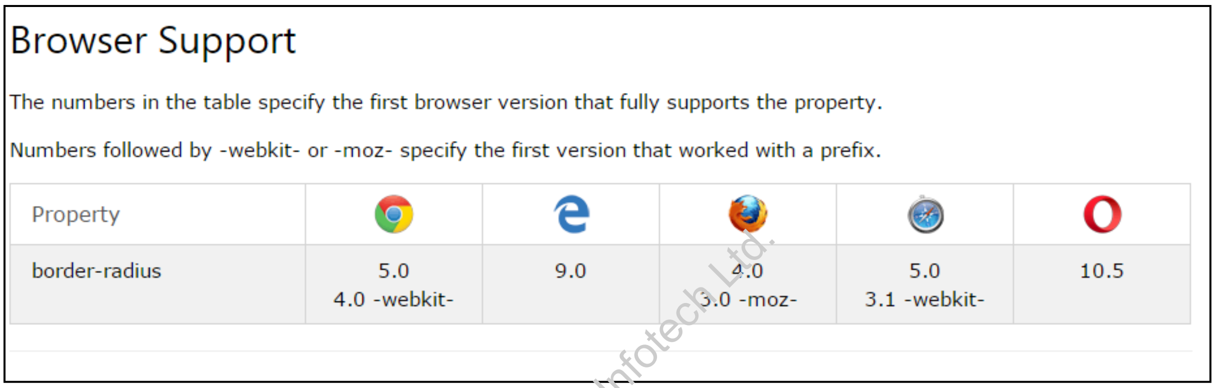
Regular image: The same image with transparency:

The CSS looks like this:

```
img
{opacity:0.4;
}
img:hover
{ opacity:1.0;}
```

Introduction to CSS3 Features

- It is used to control the style and layout of web pages.
- It is the latest version for CSS.
- It is completely backwards compatible.



Browser Support

The numbers in the table specify the first browser version that fully supports the property.

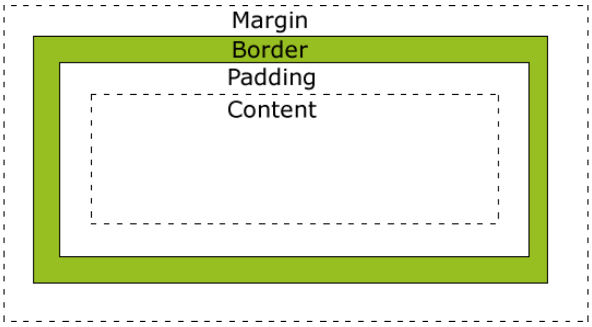
Numbers followed by -webkit- or -moz- specify the first version that worked with a prefix.

Property					
border-radius	5.0 4.0 -webkit-	9.0	4.0 3.0 -moz-	5.0 3.1 -webkit-	10.5

CSS3 Modules

- It is split up into *modules*
- Some of the most important CSS3 modules are:
  - Box Model
  - Borders
  - Text Effects

CSS Box Model



All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout. The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content. The box model allows us to place a border around elements and space elements in relation to other elements.

The image illustrates 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 affected by the background color 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

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

CSS Border

Border Style

The border-style property specifies what kind of border to display.

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Example

```
p
{
border-style:solid;
border-width:5px;
border-color:red;
}
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

```
p
{
border-top-style:dotted;
border-right-style:solid;
border-bottom-style:dotted;
border-left-style:solid;
}
```

The example above can also be set with a single property:

Example

```
border-style:dotted solid;
```

Border - Shorthand property

The shorthand property for the border properties is "border":

Example

```
border:5px solid red;
```

CSS Margin

Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.

The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser sets the margin. The result of this is dependant of the browser
length	Defines a fixed margin (in pixels, pt, em, etc.)
%	Defines a margin in % of the containing element

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides:

Example

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property. The shorthand property for all the margin properties is "margin":

Example

```
margin:100px 50px;
```

CSS Padding

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element. The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property. The shorthand property for all the padding properties is "padding":

Example

```
padding:25px 50px;
```

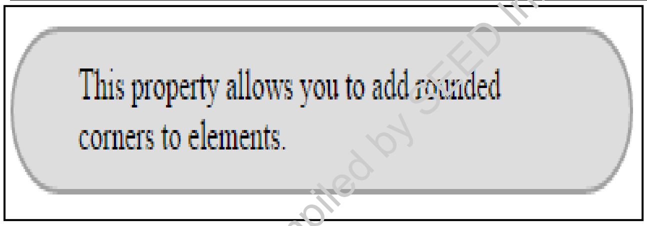
Working with border and shadow

By using below new properties

- border-radius
  - Create rounded borders for HTML element
- box-shadow
  - Adding shadow to boxes
- border-image
  - Use an image as a border

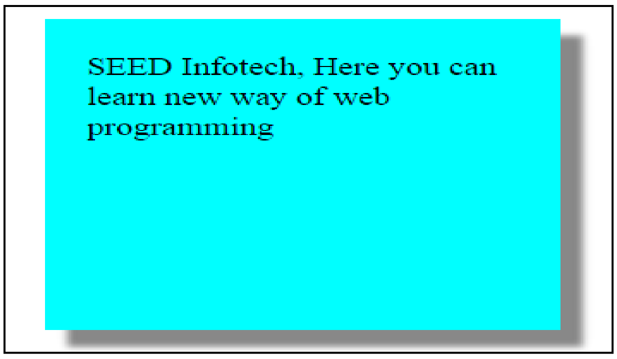
Example border-radius

```
<style type="text/css">
div
{
   border:2px solid #a1a1a1;
   padding:10px 40px;
   background:#dddddd;
   width:300px;
   border-radius:50px;
}
</style>
<div>This property allows you to add rounded
corners to elements.</div>
```


This property allows you to add rounded corners to elements.

Example box-shadow

```
<style type="text/css">
div
{   width:200px;
    height:200px;
    background-color:aqua;
    box-shadow: 10px 10px 5px #888888;
}
</style>
<div>
SEED Infotech, Here you can learn new way of web
programming
</div>
```


SEED Infotech, Here you can learn new way of web programming

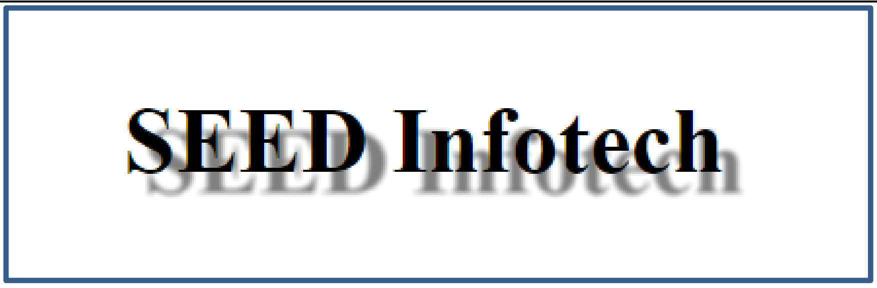
New text effects using CSS3

By using below new properties

- text-shadow
  Applying shadows to text
- word-wrap
  If a word is too long to fit within an area, it expands word-wrap property allows you to force the text to wrap-even if it means splitting it in the middle of a word

Example text-shadow

```
<style type="text/css">
h1
{
   color:black;
   text-shadow: 7px 7px 2px grey;
}
</style>
</head>
<body>
<h1>SEED Infotech</h1>
```

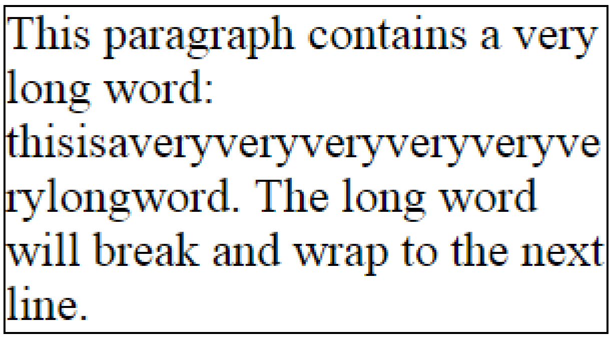


Example word-wrap

```
<style type="text/css">
p.test
{
width:200px;
border:1px solid #000000;
word-wrap:break-word;
}
</style>
</head>
<body>

<p class="test"> This paragraph contains a very
long word:
thisisaveryveryveryveryveryverylongword. The
long word will break and wrap to the next
line.</p>

</body>
```


This paragraph contains a very long word: thisisaveryveryveryveryveryve rylongword. The long word will break and wrap to the next line.

Multiple columns on a web page

- You can create multiple columns for laying out text like in newspapers
- By using below new properties
  - column-count
    - Specifies the number of columns an element should be divided into
  - column-gap

- Specifies the gap between the columns
- column-rule
    - Sets the width, style, and color of the rule between columns
    - Possible values are
    - dashed, dotted, inset, outset, hidden, solid

Example

```
<html>
    <head>
<style type="text/css">
.newspaper
{
  column-count:4;
  column-gap:40px;
  column-rule:5px dashed red;
}
</style>
        </head>
    <body>
<div class="newspaper">
Cricket is a bat-and-ball game played between
two teams of 11 players on a roughly
circular field, at the centre of which is a
rectangular 22-yard long pitch. Each team takes
it in turn to bat, in which they attempt to
accumulate as many runs as possible, while the
other team fields, attempting to prevent the
batting team scoring runs. Teams may bat once or
twice each depending upon the format of the
game. Each turn is known as an innings. </div>
        </body>
</html>
```

Chapter 3 - Introduction to Scripting

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the same?

NO!

Java and JavaScript are two completely different languages in both concept and design! Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What Can JavaScript do?

- JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

JavaScript = ECMAScript

- JavaScript is an implementation of the ECMAScript language standard. ECMA-262 is the official JavaScript standard.
- JavaScript was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all browsers since 1996.
- The official standardization was adopted by the ECMA organization (an industry standardization association) in 1997.
- The ECMA standard (called ECMAScript-262) was approved as an international ISO (ISO/IEC 16262) standard in 1998.

The development is still in progress.

How to use JavaScript

Example 1:

```
<html>
<body>
<script type="text/javascript">
document.write("hello world");
</script>
</body>
</html>
```

Example 2: We cannot use any HTML tag within the script excluding document.write

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>hello world</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another
paragraph.</p>");
</script>
</body>
</html>
```

JavaScript Comments

Comments can be added to explain the JavaScript, or to make the code more readable. Single line comments start with //. Multi line comments start with /* and end with */.

Declaration of variables in JavaScript

JavaScript Variables

As with algebra, JavaScript variables are used to hold values or expressions. A variable can have a short name, like x, or a more descriptive name, like carname.

Rules for JavaScript variable names:

- Variable names are case sensitive (y and Y are two different variables)
- Variable names must begin with a letter or the underscore character

Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables. You declare JavaScript variables with the `var` keyword:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet). However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable `x` will hold the value 5, and `carname` will hold the value Volvo.

JavaScript Arithmetic

As with algebra, you can do arithmetic operations with JavaScript variables:

```
y=x-5;
z=y+5;
```

Example

```
<html>
<body>
<h1>My First Web Page</h1>
<script type="text/javascript">
```

```
var x=5;
var y=6;
var z=parseInt(x)+parseInt(y);
document.write(z+“<br>“) ;
document.write(“addition of x and y I =” + z) ;
</script>
</body>
</html>
```

`parseInt()` is inbuilt function which is used to convert string to interger value and `parseFloat()` is inbuilt function which is used to convert string to float value.

JavaScript Operators

= is used to assign values.

+ is used to add values.

```
y=5;
z=2;
x=y+z;
```

The value of `x`, after the execution of the statements above, is 7.

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values. Given that `y=5`, the table below explains the arithmetic operators:

Operator	Description	Example	Result	
+	Addition	x=y+2	x=7	y=5
-	Subtraction	x=y-2	x=3	y=5
*	Multiplication	x=y*2	x=10	y=5
/	Division	x=y/2	x=2.5	y=5
%	Modulus (division remainder)	x=y%2	x=1	y=5

++	Increment	x=++y	x=6	y=6
--	Decrement	x=--y	x=4	y=4
		x=y--	x=5	y=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables. Given that x=10 and y=5, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together. To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day". To add a space between the two strings, insert a space into one of the strings:

```
txt1="What a very ";
txt2="nice day";
```

```
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains: "What a very nice day"

Adding Strings and Numbers

The rule is: If you add a number and a string, the result will be a string!

Example

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

Out put

```
10
55
55
55
```

The rule is: If you add a number and a string, the result will be a string.

JavaScript Comparison and Logical Operators

Comparison and Logical operators are used to test for true or false.

Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Given that x=5, the table below explains the comparison operators:

Operator	Description	Example
==	is equal to	x==8 is false x==5 is true
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How can it be used?

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```

Logical Operators

Logical operators are used to determine the logic between variables or values. Given that x=6 and y=3, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President
":"Dear ";
```

If the variable visitor has the value of "PRES", then the variable greeting will be assigned the value "Dear President" else it will be assigned "Dear".

Chapter 4 - Language Basics

JavaScript If...Else Statements

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:

- `if` statement - use this statement to execute some code only if a specified condition is true
- `if...else` statement - use this statement to execute some code if the condition is true and another code if the condition is false
- `if...else if....else` statement - use this statement to select one of many blocks of code to be executed
- `switch` statement - use this statement to select one of many blocks of code to be executed

if Statement

Use the `if` statement to execute some code only if a specified condition is true.

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
{
  document.write("<b>Good morning</b>");
}
</script>
```

if...else Statement

Use the `if....else` statement to execute some code if a condition is true and another code if the condition is not true.

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a
"Good morning" greeting.
//Otherwise you will get a "Good day" greeting.
var d = new Date();
var time = d.getHours();

if (time < 10)
{   document.write("Good morning!"); }
else
{   document.write("Good day!");  }
</script>
```

if...else if...else Statement

Use if....else if...else statement to select one of several blocks of code to be executed.

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
{
  document.write("<b>Good morning</b>");
}
else if (time>=10 && time<16)
{
  document.write("<b>Good day</b>");
}
else
{
  document.write("<b>Hello World!</b>");
}
</script>
```

JavaScript switch Statement

Conditional statements are used to perform different actions based on different conditions.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from
case 1 and 2
}
```

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
var theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this
weekend!");
}
</script>
```

JavaScript Popup Boxes

JavaScript has three kinds of popup boxes: alert box, confirm box, and prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

Example

```
<script>
 alert("welcome to javascript");
</script>
```

Confirm Box

A confirm box is often used 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. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
confirm("sometext");
```

Example

```
<script>
var r=confirm("Press a button");
if (r==true)
{
  alert("You pressed OK!");
}
else
{
  alert("You pressed Cancel!");
}
</script>
```

Prompt Box

A 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. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

Example

```
<script>
var name = prompt("Please enter your
name","Harry Potter");
if (name!=null && name!="")
{
  document.write("Hello " + name + "! How are
you today?");
}
</script>
```

JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kinds of loops:

- `for` - loops through a block of code a specified number of times
- `while` - loops through a block of code while a specified condition is true

for Loop

The `for` loop is used when you know in advance how many times the script should run.

Syntax

```
for
(variable=startvalue;variable<=endvalue;variable
=variable+increment)
```

```
{
  code to be executed
}
```

Example

The example below defines a loop that starts with `i=0`. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs.

Note: The increment parameter could also be negative, and the <= could be any comparing statement.

```
<script>
var i=0;
for (i=0;i<=5;i++)
{
  document.write("The number is " + i);
  document.write("<br>");
}
</script>
```

while Loop

The `while` loop loops through a block of code while a specified condition is true.

Syntax

```
while (variable<=endvalue)
{
  code to be executed
}
```

Note: The <= could be any comparing operator.

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as i is less than, or equal to 5. i will increase by 1 each time the loop runs:

```
<script>
var i=0;
while (i<=5)
{
  document.write("The number is " + i);
  document.write("<br>");
  i++;
}
</script>
```

The do...while Loop

The `do...while` loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

```
do
{
  code to be executed
}
while (variable<=endvalue);
```

The example below uses a `do...while` loop. The `do...while` loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<script>
var i=0;
do
{
  document.write("The number is " + i);
  document.write("<br>");
  i++;
}
while (i<=5);
</script>
```

break Statement

The `break` statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<script>
var i=0;
for (i=0;i<=10;i++)
{
  if (i==3)
  {
    break;
  }
  document.write("The number is " + i);
  document.write("<br>");
}
</script>
```

The continue Statement

The continue statement will break the current loop and continue with the next value.

Example

```
<script>
var i=0
for (i=0;i<=10;i++)
{
  if (i==3)
  {
    continue;
  }
  document.write("The number is " + i);
  document.write("<br>");
}
</script>
```

JavaScript for...in Statement

The `for...in` statement loops through the properties of an object.

Syntax

```
for (variable in object)
{
  code to be executed
}
```

Example

```
var person={fname:"John",lname:"Doe",age:25};
for (x in person)
{    document.write(person[x] + " ");
}
```

Example 1: Greater number amongst two numbers

```
<html>
<script>
var a = parseInt(prompt(“Enter first no”));
var b = parseInt(prompt(“Enter second no”));
document.write("<br>");
if(a>b)
{   document.write(a + " is greater no");
}
else
{   document.write(b + " is greater no");
}
</script>
```

Example 2: Greater number amongst three numbers

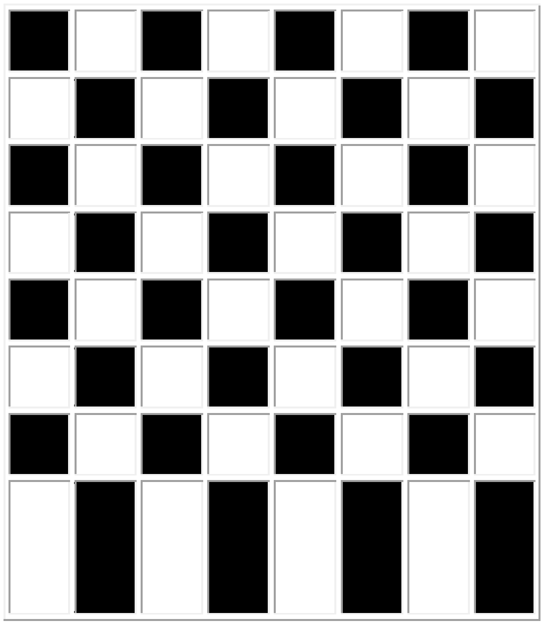
```
<script>
var a = parseInt(prompt(“Enter first no”));
var b = parseInt(prompt(“Enter second no”));
var c = parseInt(prompt(“Enter third no”));
document.write("<br>A = " +a);
document.write("<br>B = " +b);
document.write("<br>C = " + c + "<br>");
if(a>b && a>c)
{    document.write("a is greator than b and c
which is " + a);
}
```

```
else if(b>a && b>c)
{    document.write("b is greator than a and c
which is " + b);
}
else
{    document.write("c is greator than a and b
which is " + c);
}
</script>
```

Example 3: Displaying chess board

```
<html>
<style type="text/css">
table, td, tr
{
   border:2px solid purple;
   border-collapse:collapse;
     width:1000px;
     height:50px;
}
</style>
<script>
document.write("<table>");
for(i=1;i<=8;i++)
{
  document.write("<tr>");
  if(i%2==0)
  {
    for(j=1;j<=8;j++)
    {
    if(j%2==0)
    {   document.write("<td
style='background:black'></td>");
    }
    else
    {  document.write("<td
style='background:white'></td>");
    }
    }
   }
   else
   {  for(j=1;j<=8;j++)
      {
      if(j%2==0)
      {    document.write("<td
style='background:white'></td>");
      }
      else
      {    document.write("<td
style='background:black'></td>");
      }
      }
   }
   document.write("</tr>");
}
document.write("</table>");
</script>
</html>
```

Output:

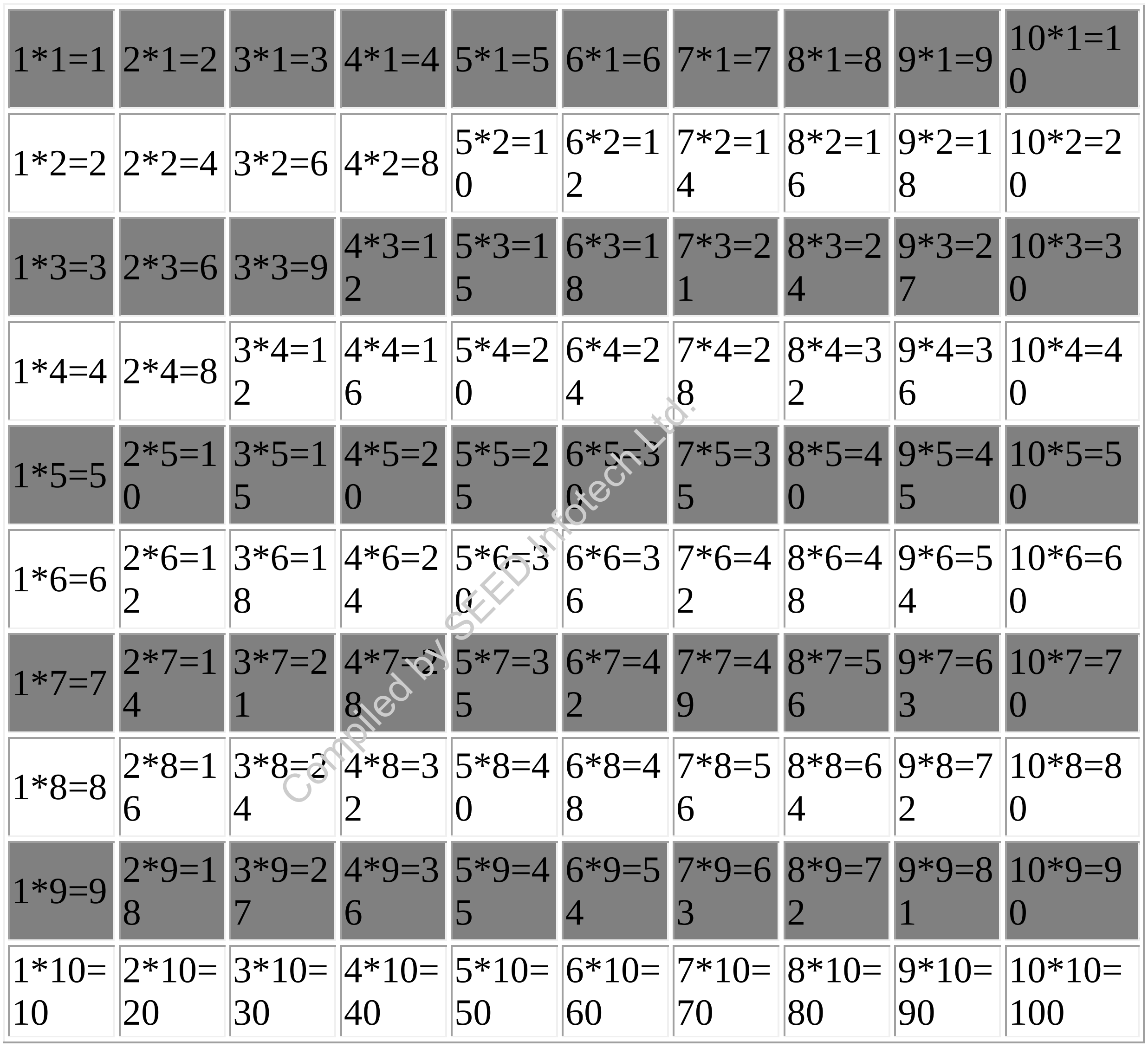


Example 4: Displaying multiplication table

```
<html>
<style type="text/css">
table, td, tr
```

```
{
     border:1px solid blue;
   border-collapse:collapse;
   margin:50px auto;
   width:1000;
   height:50;
   text-align:center;
   font-weight:bold;
   font-size:20px;
</style>
<script>
var i,j;
document.write("<table>");
for(i=1;i<=10;i++)
{
  document.write("<tr>");
  if(i%2==0)
  {    for(j=1;j<=10;j++)
   {
       document.write("<td style='background-
color:black; color:white’
           + j + "*" + i + "=" + j*i + "</td >");
   }
  }
  else
  {
   for(j=1;j<=10;j++)
   {
       document.write("<td style='background-
color:white; color:black’
           + j + "*" + i + "=" + j*i + "</td >");
   }
  }
  document.write("</tr>");
}
document.write("</table>");
</script>
</html>
```

Output:



1*1=1	2*1=2	3*1=3	4*1=4	5*1=5	6*1=6	7*1=7	8*1=8	9*1=9	10*1=10
1*2=2	2*2=4	3*2=6	4*2=8	5*2=10	6*2=12	7*2=14	8*2=16	9*2=18	10*2=20
1*3=3	2*3=6	3*3=9	4*3=12	5*3=15	6*3=18	7*3=21	8*3=24	9*3=27	10*3=30
1*4=4	2*4=8	3*4=12	4*4=16	5*4=20	6*4=24	7*4=28	8*4=32	9*4=36	10*4=40
1*5=5	2*5=10	3*5=15	4*5=20	5*5=25	6*5=30	7*5=35	8*5=40	9*5=45	10*5=50
1*6=6	2*6=12	3*6=18	4*6=24	5*6=30	6*6=36	7*6=42	8*6=48	9*6=54	10*6=60
1*7=7	2*7=14	3*7=21	4*7=28	5*7=35	6*7=42	7*7=49	8*7=56	9*7=63	10*7=70
1*8=8	2*8=16	3*8=24	4*8=32	5*8=40	6*8=48	7*8=56	8*8=64	9*8=72	10*8=80
1*9=9	2*9=18	3*9=27	4*9=36	5*9=45	6*9=54	7*9=63	8*9=72	9*9=81	10*9=90
1*10=10	2*10=20	3*10=30	4*10=40	5*10=50	6*10=60	7*10=70	8*10=80	9*10=90	10*10=100

Chapter 5 - Event Handling and Client Side Validation

Events

It describes action that occurs as the result of user interaction with a web page or other browser related activities.

For Example:

1) Clicks a hyperlink
2) Selecting input box on HTML
3) Submitting HTML form
4) A mouse click
5) A button click

Event handling

The process that is performed in response to the occurrence of an event is known as Event Handling

Event handler

The code which gets executed on occurrence of event to generate the response is called as Event handler. It is a two-step Process:

1) Defining an event that can be handled by scripts
2) Providing a standard method of connecting these events to user-supplied JavaScript code

List of events

Attribute	Description
onblur	An element loses focus
onchange	The content of a field changes
onclick	Mouse clicks an object
onfocus	An element gets focus
onkeydown	A keyboard key is pressed

onkeypress	A keyboard key is pressed or held down
onkeyup	A keyboard key is released
onload	A page or image is finished loading
onmousedown	A mouse button is pressed
onmousemove	The mouse is moved
onmouseout	The mouse is moved off an element
onmouseover	The mouse is moved over an element
onselect	Text is selected

1) onload

The onload and events are triggered when the user enters or leaves the page. The `onload` event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

```
<html>
<script>
function fun1()
{   alert("welcome to javascript"); }
</script>
<body onload="fun1()">
</body>
</html>
```

2) onclick

```
<html>
<script>
function fun1()
{   alert("welcome to javascript"); }
</script>
<body>
  <input type="button" value="Click Me!">
```

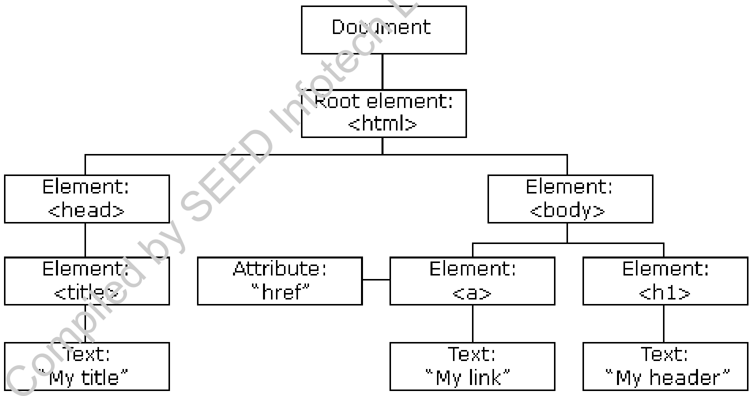
```
</body>
</html>
```

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** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

What is the HTML DOM?

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 DOM Programming Interface

The HTML DOM can be accessed with JavaScript (and with other programming languages). In the DOM, all HTML elements are defined as **objects**. The programming interface is the properties and methods of each object. A property is a value that you can get or set (like changing the content of an HTML element). A method is an action you can do (like add or deleting an HTML element).

Example:

```
<html>
<body>
<p id="para1">This is HTML DOM</p>
<script>
  var txt =
document.getElementById("para1").innerHTML;
  document.write(txt);
</script>
</body>
</html>
```

In the example above, getElementById is a **method**, while innerHTML is a **property**.

The getElementById Method

The most common way to access an HTML element is to use the id of the element. In the example above the `getElementById` method used `id="para1"` to find the element.

The innerHTML Property

The easiest way to get the content of an element is by using the **`innerHTML`** property. The `innerHTML` property is useful for getting or replacing the content of HTML elements. The `innerHTML` property can be used to get or change any HTML element, including `<html>` and `<body>`.

The HTML DOM Document

In the HTML DOM object model, the document object represents your web page. The document object is the owner of all other objects in your web page. If you want to access objects in an HTML page, you always start with accessing the document object. Below is some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements

Method	**Description**
`document.getElementById()`	Finding an element by element id
`document.getElementsByTagName()`	Finding elements by tag name
`document.getElementsByClassName()`	Finding elements by class name
`document.forms[]`	Finding elements by HTML element objects

Changing HTML Elements

Method	**Description**
`document.write(`*`text`*`)`	Writing into the HTML output stream
`document.getElementById(`*`id`*`).innerHTML=value`	Changing the inner

	HTML of an element
`document.getElementById(id).attribute=value`	Changing the attribute of an element
`document.getElementById(id).style.attribute=value`	Changing the style of an HTML element

Adding and Deleting Elements

Method	Description
`document.createElement()`	Create an HTML element
`document.removeChild()`	Remove an HTML element
`document.appendChild()`	Add an HTML element
`document.replaceChild()`	replace an HTML element

Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements. To do so, you have to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by HTML object collections

Example1: Changing background color when button is clicked

```
<html>
<body>
  <input type = "button" value = "Change
background color"
   onclick = "document.body.style.background =
'skyblue'">
</body>
</html>
```

Example2: Changing style of paragraph when button is clicked

```
<html>
<script>
function ChangeStyle()
{

document.getElementById("b").style.color="purple
";

document.getElementById("b").style.fontFamily="A
rial";

document.getElementById("b").style.fontSize=50;
}
</script>
<body>
  <b id="b"> SEED Infotech </b><hr>
  <input type="button" onclick="ChangeStyle()"
value="Change styles">
</body>
</html>
```

Example3: Addition and subtraction of two numbers

```
<html>
 <head>
  <script>
   function add()
   {
      var a =
parseInt(document.getElementById('no1').value);
    var b =
parseInt(document.getElementById('no2').value);
    var add = a + b;
    document.getElementById('add').value = 'add
is = ' + (a+b);
   }
   function sub()
   {
     var a =
parseInt(document.getElementById('no1').value);
    var b =
parseInt(document.getElementById('no2').value);
    var sub = a + b;
    document.getElementById('sub').innerHTML =
'sub is = ' + (a-b);
   }
  </script>
 </head>
<body>
  First Number  ::
  <input type='text' id='no1'><br>
  Second Number ::
  <input type='text' id='no2'><br>
  Addition ::
  <input type='text' id='add'><br>
  Subtraction ::
  <b id='sub'> </b><br>
  <input type='button' value='ADD'
onclick='add()'>
  <input type='button' value='SUB'
```

```
onclick='sub()'>
 </body>
</html>
```

Example4: Client-side validation using regular expression

```
<HTML>
 <HEAD>
  <script>
  function validation()
  {     var x = form1.fname.value;
        var str = /(^[A-Za-z]{1,1})+([a-
z]{1,19})$/;
        if(x=='')
        {   msg1.innerHTML = 'Plz Enter Your
Name';
            return false;
        }
        else
        {  msg1.innerHTML = '';
        }
        if(!x.match(str))
        {   msg1.innerHTML = 'Enter your name
such as Kedar, Mandar';
            return false;
        }
        else
        {   msg1.innerHTML = '';
        }
        if((form1.gender[0].checked==false) &&

(form1.gender[1].checked==false))
        {   msg2.innerHTML = 'Plz select your
gender';
            return false;
        }
        else
        {  msg2.innerHTML = '';
        }
        if(form1.city.selectedIndex==0)
```

```
       {   msg3.innerHTML = 'Plz select your
city group';
           return false;
       }
       else
       { msg3.innerHTML = '';
       }
       if(form1.terms.checked==false)
       {
           msg4.innerHTML = 'Plz check above
checkbox otherwise you will
                                  not be able to
create new user account';
           return false;
       }
       else
       {  msg4.innerHTML = '';   }
  }
  </script>
 </HEAD>
 <BODY>
  <form name='form1' action='#' method='post'
onsubmit='return validation()'>
  <table>
  <tr>
   <td> First Name: </td>
   <td> <input type='text' name='fname'
id='fname'> </td>
   <td> <i id='msg1'></i> </td>
  </tr>
  <tr>
   <td> Gender: </td>
   <td>
       Male<input type='radio' name='gender'
value='Male'>
       Female<input type='radio' name='gender'
value='Female'>
   </td>
   <td> <i id='msg2'></i> </td>
  </tr>
  <tr>
   <td> City: </td>
   <td>
   <select name='city'>
       <option>Select</option>
       <option>Pune</option>
       <option>Mumbai</option>
       <option>Kolhapur</option>
       <option>Nagpur</option>
   </select>
   </td>
   <td> <i id='msg3'></i> </td>
  </tr>
  <tr>
   <td> Terms and conditions: </td>
   <td> <input type='checkbox' name='terms'>
</td>
   <td> <i id='msg4'></i> </td>
  </tr>
  <tr>
   <td colspan='2'>
       <input type='submit' value='Submit'>
       <input type='reset' value='Clear'>
   </td>
  </table>
  </form>
 </BODY>
</HTML>
```

Chapter 6 - Core jQuery

What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on your website.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code. jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Tip: In addition, jQuery has plugins for almost any task out there.

Why jQuery?

There are a lots of other JavaScript frameworks out there, but jQuery seems to be the most popular, and also the most extendable. Many of the biggest companies on the Web use jQuery, such as:

- Google
- Microsoft
- IBM
- Netflix

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

Both versions can be downloaded from jQuery.com. The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (notice that the <script> tag should be inside the <head> section):

```
<head>
<script src="jquery-1.9.1.min.js"></script>
</head>
```

Tip: Place the downloaded file in the same directory as the pages where you wish to use it.

Alternatives to Downloading

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. To use jQuery from Google or Microsoft, use one of the following:

```
<script
src="//ajax.googleapis.com/ajax/libs/jquery/1.9.
1/jquery.min.js">
</script>
                   OR
<script
src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-
1.9.1.min.js">
</script>
```

One big advantage of using the hosted jQuery from Google or Microsoft: Many users already have downloaded jQuery from Google or Microsoft when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time. Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

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 allows you to have your JavaScript code before the body of your document, in the head section.

Here are some examples of actions that can fail if methods are run before the document is fully loaded:

- Trying to hide an element that is not created yet
- Trying to get the size of an image that is not loaded yet

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

```
$(function(){

   // jQuery methods go here...

});
```

Use the syntax you prefer. We think that the document ready event is easier to understand when reading the code.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s). With jQuery selectors you can find elements based on their id, classes, types, attributes, values of attributes and much more. It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.

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

The element Selector

The jQuery element selector selects elements based on their tag names. 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:

Example

```
$(document).ready(function(){
  $("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. An id should be unique within a page, so you should use the #id selector when you want to find a single, unique element. To find an element with a specific id, write a hash character, followed by the id of the element:

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

Example

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

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

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:

Example

```
$(document).ready(function(){
  $("button").click(function(){
    $(".test").hide();
  });
});
```

More Examples of jQuery Selectors

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first <li> element of the first <ul>

$("ul li:first-child")	Selects the first <li> element of every <ul>
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

Use our jQuery Selector Tester to experiment with the different selectors.

For a complete reference of all the jQuery selectors, please go to our jQuery Selectors Reference.

What are Events?

All the different visitor's actions that a web page can respond to are called events. An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires"** is often used with events. Example: "The keypress event fires the moment you press a key".

Here are some common DOM events:

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	Load
dblclick	keydown	change	Resize
mouseenter	keyup	focus	Scroll
mouseleave		blur	Unload

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!!
});
```

Commonly Used 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 dbclick() 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!");
});
```

mousedown()

The mousedown() method attaches an event handler function to an HTML element. The function is executed, when the left mouse button is pressed down, while the mouse is over the HTML element:

Example

```
$("#p1").mousedown(function(){
  alert("Mouse down 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");
});
```

jQuery Event Methods

Event methods trigger or attach a function to an event handler for the selected elements. The following table lists all the jQuery methods used to handle events.

Method	Description
blur()	Attaches/Triggers the blur event
change()	Attaches/Triggers the change event
click()	Attaches/Triggers the click event
dblclick()	Attaches/Triggers the double click event
focus()	Attaches/Triggers the focus event
focusin()	Attaches an event handler to the focusin event
focusout()	Attaches an event handler to the focusout event

hover()	Attaches two event handlers to the hover event
keydown()	Attaches/Triggers the keydown event
keypress()	Attaches/Triggers the keypress event
keyup()	Attaches/Triggers the keyup event
mousedown()	Attaches/Triggers the mousedown event
mouseenter()	Attaches/Triggers the mouseenter event
mouseleave()	Attaches/Triggers the mouseleave event
mousemove()	Attaches/Triggers the mousemove event
mouseout()	Attaches/Triggers the mouseout event
mouseover()	Attaches/Triggers the mouseover event
mouseup()	Attaches/Triggers the mouseup event
one()	Adds one or more event handlers to selected elements. This handler can only be triggered once per element
ready()	Specifies a function to execute when the DOM is fully loaded
resize()	Attaches/Triggers the resize event
scroll()	Attaches/Triggers the scroll event
select()	Attaches/Triggers the select event
submit()	Attaches/Triggers the submit event

jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the hide() and show() methods:

Example

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

$("#show").click(function(){
  $("p").show();
});
```

Syntax:

```
$(selector).hide(speed,callback);
$(selector).show(speed,callback);
```

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds. The optional callback parameter is a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).

The following example demonstrates the speed parameter with hide():

Example

```
$("button").click(function(){
  $("p").hide(1000);
});
```

jQuery Callback Functions

JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors. To prevent this, you can create a callback function. A callback function is executed after the current effect is finished.

Typical syntax: `$(`*`selector`*`).hide(`*`speed,callback`*`);`

Examples

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

Example with Callback

```
$("button").click(function(){
  $("p").hide("slow",function(){
    alert("The paragraph is now hidden");
  });
});
```

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

Example without Callback

```
$("button").click(function(){
  $("p").hide(1000);
  alert("The paragraph is now hidden");
});
```

jQuery DOM Manipulation

One very important part of jQuery is the possibility to manipulate the DOM. jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents:

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation are:

- 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

The following example demonstrates how to get content with the jQuery text() and html() methods:

Example

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

The following example demonstrates how to get the value of an input field with the jQuery val() method:

Example

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

A Callback Function for text(), html(), and val()

All of the three jQuery methods above: text(), html(), and val(), also come with a callback function. The callback function has two parameters: the index of the current element in the list of elements selected and the original (old) value. You then return the string you wish to use as the new value from the function.

The following example demonstrates text() and html() with a callback function:

Example

```
$("#btn1").click(function(){
  $("#test1").text(function(i,origText){
    return "Old text: " + origText + " New text:
Hello world!
    (index: " + i + ")";
  });
});
$("#btn2").click(function(){
  $("#test2").html(function(i,origText){
    return "Old html: " + origText + " New html:
Hello <b>world!</b>
    (index: " + i + ")";
  });
});
```

Chapter -7 Introduction to Bootstrap

What is Bootstrap

- Bootstrap is the most popular HTML, CSS, and JS framework .
- Used for developing responsive, mobile first projects on the web.
- Designed for everyone, everywhere access.
- It was originally created and used by Twitter, called as Twitter Bootstrap.
- Bootstrap is an intuitive, and powerful mobile first front-end framework
- Used for creating responsive web development. It uses HTML, CSS and JavaScript.
- Bootstrap enables creating responsive designs.

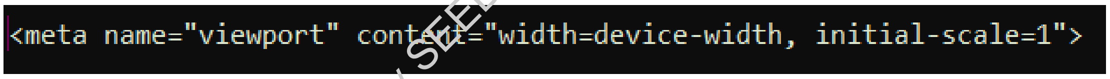
Advantages of Bootstrap

1. Rapid Application Design
2. Easy to Use
3. Responsive Features
4. Mobile-first Approach
5. Browser Compatibility
6. Open Source
7. Responsive web design

Responsive Web Design

creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

Bootstrap 3 is designed to be responsive to mobile devices.
Mobile-first styles are part of the core framework.
To ensure proper rendering and touch zooming etc.
e.g.



```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Bootstrap Packages

- Scaffolding

  Bootstrap provides a basic structure with Grid System, link styles, and background.
- CSS

  Bootstrap comes with feature of global CSS settings, fundamental HTML elements styled and enhanced with extensible classes, and an advanced grid system.
- Components

  Bootstrap contains reusable components
  Used to provide iconography, dropdowns, navigation, alerts, popovers, and much more.
- JavaScript Plugins

  Bootstrap contains over a dozen custom jQuery plugins that can make pages rich and responsive.
- Customize

  Customize Bootstrap's components, LESS variables, and jQuery plugins to get your very own version.

Bootstrap Download

There are two ways to start using Bootstrap on your own web site.

- Download Bootstrap from getbootstrap.com
- Include Bootstrap from a CDN

B
Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.
Download Bootstrap
Currently v3.3.4

Bootstrap CDN

MaxCDN provide CDN support for Bootstrap's CSS and JavaScript. Also include jQuery

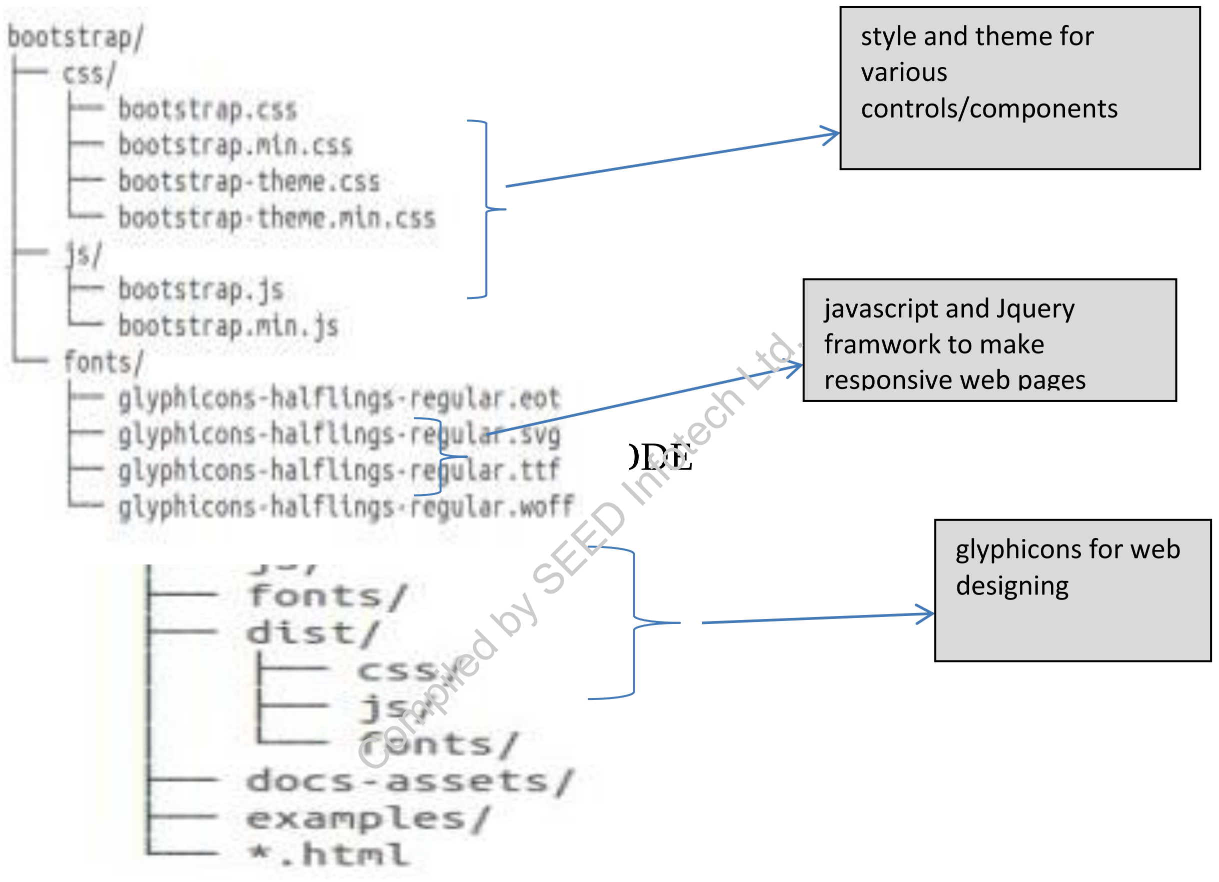
```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"href="http://maxcdn.bootst
rapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css
">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/li
bs/jquery/1.11.1/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="http://maxcdn.bootstrapcdn.com/boot
strap/3.3.4/js/bootstrap.min.js"></script>
```

Environment

File structure


bootstrap/
css/
bootstrap.css
bootstrap.min.css
bootstrap-theme.css
bootstrap-theme.min.css
js/
bootstrap.js
bootstrap.min.js
fonts/
glyphicons-halflings-regular.eot
glyphicons-halflings-regular.svg
glyphicons-halflings-regular.ttf
glyphicons-halflings-regular.woff
fonts/
dist/
css/
js/
fonts/
docs-assets/
examples/
*.html
style and theme for various controls/components
javascript and Jquery framwork to make responsive web pages
glyphicons for web designing

Basic HTML Template using Bootstrap

```
<body>
<h1>Hello, world!</h1>
<!-- jQuery (necessary for Bootstrap's
JavaScript plugins) -->
<script
src="https://code.jquery.com/jquery.js"></script
>
<!-- Include all compiled plugins (below), or
include individual files
as needed -->
<script src="js/bootstrap.min.js"></script>
</body>
    </html>
```

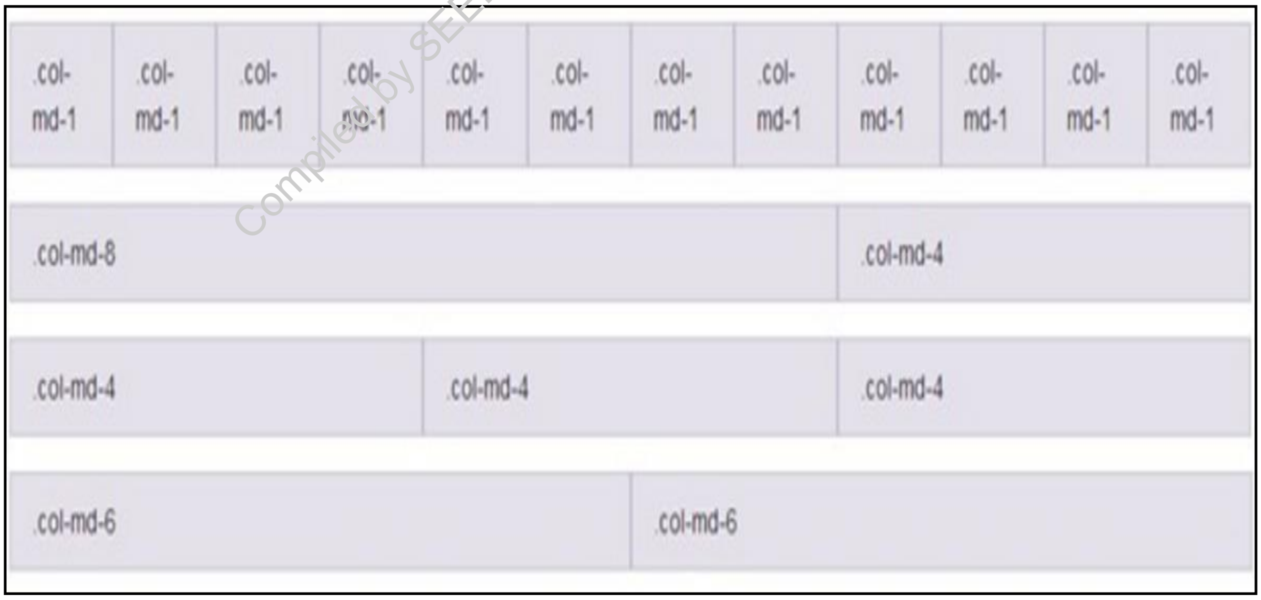
Introduction to Grid in Bootstrap

- In graphic design, a grid is a structure made up of a series of intersecting straight lines
- These are used to structure content.
- It is widely used to design layout and content structure in print design.
- In web design, it is a very effective method to create a consistent layout rapidly and effectively using HTML and CSS.

Bootstrap Grid System

Bootstrap includes a responsive, mobile fluid grid system.

- This Grid system appropriately scales up to 12 columns as the device or viewport size increases.
- It includes predefined classes for easy layout options, as well as powerful mixins for generating more semantic layouts.



Grid Classes

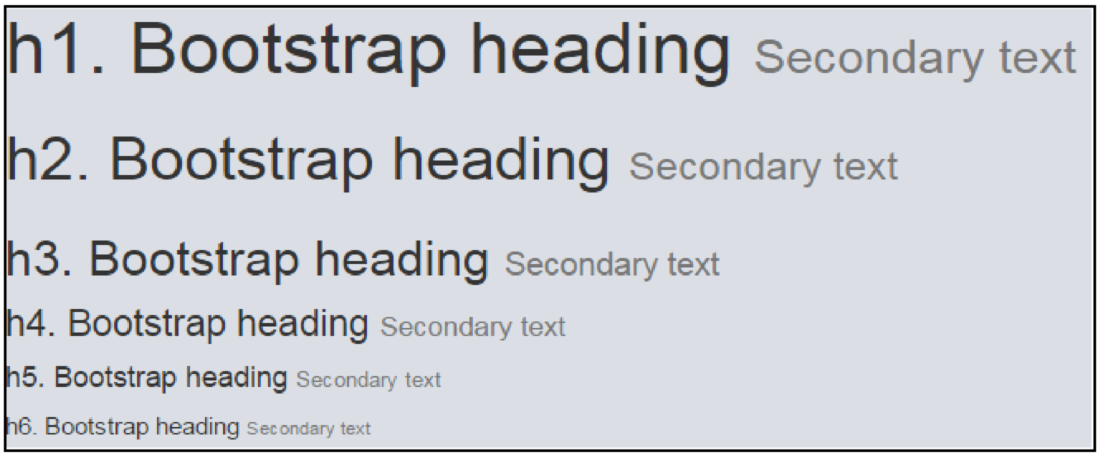
The Bootstrap grid system has four classes:

- xs (for phones)
- sm (for tablets)
- md (for desktops)
- lg (for larger desktops)

Bootstrap Typography

- The term typography is also applied to the
    - style, arrangement, and appearance of the letters, numbers, and symbols in a document.
- Bootstrap Typography features uses Helvetica Neue, Helvetica, Arial, and sans-serif in its default font stack.
- Using typography feature of Bootstrap you can create headings, paragraphs, lists and other inline elements.
- In Bootstrap the HTML <small> element is used to create a lighter.

```
 <h1>h1. Bootstrap heading <span class="small">Secondary
text</span></h1>
    <h2>h2. Bootstrap heading <span class="small">Secondary
text</span></h2>
    <h3>h3. Bootstrap heading <span class="small">Secondary
text</span></h3>
    <h4>h4. Bootstrap heading <span class="small">Secondary
text</span></h4>
    <h5>h5. Bootstrap heading <span class="small">Secondary
text</span></h5>
    <h6>h6. Bootstrap heading <span class="small">Secondary
text</span></h6>
```

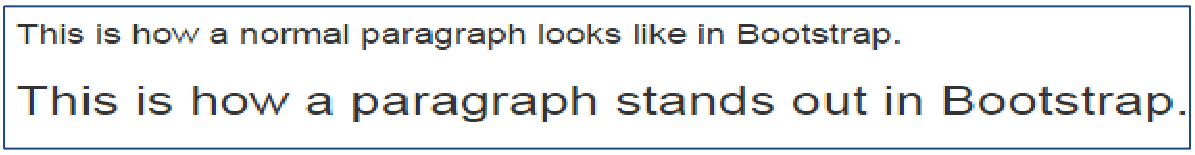


```
h1. Bootstrap heading Secondary text
h2. Bootstrap heading Secondary text
h3. Bootstrap heading Secondary text
h4. Bootstrap heading Secondary text
h5. Bootstrap heading Secondary text
h6. Bootstrap heading Secondary text
```

Lead Body

- To add some emphasis to a paragraph, add class="lead".
- This will give you larger font size, lighter weight, and a taller line height
- To add some emphasis to a paragraph, add class="lead".

- This will give you larger font size, lighter weight, and a taller line height



This is how a normal paragraph looks like in Bootstrap.

This is how a paragraph stands out in Bootstrap.

Bootstrap Tables

- Bootstrap provides a clean layout for building tables.
- Some of the table elements supported by Bootstrap are:
    - <table>
    - <thread>
    - <tbody>
    - <tr>
    - <td>
    - <th>
    - <caption>
- A basic Bootstrap table has a light padding and only horizontal dividers.
- The .table class adds basic styling to a table:

Tables with Alternate Background

- We can create table with alternate background like zebra-strips by simply adding the Bootstrap's class .table-striped to the .table base class.
- This is achieved by adding the background-color to the table row within <tbody> element via the :nth-child CSS selector

Table Decoration

- Bordered Table
    - Can also add borders to the all table cells by adding an extra Bootstrap's class .table-bordered to the .table base class.
- Hover Table
    - By adding the .table-hover class, a x-color background will be added to rows while the cursor hovers over them.

- Condensed Table
  - The .table-condensed class makes a table more compact by cutting cell padding in half.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
<script src="jquery/jquery-
3.1.1.slim.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
<body>
    <div class='container'>
       <h1>Condensed or Compact Table</h1>
      <table class="table table-condensed">
        <thead>
            <tr>
                <th>Roll No</th>
                <th>First Name</th>
                <th>Last Name</th>
                <th>Email</th>
            </tr>
         </thead>
          <tbody>
             <tr class="active">
            <td>1</td>
            <td>Anup</td>
            <td>Mehata</td>
            <td>Anup@gmail.com</td>
        </tr>
        <tr class="success">
            <td>2</td>
            <td>Priyanka</td>
            <td>Soni</td>
            <td>priyanka@gmail.com</td>
        </tr>
        <tr class="warning">
            <td>3</td>
            <td>Kartik</td>
            <td>Dube</td>
            <td>kartik@gmail.com</td>
        </tr>
        <tr class="danger">
            <td>4</td>
            <td>Amol</td>
            <td>Wagha</td>
            <td>amol@gmail.com</td>
        </tr>
         <tr class="info">
            <td>5</td>
            <td>Mery</td>
            <td>Richard</td>
            <td>mery@gmail.com</td>
        </tr>
     </tbody>
      </table>
  </div>
</body>
</html>
```

Bootstrap Forms

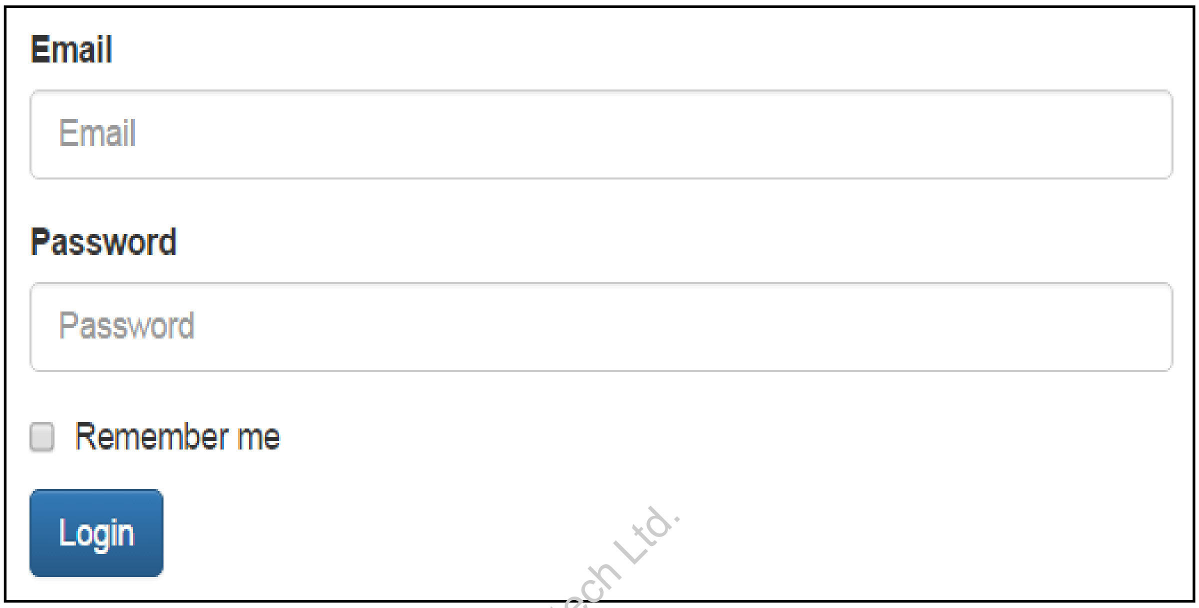
- Bootstrap makes it easy with the simple HTML markup and extended classes for different styles of forms.
- Bootstrap greatly simplifies the styling process of form controls like inputboxes, selectboxes, textareas, etc.
- Bootstrap provides three different types of form layouts:

- Vertical Form (default form layout)
- Horizontal Form
- Inline Form

Vertical Form Layout

- This is the default Bootstrap form layout in which styles are applied to form controls without adding any base class to the <form> element or any large changes in the markup.
- To create a basic form do the following:
  1. Add a role form to the parent <form> element.
  2. Wrap labels and controls in a <div> with class .form-group. This is needed for optimum spacing.
  3. Add a class of .form-control to all textual <input>, <textarea>, and <select> elements.

```
<form>
        <div class="form-group">
            <label
for="inputEmail">Email</label>
            <input type="email" class="form-
control" id="inputEmail" placeholder="Email">
        </div>
        <div class="form-group">
            <label
for="inputPassword">Password</label>
            <input type="password" class="form-
control" id="inputPassword"
placeholder="Password">
        </div>
        <div class="checkbox">
            <label><input type="checkbox">
Remember me</label>
        </div>
        <button type="submit" class="btn btn-
primary">Login</button>
    </form>
```



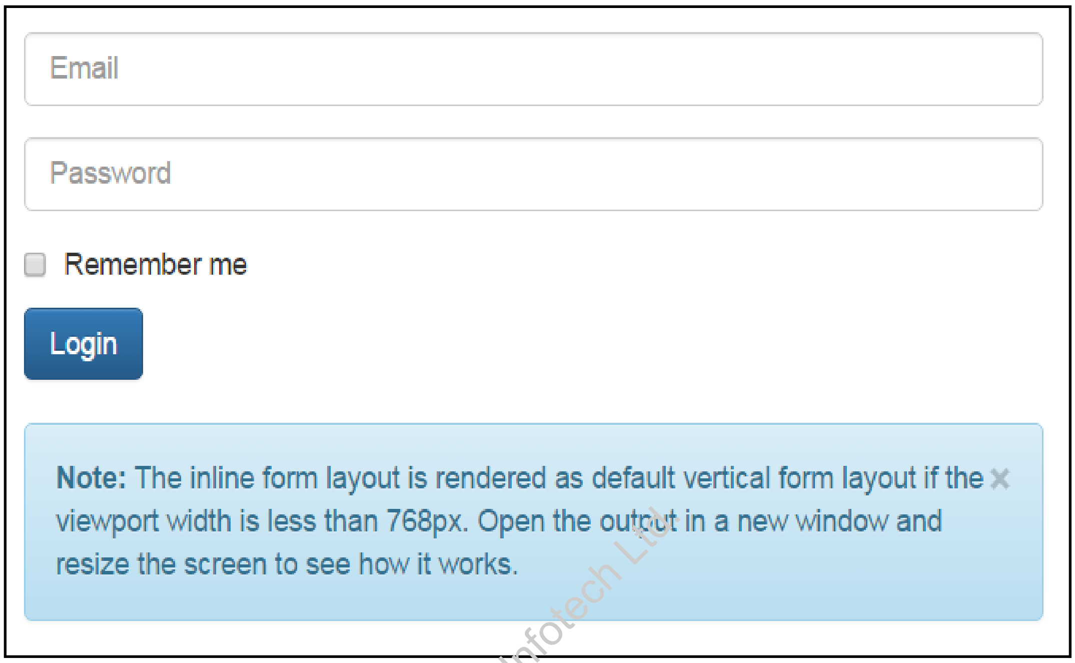
Inline Form Layout

- To place the form controls side-by-side to compact the layout.
  - Can do this easily by adding the Bootstrap class .form-inline to the <form> element.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>

    <style type="text/css">
    .bs-example{
       margin: 20px;
    }
</style>
</head>
<body>
   <div class="bs-example">
    <form>
        <div class="form-group">
            <label
for="inputEmail">Email</label>
            <input type="email" class="form-
control" id="inputEmail" placeholder="Email">
        </div>
        <div class="form-group">
            <label
for="inputPassword">Password</label>
            <input type="password" class="form-
control" id="inputPassword"
placeholder="Password">
        </div>
        <div class="checkbox">
            <label><input type="checkbox">
Remember me</label>
        </div>
        <button type="submit" class="btn btn-
primary">Login</button>
    </form>
</div>

    </body></html>
```



Note: The inline form layout is rendered as default vertical form layout if the viewport width is less than 768px. Open the output in a new window and resize the screen to see how it works.

Horizontal Form Layout

- In horizontal form layout labels are right aligned and floated to left to make them appear on the same line as form controls.
- The horizontal form layout requires the various markup changes from a default form layout.
- Steps to achieve this layout are listed below:
  1. Add the class .form-horizontal to the <form> element.
  2. Wrap labels and form controls in a <div> element and apply the class .form-group.
  3. Use Bootstrap's predefined grid classes to align labels and form controls.
  4. Add the class .control-label to the <label> element.

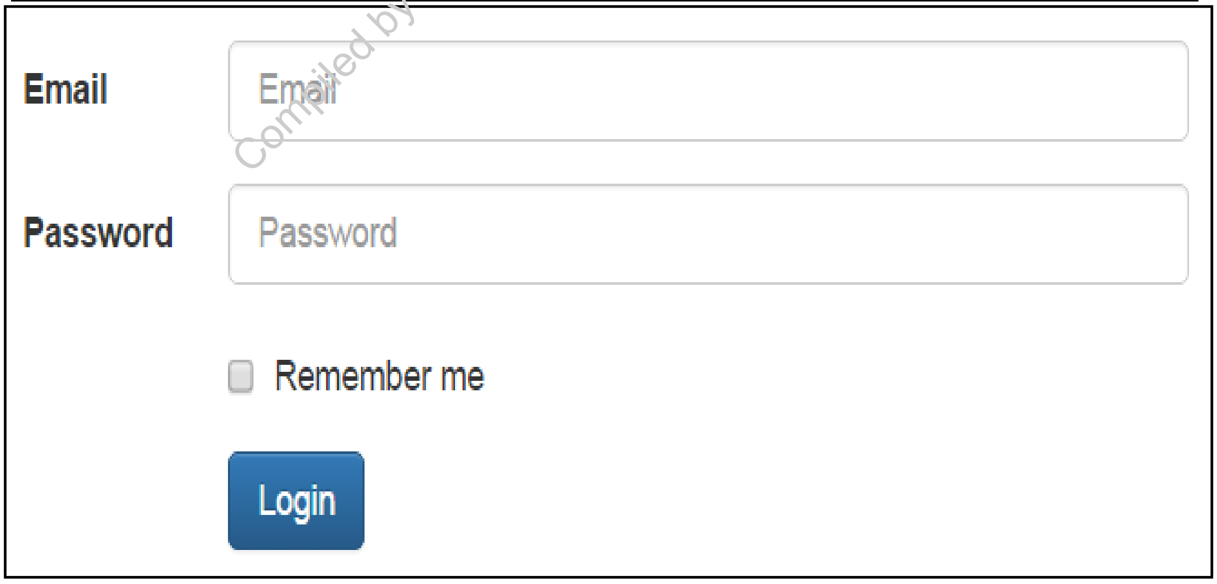
```
  <!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
```

```
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>

    <style type="text/css">
    .bs-example{
       margin: 20px;
    }
      .form-horizontal .control-label{
        padding-top: 7px;
    }
</style>
</head>
<body>
 <div class="bs-example">
    <form class="form-horizontal">
        <div class="form-group">
            <label for="inputEmail"
class="control-label col-xs-2">Email</label>
            <div class="col-xs-10">
                <input type="email" class="form-
control" id="inputEmail" placeholder="Email">
            </div>
        </div>
        <div class="form-group">
            <label for="inputPassword"
class="control-label col-xs-2">Password</label>
            <div class="col-xs-10">
                <input type="password"
class="form-control" id="inputPassword"
placeholder="Password">
            </div>
        </div>
        <div class="form-group">
            <div class="col-xs-offset-2 col-xs-
10">
                <div class="checkbox">
```

```
                    <label><input
type="checkbox"> Remember me</label>
                <div>
            </div>
        </div>
        <div class="form-group">
            <div class="col-xs-offset-2 col-xs-
10">
                <button type="submit" class="btn
btn-primary">Login</button>
            </div>
        </div>
    </form>
</div>

</body>
```



Static-Form control

- When you need to place just plain text next to a form label instead of a form control.
  - Can do this within a horizontal form layout by using the .form-control-static class on a <p> element.

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

```
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>

   <style type="text/css">
    .bs-example{
       margin: 20px;
    }
   /* Fix alignment issue of label on extra
small devices in Bootstrap 3.2 */
    .form-horizontal .control-label{
        padding-top: 7px;
    }
</style>
</head>
<body>
   <div class="bs-example">
    <form class="form-horizontal">
        <div class="form-group">
            <label for="inputEmail"
class="control-label col-xs-2">Email</label>
            <div class="col-xs-10">
                <p class="form-control-
static">harrypotter@mail.com</p>
            </div>
        </div>
      <div class="form-group">
            <label for="inputPassword"
class="control-label col-xs-2">Password</label>
```

```
            <div class="col-xs-10">
                <input type="password"
class="form-control" id="inputPassword"
placeholder="Password">
            </div>
        </div>
        <div class="form-group">
            <div class="col-xs-offset-2 col-xs-
10">
                <div class="checkbox">
                    <label><input
type="checkbox"> Remember me</label>
                </div>
            </div>
        </div>
        <div class="form-group">
            <div class="col-xs-offset-2 col-xs-
10">
                <button type="submit" class="btn
btn-primary">Login</button>
            </div>
        </div>
    </form>
</div>
</body>
</html>
```

Supported Form Controls

Bootstrap natively supports the most common form controls mainly input, textarea, checkbox, radio, and select.

- Inputs
- TextArea
- Checkboxes
- Radios
- Selects
- Text
- Password

- datetime

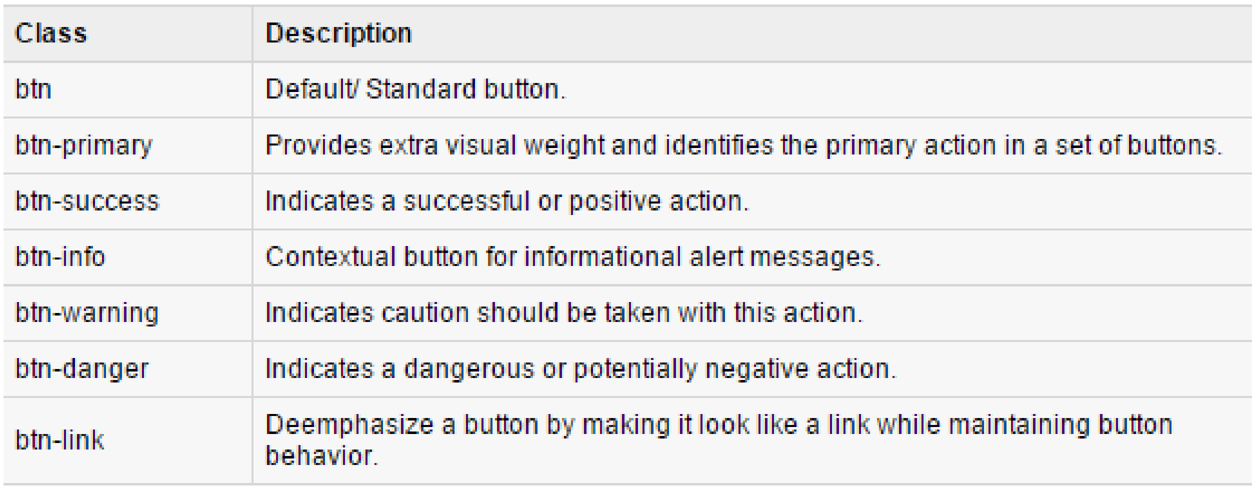
Inputs Controls

Bootstrap offers support for all native HTML5 input types:

- datetime-local
- Date
- Month
- Time
- Week
- Number
- email,
- url, search,
- tel, and color.

Bootstrap Buttons

- Anything that is given a class of '.btn' will inherit the default look of a gray button with rounded corners.
- However Bootstrap provides some options to style buttons, which are summarized in the following table:



Class	Description
btn	Default/ Standard button.
btn-primary	Provides extra visual weight and identifies the primary action in a set of buttons.
btn-success	Indicates a successful or positive action.
btn-info	Contextual button for informational alert messages.
btn-warning	Indicates caution should be taken with this action.
btn-danger	Indicates a dangerous or potentially negative action.
btn-link	Deemphasize a button by making it look like a link while maintaining button behavior.

Button Size

The following table summarizes classes used to get buttons of various sizes:

Class	Description
.btn-lg	This makes button size large.
.btn-sm	This makes button size small.
.btn-xs	This makes button size with extra small.
.btn-block	This creates block level buttons—those that span the full width of a parent.

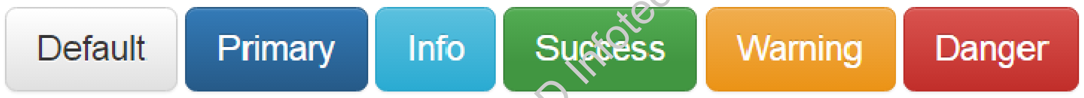
Bootstrap Buttons

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>


  <style type="text/css">
    .bs-example{
       margin: 20px;
    }
</style>

</head>
<body>
<div class="bs-example">
    <button type="button" class="btn btn-default
btn-lg">Default</button>
    <button type="button" class="btn btn-primary
btn-lg">Primary</button>
```

```
    <button type="button" class="btn btn-info
btn-lg">Info</button>
    <button type="button" class="btn btn-success
btn-lg">Success</button>
    <button type="button" class="btn btn-warning
btn-lg">Warning</button>
    <button type="button" class="btn btn-danger
btn-lg">Danger</button>
    </div>
</body>
</html>
```



Bootstrap Lists

There are three different types of List in Bootstrap

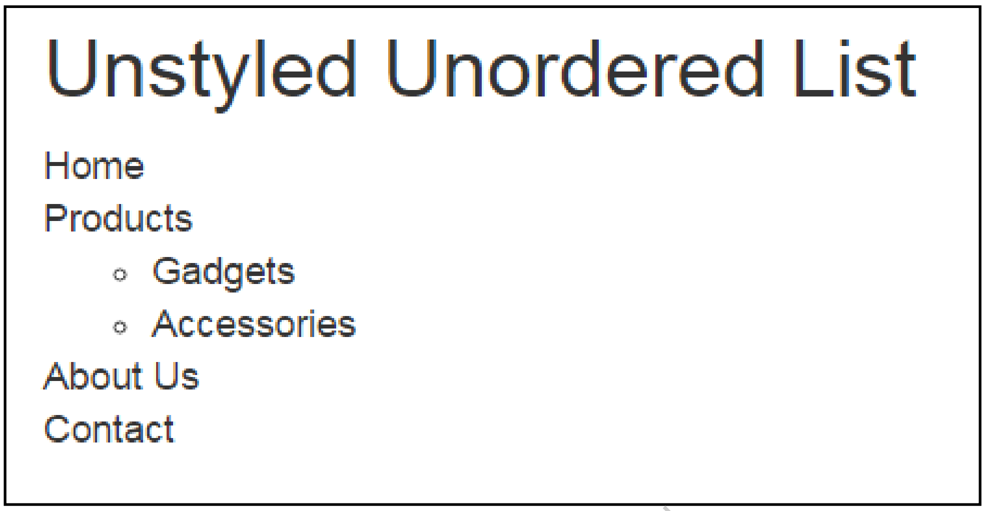
- Unordered lists — A list of items in which the order does not explicitly matter. The list items in unordered lists are marked with bullets.
- Ordered lists — A list of items in which the order does explicitly matter. The list items in ordered lists are marked with numbers.
- Definition list — A list of terms with their associated descriptions.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
```

```
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>


  <style type="text/css">
    .bs-example{
       margin: 20px;
    }
</style>

</head>
<body>
        <div class="bs-example">
<h2>Unstyled Unordered List</h2>
    <ul class="list-unstyled">
        <li>Home</li>
        <li>Products
            <ul>
                <li>Gadgets</li>
                <li>Accessories</li>
            </ul>
        </li>
        <li>About Us</li>
        <li>Contact</li>
    </ul>
</h2>
</div>
</body>
</html>
```


Unstyled Unordered List

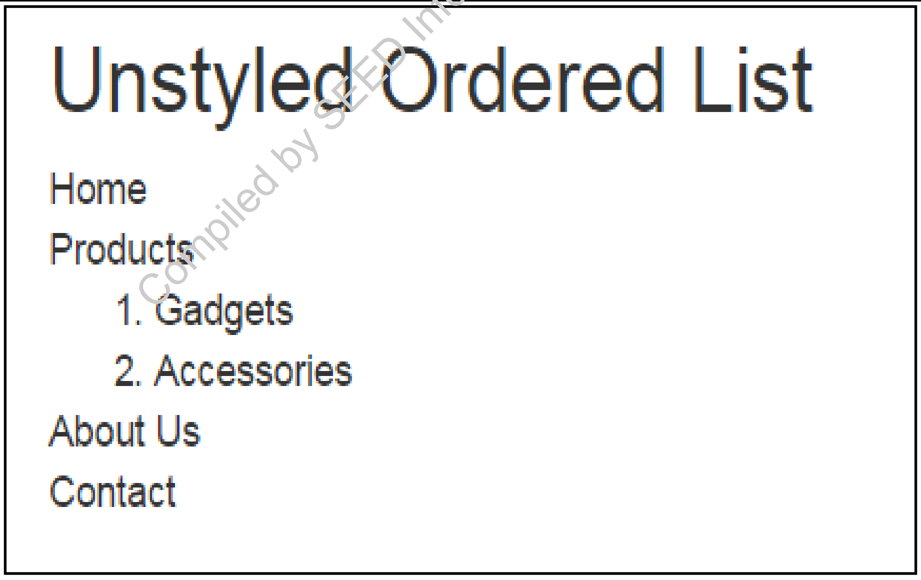
Home
Products
- Gadgets
- Accessories

About Us
Contact

Bootstrap List – Ordered list

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
   <script src="js/bootstrap.min.js"></script>

  <style type="text/css">
    .bs-example{
       margin: 20px;
    }
</style>
</head>
<body>
        <div class="bs-example">
<h2>Unstyled Ordered List</h2>
    <ol class="list-unstyled">
        <li>Home</li>
        <li>Products
            <ol>
                <li>Gadgets</li>
                <li>Accessories</li>
            </ol>
        </li>
        <li>About Us</li>
        <li>Contact</li>
    </ol>
</div>
</div>
</body>
</html>
```



Unstyled Ordered List

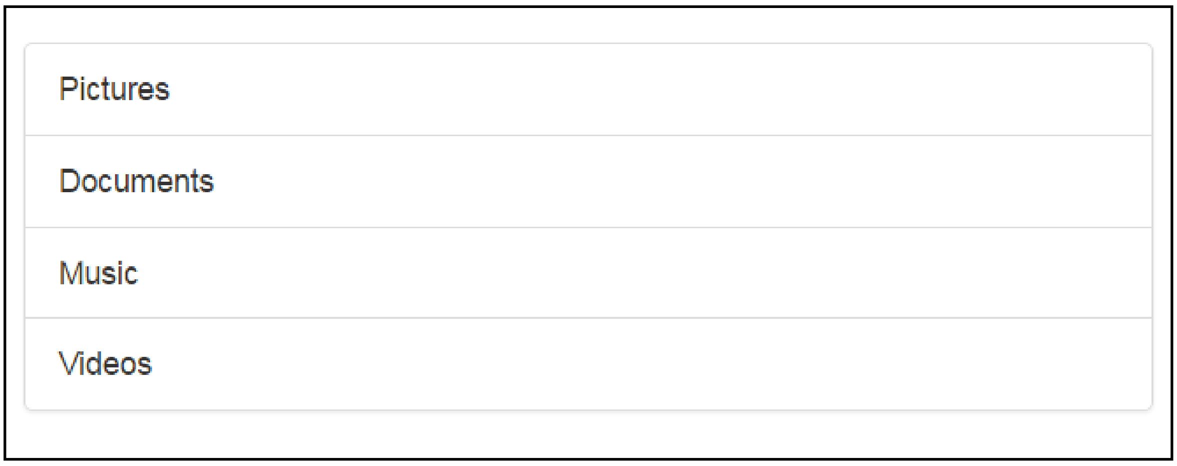
Home
Products
1. Gadgets
2. Accessories
About Us
Contact

Bootstrap List Group

- Purpose of list group component is to render complex and customized content in lists.
- To get a basic list group:
  1. Add the class **.list-group** to element <ul>.
  2. Add class **.list-group-item** to <li>.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```
<meta http-equiv="X-UA-Compatible"
content="IE=edge">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<title>Basic Bootstrap Template</title>
<link rel="stylesheet" type="text/css"
href="css/bootstrap.min.css">
<!-- Optional Bootstrap theme -->
<link rel="stylesheet" href="css/bootstrap-
theme.min.css">
<script src="jquery/jquery-
3.1.1.slim.min.js"></script>
    <script src="js/bootstrap.min.js"></script>
    <style type="text/css">
    .bs-example{
       margin: 20px;
    }
    </style>
    </head>
<body>
  <div class="bs-example">
    <ul class="list-group">
        <li class="list-group-
item">Pictures</li>
        <li class="list-group-
item">Documents</li>
        <li class="list-group-item">Music</li>
        <li class="list-group-item">Videos</li>
    </ul>
</div>
</body>
</html>
```



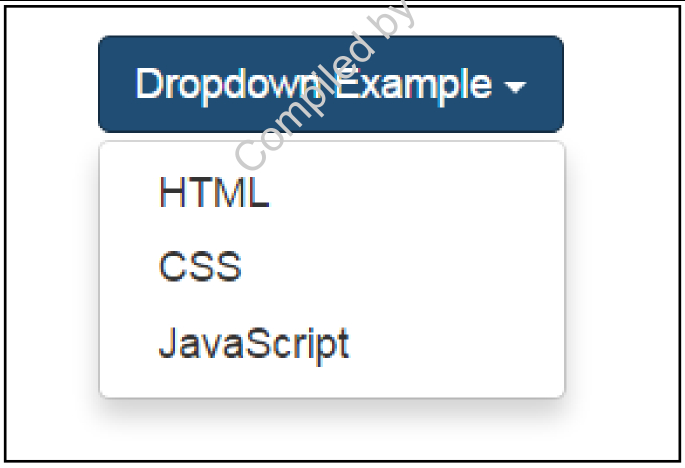
Dropdowns

- The .dropdown class indicates a dropdown menu.
- To open the dropdown menu, use a button or a link with a class of .dropdown-toggle
- The .caret class creates a caret arrow icon ( ), which indicates that the button is a dropdown.
- Add the .dropdown-menu class to a <ul> element to actually build the dropdown menu.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-
width, initial-scale=1">
  <link rel="stylesheet"
href="css/bootstrap.min.css">
  <script src="jquery/jquery.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
  <h2>Dropdowns</h2>
  <p>The .dropdown class is used to indicate a
dropdown menu.</p>
  <p>Use the .dropdown-menu class to actually
build the dropdown menu.</p>
```

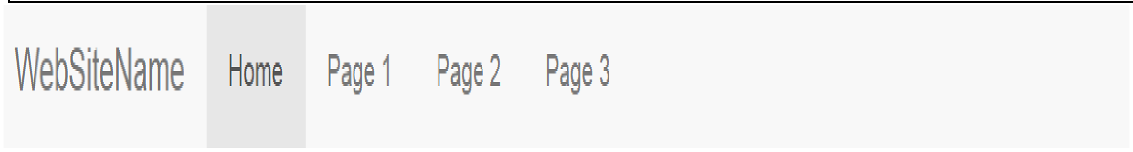
```
  <p>To open the dropdown menu, use a button or
a link with a class of .dropdown-toggle and
data-toggle="dropdown".</p>
  <div class="dropdown">
    <button class="btn btn-primary dropdown-
toggle" type="button" data-
toggle="dropdown">Dropdown Example
    <span class="caret"></span></button>
    <ul class="dropdown-menu">
      <li><a href="#">HTML</a></li>
      <li><a href="#">CSS</a></li>
      <li><a href="#">JavaScript</a></li>
    </ul>
  </div>
</div>
</body></html>
```



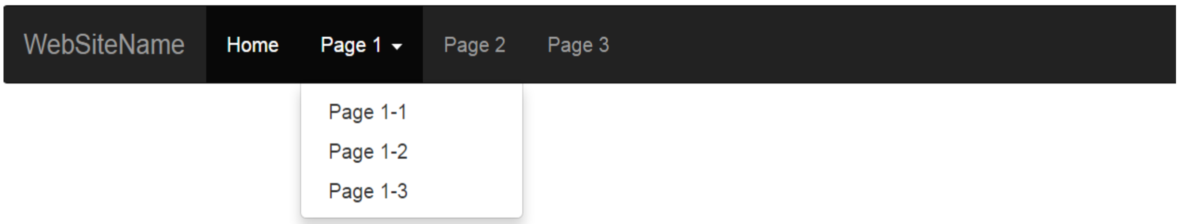
Bootstrap Navbar

- The navbar is a nice feature, and is one of the prominent features of Bootstrap sites.
- Navbars are responsive meta components that serve as navigation headers for your application or site.
- Navbars collapse in mobile views and become horizontal as the available viewport width increases.
- At its core, the navbar includes styling for site names and basic navigation.

```
<!DOCTYPE html>
<html>
<head>
  <meta name="viewport" content="width=device-
width, initial-scale=1">
  <link rel="stylesheet"
href="css/bootstrap.min.css">
  <script src="jquery/jquery.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
</head>
<body>
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand"
href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a
href="#">Home</a></li>
      <li><a href="#">Page 1</a></li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </div>
</nav>
    <nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand"
href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a
href="#">Home</a></li>
      <li><a href="#">Page 1</a></li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </div>
</nav>
  <nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand"
href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a
href="#">Home</a></li>
      <li class="dropdown">
        <a class="dropdown-toggle" data-
toggle="dropdown" href="#">Page 1
        <span class="caret"></span></a>
        <ul class="dropdown-menu">
          <li><a href="#">Page 1-1</a></li>
          <li><a href="#">Page 1-2</a></li>
          <li><a href="#">Page 1-3</a></li>
        </ul>
      </li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </div>
</nav>
</body>
</html>
```



Bootstrap Navbar with Dropdown



Lab

Chapter 1- HTML

Lab Exercise - 1

Display website links on a HTML page (like google, facebook and use images to link this websites)

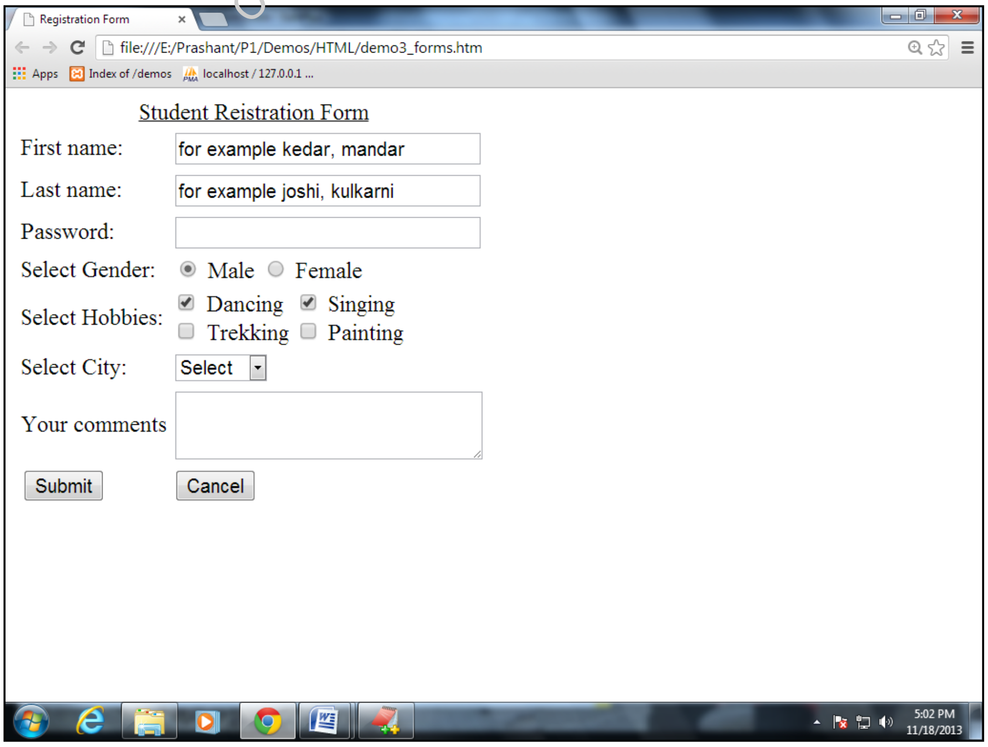
Below that display list of colors and fruits using <ul> and <ol> respectively

Below that display four different images and move them using <marquee> left-right and right-left

Set title for this web page

Lab Exercise - 2

Design a "Student Registration Form" that accepts First Name, Last Name, Password, Gender, Hobbies, Email, etc. Below is the sample look of a web page,


Registration Form
file:///E:/Prashant/P1/Demos/HTML/demo3_forms.htm
Apps
Index of /demos
localhost / 127.0.0.1 ...
5:02 PM
11/18/2013

Student Reistration Form

First name:	for example kedar, mandar
Last name:	for example joshi, kulkarni
Password:	
Select Gender:	● Male ○ Female
Select Hobbies:	☑ Dancing ☑ Singing ☐ Trekking ☐ Painting
Select City:	Select
Your comments	
Submit	Cancel

Lab Exercise-3

Create an Html page that display the tabular list as shown in the below snap short.

Product Code	Product Name	price	Company
P1	Freeze	17000	Godrej
p2	Washing Machine	18000	LG
p3	Microwave Oven	11000	Sony
p4	Dish washer	23000	IFB

Chapter 2 - Core of CSS3

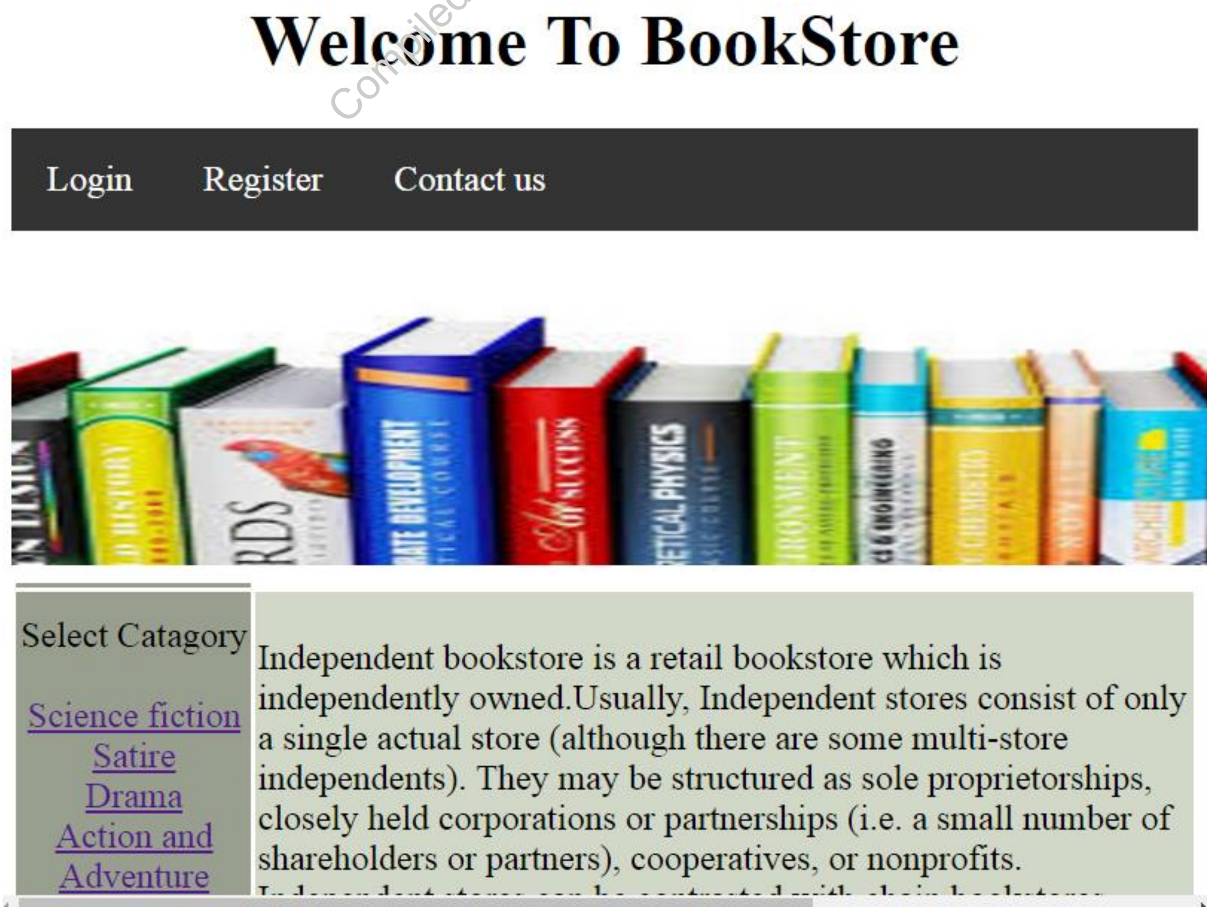
Lab Exercise - 1

Create a web page using below CSS3 properties

1) Create a block and display some contents inside it and set shadow for a box
2) After that display text message and set shadow for that text
3) Create one more block with contents and set border image to it
4) Create one more block with contents and set rounded corner border for this block
5) After that display any specific article in a columns like news-paper style

Lab Exercise -2

Create a demo webpage for 'Online Bookstore ' using HTML5, CSS3 features as shown below



Welcome To BookStore

Login Register Contact us

Select Catagory

Science fiction
Satire
Drama
Action and Adventure

Independent bookstore is a retail bookstore which is independently owned.Usually, Independent stores consist of only a single actual store (although there are some multi-store independents). They may be structured as sole proprietorships, closely held corporations or partnerships (i.e. a small number of shareholders or partners), cooperatives, or nonprofits.

Chapter 3 –Web Programming using JavaScript

Lab Exercise - 1

Write a script to do addition and subtraction of two numbers

Display addition in bold and subtraction in italic format.

Lab Exercise - 2

Write a script to:

1) Accept 3 numbers from user (use *prompt()*)
2) Do addition of these 3 numbers and calculate average of it.

Lab Exercise – 3

Refer Lab exercise -2 of Core CSS , add Login.html and Registration.html page and do the required validation .

Chapter 4- jQUery

Lab Exercise - 1

Create a HTML page as below

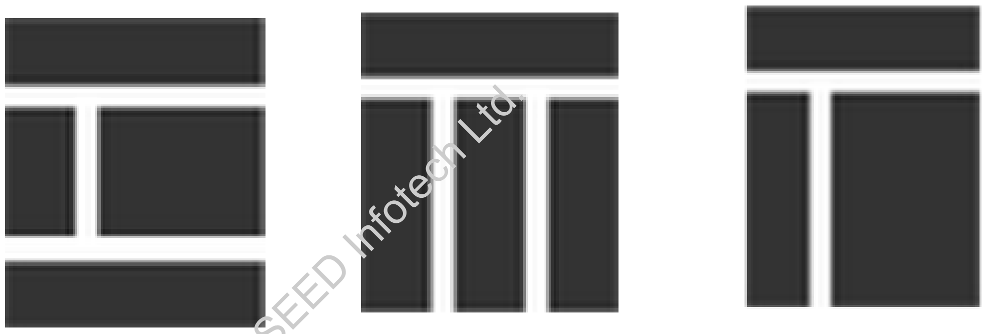
1) Create four text boxes and one paragraph with labels English, Maths, Science, Total Marks and Percentage respectively and a textbox to input marks(fourth text box should be read-only)
2) The form should also include two buttons
3) Following action should occur when the click event occurs
   a. Define a user defined function to get all the values from first three text boxes and calculate total of them and display it in fourth text box and it will be called when button 1 is clicked
   b. Define one more user defined function to get the total marks from first function and then calculate percentage of it and display it in paragraph(which is the fifth element displayed after fourth text box)

Chapter 5 - Bootstrap

Lab Exercise - 1

Design HTML page having following Grid System.

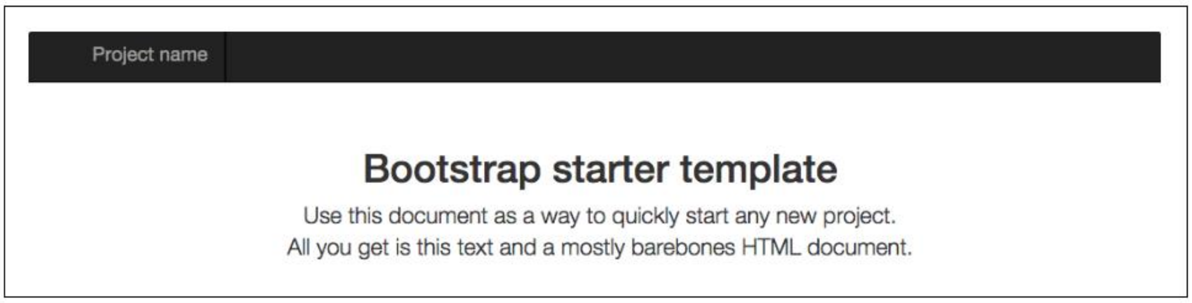
1)



2) Use Lab exercise -2 of Chapter-1 and design Student Registration form using Bootstrap Form elements

3) Use Lab exercise -3 of Chapter -1 and display the table contents by using different Bootstrap table classes .

4) Create a web page as shown below.



Project name

Bootstrap starter template

Use this document as a way to quickly start any new project.
All you get is this text and a mostly barebones HTML document.

Lab Exercise - 2

Use Lab exercise -2 of chapter - 3 and design ‘Online Bookstore’ webpage using Bootstrap.

❑❑❑

SEED Infotech Ltd.
'Panchasheel', 42/16, Erandawana, SEED Infotech Lane,
Off Karve Road, Pune - 411 004 Tel.: 020 25467484 / 92255 20000
E-mail : info@seedinfotech.com

seed®
beyond the obvious

www.seedinfotech.com