MODERN WEB APPLICATIONS (ELECTIVE –II)

B TECH (COMPUTER SCIENCE AND BUSINESS SYSTEMS)
SEM - VI

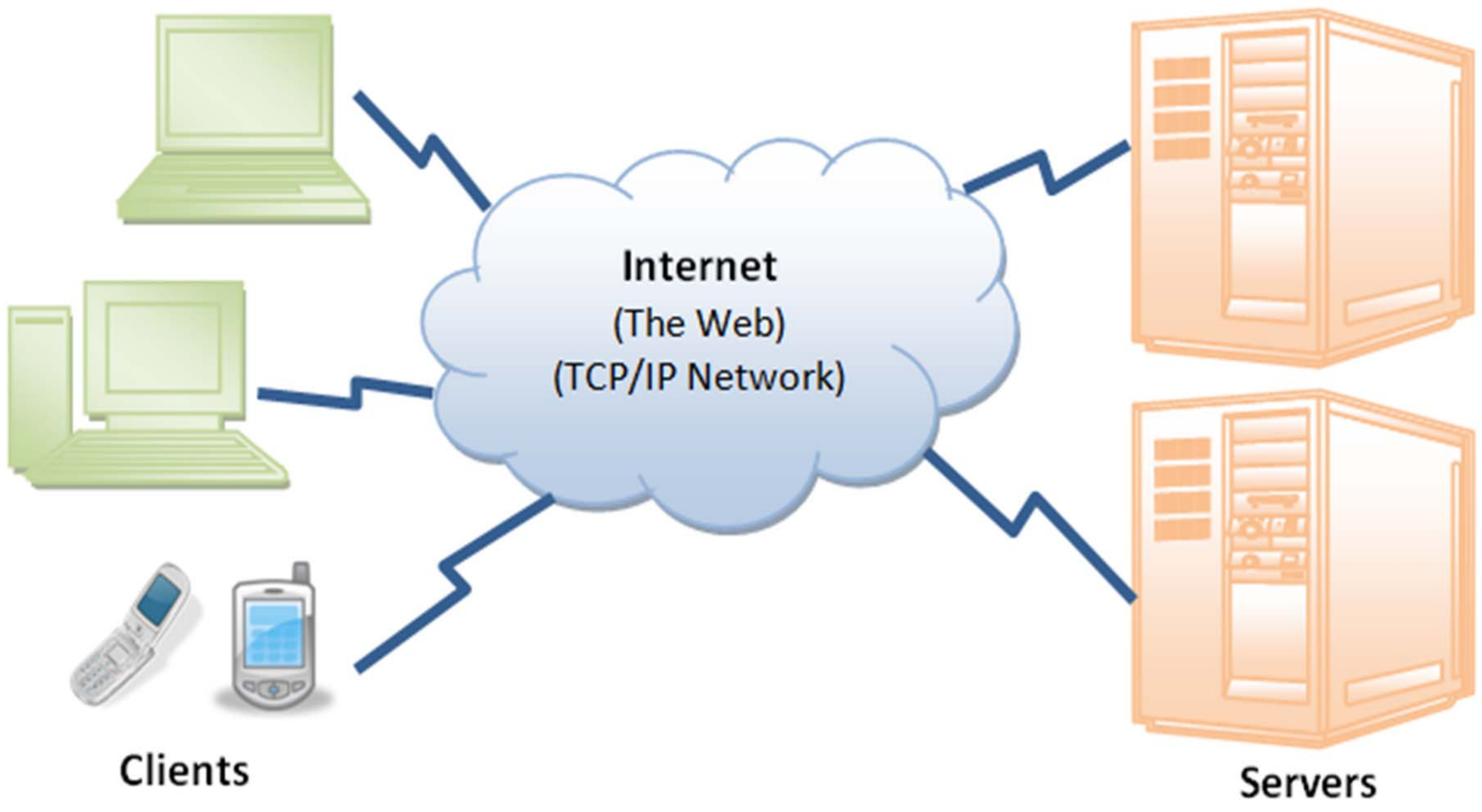
Prerequisite: Fundamentals of Programming and Networking

INTRODUCTION

What is World Wide Web?

- World Wide Web (WWW) - a lot of interconnected computers (via phone lines, cables, or satellites).
- Server Computers
- Client Computers
- Do not confuse with the WWW and Internet.
  - *Internet*: email, telnet, ftp, protocols (communications infrastructure) as well as the Web.
  - *WWW*: Just one component of the Internet.

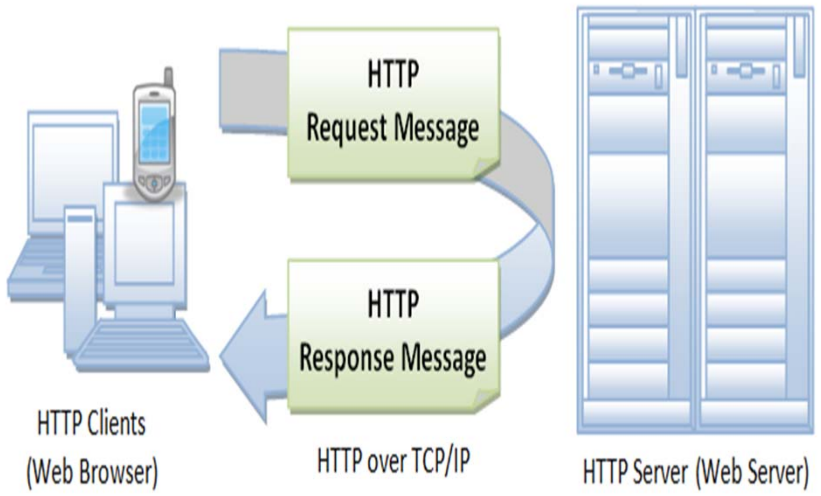
Internet / The Web

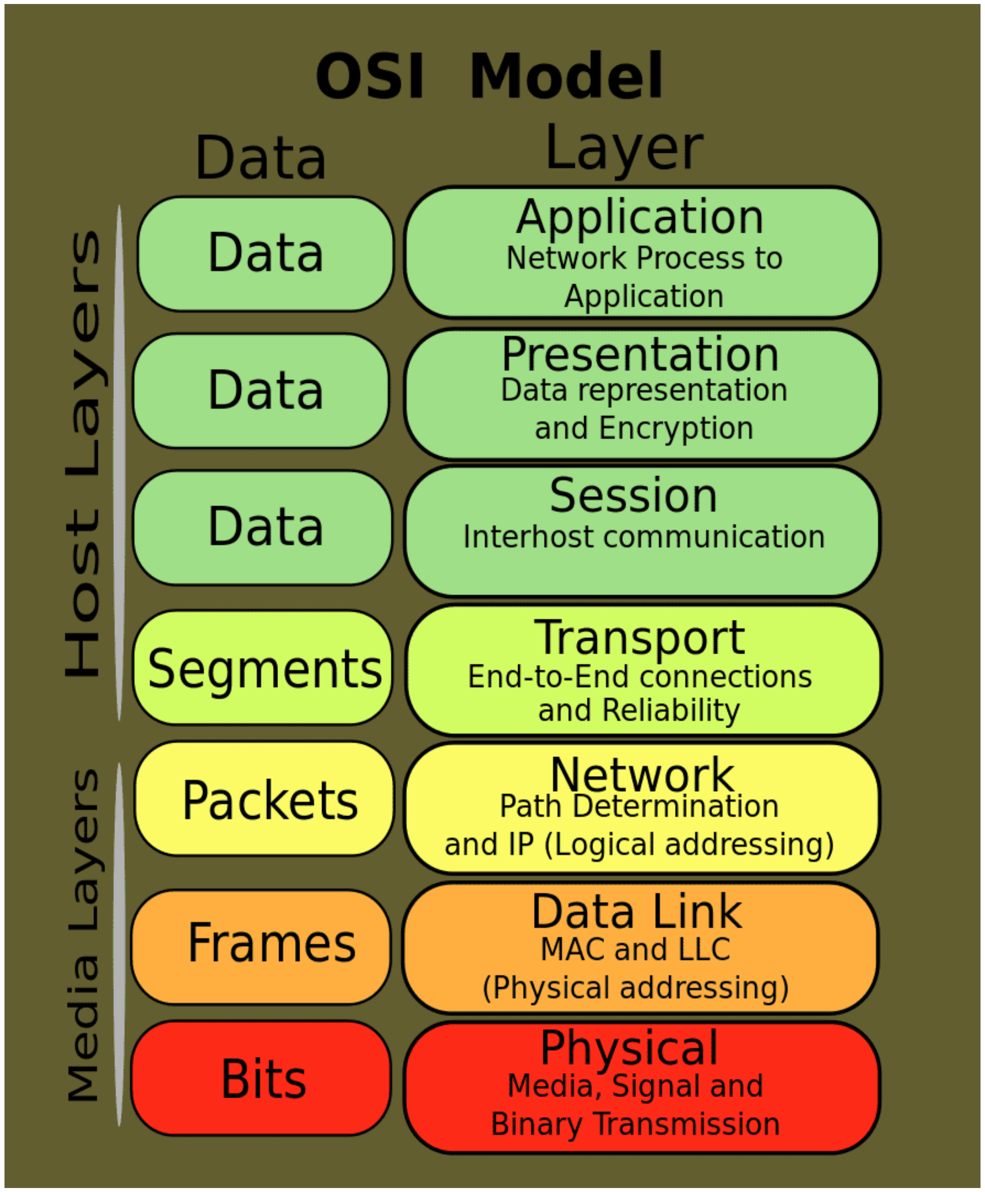


What is HTTP?

- HTTP stands for *Hypertext Transfer Protocol*. This is the protocol being used to transfer hypertext documents that makes the World Wide Web possible.

- A standard web address such as http://yahoo.com is called a *URL* and here the prefix *http* indicates its protocol


HTTP
Request Message
HTTP
Response Message
HTTP Clients
(Web Browser)
HTTP over TCP/IP
HTTP Server (Web Server)


OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data representation and Encryption
Data
Session
Interhost communication
Segments
Transport
End-to-End connections and Reliability
Packets
Network
Path Determination and IP (Logical addressing)
Frames
Data Link
MAC and LLC (Physical addressing)
Bits
Physical
Media, Signal and Binary Transmission

- HTTP is an **asymmetric** request-response client-server protocol.
- An HTTP client sends a request message to an HTTP server. The server, in turn, returns a response message.
- In other words, HTTP is a pull protocol, the client pulls information from the server (instead of server pushes information down to the client).
- HTTP is a stateless protocol. In other words, the current request does not know what has been done in the previous requests.

What is URL?

- URL stands for **U**niform **R**esource **L**ocator
- A URL will have the following format −

  protocol://hostname/other_information

- The protocol specifies how information is transferred from a link.
- Protocols: HTTP, FTP, telnet, newsgroups, and Gopher.
- Two slashes : domain name
- Single forward slashes : directories & sub-directories

What is Website?

- *svmit.ac.in*
- Collection of number of pages written in *HTML*
- Each page is called *web page*
- First page of website is called *home page*

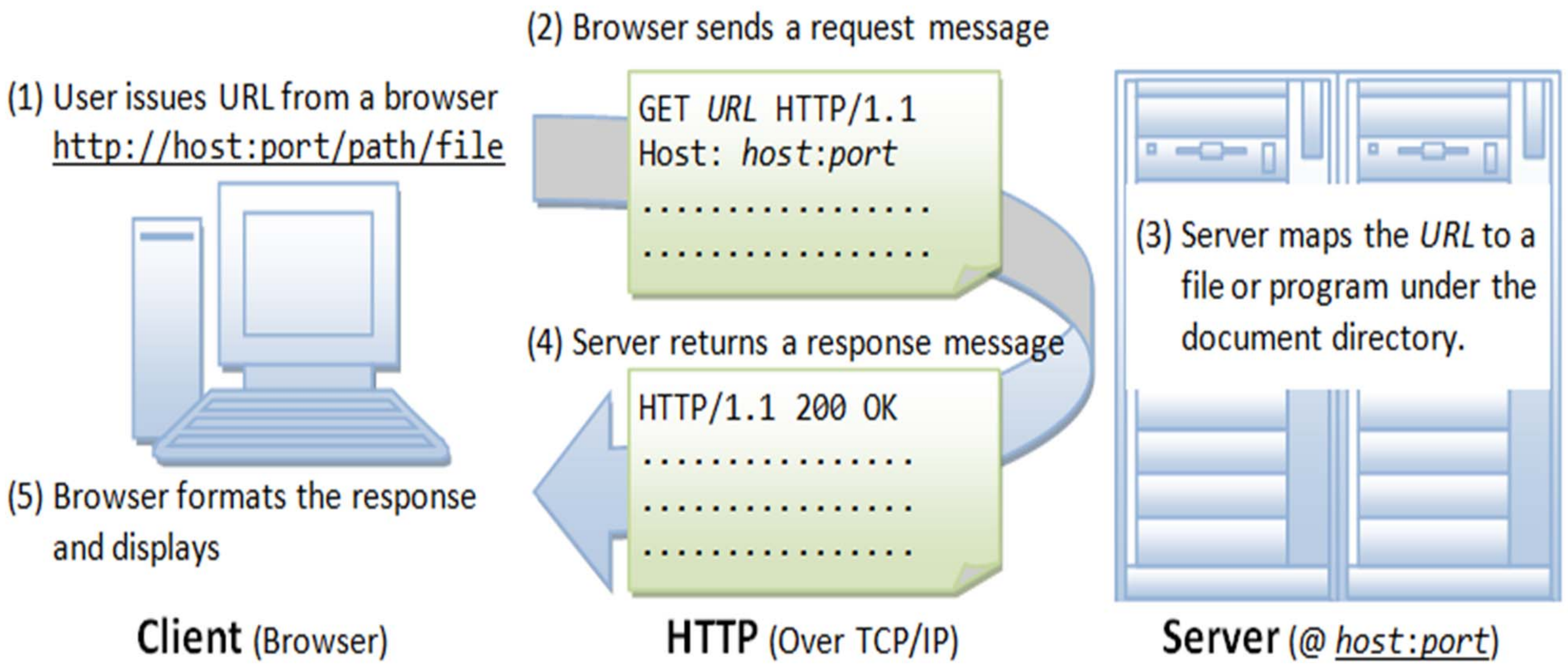
What is Web Server?

- Every website sits on a computer known as a *Web server*.
- This server is *always connected* to the internet.
- Have unique IP address: e.g. *68.178.157.132* or *68.122.35.127*
- When we register a web address, also known as a *domain name*, such as *svmit.ac.in*, we have to specify the IP address of the Web server that will host the site.
- Examples : Apache, IIS, lighttpd, Sun Java, Jigsaw

What is Web Browser?

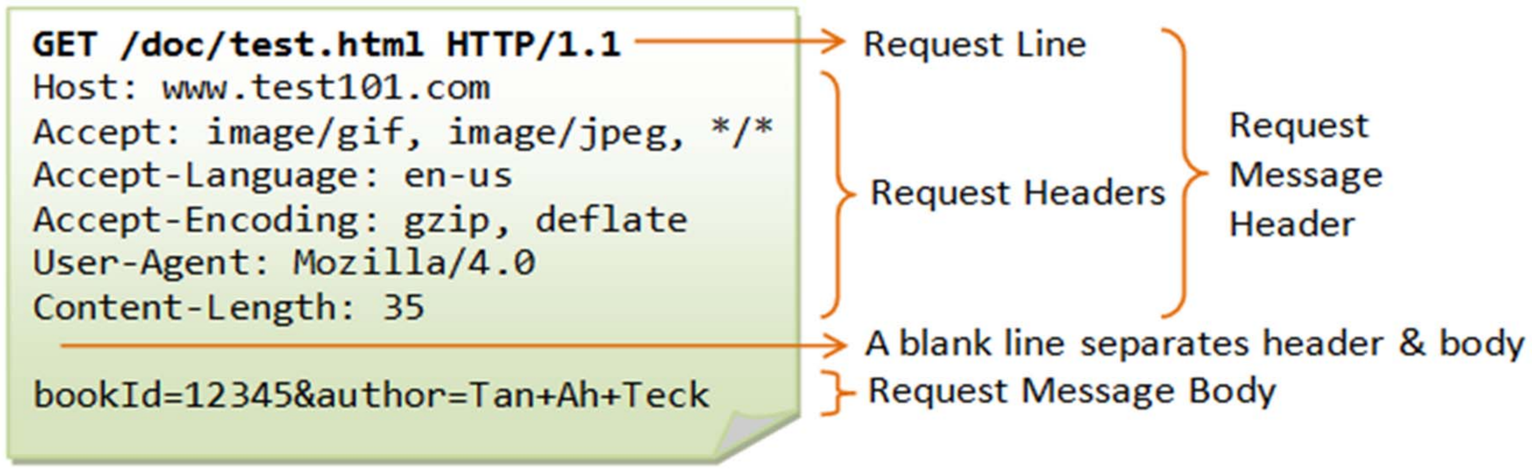
- Web Browsers are software installed on our PC.
- To access the Web you need a web browsers, such as
  Chrome, Netscape Navigator, Microsoft
  Internet Explorer or Mozilla Firefox.
- *browsing* or *surfing*.

HTTP Process


(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET URL HTTP/1.1
Host: host:port
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

browser translated the URL http://www.test101.com/doc/index.html into the following request message

sample HTTP request message

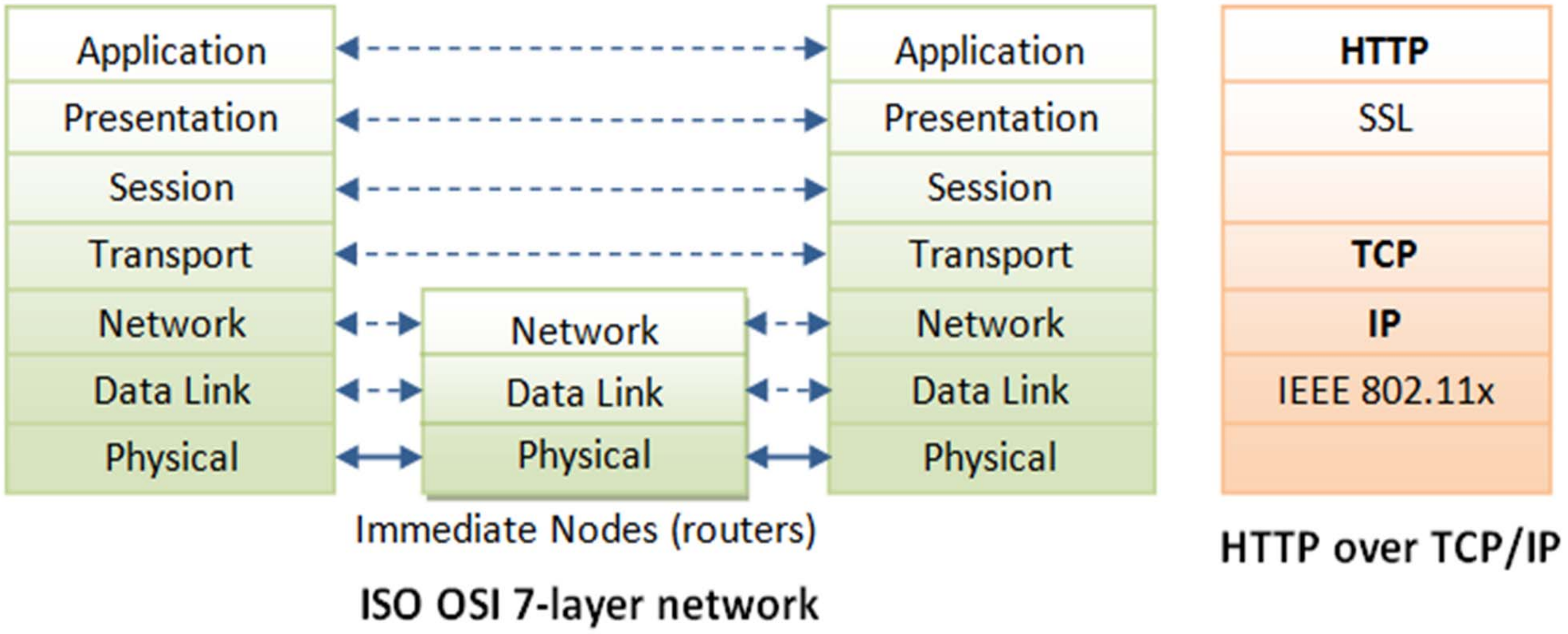


```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

(HTTP response)

```
HTTP/1.1 200 OK
Date: Sun, 18 Oct 2009 08:56:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Sat, 20 Nov 2004 07:16:26 GMT
ETag: "10000000565a5-2c-3e94b66c2e680"
Accept-Ranges: bytes
Content-Length: 44
Connection: close
Content-Type: text/html
X-Pad: avoid browser bug
```

HTTP over TCP/IP



SMTP and ISP

- SMTP : *Simple Mail Transfer Protocol*
- Emails from one server to another server

- ISP : *Internet Service Provider*
- Companies who provide internet service
  (Jio, MTNL, MTS, YOU, Hathway, Tikona)

- Buy space on a Web Server from any Internet Service Provider. This space will be used to host your Website.

Hyperlink and DNS

- *hyperlink* or simply a link is a selectable element in an electronic document that serves as an *access point* to other electronic resources

- DNS stands for *Domain Name System*.
- When someone types, *www.example.com*, browser ask the *Domain Name System* to find the *IP* that hosts the site.

Summary

- A user enters a URL into a browser (for example, Google.com. This request is passed to a domain name server.
- The domain name server returns an IP address for the server that hosts the Website (for example, 68.178.157.132).
- The browser requests the page from the Web server using the IP address specified by the domain name server.
- The Web server returns the page to the IP address specified by the browser requesting the page. The page may also contain links to other files on the same server, such as images, which the browser will also request.
- The browser collects all the information and displays to your computer in the form of Web page.

Web Design Issues

a) Browser & Operating Systems
b) Bandwidth and Cache
c) Display Resolution
d) Look & Feel
e) Page Layout and Linking
f) Locating Information
g) Making Design user-Centric
h) Sitemap

a) Browser & Operating Systems

- Web pages are written using different HTML tags and viewed in browser window.
- The **different browsers** and their **versions** greatly affect the way a page is **rendered**, as different browsers sometimes interpret same HTML tag in a **different way**.
- **Different versions** of **HTML** also support different sets of tags.
- The support for different tags also varies across the different browsers and their versions.
- Same browser may work slightly different on different **operating system** and **hardware platform**.
- To make a web page portable, **test it on different browsers** on different operating systems.

b) Bandwidth and Cache

- Users have different **connection speed**, i.e. bandwidth, to access the Web sites.
- Connection speed plays an important role in designing web pages, if user has low bandwidth connection and a web page contains too many images, it takes more time to download.
- Generally, users have **no patience** to **wait** for longer time than **10-15 seconds** and move to other site without looking at contents of your web page.
- Browser provides temporary memory called **cache** to store the graphics.
- When user gives the URL of the web page for the first time, HTML file together with all the graphics files referred in a page is downloaded and displayed.

c) Display Resolution

- Display resolution is another important factor affecting the Web page design, as we do not have any control on display resolution of the monitors on which user views our pages.
- Display or screen resolution is measured in terms of **pixels** and common resolutions are 800 X 600 and 1024 X 786.
- We have three choices for Web page design.
  - Design a web page with **fixed resolution**.
  - Make a **flexible design** using HTML table to fit into different resolution.
  - If the page is displayed on a monitor with a higher resolution, the page is displayed on left hand side and some part on the right-hand side remains blank. We can use centered design to display page properly.
  - Ideally we should use some frameworks for designing like **Bootstrap/Material design**.

d) Look & Feel

- Look and feel of the website decides the overall appearance of the website.
- It includes all the design aspects such as
  - Web site theme
  - Web typography
  - Graphics
  - Visual structure
  - Navigation etc…

e) Page Layout and Linking

- Website contains of **individual web pages** that are linked together using various **navigational links.**
- Page layout defines the visual structure of the page and divides the page area into different parts to present the information of varying importance.
- Page layout allows the designer to distribute the contents on a page such that visitor can view it easily and find necessary details.

f) Locating Information

- Webpage is viewed on a computer screen and the screen can be divided into **five major areas** such as **center**, **top**, **right**, **bottom** and **left** in this particular order.
- The first major area of importance in terms of users viewing pattern is the center, then top, right, bottom and left in this particular order

g) Making Design user-Centric

- It is very difficult for any Web designer to predict the exact behavior of the Web site users.
- However, idea of general behavior of common user helps in making design of the Web site user centric.
- Users either scan the information on the web page to find the section of their interest or read the information to get details.

h) Sitemap

- A **Sitemap** is a model of a website's content designed to help both users and search engines navigate the site.
- Many a times Web sites are too complex as there are a large number of sections and each section contains many pages.
- It becomes difficult for visitors to quickly move from one part to other.
- Once the user selects a particular section and pages in that section, user gets confused about where he/she is and where to go from there.
- To make it simple, keep your hierarchy of information to few levels or provide the navigation bar on each page to jump directly to a particular section.

Planning a Website

- The most important activity in a website development is **planning**.
- To achieve **higher success** of the website in terms of user satisfaction, better planning is needed.
- Before we start developing a website, we should ask question such as
    - **Why** are we developing this website?
    - **What** do we achieve by developing this website?
    - **Who** are the people who will use this website?
    - What are the information **contents**?
    - How are these contents **organized**? What are the **possible ways**?

Effective Navigation

- The most important design element in the web design after page layout is navigation design.
- Navigation means the **ways to move from one page to another page** in a Web site using hyperlinks provided on the page.
- If navigation design is not proper then user feels the problem in moving around the pages in your site in a desired manner or gets confused and leaves the site.

Tips for Effective Navigation

- Navigation links are either **text based**, i.e. a word or a phrase is used as a link, or **graphical**, i.e. a image, a icon or a logo is used as a link.
- Navigation links should be **clear** and **meaningful**.
- It should be **consistent**.
- Link should be **understandable**.
- Organize the links such that contents are **grouped** logically.
- Provide **search** link, if necessary, usually on top of the page.
- Use **common** links such as ‘about us’ or ‘Contact us’.
- Provide the way to **return** to **first page**.
- Provide the user with **information** regarding **location**
- Horizontal navigation bar can be provided on each page to **directly jump** to any section

Basic HTML tags

Structural Tags

<HTML>

These tags enclose the entire Web page document.

</HTML>

<HEAD>

These tags enclose the Head part of the document

</HEAD>

<TITLE>

These tags enclose the title of the document. This text appears in the title bar in the browser and on the bookmark list if someone bookmarks your web page.

</TITLE>

Sample Structure of a Web Site

```
<HTML>
    <HEAD>
        <TITLE> SVMIT Bharuch</TITLE>
    </HEAD>

    <BODY>
        This is SVMIT Webpage!
    </BODY>
</HTML>
```

Header Tags

Header Tags -- Used for marking sections and subsections in a document.

<H1>Header 1 -- Giant-sized and bold </H1>
<H2>Header 2 -- Large and bold </H2>
<H3>Header 3 -- Normal-sized and bold </H3>
<H4>Header 4 -- Small and bold </H4>
<H5>Header 5 -- Very Small and bold </H5>
<H6>Header 6 -- Tiny and bold </H6>

Header Tags (cont.)

H1 = Giant-sized and bold

H2 = Large and bold

H3 = Normal-sized and bold

H4 = Small and bold

H5 = Very Small and bold

H6 = Tiny and bold

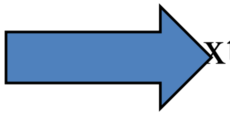
Breaking Lines and Paragraphs

- <P> text </P>
  - Paragraph tag
  - Most browsers render (process) this with blank lines between each paragraph
- <BR>
  - Line break tag
  - Used when the webmaster wants a carriage return but doesn't want a blank line to follow

---

Example:

```
<p>text a</p>
<p>text b</p>
<br>text c
<br>text d
```



```
text a

text b

text c
text d
```

Horizontal Rule

The <HR> tag puts a graphical line across the page.

Ex:

Horizontal Rule Attributes:

NOSHADE -- A solid line with no shading

WIDTH="xx%/xx" -- Controls the width of the line. You may specify either percentage of the width of a page or actual pixel length

SIZE="xx" -- Controls the height of the line. You need to specify the dimension in pixels.

ALIGN="left/center/right" -- This allows the line to be aligned to the left, right, or center of the page

Text Formatting Tags

Some basic text formatting styles:

Tag	**Result**
<I> Italics </I>	*Italics*
<B> Bold </B>	**Bold**
<PRE> Preformatted Text </PRE>	`Preformatted Text`
<STRONG> Strong </STRONG>	**Strong**
<ADDRESS> Address </ADDRESS>	*Address*
<CITE> Citations </CITE>	*Citations*
<CODE> Source Code </CODE>	`Source Code`

Font modifications

Web creators can also change the way text looks by using the <FONT> tag

SIZE="number" - changes size of the font; 1=smallest, 7 = largest

```
<FONT SIZE="7">Big</FONT> <FONT SIZE="1">Small</FONT>
```

Big Small

COLOR="color-name" - changes text color

```
<FONT COLOR="red">This is red</FONT>
```

This is red

FACE="font-name" - changes font

```
<FONT FACE="verdana">This is the verdana font;</FONT> <FONT FACE="chicago">this is the chicago font.</FONT>
```

This is the verdana font; this is chicago font.

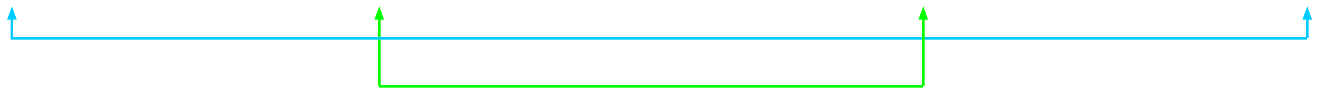
<Font> modifications (cont.)

One can combine font modifications:

```
<FONT SIZE="7" FACE="courier" COLOR="red">Big, Courier & Red</FONT>
```

Big, Courier & Red

```
<FONT SIZE="7"><FONT FACE="courier">Big & Courier</FONT> - Just Big</FONT>
```



Big & Courier - Just Big

Lists -- Unordered Lists

Unordered lists:

```
<UL>
    <LI>Item One
    <LI>Item Two
    <LI>Item Three
    <LI>Item Four
</UL>
```

→

- Item One
- Item Two
- Item Three
- Item Four

Unordered List Attributes:

type="disc/circle/square"

• Disc (default) ○ Circle ■ Square

Lists -- Ordered Lists

Ordered (Numbered) Lists:

```
<OL>
        <LI> Item One
        <LI> Item Two
        <LI> Item Three
        <LI> Item Four
</OL>
```

1. Item One
2. Item Two
3. Item Three
4. Item Four

***Ordered List Attributes*:**

type="i/I/a/A/1" (default)

i =	i. Item One	I =	I. Item One	a =	a. Item One	A =	A. Item One	1 =	1.Item One
	ii. Item Two		II. Item Two		b. Item Two		B. Item Two		2. Item Two
	iii. Item Three		III. Item Three		c. Item Three		C. Item Three		3. Item Three
	iv. Item Four		IV. Item Four		d. Item Four		D. Item Four		4. Item Four

start="xx"

Lists -- Definition Lists

Definition Lists:

```
<DL>
      <DT>List Name One
            <DD>This is where information about List Name One would go</DD>
      </DT>
      <DT>List Name Two
            <DD>This is where information about List Name Two would go</DD>
      </DT>
</DL>
```

⬇

List Name One

    This is where information about List Name One would go

List Name Two

    This is where information about List Name Two would go

Links

The anchor tag <A> is used to link one document to another or from one part of a document to another part of the same document.

Basic Links:
```
<A HREF="http://www.stanford.edu/">Stanford University</A>
```

Inter-document Links:
```
<A HREF="#spot">Point to 'spot' in this document</A>
```

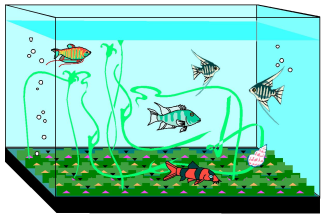
Defining a point in a document:
```
<A NAME="spot">Spot</A>
```

Email links:
```
<A HREF="mailto:someone@somehost.com">Email someone@somehost.com</A>
```

Graphics

To have a graphic appear on a webpage, web designers must put the <IMG> tag in with the address where the graphic "lives":

<IMG SRC="http://www.someplace.com/images/fish.gif">



Graphics attributes:

alt="text": insert a description of the graphic for those who are using browsers that cannot process images (e.g., page readers for the blind)

width="xx/xx%": width in pixels/percentage

height="xx/xx%": height in pixels/percentage

border="xx": pixel length of the border surrounding the image.

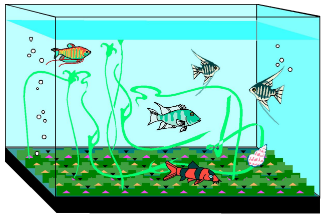
hspace="xx": places a buffer of space horizontally around the image

vspace="xx": places a buffer of space vertically around the image

align="top/middle/bottom/right/left": aligns image in relation to the text (see next 2 slides)

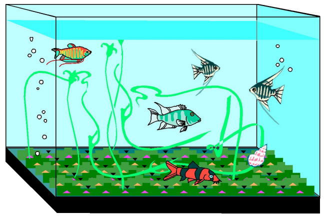
Graphics (cont.)

```
<img src="http://www.someplace.com/images/fish.gif" align="top">All about Fish
```



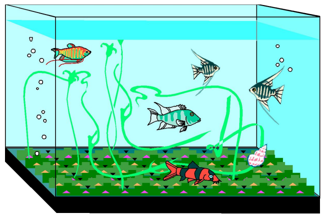
All about Fish

```
<img src="http://www.someplace.com/images/fish.gif" align="middle">All about Fish
```



All about Fish

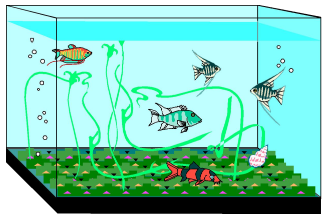
```
<img src="http://www.someplace.com/images/fish.gif" align="bottom">All about Fish
```



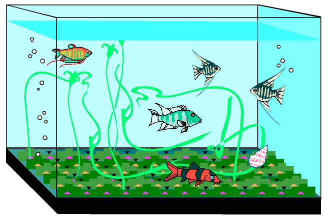
All about Fish

Graphics (cont.)

```
<img src="http://www.someplace.com/images/fish.gif" align="left">
```



```
<img src="http://www.someplace.com/images/fish.gif" align="right">
```

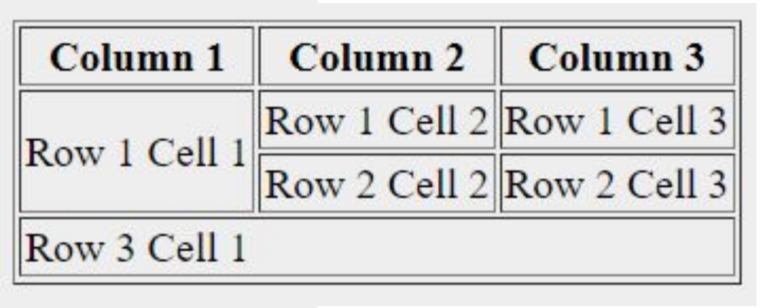


HTML Tables

```
<table>
    <tr>
            <th>Company</th>
            <th>Contact</th>
            <th>Country</th>
   </tr>
    <tr>
           <td>Google</td>
           <td>Sundar Pichai</td>
           <td>USA</td>
    </tr>
   <tr>
         <td>Microsoft</td>
         <td>Satya Nadella</td>
         <td>USA</td>
     </tr>
</table>
```

Table: border, rowspan, colspan

```
<body>
<table border = "1">
  <tr>
    <th>Column 1</th>
    <th>Column 2</th>
    <th>Column 3</th>
  </tr>
<tr>
   <td rowspan = "2">Row 1 Cell 1</td>
   <td>Row 1 Cell 2</td>
   <td>Row 1 Cell 3</td>
</tr>
<tr>
   <td>Row 2 Cell 2</td>
   <td>Row 2 Cell 3</td>
</tr>
<tr>
   <td colspan = "3">Row 3 Cell 1</td>
</tr>
</table>
</body>
```



Column 1	Column 2	Column 3
Row 1 Cell 1	Row 1 Cell 2	Row 1 Cell 3
	Row 2 Cell 2	Row 2 Cell 3
Row 3 Cell 1		

OUTPUT ?

ALL HTML Tags

https://way2tutorial.com/html/tag/index.php

HTML Forms

Parts of a Web Form

- A Form is an area that can contain Form Control/Elements.
- Each piece of information for a form is stored in a Field.
- The value itself is called Field Value.

Parts of a Web Form

- Users enter or select a field using Form Control/ Elements.
  - Form/Control elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
  - A form usually contains a Submit button to send the information in the form elements to the server

Control Elements

- Input Boxes – for text and numerical entries
- Option Buttons, also called Radio Buttons – for selecting a single option from a predefined list
- Selection Lists – for long lists of options, usually appearing in a Drop-Down List Box
- Check Boxes – for specifying yes or no
- Text Areas – for extended entries that can include several lines of text

HTML Forms

- The basic construction of a HTML form is this...

  <form> **-** begin a form

  <input> **-** ask for information in one of several different ways

  <input> **-** there can be as many input areas as you wish

  </form> **-** end a form HTML form

Forms and Server-Based Programs

- Forms are used to collect information.
- The information is then sent back to the server.
- Information is stored and analyzed using a program on the server.
- By giving users access to programs that react to user input, the Web became a more dynamic environment where companies and users could interact.

Forms and Server-Based Programs

- Server-Based programs provide:
  - Online databases containing customer information
  - Online catalogs for ordering and purchasing merchandise
  - Dynamic Web sites with content that is constantly modified and updated
  - Message boards for hosting online discussion forums

Forms and Server-Based Programs

- Because these programs run on Web servers, rather than locally, users might not have permission to create or edit them. Instead, users will receive information about how to interact with the programs on the server.

- Several reason to restrict direct access:
  - When you run a server-based program, you are interacting directly with the server
    - Security risks (computer hackers)
    - Drain on system resources caused by large number of programs running simultaneously

Forms and Server-Based Programs

- Server-Based Programs
  - Common Gateway Interface (CGI) Scripts
    - Most common
  - ASP
  - Cold Fusion
  - C/C++
  - PHP
  - VBScript
- The Web server determines which language your Web form will use.

Getting Started

- The first step in creating a form is to specify the name and location of the CGI script that will be used to process the form data. To do this, type the following code within your HTML file, and note that there are no spaces:
- <form METHOD="Post" ACTION=http://www.temple.edu/cgi-in/mail?*your-e-mail-address*@temple.edu>
- For example, if your e-mail address is jsmith@temple.edu, you would enter:
- <form METHOD="Post" ACTION="http://www.temple.edu/cgi-bin/mail?jsmith@temple.edu">

Text Input (type="text")

- **A Text Field:**
  - used to create one line fields that viewers can type text. The **default width is 20 characters**, and you can create fields of other sizes by the value in the size option. You can limit the number of characters by the value of the **MAXLENGTH option**. Text input fields will be empty when the page loads, unless you provide an initial text string for its VALUE option
  - **<input type="text" name="textfield" size="value" value="with an initial value">**

Text Input (type=“text”)

- **Example 1: A text field named "text1" that is 30 characters wide.**
- **<input type="text" name="text1" size="30">**

- **Example 2: A text field named "text2" that is 30 characters wide but will only accept 20 characters.**
- **<input type="text" name="text2" size="30" maxlength="20">**

- **Example 3: A text field named "text3" that is 40 characters wide with default value.**
- **<input type="text" name="text3" size="40" value="We are not alone">**

Password Input (type=“password”)

- are exactly the same as text input elements, except that when the viewer types in, they see "bullet" characters rather then the letter they are typing. Password text is scrambled during transmission and then unscramble when the form data is received at the server end.

- **Example 4: A password field named "pass1" that is 30 characters wide**
- **<input type="password" name="pass1" size="30">**

- **Example 5: A password field named "pass2" that is 30 characters wide but will only accept 20 characters**
- **<input type="password" name="pass2" size="30" maxlength="20">**

Text Input (type=“textarea”)

- Text fields that have more than one line and can scroll as the viewer enters more text. The tag options define the size of the field by the number of rows and character columns. By adding the option **WRAP=VIRTUAL**, the text entered will automatically wrap at the right hand side of the field. You can also include default text to appear in the field

- **Example 6: A textarea field named "comments" that is 45 characters wide and 6 lines high**
- **<textarea name="comments" rows="6" cols="45" wrap="virtual"> The first time I ever saw a web page, I thought.... (continue) </textarea>**

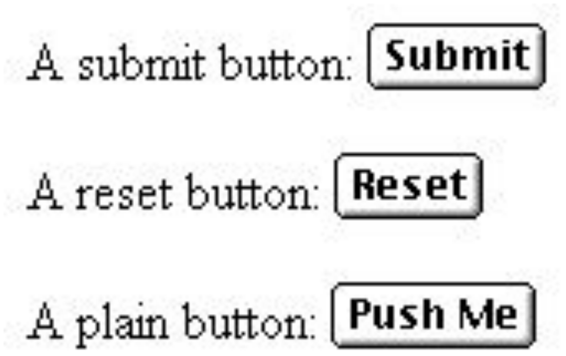
Adding Control Buttons

- A form must include at least one control button for submitting the information once it is filled out. In addition, forms often include a button for resetting all the entries if a person wants to start over.

- When a person presses the submit button, he or she will receive confirmation that the form results were sent to your e-mail address. You will then see an e-mail message in your Inbox with the subject *FORM results*.

Adding Control Buttons

- A submit button:
  `<input type="submit" name="Submit" value="Submit">`
- A reset button:
  `<input type="reset" name="Submit2" value="Reset">`



- submit: send data
- reset: restore all form elements to their initial state

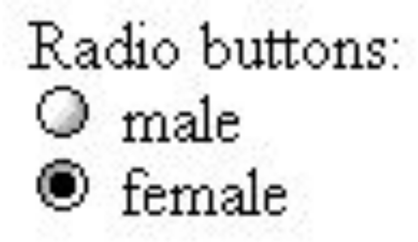
- Note that the type is input, not “button”

Radio buttons (type=“radio”)

- Are sets of controls that are linked so that only one radio button among each set is selected at a time
- If you click one radio button, the others in the set are automatically de-selected
- A set of radio buttons is defined by providing them the <u>same</u> name
- The value sent in the web form is the value of the radio button that was last selected
- Adding the option CHECKED to one of the buttons in a set will make that button highlighted when the page loads
- Radio buttons do not contain any text

Radio buttons (type="radio")

```
Radio buttons:<br>
    <input type="radio" name="radiobutton" value="myValue1">
    male<br>
    <input type="radio" name="radiobutton" value="myValue2" checked>
    female
```



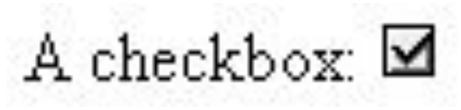
Checkboxes (type="checkbox")

- Are similar to radio buttons, but are not affected by other buttons, so you can have more than one in a group checked at a time
- Note that every checkbox has a unique name. If there is no check in the box, clicking it will place an X or a check mark there
- If the box is checked, clicking it again will remove the mark. The value sent in the web form is the value of the checkbox if it was selected; otherwise the value will be empty
- Adding the option **CHECKED** to a checkbox will make that checkbox highlighted when the page loads.

Checkboxes (type="checkbox")

- A checkbox:

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

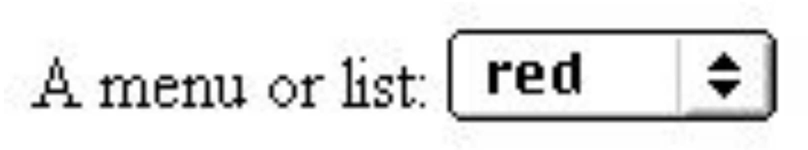

A checkbox: ☑

- type: “checkbox”
- name: used to reference this form element from JavaScript
- value: value to be returned when element is checked
- Note that there is *no text* associated with the checkbox—you have to supply text in the surrounding HTML

Drop-down menu or list

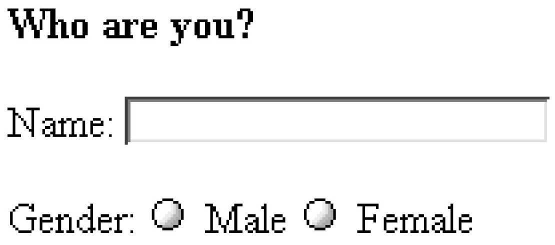
- A menu or list:

```
<select name="select">
    <option value="red">red</option>
    <option value="green">green</option>
    <option value=“blue">blue</option>
</select>
```



- Additional arguments:
  - size: the number of items visible in the list (default is "1")
  - multiple: if set to "true", any number of items may be selected (default is "false")

Practice Exercise

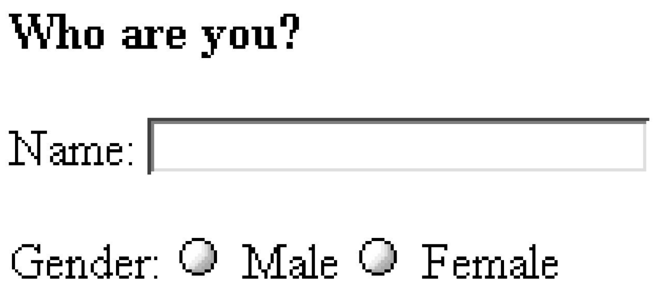

Who are you?

Name:

Gender: Male Female

A complete example

```
<html>
<head>
<title>Get Identity</title>
</head>
<body>
<p><b>Who are you?</b></p>
<form method="post" action="">
  <p>Name:
    <input type="text" name="textfield">
  </p>
  <p>Gender:
    <input type="radio" name="gender" val
    <input type="radio" name="gender" value="f">Female</p>
  </form>
</body>
</html>
```



Who are you?

Name:

Gender: ○ Male ○ Female

GET Method

```
<?php
  if( $_GET["name"] || $_GET["age"] ) {
    echo "Welcome ". $_GET['name']. "<br />";
    echo "You are ". $_GET['age']. " years old.";
    exit();
  }
?>

<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "GET">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>
  </body>
</html>
```

POST Method

```
<?php
  if( $_POST["name"] || $_POST["age"] ) {
    echo "Welcome ". $_POST['name']. "<br />";
    echo "You are ". $_POST['age']. " years old.";
    exit();
  }
?>

<html>
  <body>
    <form action = "<?php $_PHP_SELF ?>" method = "POST">
      Name: <input type = "text" name = "name" />
      Age: <input type = "text" name = "age" />
      <input type = "submit" />
    </form>
  </body>
</html>
```

GET and POST method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

- The GET method produces a long string that appears in your server logs, in the browser's Location: box.
- The GET method is restricted to send upto 1024 characters only.
- Never use GET method if you have password or other sensitive information to be sent to the server.
- GET can't be used to send binary data, like images or word documents, to the server.
- The data sent by GET method can be accessed using QUERY_STRING environment variable.
- The PHP provides **$_GET** associative array to access all the sent information using GET method.

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

- The POST method does not have any restriction on data size to be sent.
- The POST method can be used to send ASCII as well as binary data.
- The data sent by POST method goes through HTTP header so security depends on HTTP protocol. By using Secure HTTP you can make sure that your information is secure.
- The PHP provides **$_POST** associative array to access all the sent information using POST method.

HTML Frames

Frames

- HTML frames are used to divide your browser window into multiple sections
- Each section can load a separate HTML document.
- A collection of frames in the browser window is known as a frameset.
- The window is divided into frames in a similar way the tables are organized: into rows and columns.

Disadvantages of Frames

- Some smaller devices cannot cope with frames - size
- Sometimes your page will be displayed differently - resolution
- The browser's *back* button might not work
- There are still few browsers that do not support

<frameset> Attributes

- cols : specify how many columns
- rows : specify how many rows
- border : the border of each frame
- frameborder :
- framespacing :

<frame> Attributes

- src : Source
- Name : location name
- frameborder : overrides frameset, frameborder="0"
- marginwidth: marginwidth = "100"
- marginheight: marginheight = "100"
- noresize : noresize = "noresize"
- scrolling : scrolling = "yes"
- longdesc: longdesc = "framedescription.html"

navigation bars

- Test.html
- Menu.html
- Main.html

CSS

Cascading Style Sheets

Introduction

- **CSS** stands for **C**ascading **S**tyle **S**heets
- CSS defines how HTML elements are to be displayed
- CSS saves a lot of work
- External Style Sheets are stored in CSS files

What CSS does?

- CSS performs the following functions:

  1. Controls the text color of any element.
  2. Controls the background colors.
  3. Controls the border around elements.
  4. Spacing between elements and borders.
  5. Text manipulation and decoration.

Example

- HTML
  - <h1>This is a heading</h1>
  - <p>This is a paragraph.</p>
- CSS
  - h1 { color:blue; font-size 12px}
  - p {color:red; text-align:center;}

CSS Selectors

- CSS selectors allow you to select and manipulate HTML elements.

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

- There are two types of CSS selectors.
  - Element selector
  - id selector
  - class Selector

1. CSS element Selector

- The element selector selects elements based on the element name.
- You can select all <p> elements on a page like this: (all <p> elements will be center-aligned, with a red text color)

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

2. CSS ID Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- An id should be unique within a page, so the id selector is used if you want to select a single, unique element.

```
<p id="para1">Hello World!</p>
```

Example

```
<!DOCTYPE html>
<html> <head>
<style>
#para1 {
   text-align: center;
   color: red;
    font-size:20px;
}
h3{  color: blue; }
</style>
</head>
<body>
  <p id="para1">Hello World!</p>
  <p>This paragraph is not affected by the
    style.</p>
  <h3> this is heading 3 line one..</h3>
  <h3> this is heading 3 line two..</h3>
  <h3> this is heading 3 line three..</h3>
</body>
</html>
```

Hello World!

This paragraph is not affected by the style.

this is heading 3 line one..

this is heading 3 line two..

this is heading 3 line three..

The class Selector

- The class selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period character, followed by the name of the class:

Example

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

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

</body>
</html>
```

Red and center-aligned heading

Red and center-aligned paragraph.

Specific Element class selector

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

<h1 class="center">This heading will not be
      affected</h1>
<p class="center">This paragraph will be red
      and center-aligned.</p>

</body>
</html>
```

This heading will not be affected

This paragraph will be red and center-aligned.

Grouping the selectors

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

h2 {
    text-align: center;
    color: red;
}

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

```
h1, h2, p {

    text-align: center;

    color: red;

}
```

Border Style

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

CSS Margins

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

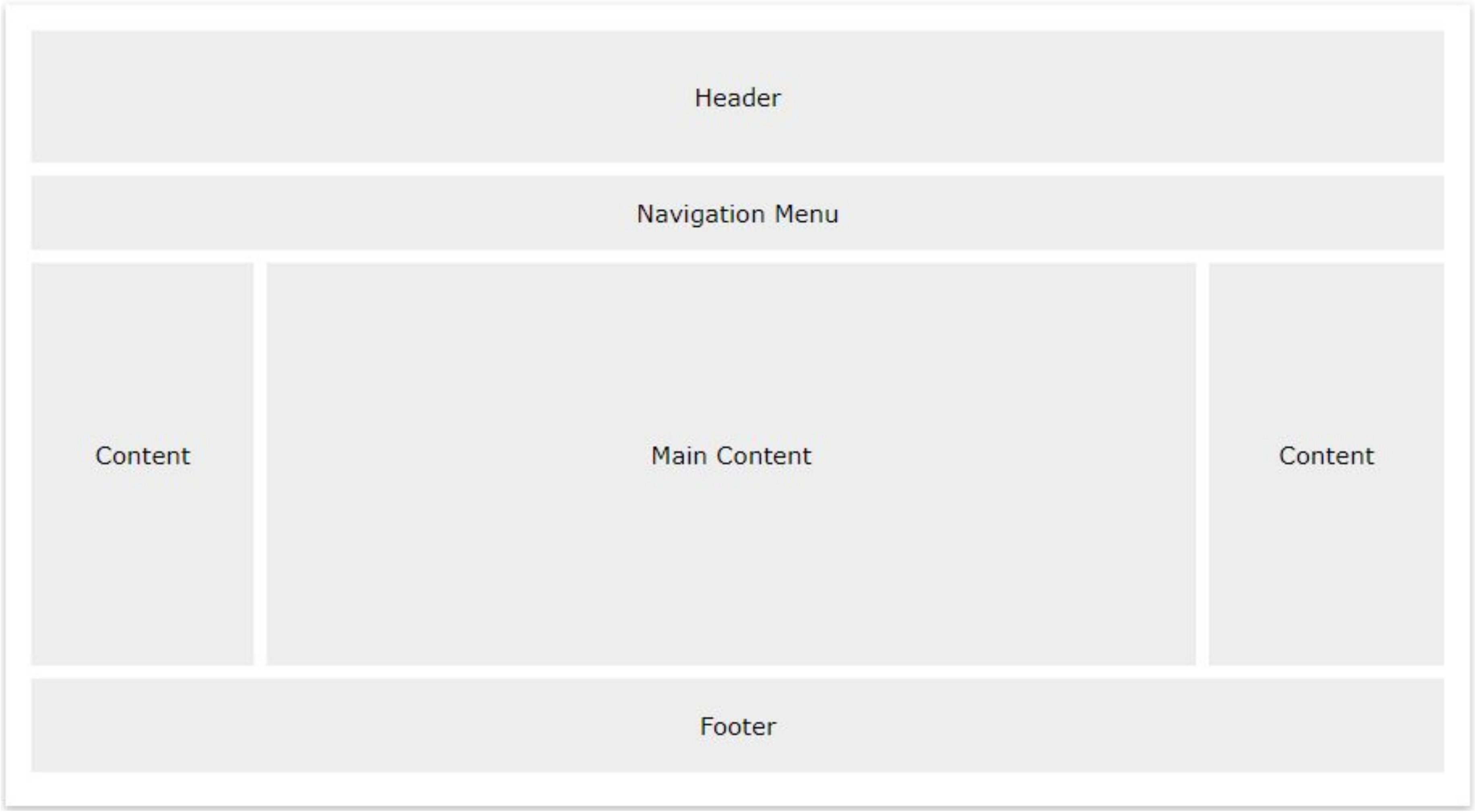
CSS Lists

```
ul.a {
  list-style-type: circle;
}
ul.b {
  list-style-type: square;
}
ol.c {
  list-style-type: upper-roman;
}
ol.d {
  list-style-type: lower-alpha;
}
```

CSS Animation

```
<!DOCTYPE html>
<html>
<head>
<style>
#myDIV {
  width: 300px;
  height: 200px;
  background: red;
  animation: mymove 5s infinite;
}
/* Standard syntax */
@keyframes mymove {
  from {background-color: red;}
  to {background-color: blue;}
}
```

CSS Layout (CSS6.HTML)



Way to add CSS

- There are three ways of inserting a style sheet:

1. External style sheet

2. Internal style sheet

3. Inline style

External StyleSheet

- 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 just one file.

- Each page must include a link to the style sheet with the <**link**> tag. The <link> tag goes inside the head section.

- An external style sheet can be written in any text editor. The file should not contain any **html** tags.

- The style sheet file must be saved with a **.css** extension.

Example: External StyleSheet

SAMPLE.HTML

```
<!DOCTYPE html>
<html>

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

<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

MYSTYLE.CSS

```
body{
        background-color:rgb(196, 230, 173);
}

h1{
        color:navy;
        margin-left:20px;
}

p{
        color:#5E21DD;
}
```

Internal Style Sheet

- An internal style sheet should be used when a single document has a unique style.

- You define internal styles in the head section of an HTML page, inside the <style> tag.

Example

```
<head>
    <style>
    body {
        background-color: linen;
    }
    h1 {
        color: maroon;
        margin-left: 40px;
    }
    </style>
</head>
```

Inline Styles

- An inline style loses many of the advantages of a style sheet (by mixing content with presentation).

- To use inline styles, add the style attribute to the relevant tag. The style attribute can contain any CSS property.

- **<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>**

CSS Background

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

- CSS properties used for background effects:
  1. background-color
  2. background-image
  3. background-repeat
  4. background-attachment
  5. background-position

Background Color

- The background-color property specifies the background color of an element.

- body {
      background-color: #ff0000;
  }

- With CSS, a color is most often specified by:
  - a HEX value - like "#ff0000"
  - an RGB value - like "rgb(255,0,0)"
  - a color name - like "red"

Background Image

- The background-image property specifies an image to use as the background of an element.
- By default, the image is repeated so it covers the entire element.

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

CSS Text

- **Text Color**
- The color property is used to set the color of the text.

- body {
      color: blue;
  }

- h1 {
      color: rgb(255,0,0);
  }

Text Alignment

- The text-align property is used to set the horizontal alignment of a text.
- Text can be centered, or aligned to the left or right, or justified.

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

  p.date {
     text-align: right;
  }

  p.main {
     text-align: justify;
  }
  ```

Text Decoration

- The text-decoration property is used to set or remove decorations from text.

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

  h2 {
      text-decoration: line-through;
  }

  h3 {
      text-decoration: underline;
  }
  ```

Text Transformation

- The text-transform property is used to specify uppercase and lowercase letters in a text.

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

  p.lowercase {
      text-transform: lowercase;
  }

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

Text Indentation

- the text-indent property is used to specify the indentation of the first line of a text.

-
  ```
  p {
          text-indent: 50px;
  }
  ```

CSS Font

- CSS font properties define the font family, boldness, size, and the style of a text.

- CSS Font Families
- In CSS, there are two types of font family names:

1. **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
2. **font family** - a specific font family (like "Times New Roman" or "Arial")

Font family and Font Size and Font Style

-
  ```
  p {
      font-family: "Times New Roman", Times, serif;
      font-size: 13px
       }
  p.normal {
      font-style: normal;
  }
  p.italic {
      font-style: italic;
  }
  p.oblique {
      font-style: oblique;
  }
  ```

Font Example

```
<html><head><style>
p.normal {
    font-family: Times New Roman;
    font-size: 40px;
    font-style: normal;
}
p.italic {
    font-family: Arial;
    font-size: 30px;
   font-style: italic;
}
p.oblique {
    font-family: Courier;
   font-style: oblique;
}
</style></head>
```

```
<body>
<p class="normal">This is a paragraph in
    normal style.</p>

<p class="italic">This is a paragraph in
    italic style.</p>

<p class="oblique">This is a paragraph
    in oblique style.</p>
</body>
</html>
```

UNIT 4: JAVASCRIPT

Features of JavaScript, extension of JavaScript, Syntax of JavaScript: data types, operators, variables, tag, Document Object Model (DOM) with JavaScript, Selection Statement using if and Switch, Iterative statement: for, for/in, while, do while, break and continue

What is JavaScript?

- JavaScript was once upon a time used only in *client* side (browser).
- But *node js* (execution engine/run time/web server) have made possible to run JavaScript on server side.
- JavaScript is everywhere – on Desktop/Server/Mobile.

Introduction

- It is a client side scripting language.
- It is basically used for validations.
- It was developed by Netscape Communications.
- JavaScript is object based language not object oriented language.
- JavaScript is embedded in the web pages and interpreted by the web browser.

Advantages and Disadvantages of client side scripting

- Advantages:
    - The Web browser uses its own resources so less load on the server.

- Disadvantages
    - Code is usually visible
    - Local files and databases cannot be accessed as they are located on to the server..

JavaScript Features

- Light Weight Scripting language
- Dynamic Typing
- Object-oriented programming support
- Functional Style
- Platform Independent
- Prototype-based
- Interpreted Language
- Async Processing
- Client-Side Validation
- More control in the browser

Where to place JavaScript

- Embedded Script
- JavaScript can be embedded in the HTML document.
- It begins with

  <script type="text/javascript>

  - and ends with

  </script>

- You can also write

  <script language="Javascript">

  - And end with

  </script>

JavaScript example

```
<html>
  <head>
     <title>A small piece of JavaScript</title>
  </head>
  <body>
     <script type="text/javascript">
          document.write("hello, JavaScript user!");
     </script>
  </body>
</html>
```

- **Javascript can also be placed in the head section of HTML document.**

External Scripts

- If you want to use same script on several pages, it is better to create a script in the separate file.

- For eg:

```
<html>
  <head>
     <script src=“myscript.js”>
     </script>
  </head>
<html>
```

myscript.js

```
document.write("hello, JavaScript user!");
```

The external js file can not use <script>

JavaScript Display Possibilities

- **JavaScript can "display" data in different ways:**

- Writing into an HTML element, using **innerHTML**.
- Writing into the HTML output using **document.write()**.
- Writing into an alert box, using **window.alert()**.
- Writing into the browser console, using **console.log()**.

Using innerHTML

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

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

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Using window.alert()

-
  ```
  <!DOCTYPE html>
  <html>
  <body>

  <h1>My First Web Page</h1>
  <p>My first paragraph.</p>

  <script>
  window.alert(5 + 6);
  </script>

  </body>
  </html>
  ```
- **You can skip the window keyword**

Using console.log()

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Check LOG:

1. F12 (or inspect) on your keyboard to activate debugging.
2. Then select "Console" in the debugger menu.
3. Click Run again.

Variables

- The primitive data types in JS are as follows:
  1. number
  2. string
  3. boolean
  4. null
  5. function
  6. object
- Some conditions for variables:
  1. It is case sensitive.
  2. It cannot contain punctuations, spaces etc.
  3. It should not be reserved word.

Scope of the variable

- A JS variable can either have local or global scope.
- A variable is declared with the var keyword.

```
var  a,b;
```

- The above variables if defined outside function, have a global scope.
- If a variable is defined within the function, it has the local scope.

An Example

```
<html>
<body>
<script type="text/javascript">
        var a= "abc";
        test();
        document.write(a, "<br>");
        test();
        document.write(c, "<br>");
        document.write(b, "<br>");

                function test()
                {
                        var a="XYZ";
                        var b="def";
                        var c="pqr";
                        document.write(a, "<br>");
                }
</script>
</body>
</html>
```

Here variable **a** is defined globally so it is accessible.

Again three more variables **a,b** and **c** defined in function with local scope

The output is

```
XYZ
abc
XYZ
```

Variable: Assignments

- Assignment is used with the help of "=" operator.

  - X=123;
  - X="abc";
  - X=X+1;
  - X++;
  - X--;
  - X+=y;
  - X-=y;

Strings

- To **define** strings we can use

  var a="abc";

  a="abc";

- To **concat** strings we can use + operator

  a ="abc"+"def";

  b= "def"+a;

- To find the **length**

  length=a.length;

- To find a **particular** character

  nchar=a.charAt(5);

Arrays

- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

- EX (using literal):

```
<script>
var emp=[“One",“Two",“Three"];
for (i=0;i<emp.length;i++) {
   document.write(emp[i] + "<br/>");
   }
</script>
```

- EX (using constructor):

```
var myarray=new Array();
var myarray= new Array(1,2,”three”,false);
var myarray=new Array(15);
```

Arrays: Example

Using ‘new’ Keyword

```
<script type="text/javascript">
var i;
var myArray = new Array(3);
myArray[0] = “NMIMS";
myArray[1] = “University";
myArray[2] = “Navi Mumbari";

for (i=0; i<myArray.length ;i++)  {
  document.write (myArray[i] + "<br>");
        }
</script>
```

Comment Style in HTML

```
<! --
        insert your comment here….
-->
```

Using Constructor

```
<script>
var emp=new Array(“ABC",“PQR",“XYZ");

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Comment Style in JS:

Same as C Language

JavaScript: Array Methods
For Examples, visit https://www.javatpoint.com/javascript-array

concat()	It returns a new array object that contains two or more merged arrays.
entries()	It creates an iterator object and a loop that iterates over each key/value pair.
every()	It determines whether all the elements of an array are satisfying the provided function conditions.
flat()	It creates a new array carrying sub-array elements concatenated recursively till the specified depth.
flatMap()	It maps all array elements via mapping function, then flattens the result into a new array.
fill()	It fills elements into an array with static values.
from()	It creates a new array carrying the exact copy of another array element.
find()	It returns the value of the first element in the given array that satisfies the specified condition.
findIndex()	It returns the index value of the first element in the given array that satisfies the specified condition.
forEach()	It invokes the provided function once for each element of an array.
includes()	It checks whether the given array contains the specified element.
indexOf()	It searches the specified element in the given array and returns the index of the first match.
isArray()	It tests if the passed value ia an array.
join()	It joins the elements of an array as a string.
push()	It adds one or more elements to the end of an array.
reverse()	It reverses the elements of given array.
some()	It determines if any element of the array passes the test of the implemented function.
shift()	It removes and returns the first element of an array.
slice()	It returns a new array containing the copy of the part of the given array.
sort()	It returns the element of the given array in a sorted order.
splice()	It add/remove elements to/from the given array.
toString()	It converts the elements of a specified array into string form, without affecting the original array.
unshift()	It adds one or more elements in the beginning of the given array.
values()	It creates a new iterator object carrying values for each index in the array.

Methods used with character arrays

1. charAt
2. Indexof
3. lastIndexOf
4. substring
5. valueOf
6. toLowerCase
7. toUpperCase

```
<!DOCTYPE html>
<html> <body> <script>

//charAt  finds the position in a string
var str = "HELLO NMIMS";
var res = str.charAt(0);
document.write("char at 0: ",res);

//indexOf variable in string..
var str = "Hello, welcome to NMIMS ";
var n1 = str.indexOf("welcome");
document.write("<br>index of: ",n1);

//lastIndexOf a charater in string
var str = "Hello NMIMS Friends";
var n2 = str.lastIndexOf("e");
document.write("<br>last index of: ",n2);

//substring from a string
var str = "Hello world!";
var res = str.substr(1, 4);
document.write("<br>substring: ",res);
```

Output

```
1.  char at 0: H
2.  index of: 7
3.  last index of: 15
4.  substring: ello
5.  valueof: Hello World!
6.  lower case: hello world!
7.  Upper case: HELLO WORLD!
```

Functions

- A function is a section of code that is separated from the main program.

```
function abc
{
        Document.write(“hello”);
}
abc();
```

Functions: Example

```
<html>
<body>
   <script type="text/javascript">
   var z = multXbyY(10, 15);
   document.write("<br>The result is "+z);
   function multXbyY(x,y) {
        document.write("x is "+x);
        document.write("<br>y is "+y);
        return x*y;
   }
</script>
</body>
</html>
```

Conditions

if –else

```
if(a==0)
{
  …
}
else
{
  …
}
```

Switch Case

```
<html> <body>
<script type="text/javascript">
   var m=new Date();
   theMonth=m.getMonth();
   document.write(theMonth);
   switch (theMonth)
   {
        case 4:  document.write("The Merry Month of May");
                 break;
        case 0:  document.write("Cold January!");
                 break;
        case 6:  document.write("Summer Time!");
                 break;
        default:  document.write("When is it holiday time?");
   }
</script> </body> </html>
```

Loops

```
for(i=0;i<10;i++)
{
        ....
}



while(i>0)
```

The “for…in” loop

- JavaScript supports different kinds of loops:
  - for - loops through a block of code a number of times
  - for/in - loops through the properties of an object

```
var person = {fname:"John", lname:"Doe", age:25};

var text = "";
var x;
for (x in person) {
  text += person[x] + " ";
}
```

- More on objects in next session…

OBJECTS IN JS

Objects are Variables

- Objects are variables too. But objects can contain many values.
- Object values are written as name : value pairs

  let person = {firstName:“A”, lastName:“B”, age:50, eyeColor:"blue"};

```
<script>
let person = {
  firstName      : “A”,
  lastName       : “B”,
  age            :  50,
  eyeColor       : "blue”
};

document.getElementById("demo").innerHTML = person.firstName + " " + person.lastName;
</script>
```

A JavaScript object is a collection of **named values**

The named values, in JavaScript objects, are called **properties (e.g firstname)** the value of **firstname** property is **“A”** in above example

let keyword

- The **let** keyword was introduced in ES6 (2015).
- Variables defined with **let** cannot be Re-declared.
- Variables defined with **let** must be Declared before use.
- Variables defined with **let** have Block Scope.

- It is a common practice to declare objects with the const keyword.

const person = {firstName:“A”, lastName:“B”, age:50, eyeColor:"blue"};

JavaScript Objects are Mutable

- Objects are **mutable**: They are addressed by **reference**, not by **value**.

  const x = person; // Will not create a copy of person.

- The object x is **not a copy** of person. It **is** person. Both x and person are the same object.
- Any changes to x will also change person, because x and person are the **same** object.

```
const person = {
  firstName:“A",
  lastName:“B",
  age:50, eyeColor:"blue"
}

const x = person;
x.age = 10;      // Will change both x.age and person.age
```

Objects

- A javaScript object is an entity having state and behavior
  - properties and method
- JavaScript is an object-based language.
  - Everything is an object in JavaScript.
- JavaScript is template based not class based.
  - Here, we don't create class to get the object. But, we directly create objects.

- **Math Object (Inbuilt Object)**

```
Math.PI  is property
document.write(Math.PI);
value=Math.round(10.2);
```

Nested Objects

```
myObj = {
  name:"John",
  age:30,
  cars: {
    car1:"Ford",
    car2:"BMW",
    car3:"Fiat"
  }
}
```

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

<script>
const myObj = {
  name: "John",
  age: 30,
  cars: {
     car1: "Ford",
     car2: "BMW",
     car3: "Fiat"
     }
}
document.getElementById("demo").innerHTML =
myObj.cars.car2;
</script>
```

Using new keyword to create objects

```
<script type="text/JavaScript">
        var  person = new Object();
        person.firstName = "Jane";
        person.lastName = "Smith";
        person.age = 32;

        person.hair = new Object();
        person.hair.length = "long";
        person.hair.colour = "red";

        document.write(person.firstName+" "+person.lastName+" "+person.age);
        document.write("</br>");
        document.write(person.hair.length+" "+person.hair.colour);
</script>
```

DOM (Document Object Model)

- The base class in JavaScript is the window object.
- Whenever we write

  document.write(“hello”);

- We are actually writing

  window.document.write(“hello”);

- The window object is there by default.

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 DOM?

- The DOM is a W3C (World Wide Web Consortium) standard.
- The DOM defines a standard for accessing 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."
- The W3C DOM standard is separated into 3 different parts:
  - Core DOM - standard model for all document types
  - XML DOM - standard model for XML documents
  - HTML DOM - standard model for HTML documents

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 following example changes the content (the innerHTML) of the <p> element with id="demo"

```
<html>
<body>

<p id="demo">This is SVMIT College</p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

JavaScript HTML DOM - Changing CSS

```
<html> <body>
<h1 id="id1">My Heading 1</h1>

<button type="button"
onclick="document.getElementById('id1').style.color = 'red'">
Click Me!</button>

<button type="button"
onclick="document.getElementById('id1').innerHTML = 'Now, Not Allowed
!!!'">
Ohh No!</button>

<button type="button"
onclick="document.getElementById('id1').style.color = 'black'">
Black?</button>

<button type="button"
onclick="document.getElementById('id1').innerHTML = 'My Heading 1'">
Origional!</button>
</body> </html>
```

Finding HTML Elements by HTML Object Collections

```
<html> <body>
<form id="frm1" action="/action_page.php">
  First name: <input type="text" name="fname" value=“NMIMS"><br>
  Last name: <input type="text" name="lname" value=“NM"><br><br>
  <input type="submit" value="Submit">
</form>
<p>Click "Try it" to display the value of each element in the form.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x = document.forms["frm1"];
  var text = "";      var i;
  for (i = 0; i < x.length ;i++) {
    text += x.elements[i].value + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
}
</script> </body> </html>
```

JavaScript Errors - Throw and Try to Catch

```
<html>  <body>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="p01"></p>
<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if (x == "")  throw "empty";
    if (isNaN(x)) throw "not a number";x = Number(x);
    if (x < 5)  throw "too low";              if(x > 10)   throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}             </script>  </body>  </html>
```

JavaScript Debugging

```
<html> <body>

<h2>My First Web Page</h2>

<p>Activate debugging in your browser (Chrome, IE, Firefox) with
F12, and select "Console" in the debugger menu.</p>

<script>
a = 5;
b = 6;
c = a + b;
console.log(c);
</script>    </body>     </html>
```

JAVASCRIPT VALIDATIONS

Validate Numeric Input

```
<html>
<body>

<h2>JavaScript Can Validate Input</h2>

<p>Please input a number between 1 and 10:</p>

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

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

<script>
function myFunction() {
  var x, text;



 // Get the value of the input field with id="numb"
 x = document.getElementById("numb").value;
```

Automatic HTML Form Validation

```
<html>
<body>

<form action="/action_page.php" method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>

<p>If you click submit, without filling out the text field,
your browser will display an error message.</p>

</body>
</html>
```

HTML Constraint Validation

Attribute	Description
disabled	Specifies that the input element should be disabled
max	Specifies the maximum value of an input element
min	Specifies the minimum value of an input element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of an input element

```
<html><body>
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
</body></html>
```

Constraint Validation DOM Methods

```
<html> <body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" min="100" max="300" required>
<button onclick="myFunction()">OK</button>

<p>If the number is less than 100 or greater than 300, an error message will be
displayed.</p>
<p id="demo"></p>

<script>
function myFunction() {
  var inpObj = document.getElementById("id1");
  if (!inpObj.checkValidity()) {
    document.getElementById("demo").innerHTML = inpObj.validationMessage;
  } else {
    document.getElementById("demo").innerHTML = "Input OK";
  }
}
</script> </body> </html>
```

Constraint Validation DOM Properties

Property	**Description**
validity	Contains boolean properties related to the validity of an input element.
validationMessage	Contains the message a browser will display when the validity is false.
willValidate	Indicates if an input element will be validated.

Validity Properties

Property	**Description**
customError	Set to true, if a custom validity message is set.
patternMismatch	Set to true, if an element's value does not match its pattern attribute.
rangeOverflow	Set to true, if an element's value is greater than its max attribute.
rangeUnderflow	Set to true, if an element's value is less than its min attribute.
stepMismatch	Set to true, if an element's value is invalid per its step attribute.
tooLong	Set to true, if an element's value exceeds its maxLength attribute.
typeMismatch	Set to true, if an element's value is invalid per its type attribute.
valueMissing	Set to true, if an element (with a required attribute) has no value.
valid	Set to true, if an element's value is valid.

The rangeOverflow Property

```
<html> <body>
<p>Enter a number and click OK:</p>
<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>
<p>If the number is greater than 100 (the input's max attribute), an error message
will be displayed.</p>
<p id="demo"></p>

<script>
function myFunction() {
  var txt = "";
  if (document.getElementById("id1").validity.rangeOverflow) {
    txt = "Value too large";
  } else {
    txt = "Input OK";
  }
  document.getElementById("demo").innerHTML = txt;
}
</script> </body> </html>
```

JavaScript Form Validation Example

```
<script>
function validateform(){
var name=document.myform.name.value;
var password=document.myform.password.value;

if (name==null || name==""){
  alert("Name can't be blank");
  return false;
}else if(password.length<6){
  alert("Password must be at least 6 characters long.");
  return false;
  }
}  </script>
<body>
<form name="myform" method="post" action="abc.jsp" onsubmit="return
validateform()" >
Name: <input type="text" name="name"><br/>
Password: <input type="password" name="password"><br/>
<input type="submit" value="register">
</form>
```

Validation with IMAGE (HTML form on next slide)

```
<script>
function validate(){
var name=document.f1.name.value;
var password=document.f1.password.value;
var status=false;

if(name.length<1){
document.getElementById("nameloc").innerHTML=  " <img src='uncheck.png'/> Please enter 
your name";
status=false;
}else{
document.getElementById("nameloc").innerHTML=" <img src='check.png'/>";  status=true;
}
if(password.length<6){
document.getElementById("passwordloc").innerHTML= " <img src='uncheck.png'/>
Password must be at least 6 char long";   status=false;
}else{
document.getElementById("passwordloc").innerHTML=" <img src='check.png'/>";
}
return status;
}
</script>
```

HTML form for previous slide

```
<form name="f1" action="#" onsubmit="return validate()">
<table>
<tr><td>Enter Name:</td><td><input type="text" name="name"/>
<span id="nameloc"></span></td></tr>
<tr><td>Enter Password:</td><td><input type="password"
name="password"/>
<span id="passwordloc"></span></td></tr>
<tr><td colspan="2"><input type="submit"
value="register"/></td></tr>
</table>
</form>
```

Regular Expressions: Modifiers (g, i, m)

```
<html> <body>
<p>Click the button to do a global (g), multiline(m), case insensitive (i) search for
"is" at the beginning of each line in a string.</p>
<button onclick="myFunction()">Try it</button>

<p id="strdisp">input string is:</p>

<p id="demo"> Regular Expression result: </p>

<script>
function myFunction() {
  var str = "\nIs th\nis h\nis?";
  document.getElementById("strdisp").innerHTML += str;
  var patt1 = /is/gim;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML += result;
}
</script>  </body> </html>
```

Brackets

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

*** Example of [abc]**

```
<html> <body>
<p>Click the button to do a global search for the character "h" in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var str = "Is this all there is?";
  var patt1 = /[h]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script> </body> </html>
```

[^abc]

```
<html> <body>

<p>Click the button to do a global search for characters that are NOT "i" and "s" in a
string.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var str = "Do you know if this is all there is?";
  var patt1 = /[^is]/gi;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>  </body>  </html>
```

[^abc] another example

```
<html> <body>

<p>Click the button to do a global search for the character-span NOT from uppercase A to
uppercase E.</p>

<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var str = "I SCREAM FOR ICE CREAM!";
  var patt1 = /[^A-E]/g;
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>
</body> </html>
```

Meta Characters for Regular Expression

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character

\d – Find a digit (\D for non-digit)

```
<html> <body>

<p>Click the button to do a global search for digits in a string.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>

<script>
function myFunction() {
  var str = "Give 100%!";
  var patt1 = /\d/g;   //use d+ for complete number e.g. 100
  var result = str.match(patt1);
  document.getElementById("demo").innerHTML = result;
}
</script>  </body> </html>
```

JAVASCRIPT EVENTS

Mouse events:

Event Performed	**Event Handler**	**Description**
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Mouse event: onclick, onmouseover, onmouseout (HTML)

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

<button onclick="document.getElementById('d1').innerHTML=Date()"> The time is?
</button>
<p id="d1"></p>
<button onmouseover= "display()" onmouseout= "displayout()"> Over ME </button>

<p id="d2"></p>
<script>
function display() {
  document.getElementById('d2').innerHTML="mouse over";
}

function displayout() {
  document.getElementById('d2').innerHTML="mouset out";
}
</script>  </body>   </html>
```

Mouse event: onclick (JavaScript)

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

<p>This example uses the HTML DOM to assign an "onclick" event to
a p element.</p>
<p id="demo">Click me.</p>

<script>
document.getElementById("demo").onclick = function()
{myFunction()};

function myFunction() {
  document.getElementById("demo").innerHTML = "CLICKED !!!";
}
</script>  </body>  </html>
```

Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

HTML EXAMPLE

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

<p>A function is triggered when the user is pressing a key in the input field.</p>

<input type="text" onkeydown="myFunction()">

<script>
function myFunction() {
  alert("You pressed a key inside the input field");
}
</script>

</body>  </html>
```

Keyboard Event: JavaScript Example

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

<p>This example uses the HTML DOM to assign an "onkeypress" event to
an input element.</p>

<p>Press a key inside the text field to set a red background color.</p>

<input type="text" id="demo">

<script>
document.getElementById("demo").onkeypress = function()
{myFunction()};

function myFunction() {
  document.getElementById("demo").style.backgroundColor = "red";
}
</script>  </body>  </html>
```

Form events:

Event Performed	**Event Handler**	**Description**
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

form Event (onsubmit): JavaScript Example

```
<!DOCTYPEhtml>
<html> <body>
<p>Welcome To My Page.</p>
<p>Have a Nice Day.</p>

<form id="first" action="yourpage.html">
Enter your name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>

<script>
document.getElementById("first").onsubmit = function() {demo()};
function demo() {
alert("The form was submitted successfully");
}
</script>

</body> </html>
```

Window/Document events:

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

onload

```
<!DOCTYPE html>
<html>
<body onload="myFunction()">

<h1>Hello World!</h1>

<script>
function myFunction() {
  alert("Page is loaded");
}
</script>

</body>
</html>
```

addEventListener

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

<p>This example uses the addEventListener() method to attach a "load" event to an iframe element.</p>

<iframe id="myFrame" src="/default.asp"></iframe>

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

<script>
document.getElementById("myFrame").addEventListener("load", myFunction);

function myFunction() {
  document.getElementById("demo").innerHTML = "Iframe is loaded.";
}
</script>

</body> </html>
```

jQuery

https://jquery.com/

What is jQuery?

- jQuery is a fast and concise JavaScript **Library** created by John Resig in 2006 with a nice motto: Write less, do more.

- jQuery **simplifies** HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

- jQuery is a JavaScript **toolkit** designed to simplify various tasks by writing less code.

Important core features supported by jQuery

- **DOM manipulation** − The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.
- **Event handling** − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
- **AJAX Support** − The jQuery helps you a lot to develop a responsive and featurerich site using AJAX technology.
- **Animations** − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
- **Lightweight** − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
- **Cross Browser Support** − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology** − The jQuery supports CSS3 selectors and basic XPath syntax.

How to use jQuery?

There are two ways to use jQuery.

- **Local Installation** − Download jQuery library on local machine and include it in HTML code.
- **CDN Based Version** − Include jQuery library into HTML code directly from Content Delivery Network (CDN).

```
CDN:
<script type = "text/javascript"
  src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
```

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

- How to get CDN path?

https://code.jquery.com/

```
<script src="https://code.jquery.com/jquery-2.2.4.js"
integrity="sha256-iT6Q9iMJYuQiMWNd9lDyBUStIq/8PuOW33aOqmvFpqI="
crossorigin="anonymous"></script>
```

jQuery Syntax

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".

First working example

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide();
  });
});
</script>
</head>
<body>
```

Use CDN - https://ajax.googleapis.com/ajax/libs/jquery/3.6/jquery.min.js

```
<h2>This is a heading</h2>
```

The Document Ready Event

- jQuery methods in example, is 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).
- The jQuery team has also created an even shorter method for the document ready event

```
$(function(){

  // jQuery methods go here...

});
```

jQuery Selectors

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

The element Selector

- You can select all <p> elements on a page like this: $("p")

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

The #id Selector

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

```
<p id="test">This is another paragraph.</p>
```

The .class Selector

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

```
<h2 class="test">This is a heading</h2>
```

href example

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("[href]").hide();
  });
});
</script>
```

```
<p><a href="www.example1.com/">HTML Tutorial</a></p>
<p><a href="www.example2.com/">CSS Tutorial</a></p>
```

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

$("ul li:first-child")

```
<!DOCTYPE html>
<html>
<head>
<script
src="jquery-3.6.0.js">
</script>

<script>
$(document).ready(function(){
  $("button").click(function(){
    $("ul li:first-child").hide();
  });
});
</script>
</head>
<body>
```

What about second child?

```
$("ul li:nth-child(2)").hide();
```

```
<p>List 1:</p>
```

$("ul li:first")

```
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("ul li:first").hide();
  });
});
</script>
```

```
<script>
$(document).ready(function(){
```

jQuery 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

click() function is executed when the user clicks on the HTML element

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js">
</script>
<script>
$(document).ready(function(){
  $("p").click(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>
```

dblclick() is executed when the user double clicks HTML element

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js">
</script>
<script>
$(document).ready(function(){
  $("p").dblclick(function(){
    $(this).hide();
  });
});
</script>
</head>
<body>
```

mouseenter()

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js"></script>
<script>
$(document).ready(function(){
  $("#p1").mouseenter(function(){
    alert("You entered p1!");
  });
});
</script>
</head>
<body>

<p id="p1">Enter this paragraph and you will get the alert !!!</p>
```

mouse hover example

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>hover demo</title>
  <style>
  ul {
    margin-left: 20px;
    color: blue;
  }
  li {
    cursor: default;
  }
  span {
    color: red;
  }
  </style>
  <script src=" jquery-3.6.0.js "></script>
</head>
<body>

<ul>
  <li>Milk</li>
```

Working with CSS

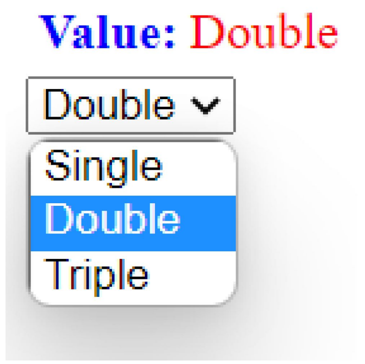
```
<!DOCTYPE html>
<html>
<head>
<script type="jquery-3.6.0.js">
</script>

 <script>
    $(document).ready(function() {
      $("p").css("background-color", "pink");
    });
 </script>
 </head>



<body>
<p>This is first paragraph.</p>
```

Working with Animation

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").animate({left: '450px'});
    });
});
</script>
</head>


<body>
```

Working with Select

```
<!DOCTYPE html>
<html lang="en">
<head>

<title>val demo</title>
  <style>
  p {
    color: red;
    margin: 4px;
  }
  b {
    color: blue;
  }
  </style>
  <script src="jquery-3.6.0.js"></script>
</head>
<body>

<select id="single">
```


Value: Double

Working with class (addClass)

```
<!DOCTYPE html>
<html>
<head>
<script src="jquery-3.6.0.js"></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("p:first").addClass("intro");
    });
});
</script>
```

Also Check: hasClass()

```
<style>
```

jQuery wrap()

- jQuery wrap() method is used to wrap specified HTML elements around each selected element. The wrap () function can accept any string or object that could be passed through the $() factory function.

```
<!DOCTYPE html>
<html>
<head>
<script src=" "></script>

<script>
$(document).ready(function(){
   $("button").click(function(){
      $("p").wrap("<div></div>");
   });
});
</script>

<style>
div{background-color: pink;}
</style>
```

jQuery serialize()

- jQuery **serialize()** method is used to create a text string in standard URL-encoded notation. It is used in form controls like <input>, <textarea>, <select> etc. It serializes the form values so that its serialized values can be used in the URL query string while making an AJAX request.

```
<!DOCTYPE html>
<html>
<head>
  <script src=" "></script>
<script>
$(document).ready(function(){
    $("button").click(function(){
        $("div").text($("form").serialize());
    });
});
</script>
</head>

<body>
<form action="">
```

jQuery Traversing

jQuery find() method

- The **find()** method in jQuery finds the descendant elements of the given selector. A descendant can be a child, grandchild, and so on. It is an inbuilt method in jQuery.
- For searching the descendant, the **find()** method traverse downwards from the selected element in the DOM tree.
- We can use the **"*"** selector for returning all descendant elements.
- To return all descendant elements of the given selector, we have to write: `$(selector).find("*")`

- The **children()** method works similarly to the find() method. Unlike the find() method, the children() method traverse a **single level down** the **DOM** tree, i.e., it returns the direct children.

```
<!DOCTYPE html>
<html>
<head>
<style>
.main * {
  display: block;
  font-size: 20px;
  position: relative;
  border: 2px solid black;
  color: black;
  padding: 10px;
  margin: 17px;
}
</style>
<script src=" "></script>
<script>
function fun(){
$(document).ready(function(){
  $("#div1").find("ul").css({ "font-size": "30px", "color": "blue", "border": "6px dashed blue"});
});
```

jQuery children() method

```
<!DOCTYPE html>
<html>
<head>
<style>
.main * {
  display: block;
  font-size: 20px;
  position: relative;
  border: 2px solid black;
  color: black;
  padding: 10px;
  margin: 17px;
}
</style>
<script src=" "></script>
<script>
function fun(){
$(document).ready(function(){
  $("div").children().css({ "font-size": "30px", "color": "blue", "border": "6px dashed blue"});
});
```

With jQuery you can traverse up the DOM tree

An ancestor is a parent, grandparent, great-grandparent, and so on.

Traversing Up the DOM Tree

Three useful jQuery methods for traversing up the DOM tree are:

- parent()
- parents()
- parentsUntil()

```
<script>
$(document).ready(function(){
  $("span").parent().css({"color": "red", "border": "2px solid
red"});
});
</script>
```

```
<script>
$(document).ready(function(){
  $("span").parents().css({"color": "red", "border": "2px solid
red"});
});
</script>
```

jQuery AJAX

jQuery ajax() method

- AJAX is an acronym for **Asynchronous JavaScript and XML**.
- It is a group of inter-related technologies like JavaScript, DOM, XML, HTML, XHTML, CSS, XMLHttpRequest etc.
- It allows us to send and receive data asynchronously without reloading the web page. So it is fast.
- The **ajax()** method in jQuery performs an AJAX request. It sends an synchronous HTTP request to the server.
- JQuery provides a rich set of AJAX methods for developing web applications. It is widely used for the requests.
- The syntax of using the ajax() method is: **$.ajax({name:value, name:value, ... })**

jQuery AJAX Example (test.html file is given)

```
<!DOCTYPE html>
<html>
<head>
<script src=" "></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
jQuery.ajax({
url: 'test.html',
type: 'GET',
  success: function(data) {
    $("#para").html(data);
  }
});
```

PHP

Hypertext Preprocessor

Introduction

- PHP stands for Hypertext Preprocessor.
- PHP is a scripting language that is embedded with the HTML page.
- It is a server side scripting language.
- A PHP code inside HTML page starts with **<?php** tag and ends with **?>** tag

A simple program

```
<html>
    <head>
     <title>hello world</title>
    </head>

    <body>
     <?php
          print(“hello world”);
     ?>
    </body>
</html>
```

PHP comments

- Comments can be applied same as in C and C++.

```
<html>  <head>  <title>Comments</title>  </head>
    <body>
     <?php
          /*
              We can print today's date using PHP
          */
          //print ("hello");
          print("hello world");
     ?>
</body>
</html>
```

Variables

- Whenever a variable is encountered for the first time, a memory space is set aside for the contents.
- You do not need to specify the data types for the variables.
- In PHP, all variables are prefaced with the ‘$’ sign.

Example: Variables

l = A full textual representation of the day of the week	*Sunday* through *Sat urday*
F = A full textual representation of a month, such as January or March	*January* through *D ecember*
d = Day of the month, 2 digits with leading zeros	*01* to *31*
Y = A full numeric representation of a year, 4 digits	Examples: *1999* or *2003*

```
<html>
<head>
        <title>Variables</title>
</head>
<body>
<?php
        $Name = "A D";
        $Surname = "Patel";
        $id_no= "N22";
        $todaysDate = date("l F d, Y");

        print("Name is ".$Name );
        print("<br> Surname is ".$Surname);
        print("<br> Id is ".$id_no );
        print("<br> Date is ".$todaysDate);
?>
</body>
</html>
```

Send and Receive

```
<html> <head> <title>form</title>   </head>
<body>       <form action="retrieve.php" method="post">
            Name: <input type="text" name="aName"><br>
            Age: <input type="text" name="anAge"><br>
            Date of Birth: <input type="text" name="dob"><br>
            <input type="submit" value="Send Data">
</form> </body> </html>
```

```
<html> <body>
<?php
print("Name is ".$_REQUEST['aName']."<br>");
print("Age is ".$_REQUEST['anAge']."<br>");
print("Birth date is ".$_REQUEST['dob']);
?>
</body> </html>
```

Decisions

L = Whether it's a leap year
1 if it is a leap year, 0 otherwise.

```
<html>    <body>
<?php
    $Today = date("l F d, Y");
    print("Today is $Today");
    $Today = date("L");
    if($Today == 1)
        print("<br>This year is a leap year!");
    else
        print("<br>This year is not a leap year");
?>
</body> </html>
```

Operators for Decisions

Operator	Operation Performed	Example
<	Less than	$num < 12
>	Greater than	$num > 12
<=	Less than equal to	$num <= 12
>=	Greater than equal to	$num >= 12
==	Equal to	$num == 12
!=	Not Equal to	$num != 12
AND, &&	Logical And	$num1 AND $num2 $num1 && $num2
OR, ||	Logical Or	$num1 OR $num2 $num1 || $num2
XOR	Exclusive OR	$num1 XOR $num2
!	Not	!$num

Switch Case

```
<html>
<body>
<?php
$Today = date("l F d, Y");
print("Today is $Today, <br>\n");
$diaryDate = date("d");
switch($diaryDate)
{
     case 03 : print("meeting");
       break;
     case 10 : print("appointment ");
       break;
     case 23 : print("club");
               break;

case 25 : print("conference");
               b
```

d	Day of the month, 2 digits with leading zeros	*01* to *31*

For Loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
    <?php
        for($count = 1; $count <= 10; $count++)
        {
            print("7 * $count =".(7*$count)."<br>");
        }
    ?>
</body>
</html>
```

While loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
      <?php
      $count=1;
      while( $count<=10 )
      {
       print("7 * $count =".(7*$count)."<br>");
       $count++;
      }
      ?>
</body>
</html>
```

do-while loop

```
<html>
<body>
<h1>I must learn my 7 times table</h1><br>
    <?php
    $count=1;
    do
    {
     print("7 * $count =".(7*$count)."<br>");
     $count++;
    }while( $count<=10 )
    ?>
</body>
</html>
```

break and continue

```
<html> <body>
    <?php
        for ($i = 0; $i <= 5; $i++)
        {
            if ($i == 2)
             continue;
            print "$i<br>";
            if($i==4)
             break;
        }
    ?>
</body> </html>
```

Arrays

Arrays are the variables that can hold many values under a single name

```
<html>
<body>
<?php
    $myarray = array();
    $myarray[0] = "This";
    $myarray[1] = " is ";
    $myarray[2] = " my array";
    echo($myarray[0].$myarray[1].$myarray[2]);
?>
</body>
</html>
```

```
$myarray = array(“This”, “is”, “myarray”);
$myarray = array(“This”, 1, 2.5);
$total = myarray[1]+myarray[2];

$preference= array(“red”, “white”, “blue”);
$preference[1]=“green”;
```

Two-dimensional Arrays

```
<html> <body>
<?php
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );

echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
</body> </html>
```

OUTPUT:

```
Volvo: In stock: 22, sold: 18.
BMW: In stock: 15, sold: 13.
Saab: In stock: 5, sold: 2.
Land Rover: In stock: 17, sold: 15.
```

Arrays with loop

```
<html><body> <ol>
    <?php
    $preferences = array ("red", "white", "blue",
    "silver", "aqua", "cyan", "yellow");
    echo("The current preferences are");
    foreach($preferences as $value)
    {
     echo("<li>This preference is: $value </li>");
    }
    ?>
</ol> </body> </html>
```

Arrays with loop

```
<html> <body> <ol>
    <? php
    $preferences = array
    ("red","white","blue","silver","aqua","cyan", "yellow");
    for ($i=0; $i<sizeof($preferences); $i++)
    {
     $value = $preferences[$i];
     echo("<li>This preference is: $value</li>");
    } ?>
</ol> </body> </html>
```

Array Operations

- To add values at end of the array
  array_push($preferences, “black”, “gold”);

- To add values in beginning of the array
  array_unshift($preferences, “black”, “gold”);

- To remove an item from the start of the array
  array_shift($preferences);

- To remove an item from the end of the array
  array_pop($preferences);

Cont’d

- To sort an array
- **sort($preferences);**
- **rsort()** - sort arrays in descending order
- SORT_REGULAR   compare items normally
- SORT_NUMERIC   compare items numerically
- SORT_STRING   compare items as strings

- For merging two arrays we have
- **$endarray=array_merge($array1,$array2);**
- To slice up an array
- **$endarray=array_slice($prefs,2,6);**

PHP Global Variables - Superglobals

- Some predefined variables in PHP are "superglobals", which means that they are always accessible, regardless of scope - and you can access them from any function, class or file without having to do anything special.
- The PHP superglobal variables are:
  - $GLOBALS
  - $_SERVER
  - $_REQUEST
  - $_POST
  - $_GET
  - $_FILES
  - $_ENV
  - $_COOKIE
  - $_SESSION

$GLOBALS

```php
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

declare global variable in PHP

```
<?php
// Demonstrate how to declare global variable
// Declaring global variable
$x = “NMIMS";
$y = “STME";
$z = “Computer";
$a = 55;
$b = 100;

function concatenate() {
   // Using global keyword
   global $x, $y, $z;
   return $x.$y.$z;
}
function add() {
   // Using GLOBALS['var_name']
   $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
```

$_SERVER

```
<html> <body>
<?php
   echo $_SERVER['PHP_SELF']; //page
   echo "<br>";
   echo $_SERVER['SERVER_NAME']; //localhost
   echo "<br>";
   echo $_SERVER['HTTP_HOST'];  //host name, here, localhost
   echo "<br>";
   echo $_SERVER['HTTP_REFERER'];  //complete URL
   echo "<br>";
   echo $_SERVER['HTTP_USER_AGENT'];  //Mozilla / chrome / safari
   echo "<br>";
   echo $_SERVER['SCRIPT_NAME']; //php script name
?>
</body> </html>
```

PHP Form Handling

exmple.php

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

welcome.php

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo
$_POST["email"]; ?>

</body>
</html>
```

The same result could also be achieved using the HTTP GET method

PHP Form Validation

- The Form Element
  - <form method="post" action="<?php echo htmlspecialchars ($_SERVER["PHP_SELF"]);?>">
- $_SERVER["PHP_SELF"]
  - The $_SERVER["PHP_SELF"] is a super global variable that returns the filename of the currently executing script.
- htmlspecialchars()
  - The htmlspecialchars() function converts special characters to HTML entities. This means that it will replace HTML characters like < and > with &lt; and &gt;. This prevents attackers from exploiting the code by injecting HTML or Javascript code (Cross-site Scripting attacks) in forms.

PHP Form Security

- The $_SERVER["PHP_SELF"] variable can be used by hackers!
- If PHP_SELF is used in your page then a user can enter a slash (/) and then some Cross Site Scripting (XSS) commands to execute.
- Cross-site scripting (XSS) is a type of computer security vulnerability typically found in Web applications. XSS enables attackers to inject client-side script into Web pages viewed by other users.

Vulnerability Example

- Assume we have the following form in a page named "test_form.php":

```
<form method="post" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

- Now, if a user enters the normal URL in the address bar like "http://www.example.com/test_form.php", the above code will be translated to:

```
<form method="post" action="test_form.php">
```

- However, consider that a user enters the following URL in the address bar:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ea
lert('hacked')%3C/script%3E
```

- In this case, the above code will be translated to:

```
<form method="post" action="test_form.php/"><script>alert('hack
ed')</script>
```

How To Avoid $_SERVER["PHP_SELF"] Exploits?

- $_SERVER["PHP_SELF"] exploits can be avoided by using the htmlspecialchars() function.
- The form code should look like this:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

- The htmlspecialchars() function converts special characters to HTML entities.
- Now if the user tries to exploit the PHP_SELF variable, it will result in the following output:

```
<form method="post" action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

Validate Form Data With PHP

```
<html> <head> </head>
<body>
<?php
// define variables and set to empty values
$name = $email = $gender = $comment = $website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data); //Strip unnecessary characters (extra space, tab,
newline) from the user input data
  $data = stripslashes($data); // Remove backslashes (\) from the user
input data
  $data = htmlspecialchars($data);
  return $data;
} ?>
```

```
<h2>PHP Form Validation Example</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  Name: <input type="text" name="name">
  E-mail: <input type="text" name="email">
 Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></textarea>
Gender:
  <input type="radio" name="gender" value="female">Female
  <input type="radio" name="gender" value="male">Male
  <input type="radio" name="gender" value="other">Other
<input type="submit" name="submit" value="Submit">
</form>

<?php
echo "<h2>Your Input:</h2>";
echo $name;        echo "<br>";
echo $email;       echo "<br>";
echo $website;          echo "<br>";
echo $comment;          echo "<br>";
echo $gender;
?>   </body> </html>
```

In continuation with previous slide

PHP Advanced

PHP Include Files

- Assume we have a standard menu file called "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a> -
<a href="default.asp">PHP Tutorial</a>';
?>
```

- All pages in the Web site should use this menu file. Here is how it can be:

```
<html> <body>
<div>        <?php include 'menu.php';?> </div>
<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
</body> </html>
```

Cont’d

- The include (or require) statement takes all the text/code/markup that exists in the specified file and copies it into the file that uses the include statement.
- require will produce a fatal error (E_COMPILE_ERROR) and stop the script
- include will only produce a warning (E_WARNING) and the script will continue

PHP File Handling

- PHP has several functions for creating, reading, uploading, and editing files.
- PHP readfile() Function
  - The readfile() function reads a file and writes it to the output buffer.

```
<?php
echo readfile(“file.txt");
?>
```

file.txt

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

PHP File Open/Read/Close

```
<?php
$myfile = fopen("file.txt", "r") or die("Unable to open file!");
echo fread($myfile,filesize("file.txt"));
fclose($myfile);
?>
```

Modes	**Description**
r	**Open a file for read only**. File pointer starts at the beginning of the file
w	**Open a file for write only**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a	**Open a file for write only**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist
x	**Creates a new file for write only**. Returns FALSE and an error if file already exists
r+	**Open a file for read/write**. File pointer starts at the beginning of the file
w+	**Open a file for read/write**. Erases the contents of the file or creates a new file if it doesn't exist. File pointer starts at the beginning of the file
a+	**Open a file for read/write**. The existing data in file is preserved. File pointer starts at the end of the file. Creates a new file if the file doesn't exist

- PHP Read Single Line - fgets()

```
<?php
$myfile = fopen(“file.txt", "r") or die("Unable to open file!");
echo fgets($myfile);
fclose($myfile);   ?>
```

- PHP Check End-Of-File - feof()

```
<?php
$myfile = fopen(“file.txt", "r") or die("Unable to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);   ?>
```

- PHP Read Single Character - fgetc()

```
<?php
$myfile = fopen("webdictionary.txt", "r") or die("Unable to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);   ?>
```

- **PHP Write to File - fwrite()**

fwrite(file, string, length)

Parameter	**Description**
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

```
<?php
$myfile = fopen("newfile.txt", "w") or die("Unable to open file!");
$txt = “NMIMS\n";
fwrite($myfile, $txt);
$txt = “STME\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

PHP file upload: Rules

- First, ensure that PHP is configured to allow file uploads.
  - In your "php.ini" file, search for the file_uploads directive, and set it to On

    file_uploads = On
- Create The HTML Form
  - Make sure that the form uses method="post"
  - The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form
- Create The Upload File PHP Script

PHP file upload: HTML Form

```
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```

PHP file upload: Adding restrictions

```
// Check if file already exists
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
```

```
// Check file size
if ($_FILES["fileToUpload"]["size"] >
500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType != "png"
            && $imageFileType != "jpeg“ && $imageFileType != "gif" )
{
   echo "Sorry, only JPG, JPEG, PNG & GIF files are allowed.";
   $uploadOk = 0;
}
```

Upload File PHP Script

```
$target_dir = "uploads/";
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType
= strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
// Check if image file is a actual image or fake image
if(isset($_POST["submit"])) {
    $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
    if($check !== false) {
        echo "File is an image - " . $check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
```

Click for the complete PHP script

Functions used in PHP file upload script

- **isset():** Check whether a variable is empty. Also check whether the variable is set/declared
- **basename():** Return filename from the specified path
- **pathinfo():** returns information about path: either an associative array or a string, depending on options
  - pathinfo (string $path [, int $options = PATHINFO_DIRNAME | PATHINFO_BASENAME | PATHINFO_EXTENSION | PATHINFO_FILENAME])
- **strtolower():** Convert all characters to lowercase
- **getimagesize():** function will determine the size of any supported given **image** file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML IMG tag and the correspondent HTTP content type
- **$_FILES**: HTTP File Upload variables

PHP Cookies

- What is a Cookie?
  - A cookie is often used to identify a user.
  - A cookie is a small file that the server embeds on the user's computer.
  - Each time the same computer requests a page with a browser, it will send the cookie too.
  - With PHP, you can both create and retrieve cookie values.
  - A cookie is created with the **setcookie()** function.

setcookie(name, value, expire, path, domain, secure, httponly);

Cookie example

```
<?php
$cookie_name = “NMIMS";
$cookie_value = “STME";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); // 86400 = 1 day
?>    //before HTML tag

<html>    <body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is set!<br>";
    echo "Value is: " . $_COOKIE[$cookie_name];
}
?>
<p><strong>Note:</strong> You might have to reload the page to see the value of the
cookie.</p>
</body>  </html>
```

PHP Sessions

- A session is a way to store information (in variables) to be used across multiple pages.
  - Unlike a cookie, the information is not stored on the users computer.
- When you work with an application, you open it, do some changes, and then you close it.
  - This is much like a Session. The computer knows who you are. It knows when you start the application and when you end.
  - But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
- Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc).
- By default, session variables last until the user closes the browser.
- So; Session variables hold information about one single user, and are available to all pages in one application.
- If you need a permanent storage, you may want to store the data in a database.

Start a PHP Session

sessionstart.php

```
<?php
// Start the session ; must before HTML tag
session_start();
?>

<html>  <body>
<?php
// Set session variables
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>  </html>
```

Get PHP Session Variable Values

sessionretrieve.php

```
<?php
session_start();
?>
<html> <body>

<?php
// Echo session variables that were set on previous page
echo "Favorite color is " . $_SESSION["favcolor"] . ".<br>";
echo "Favorite animal is " . $_SESSION["favanimal"] . ".";
?>

</body> </html>
```

More on PHP session

- Another way to show all the session variable values for a user session is to run the following code

```
<?php
print_r($_SESSION);
?>
```

- Modify a PHP Session Variable

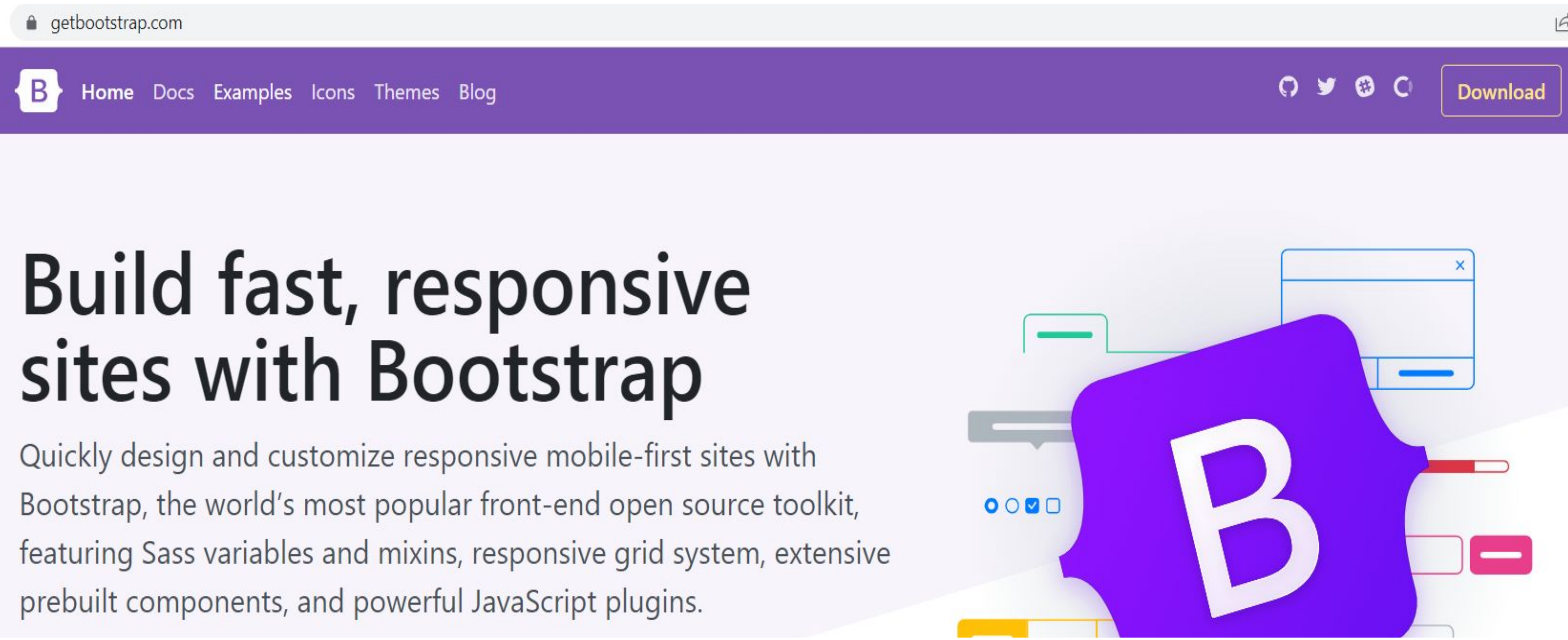
```
<?php
// to change a session variable, just overwrite it
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
```

- Destroy a PHP Session

```
<?php
// remove all session variables
session_unset();
// destroy the session
session_destroy();
?>
```

Bootstrap

getbootstrap.com



getbootstrap.com

Build fast, responsive sites with Bootstrap

Quickly design and customize responsive mobile-first sites with Bootstrap, the world's most popular front-end open source toolkit, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins.

What is Responsive Web Design?

- Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.
- Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.
- Bootstrap is completely free to download and use!

- Other free web development frameworks?
  - Laravel, Django, ASP.NET, Spring Web MVC, Cake PHP, Ruby on Rails, Code Ignighter, Symfony and many more…

What is Bootstrap?

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

Bootstrap History

- Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter, and released as an open source product in August 2011 on GitHub.
- Advantages of Bootstrap:
  - **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
  - **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops
  - **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework
  - **Browser compatibility:** Bootstrap is compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera)

Where to Get Bootstrap?

- There are two ways to start using Bootstrap on your own web site.
  - Download Bootstrap from getbootstrap.com
    - If you want to download and host Bootstrap yourself, go to getbootstrap.com, and follow the instructions there.
  - Include Bootstrap from a CDN
    - If you don't want to download and host Bootstrap yourself, you can include it from a CDN (Content Delivery Network).
    - MaxCDN provides CDN support for Bootstrap's CSS and JavaScript. You must also include jQuery.

Bootstrap CDN

- You must include the following Bootstrap’s CSS, JavaScript, and jQuery from MaxCDN into your web page.

```
<!-- Latest compiled and minified Bootstrap CSS -->
<link rel="stylesheet"href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">


<!-- Latest compiled Bootstrap JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>


<!-- latest jQuery library -->
<script src="https://code.jquery.com/jquery-latest.js"></script>
```

- Advantage of using the Bootstrap CDN:
  - Many users already have downloaded Bootstrap from MaxCDN 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.

Create Web Page with Bootstrap (1)

- Add the HTML5 doctype
  - Bootstrap uses HTML elements and CSS properties that require the HTML5 doctype.
  - Always include the HTML5 doctype at the beginning of the page, along with the lang attribute and the correct character set:

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

Create Web Page with Bootstrap (2)

- Bootstrap is mobile-first
  - 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, add the following <meta> tag inside the <head> element:

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

  - The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
  - The initial-scale=1 part sets the initial zoom level when the page is first loaded by the browser.

Create Web Page with Bootstrap (3)

- Containers
  - Bootstrap also requires a containing element to wrap site contents.
  - There are two container classes to choose from:
    - The .container class provides a responsive **fixed width container.**
    - The .container-fluid class provides a **full width container**, spanning the entire width of the viewport.
- **Note:** Containers are not nestable (you cannot put a container inside another container).

Check the working example

1. Create an HTML Page

As a first step, we will create a simple **HTML** template as a base where we will use Bootstrap.

For that, the first thing you want to do is create a **folder** on your computer for the project files.

In this case, we will simply call it **bootstrap**.

Here, create a new text file and call it **index.html**. Open it with a text editor of your choice (e.g. Notepad++) and then use the code.

```
!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap Tutorial Sample Page</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

  </head>

  <body>
  </body>

</html>
```

2a. Load Bootstrap via CDN

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
crossorigin="anonymous">
```

https://getbootstrap.com/docs/3.3/getting-started/

2b. Host Bootstrap Locally

```
<link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
```

3. Include jQuery

- download jQuery, unzip, and put it into the project folder. Then, include it in the same place of your file in this way:

```
<script src="jquery-3.6.0.min.js"></script>
```

You can use it to load the library into your page by putting the line of code below right before where it says </body> on your page.

4. Load Bootstrap JavaScript

- The last step in setting up Bootstrap is to load the Bootstrap JavaScript library. However, we will load it in a different place than the style sheet. Instead of the header, it goes into the page footer, right after the call for jQuery.

```
<script src="bootstrap/js/bootstrap.min.js"></script>
```

Put it All Together

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap Tutorial Sample Page</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="bootstrap/css/bootstrap.min.css">
  </head>

  <body>
    <script src="jquery-3.6.0.min.js"></script>
    <script src="bootstrap/js/bootstrap.min.js"></script>
  </body>
</html>
```

5. Add a Navigation Bar

You can start by posting this just after the <body> tag:

```
<nav class="navbar navbar-expand-md">
        <a class="navbar-brand" href="#">Logo</a>
        <button class="navbar-toggler navbar-dark" type="button" data-toggle="collapse" data-target="#main-navigation">
                <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="main-navigation">
                <ul class="navbar-nav">
                        <li class="nav-item">
                                <a class="nav-link" href="#">Home</a>
                        </li>
                        <li class="nav-item">
                                <a class="nav-link" href="#">About</a>
                        </li>
                        <li class="nav-item">
                                <a class="nav-link" href="#">Contact</a>
                        </li>
                </ul>
        </div>
</nav>
```

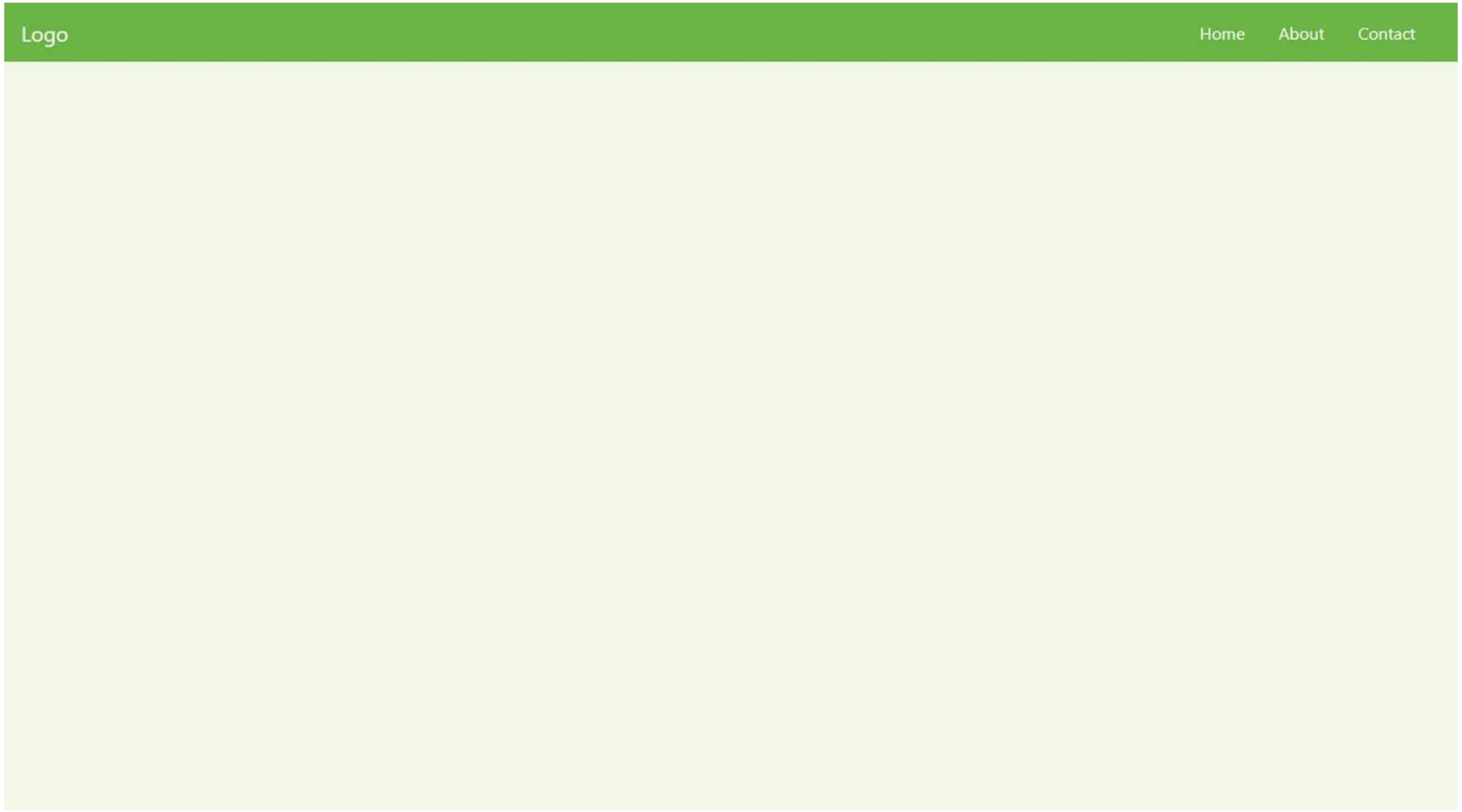
6. Include Custom CSS

- For that, simply create a blank file with your text editor and call it *main.css*. Save it, then add it to the head section of your Bootstrap site like this:

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

```
body {
   padding: 0;
   margin: 0;
   background: #f2f6e9;
}
.navbar {
   background:#6ab446;
}
.nav-link,
.navbar-brand {
   color: #fff;
   cursor: pointer;
}
.nav-link {
   margin-right: 1em !important;
}
.nav-link:hover {
   color: #000;
}
.navbar-collapse {
   justify-content: flex-end;
}
```

The output



7. Create a CTA Button

- we add the following markup right below the page content inside the <description> container:

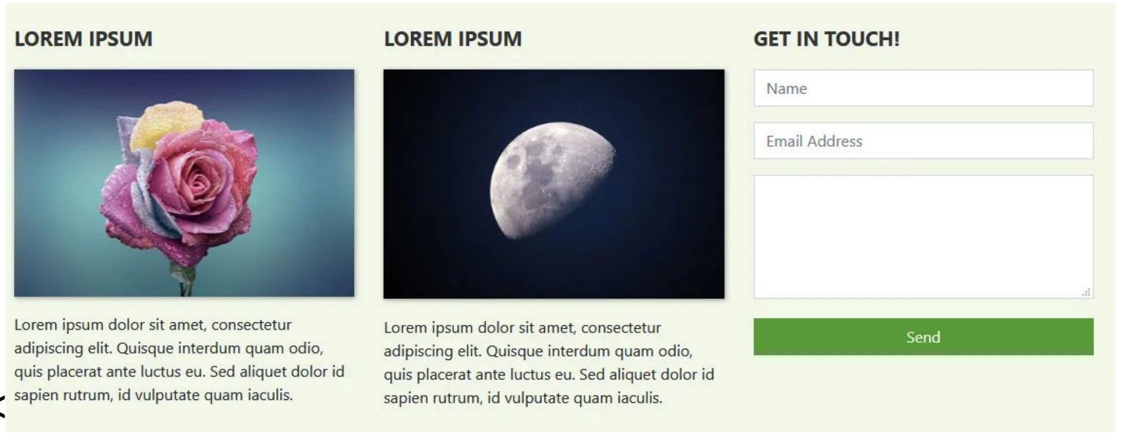
```
<button class="btn btn-outline-secondary btn-lg">Tell Me More!</button>
```

- In addition to that, we add this CSS to main.css:

```
.description button {
   border:1px solid #6ab446;
   background:#6ab446;
   border-radius: 0;
   color:#fff;
}
.description button:hover {
   border:1px solid #fff;
   background:#fff;
   color:#000;
}
```

8. Set Up a Three-Column Section

```
<div class="container features">
  <div class="row">
    <div class="col-lg-4 col-md-4 col-sm-12">
      <h3 class="feature-title">Lorem ipsum</h3>
      <img src="images/column-1.jpg" class="img-fluid">
      <p>text for sample para1</p>
    </div> /* 1st column  */
```



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.

```
<
      <h3 class="feature-title">Lorem ipsum</h3>
```

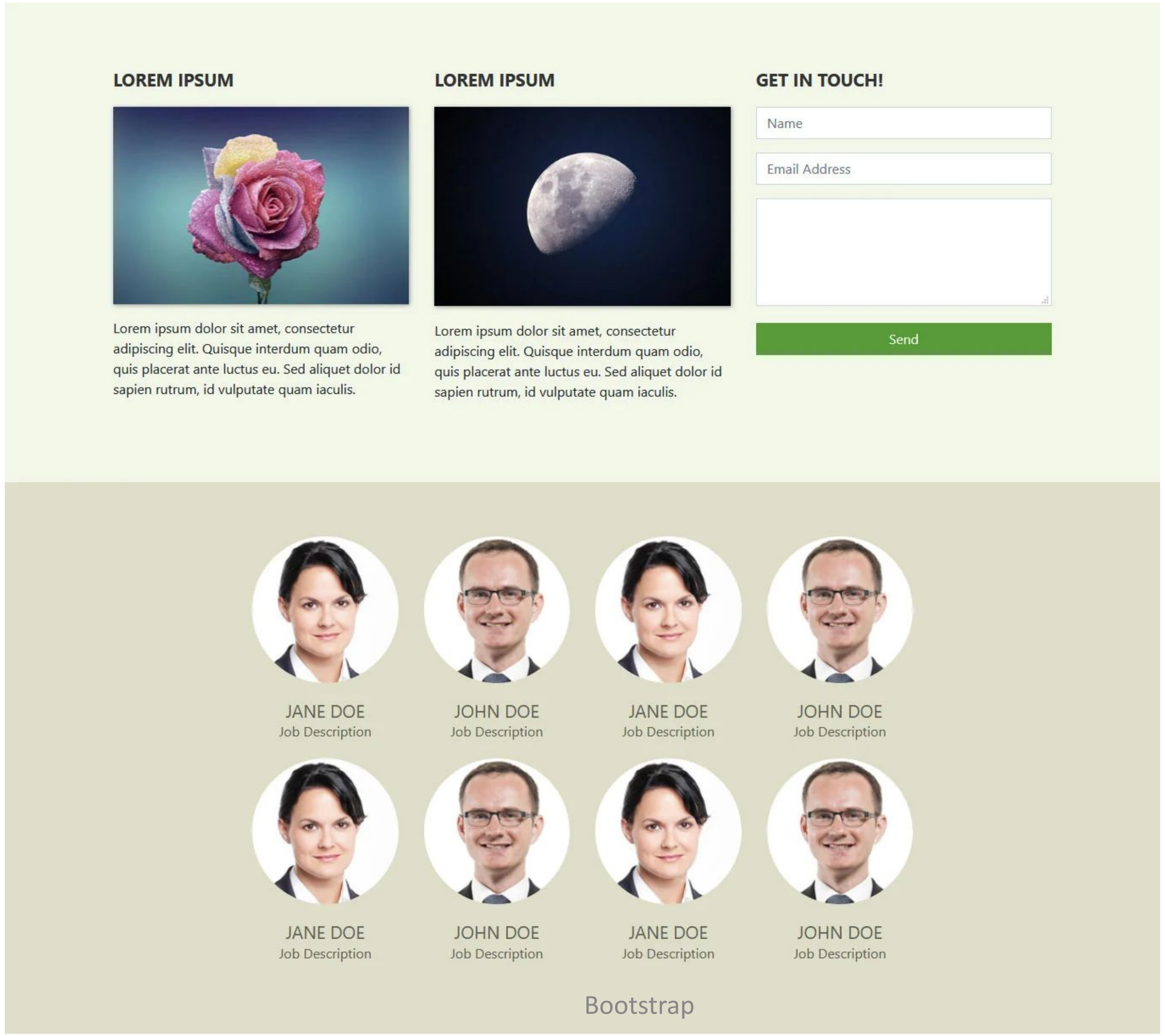
9. Add a Contact Form

Add in 3rd div of last slide…

```
<h3 class="feature-title">Get in Touch!</h3>
<div class="form-group">
        <input type="text" class="form-control" placeholder="Name" name="">
</div>
<div class="form-group">
   <input type="email" class="form-control" placeholder="Email Address" name="email">
</div>
<div class="form-group">
   <textarea class="form-control" rows="4"></textarea>
</div>
<input type="submit" class="btn btn-secondary btn-block" value="Send" name="">
```

```
.features .form-control,
.features input {
        border-radius: 0;
}
.features .btn {
   background-color: #589b37;
   border: 1px solid #589b37;
   color: #fff;
   margin-top: 20px;
}
.features .btn:hover {
   background-color: #333;
   border: 1px solid #333;
}
```

10. Include a Team Section (Card layouts)



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque interdum quam odio, quis placerat ante luctus eu. Sed aliquet dolor id sapien rutrum, id vulputate quam iaculis.

Step1

- we first include another <div> element as its background. Inside, there is another container with a custom class for more customizability, followed by a row.

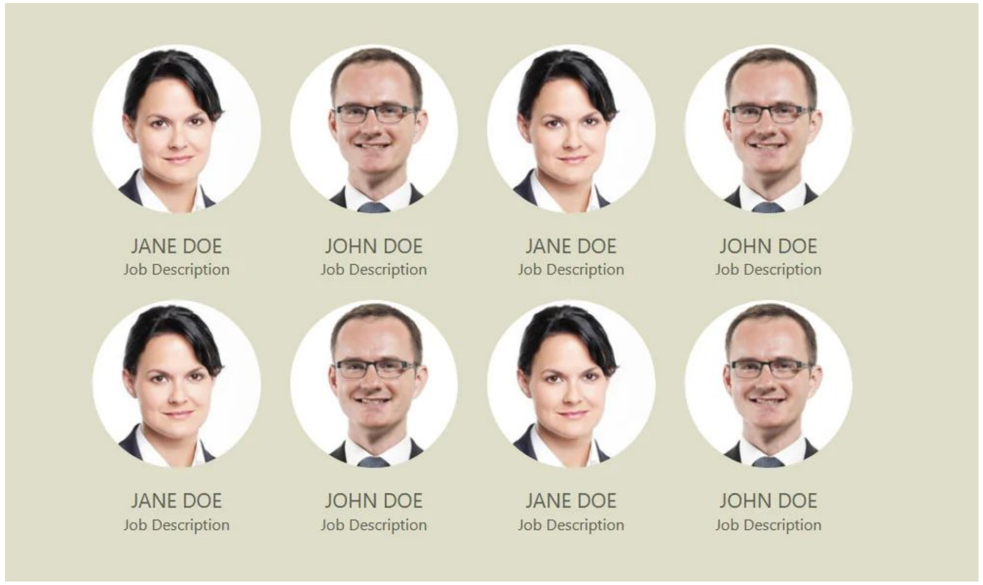
```
<div class="background">
   <div class="container team">
      <div class="row">

      </div>
   </div>
</div>
```

step2

- Cards can contain different elements. In our case, we want an image at the top, followed by a title (the name of the person) and a description that says what they do.

```
<div class="card col-lg-3 col-md-3 col-sm-4 text-center">
   <img class="card-img-top rounded-circle"
src="images/team-image-1.png" alt="Card image">
   <div class="card-body">
      <h4 class="card-title">Jane Doe</h4>
      <p class="card-text">Job Description</p>
   </div>
</div>
```

Step3: The CSS addition

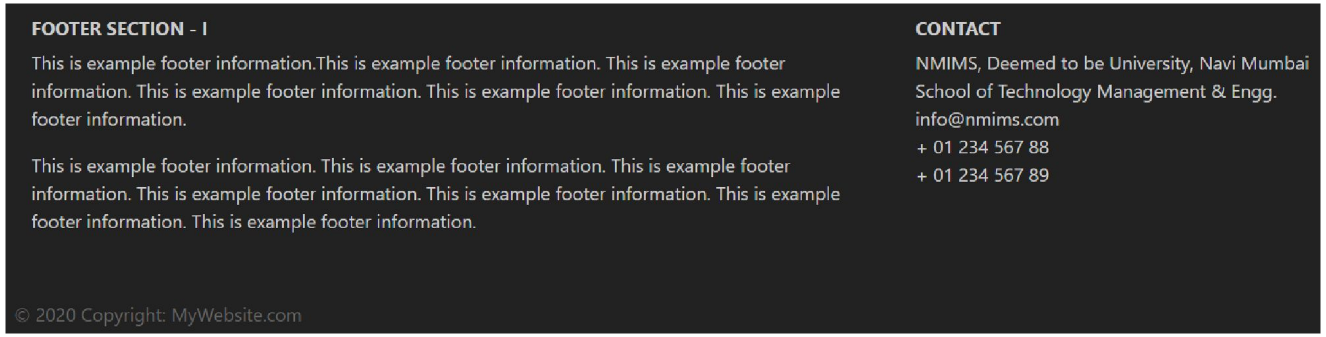


```
.background {
        background: #dedec8;
        padding: 4em 0;
}
.team {
        color: #5e5e55;
        padding: 0 180px;
}
.team .card-columns {
        -webkit-column-count: 4;
        -moz-column-count: 4;
        column-count: 4;
}


.team .card {
        background:none;
        border: none;
}
.team .card-title {
        font-size: 1.3rem;
        margin-bottom: 0;
        text-transform: uppercase;
}
```

11. Create a Two-Column Footer

```
<footer class="page-footer">
    <div class="container">
        <div class="row">
            <div class="col-lg-8 col-md-8 col-sm-12">
                <h6 class="text-uppercase font-weight-bold">Footer Section - I</h6>
                <p>Footer paragraph one. Footer paragraph one. Footer paragraph one.</p>
                <p>Footer paragraph two. Footer paragraph two. Footer paragraph two. </p>
        </div>
        <div class="col-lg-4 col-md-4 col-sm-12">
            <h6 class="text-uppercase font-weight-bold">Contact</h6>
            <p>NMIMS, Deemed to be University, Navi Mumbai
            <br/> School of Technology Management & Engg.
            <br/>info@mywebsite.com
            <br/>+ 01 234 567 88
            <br/>+ 01 234 567 89</p>
        </div>
    </div>

<div class="footer-copyright text-center">© 2020 Copyright: MyWebsite.com</div>
</footer>
```

```
.page-footer {
    background-color: #222;
    color: #ccc;
    padding: 60px 0 30px;
}
.footer-copyright {
    color: #666;
    padding: 40px 0;
}
```


FOOTER SECTION - I

This is example footer information.This is example footer information. This is example footer information. This is example footer information. This is example footer information. This is example footer information.

This is example footer information. This is example footer information. This is example footer information. This is example footer information. This is example footer information. This is example footer information. This is example footer information.

CONTACT

NMIMS, Deemed to be University, Navi Mumbai
School of Technology Management & Engg.
info@nmims.com
+ 01 234 567 88
+ 01 234 567 89

© 2020 Copyright: MyWebsite.com

12. Create Additional Pages

Find the about in html, replace the line with this:

```
<a class="nav-link" href="about.html">About</a>
```

Bootstrap Grids

- Bootstrap’s grid system allows up to 12 columns across the page.
- If you do not want to use all 12 columns individually, you can group the columns together to create wider columns:

```
<div class="col-md-12">Span 12 columns</div>

<div class="col-md-6">Span 6</div><div class="col-md-6">Span 6</div>

<div class="col-md-4">Span 4</div><div class="col-md-8">Span 8</div>

<div class="col-md-4">Span 4</div><div class="col-md-4">Span 4</div> <div class="col-md-4">Span
  4</div>
```

- Bootstrap's grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Grid Classes

- The Bootstrap grid system has four classes:
  - xs (for phones)
  - sm (for tablets)
  - md (for desktops)
  - lg (for larger desktops)
- The classes above can be combined to create more dynamic and flexible layouts.

Basic Structure of a Bootstrap Grid

```
<div class="row">
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  ...
</div>
```

- First; create a row (<div class="row">). Then, add the desired number of columns (tags with appropriate .col-*-*classes). Note that numbers in .col-*-* should always add up to 12 for each row.

Three Equal Columns

- Three equal columns (desktop version):
- Three equal columns (tablet version):
- Three equal columns (smart phone version):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Grid Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
  <script src="https://code.jquery.com/jquery-latest.js"></script>
</head>
<body>

<div class="container">
  <h1>Three Equal Columns</h1>
  <p>Desktops version: .col-md-4</p>

      <div class="row">
         <div class="col-md-4">.col-md-4</div>
         <div class="col-md-4">.col-md-4</div>
         <div class="col-md-4">.col-md-4</div>
      </div>
 </div>
</body>
</html>
```

Two Unequal Columns

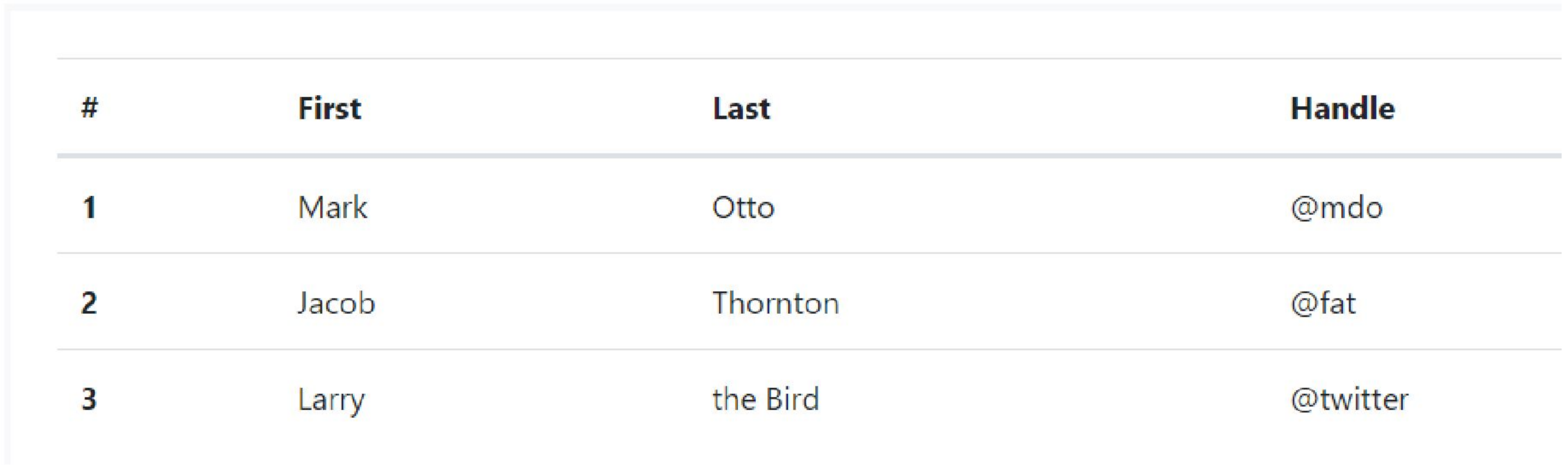
- Two unequal columns (desktop version):
- Two unequal columns (tablet version):
- Two unequal columns (smart phone version):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Grid Example</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
  <script src="https://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
<div class="container">
  <h1>Two Unequal Columns</h1>
  <p>Desktops version: .col-md-4 and .col-md-8</p>

  <div class="row">
        <div class="col-md-4">.col-md-4</div>
        <div class="col-md-8">.col-md-8</div>
  </div>
 </div>
</body>
</html>
```

Bootstrap Tables

- A **basic** Bootstrap table has a light padding and only horizontal dividers.
  - The .table class adds basic styling to a table:
- **Striped Rows**
  - The .table-striped class adds zebra-stripes to a table:
- **Bordered Table**
  - The .table-bordered class adds borders on all sides of the table and cells:
- **Hover Rows**
  - The .table-hover class enables a hover state on table rows:
- **Responsive Tables**
  - The .table-responsive class creates a responsive table. The table will then scroll horizontally on small devices (under 768px). When viewing on anything larger than 768px wide, there is no difference:

```
<table class="table">
  <thead>
    <tr>
      <th scope="col">#</th>
      <th scope="col">First</th>
      <th scope="col">Last</th>
      <th scope="col">Handle</th>
    </tr>
  </thead>

<tbody>
    <tr>
      <th scope="row">1</th>
      <td>Mark</td>
      <td>Otto</td>
      <td>@mdo</td>
    </tr>
    <tr>
      <th scope="row">2</th>
      <td>Jacob</td>
      <td>Thornton</td>
      <td>@fat</td>
```



#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry	the Bird	@twitter

For dark theme, use class="table table-dark"

Bootstrap Images

- Rounded Corners
  - The .img-rounded class adds rounded corners to an image (IE8 does not support rounded corners):
- Circle
  - The .img-circle class shapes the image to a circle (IE8 does not support rounded corners):
- Thumbnail
  - The .img-thumbnail class shapes the image to a thumbnail:
- Responsive Images
  - Images comes in all sizes. So do screens. Responsive images automatically adjust to fit the size of the screen.
  - Create responsive images by adding an .img-responsive class to the <img> tag. The image will then scale nicely to the parent element.
  - The .img-responsive class applies display: block; and max-width: 100%; and height: auto; to the image:

Bootstrap Buttons

- Button Styles
  - Bootstrap provides seven styles of buttons with the following classes:

.btn-default

.btn-primary

.btn-success

.btn-info

.btn-warning

.btn-danger

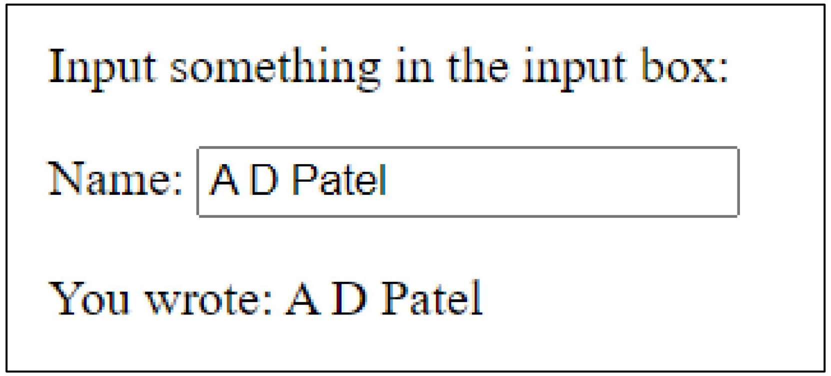
.btn-link

Angular JS

https://angularjs.org/

First Example: Angular JS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
  <div ng-app=" " ng-init=" firstName='A D Patel'">
    <p>Input something in the input box:</p>
    <p>Name: <input type="text" ng-model="firstName"></p>
    <p>You wrote: {{ firstName }}</p>
  </div>
</body>
</html>
```



Input something in the input box:

Name: A D Patel

You wrote: A D Patel

Introduction

- **Angular** is a development platform, built on **TypeScript**.
- As a platform, Angular includes:
  - A **component-based framework** for building scalable web applications
  - A collection of well-integrated **libraries** that cover a wide variety of features, including routing, forms management, client-server communication, and more
  - A suite of developer tools to help you **develop**, **build**, **test**, and **update** your code
- With Angular, you're taking advantage of a platform that can **scale** from single-developer projects to enterprise-level applications.
- Angular is designed to make updating as **straightforward** as possible.
- The Angular ecosystem consists of a diverse group of over **1.7 million** developers, library authors, and content creators.

Cont’d

- Angular JS is an **open source** JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone.
- Angular JS is maintained by Google.
- Following are the advantages of AngularJS over other JavaScript frameworks:
  - **Dependency Injection:** Dependency Injection specifies a design pattern in which components are given their dependencies instead of hard coding them within the component.
  - **Two way data binding:** AngularJS creates a two way data-binding between the select element and the orderProp model. orderProp is then used as the input for the orderBy filter.
  - **Testing:** Angular JS is designed in a way that we can test right from the start. So, it is very easy to test any of its components through unit testing and end-to-end testing.
  - **Model View Controller:** In Angular JS, it is very easy to develop application in a clean MVC way. You just have to split your application code into MVC components i.e. Model, View and the Controller.

It’s all to avoid postback…

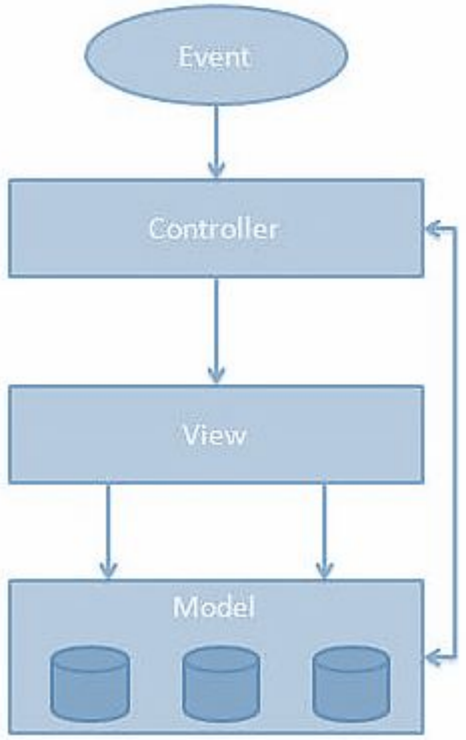
- Check the Angular JS website.

AngularJS MVC Architecture

- MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.

The MVC pattern is made up of the following three parts:

- **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
- **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
- **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.



```
<!DOCTYPE html>
<html lang="en">
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.5/angular.min.js">  </script>
</head>

<body ng-app="myapp">

<div ng-controller="HelloController" >
<h2>Hello {{helloTo.title}} !</h2>
</div>
```

View Part

```
<script>
angular.module("myapp", [])
   .controller("HelloController", function($scope) {
      $scope.helloTo = {};
      $scope.helloTo.title = "World, AngularJS";
   } );
</script>
```

Controller Part

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

AngularJS Directives

- AngularJS *directives* are used to extend HTML. Directives are markers on HTML DOM element that tell AngularJS to attach a *specified behavior* to that HTML element. Directives in AngularJS are special attributes starting with *ng- prefix* where *ng* stands for Angular. AngularJS includes various *built-in* directives, you may also create your *own directive* in AngularJS. Some built-in directives are listed here.

Directive	Description
ng-app	This directive starts an AngularJS Application
ng-init	This Initializes AngularJS variables i.e. application data.
ng-repeat	This directive repeats HTML elements for each item in a collection.
ng-model	This directive binds the value of HTML controls (input, select, textarea) to application data.
ng-bind	This directive Replaces the value of HTML control with the value of specified AngularJS expression.
Other built-in directive-	
ng-controller	Attaches the controller of MVC to the view.
ng-show	Display HTML element based on the value of the specified expression.
ng-readonly	Makes HTML element read-only based on the value of the specified expression.
ng-disabled	Sets the disable attribute on the HTML element if specified expression evaluates to true.
ng-if	Removes or recreates HTML element based on an expression.
ng-click	Specifies custom behavior when an element is clicked.

ng-app directive

- The ng-app directive defines the root element of an AngularJS application and starts an AngularJS Application.
- The ng-app directive will automatically initialize the application when a web page is loaded.
- It is also used to load various AngularJS modules in AngularJS Application.

Example

```
<div ng-app = "">
...
</div>
```

In the above example code, we have defined a default AngularJS application using **ng-app** attribute of a **<div> element**.

ng-init directive

- ng-init directive initializes an AngularJS Application data. It is used to declare and assign values to the variables for an AngularJS application.

In following example, we initialize variable firstName and lastName and assign values to it.

```
<div ng-app = "" ng-init="firstName='A D';lastName=‘Patel' ">
...
</div>
```

ng-model directive

- The ng-model directive is used for two-way data binding in AngularJS. It is used to get value of input controls like textbox, label, etc and use these values in web pages.

In the following example, we define a model named my*name*.

```
<div ng-app = " ">
...
<input type = "text" ng-model = "myname">
<p>My name is(enter it): {{myname}} </p>

</div>
```

ng-repeat directive

- The ng-repeat directive repeats HTML elements for each item in a collection. Or simply say that it is used to loop through items in collection element and it will act as for loop. In the following example, we iterate over the name of the students:

```
<div ng-app="" ng-init="students=[‘Ramesh',‘Mahesh',‘Suresh']">
  <ul>
    <li ng-repeat="name in students">
      {{ ‘Student Name: ’ +name }}
    </li>
  </ul>
</div>
```

ng-bind directive

- The ng-bind directive binds the model property declared via ng-model directive or the result of an expression to the HTML element. It also updates an element if the value of an expression changes.

```
<html >
  <head>
    <title>
      ng-bind Example
    </title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
</head>
<h1> Showing ng-bind Directive</h2>
<body ng-app="">
<div>
  Sum of 10 + 5 = <span ng-bind="10 + 5"></span> <br />
  Please Enter your Mobile no: <input type="text" ng-model="mobileno" /><br />
  Hi, my Mobile no is <span ng-bind="mobileno"></span>
</div>
</body>
</html>
```

AngularJS Expressions

- AngularJS expressions can be written inside double braces: *{{ expression }}*.
- AngularJS expressions can also be written inside a directive: *ng-bind="expression".*
- AngularJS will resolve the expression, and *return the result* exactly where the expression is written.
- AngularJS expressions are much like JavaScript expressions: They can contain *literals, operators,* and *variables*.
- Example *{{ 5 + 5 }}* or *{{ firstName + " " + lastName }}*

```
<!DOCTYPE html>
<html>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">    </script>
<body>
 <div ng-app>
  <p> Expression: {{ 5 + 5 }} </p>
 </div>
</body>
</html>
```

If you remove the ng-app directive, HTML will display the expression as it is, without solving it

Change color of input box

- You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.
- Example: Let AngularJS change the value of CSS properties.
- Change the color of the input box below, by changing its value

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
 <p>Change the value of the input field:</p>
  <div ng-app="" ng-init="myCol='lightblue'">
    <input style="background-color:{{myCol}}" ng-model="myCol">
  </div>
 <p>AngularJS resolves the expression and returns the result.</p>
 <p>The background color of the input box that you write in the input field.</p>
</body>
</html>
```

AngularJS Numbers

- AngularJS numbers are like JavaScript numbers

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
  <div ng-app="" ng-init="quantity=3;cost=5">
  <p>Total in dollar: {{ quantity * cost }}</p>
  </div>
</body>
</html>
```

- The same can be achieved using *ng-bind*

```
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar: <span ng-bind="quantity * cost"></span></p>
</div>
```

AngularJS Strings

- String example

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='NMIMS';lastName='Navi Mumbai'">
<p> Example: {{ firstName + " " + lastName }} </p>
</div>
</body>
</html>
```

- Same using *ng-bind*

```
<div ng-app="" ng-init="firstName='NMIMS';lastName='Navi Mumbai'">
<p>The name is <span ng-bind="firstName + ' ' + lastName"></span></p>
</div>
```

AngularJS Objects

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="person={firstName:‘NMIMS',lastName:‘Mumbai'}">
<p>My name is {{ person.firstName }}</p>
</div>
</body>
</html>
```

Same with ng-bind

```
<div ng-app="" ng-init="person={firstName:‘NMIMS',lastName:‘Navi Mumbai'}">
<p>The name is <span ng-bind="person.firstName"></span></p>
</div>
```

AngularJS Arrays

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The third element is {{ points[2] }}</p>
</div>
</body>
</html>
```

Same using ng-bind:

```
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The first result is <span ng-bind="points[0]"></span></p>
</div>
```

Cost Calculator using Angular JS

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div data-ng-app="" data-ng-init="quantity=1;price=20">
<h2>Cost Calculator</h2>
Quantity: <input type="number" ng-model="quantity">
Price: <input type="number" ng-model="price">
<p><b>Total in rupees:</b> {{quantity * price}}</p>
</div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.
min.js">
</script>
<body>
<div ng-app>
<p>A simple expression example: {{ 5 + 5 }}</p>
</div>
</body>
</html>
```

Changing the color dynamically

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<p>Change the value of the input field:</p>
<div ng-app="" ng-init="myCol='pink'">
<input style="background-color:{{myCol}}" ng-model="myCol" value="{{myCol}}">
</div>
<p>AngularJS resolves the expression and returns the result.</p>
<p>The background color of the input box will be whatever you write in the input field.</p>
</body>
</html>
```

ng-list (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="">
<p>Write some names in the input field, use a comma to separate them:</p>
<input ng-model="customers" ng-list/>
<p>This example will convert your input into an array, one item for each name:</p>
<pre>{{customers}}</pre>
</div>
</body>
</html>
```

ng-keydown (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<p>Press any key within the input box.</p>
<input ng-keydown="count = count + 1" ng-init="count=0" />
<h1>{{count}}</h1>
<p>Every time you press a key in the input field, this will increase the value of the variable "count".</p>
</body>
</html>
```

ng-style (directive)

```
<!DOCTYPE html>
<html>
<script src="   "></script>
<body ng-app="myApp" ng-controller="myCtrl">
<h1 ng-style="myObj">Welcome to NMIMS Navi Mumbai!</h1>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
  $scope.myObj = {
    "color" : "white",
    "background-color" : "brown",
    "font-size" : "40px",
    "padding" : "40px"
  }
});
```

ng-required (directive)

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body ng-app="">
<form name="myForm">
Click here to make the input field required:
<input type="checkbox" ng-model="myVar"><br><br>
<input name="myInput" ng-model="myInput" ng-required="myVar">
<h1 ng-if="!myForm.myInput.$valid">The input field cannot be empty</h1>
</form>
<p><strong>Note:</strong> This example shows an error if the input field is empty <strong>and</strong>
 the checkbox is checked for "required".</p>
</body>
</html>
```

ng-submit (directive)

```
<!DOCTYPE html>
<html>
<script src="   "></script>
<body ng-app="myApp" ng-controller="myCtrl">
<form ng-submit="myFunc()">
  <input type="text">
  <input type="submit">
</form>
<p>{{myTxt}}</p>
<p>ng-submit example.</p>



<script>
var app = angular.module("myApp", []);
```

AngularJS Modules & Controllers

Introduction

- An AngularJS module defines an application.
- The angular object's module() method is used to create a module.
- It is also called AngularJS function angular.module

```
<div ng-app="myApp">...</div>

<script>

var app = angular.module("myApp", []);

</script>
```

- The "myApp" parameter refers to an HTML element in which the application will run.
- Now we can add controllers, directives, filters, and more, to AngularJS application.

How to add controller to a module

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "NMIMS";
    $scope.lastName = "Navi Mumbai";
});
</script>
</body>
</html>
```

How to add directive to a module

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>

<div ng-app="myApp" w3-test-directive></div>

<script>
var app = angular.module("myApp", []);
app.directive("w3TestDirective", function() {
    return {
        template : "This is a directive constructor. "
    };
});
</script>
</body>
</html>
```

Modules and controllers in file

Example.html

```
<!DOCTYPE html>
<html>
<script src=“cdn-angular.min.js"></script>
<body>
<script src="myApp.js"></script>
<script src="myCtrl.js"></script>
</body>
</html>
```

myApp.js **(for controller)**

What is Scope?

- The Scope is an object that is specified as a binding part between the HTML (view) and the JavaScript (controller).
- It plays a role of joining controller with the views. It is available for both the view and the controller.
- To make a controller in AngularJS, you have to pass the $scope object as an argument.

```
<div ng-app="myApp" ng-controller="myCtrl">
<h1>{{carname}}</h1>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.carname = "Volvo";
});
</script>
```

AngularJS Filters

Filter	**Description**
Currency	It formats a number to a currency format.
Date	It formats a date to a specified format.
Filter	It select a subset of items from an array.
Json	It formats an object to a Json string.
Limit	It is used to limit an array/string, into a specified number of elements/characters.
Lowercase	It formats a string to lower case.
Number	It formats a number to a string.
Orderby	It orders an array by an expression.
Uppercase	It formats a string to upper case.

How to add filters to expressions

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="personCtrl">
<p>The name is {{ firstName | uppercase }}</p>
</div>
<script>
angular.module('myApp', []).controller('personCtrl', function($scope) {
    $scope.firstName = "nmims",
    $scope.lastName = "navi mumbai"
});
</script>
</body>
</html>
```

AngularJS Validation

- AngularJS provides client-side form validation. It checks the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.

- It also holds the information about whether the input fields have been touched, or modified, or not.

- Following directives are generally used to track errors in an AngularJS form:

  $dirty - states that value has been changed.
  $invalid - states that value entered is invalid.
  $error - states the exact error.

Example: AJS Validation

Required field Validation

```
<input name = "firstname" type = "text" ng-model = "firstName" required>
<span style = "color:red" ng-show = "studentForm.firstname.$dirty && studentForm.firstname.$invalid">
    <span ng-show = "studentForm.firstname.$error.required">First Name is required.</span>
 </span>
```

Email Validation

```
<input name = "email" type = "email" ng-model = "email" length = "100" required>
<span style = "color:red" ng-show = "studentForm.email.$dirty && studentForm.email.$invalid">
    <span ng-show = "studentForm.email.$error.required">Email is required.</span>
    <span ng-show = "studentForm.email.$error.email">Invalid email address.</span>
</span>
```

Click for Working Example