Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-01

Introduction to WEB

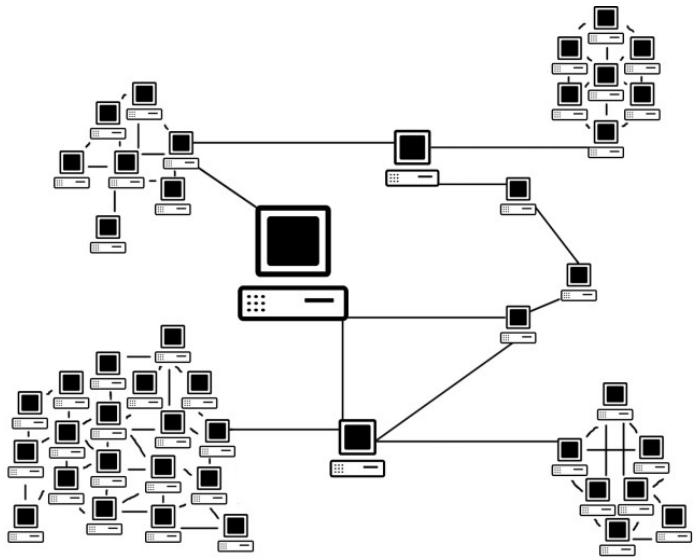
Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline

- ✓ Introduction to WEB
  - ✓ What is Internet?
  - ✓ What is WWW?
- ✓ How the WEB works?
- ✓ HTTP Protocol
- ✓ Client Server architecture
- ✓ Web Servers
- ✓ Installing Web Server
- ✓ Configuring Web Server

What is Internet?

- The Internet is a massive **network of networks**, a networking infrastructure.
- It connects millions of computers together globally, forming a network in which any computer can communicate with any other computer as long as they are both connected to the Internet.
- Information that travels over the Internet uses many different set of rules which are known as **protocols**.



What is WWW?

- WWW stands for **W**orld **W**ide **W**eb.
- A technical definition of the WWW is – All the resources and users on the Internet that are using HTTP.
- HTTP stands for **H**yper**t**ext **T**ransfer **P**rotocol which is a text-based request-response protocol.
- HTTP is an application layer protocol that allows web-based applications to communicate and exchange the data.
- HTTP is a TCP/IP based protocol.
- After making the request, the client disconnects from the server, then when the response is ready the server re-establish the connection again and deliver the response, so it is a connectionless and stateless protocol.
- HTTP is the protocol being used to transfer hypertext documents that makes the World Wide Web possible.

How the Web Works?

- World Wide Web (WWW) use classical client / server architecture.

Client running a Web Browser

HTTP

Page request

HTTP

Server response

Server running **Web Server** Software (Apache, IIS, Tomcat, etc.)

HTTP request

- The HTTP request message consist of following,
  - A request line (e.g. GET /index.php HTTP1.1)
  - Request header fields (e.g. Accept-Language: en)
  - An empty line (CRLF)
  - An optional message body
- The request line and other header fields must end with CRLF (Carriage return, Line Feed) (/r/n)
- A **request line** contains the **method** of request followed by the **resource** we want and at the end protocol **version** used.
  - HTTP Request Methods: GET, POST, PUT, DELETE etc...
- There are many **request header fields** available with HTTP Request, some of are listed below
  - **Accept** : Media type(s) that is/are acceptable for the response. (e.g. Accept: text/html)
  - **Accept-Charset**: Character sets that are acceptable. (e.g. Accept-Charset: utf-8)
  - **Date**: The date and time at which the message was originated (e.g. Date: Tue, 15 Nov 1994 08:12:31 GMT)
  - **Host**: The domain name of the server (e.g. Host: darshan.ac.in)
  - **User-Agent**: details of the browser used (e.g. User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36)

HTTP Request (Example)

GET /index.php HTTP/1.1		index.php is requested from server using GET method of HTTP version 1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36		Version 5.0 of mozila browser is used on windows 10 (64 bit) while requesting the page from the server.
Host: www.darshan.ac.in		Host of the requested page is www.darshan.ac.in
Accept-Language: en-us		Client accepts US English locale while receiving the response from the server.
Accept: text/html		Client accepts text file containing the HTML in it while receiving the response from the server.

HTTP Response

- After receiving and interpreting a HTTP request message, a server responds with an HTTP response message.
- The HTTP response message consist of following,
  - Status-Line (format= HTTP-Version SP Status-Code SP Reason-Phrase CRLF)
  - *((general-header | response-header | entity-header) CRLF)
  - An empty line (CRLF)
  - An optional message body
- Status-Line consist of
  - HTTP-Version, which can be HTTP/1.1
  - Status-Code is a 3 digit code which is in below format
    - **1**xx: **Informational** - Request received, continuing process
    - **2**xx: **Success** - The action was successfully received, understood, and accepted
    - **3**xx: **Redirection** - Further action must be taken in order to complete the request
    - **4**xx: **Client Error** - The request contains bad syntax or cannot be fulfilled
    - **5**xx: **Server Error** - The server failed to fulfill an apparently valid request
  - Reason-Phase is a textual representation of the status code in human readable format.

HTTP Status Codes with reason phrase

"100": Continue
"101": Switching Protocols
"200": OK
"201": Created
"202": Accepted
"203": Non-Authoritative Information
"204": No Content
"205": Reset Content
"206": Partial Content
"300": Multiple Choices
"301": Moved Permanently
"302": Found
"303": See Other
"304": Not Modified
"305": Use Proxy
"307": Temporary Redirect
"400": Bad Request
"401": Unauthorized
"402": Payment Required
"403": Forbidden
"404": Not Found
"405": Method Not Allowed
"406": Not Acceptable
"407": Proxy Authentication Required
"408": Request Time-out
"409": Conflict
"410": Gone
"411": Length Required
"412": Precondition Failed
"413": Request Entity Too Large
"414": Request-URI Too Large
"415": Unsupported Media Type
"416": Requested range not satisfiable
"417": Expectation Failed
"500": Internal Server Error
"501": Not Implemented
"502": Bad Gateway
"503": Service Unavailable
"504": Gateway Time-out
"505": HTTP Version not supported

Refer: https://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html for more details on HTTP Status Code

HTTP Response (Example)

HTTP/1.1 200 OK		Response is 200 status code with OK message using HTTP1.1
Date: Mon, 27 Jul 2009 12:28:53 GMT		Response Date & Time
Server: Apache/2.2.14 (Win32)		Webserver used by server is Apache and version is 2.2.14 built for 32bit OS
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT		Last modified at Date & Time.
Content-Length: 88		Content size of the response is 88 bytes
Content-Type: text/html		Content type of the response is text file containing HTML

Web Server

- A **web server** is server **software**, that can satisfy client HTTP requests on the public World Wide Web or also on private LANs and WANs.
- The primary function of a web server is to store, process and deliver web pages to clients.
- This primary function definition was good a few decades ago but nowadays it is better to use the terms of **Web contents** and / or **Web resources** instead of Web Pages because it cover all kind of contents that can be delivered to clients by web server.
- Examples of Web contents may be HTML files, XHTML files, image files, style sheets, scripts, other types of generic files that may be downloaded by clients, etc…
- A user agent, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so.
- Commonly used web servers,
  - For PHP: Apache
  - For ASP: IIS
  - For JSP: Tomcat, Glassfish

Apache web server

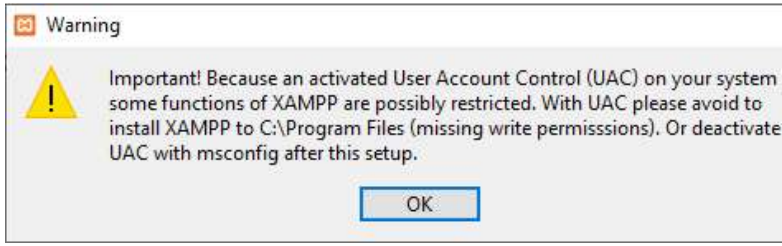
- The Apache HTTP Server ("httpd") was launched in 1995 and it has been the most popular web server on the Internet since April 1996.
- We can download and install apache web server separately from http://httpd.apache.org/download.cgi
- We can also download Apache web server with bundle like XAMPP, WAMP or LAMP.
  - XAMPP is the most popular PHP development environment consist of Apache, MariaDB, PHP, Perl and many other packages.
  - WAMP is popular PHP development environment for Windows OS.
  - LAMP is popular PHP development environment for Linux based OS.
- We are going to use XAMPP bundle as it has installable versions of Windows as well as Linux.
- We can download XAMPP and install XAMPP from https://www.apachefriends.org/index.html
- In the next few slides we are going to see how to install and configure the XAMPP.

Installing XAMPP

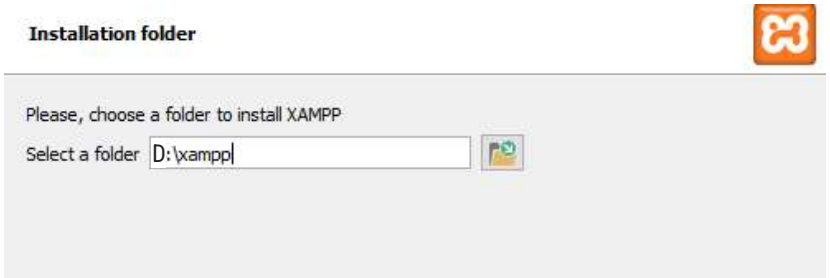
- Step 01: Download XAMPP installable from https://www.apachefriends.org/index.html

Installing XAMPP (Cont.)

- Step 02: start the installation by double clicking on the downloaded package
- Note: you might get this warning before installation begins, which suggest that UAC policy may restrict XAMPP to perform some functions if we install it in C drive.

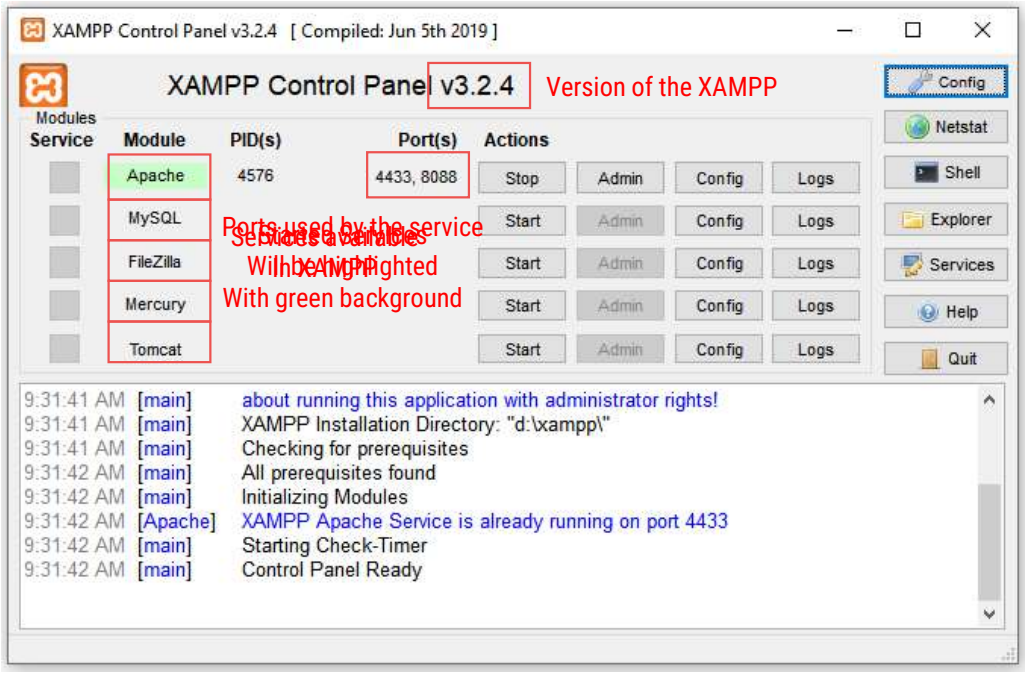


- Press OK in this warning, as we are not going to install XAMPP on C drive.
- After some screen you will get below screen, change the destination to other then C Drive.

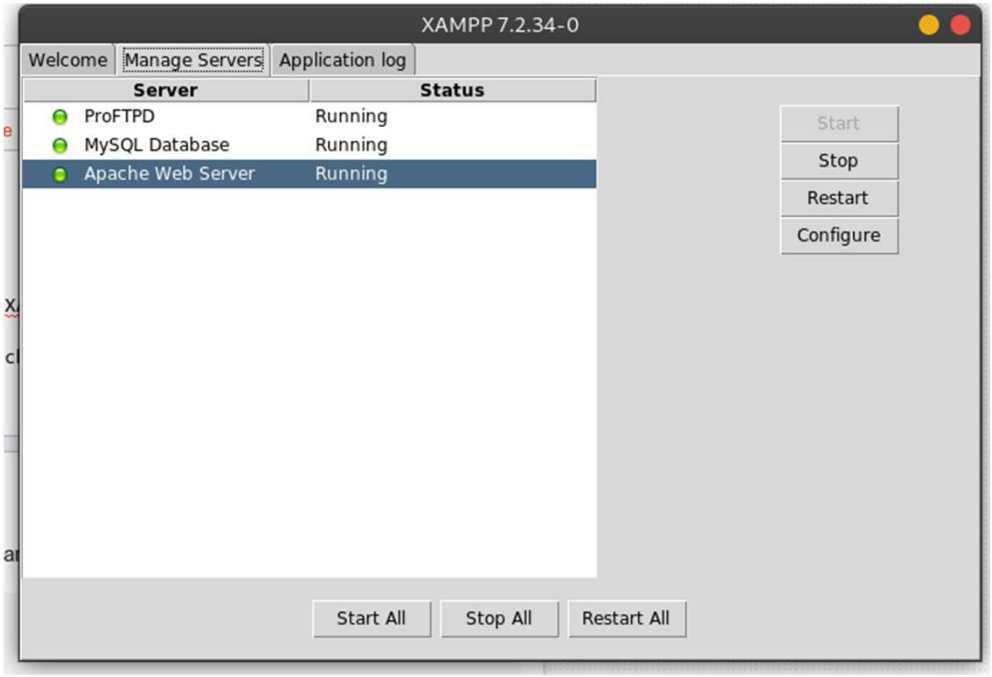


- After changing the destination of installation, just follow the default installation process.

XAMPP Control Panel



XAMPP Control Panel (Windows)



XAMPP Control Panel (Linux)

Configuring Server (Apache)

- Sometimes we need to change the server configuration in some specific conditions like,
  - Port is occupied
  - To add/load modules
  - To change max upload size
  - To change root directory
  - Etc…
- We can configure Apache server using **httpd.conf** file located at **xampp/apache/conf** folder.
- We can configure some parameters of the server using **php.ini** file located at **xampp/php** folder.
- Important settings in **httpd.conf** file are,
  - Changing **port number** for apache : to change the port number for apache go to httpd.conf file and change Listen:80 to any other available port number, we can also change port for https using httpd-ssl.conf file.
  - **Loading/Unloading Modules** in apache : to load/unload modules in apache we can go to httpd.conf and remove/add the comment code(#) from the module we want to load/unload.
  - **Changing Root Directory**: to change root directory we can change **DocumentRoot** and **<Directory>** tag to point new directory.

Configuring Server (Apache) (Cont.)

- Important settings in **php.ini** file are,
  - **`upload_max_filesize`** setting is for the maximum allowed size for uploaded files in the scripts.
  - **`post_max_size`** setting is for the maximum allowed size of POST data that PHP will accept.
  - **`error_reporting = E_ALL & ~E_NOTICE`** setting has default values as `E_ALL` and `~E_NOTICE` which shows all errors except notices.
  - **`max_execution_time`** = 30, Maximum execution time is set to seconds for any script to limit the time in production servers.
  - **`mysql.default_host = hostname`** setting is done to connect to MySQL with default server if no other server host is mentioned.
  - **`mysql.default_user = username`** setting is done to connect MySQL with default username, if no other name is mentioned.
  - **`mysql.default_password = password`** setting is done to connect MySQL with default password if no other password is mentioned.

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-02

WEB Design

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline

- ✓ Concepts of Effective Web Design
- ✓ Web Design Issues
- ✓ Planning a website
- ✓ Effective Navigation.

Concept of Effective Web Design

- It's a good idea to first think about and design your site. That way, you'll give yourself direction and you'll need to reorganize less later.
- To design your site:
  1. Figure out why you're creating this site. What do you want to convey?
  2. Think about your audience. How can you tailor your content to appeal to this audience? For example, should you add lots of graphics or is it more important that your page download quickly?
  3. How many pages will you need? What sort of structure would you like it to have? Do you want visitors to go through your site in a particular direction, or do you want to make it easy for them to explore in any direction?
  4. Sketch out your site on paper.
- Devise a simple, consistent naming system for your pages, images and other external files.

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 & e) Page Layout and Linking

- 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…
- 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 & g) Making Design user-centric

- 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
- 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?

Publishing Website

- This topic will be covered in detail once we complete PHP.

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, or **graphical**, i.e. a image.
  - 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.

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-03

Basics of HTML and CSS

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline HTML

- ✓ Introduction to HTML
  - ✓ What is a Web Page?
  - ✓ My First HTML Page
  - ✓ HTML Code Formatting
- ✓ Basic HTML Tags
  - ✓ Heading
  - ✓ Paragraph
  - ✓ Color
  - ✓ Font
  - ✓ List
  - ✓ Anchor
  - ✓ Image
- ✓ HTML Tables
- ✓ HTML Forms
- ✓ HTML Meta tags
- ✓ Media tags
- ✓ HTML 5 tags and validation

Outline CSS

- ✓ What is CSS?
- ✓ Syntax and structure of CSS
- ✓ CSS rules for Backgrounds, Colors and properties
- ✓ Manipulating texts, Fonts
- ✓ The Box Model
  - ✓ borders and boxes
  - ✓ Margins and Padding
- ✓ Lists
- ✓ CSS Positioning and Float Property
- ✓ Animations
- ✓ Tooltip
- ✓ Style images
- ✓ Variables
- ✓ Media Queries
- ✓ Wildcard Selectors (*, ^ and $) in CSS
- ✓ Working with Gradients, Pseudo Class
- ✓ Pseudo elements
- ✓ basic of frameworks like Bootstrap

What is a Web Page?

- A webpage is a document, commonly written in HTML, that is viewed in an Internet browser.
- HTML – Hyper Text Markup Language is the notation for describing
  - document structure (semantic markup)
  - formatting (presentation markup)
- A web page can be accessed by entering a URL into a browser's address bar.
- A web page may contain text, graphics, and hyperlinks to other web pages and files.
- The first web page was created at CERN by Tim Berners-Lee on August 6, 1991.
- You can visit and browse the first website and the first web page at the info.cern.ch address.

Creating HTML Pages

- An HTML file must have an .htm or .html file extension
- HTML files can be created with text editors:
  - NotePad, NotePad ++, PSPad
- Or HTML editors (WYSIWYG Editors):
  - Microsoft FrontPage
  - Macromedia Dreamweaver
  - Netscape Composer
  - Visual Studio
- Open any above mentioned editors and create a new file with .html extension and save the file.
- After saving the file you can open the file with any Web Browser in order to view the output.

First HTML Page

test.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>My First HTML Page</title>
  </head>
  <body>
     <p>This is some text...</p>
  </body>
</html>
```

HTML Structure

- HTML is comprised of “elements” and “tags”
  - Begins with `<html>` and ends with `</html>`
- Elements (tags) are nested one inside another:

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

- Tags have attributes:

```
<img src="logo.jpg" alt="logo" />
```

- HTML describes structure using two main sections: `<head>` and `<body>`
- The HTML source code should be formatted to increase readability and facilitate debugging.
  - Every block element should start on a new line.
  - Every nested (block) element should be indented.
  - Browsers ignore multiple whitespaces in the page source, so formatting is harmless.
- For performance reasons, formatting can be sacrificed

First HTML Page

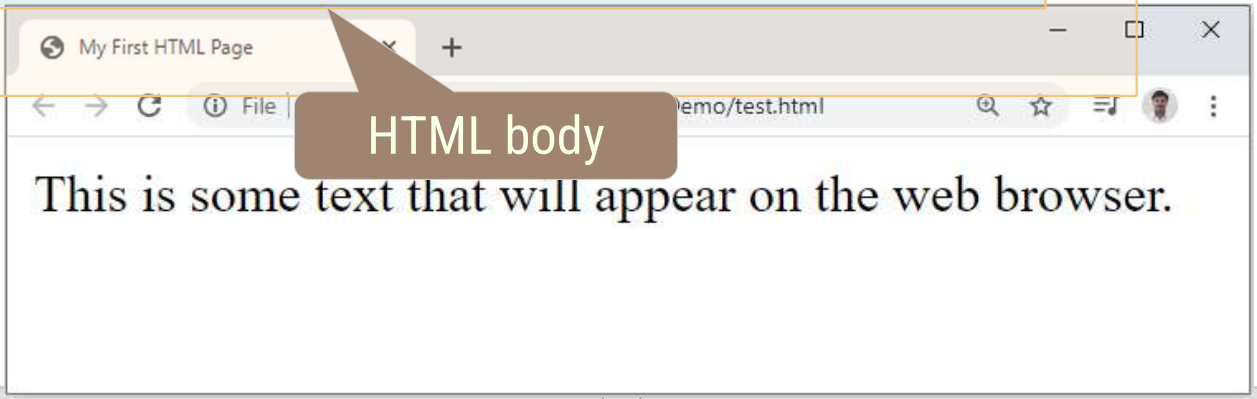
```
<!DOCTYPE HTML>
<html>
  <head>
    <title>My First HTML Page</title>
  </head>
  <body>
     <p>This is some text...</p>
  </body>
</html>
```

HTML header

Opening tag

Closing tag

HTML body



Basic HTML Tags

1. Headings
2. Paragraph
3. Colors
4. Fonts
5. List
6. Anchor Tag
7. Image
8. Table
9. Form

1) Headings

- Headings are important because search engines use the headings to index the structure and content of your web pages.

<h1> text </h1> -- largest of the six
<h2> text </h2>
<h3> text </h3>
<h4> text </h4>
<h5> text </h5>
<h6> text </h6> -- smallest of the six

align="*position*" --left (default), center or right

2) Paragraph

- The HTML element represents a paragraph.
- Paragraphs are usually represented in visual media as blocks of text separated from adjacent blocks by blank lines and/or first-line indentation, but HTML paragraphs can be any structural grouping of related content, such as images or form fields.
- Paragraphs are block-level elements, and notably will automatically close if another block-level element is parsed before the closing tag.
- We can use align attribute of the paragraph tag to specify the text alignment for the text inside the paragraph, ex. our test

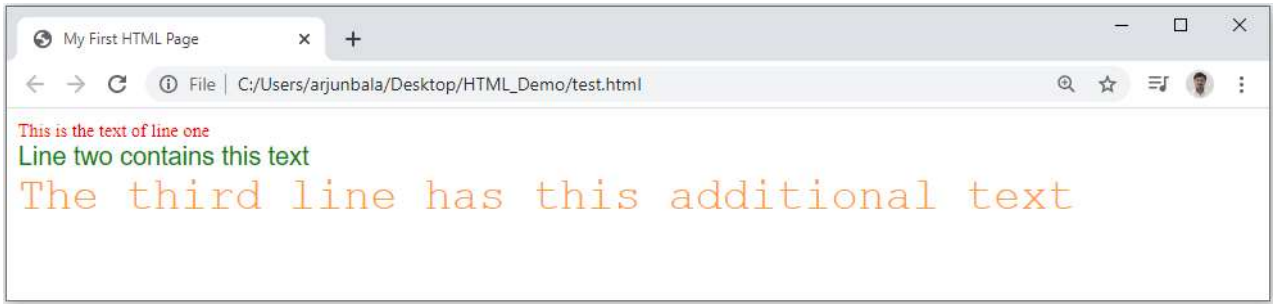
3) Colors

- We can use color values for mainly two attributes named **bgcolor** and **color.**
- Possible values for the color are,
  - many are predefined (red, blue, green, ...)
  - all colors can be specified as a six character hexadecimal value: #RRGGBB
    - #FF0000 – red
    - #888888 – gray
    - #00FF00 –green
    - #000000 – black
- For example, <body bgcolor="red"> or <body bgcolor="#888888">

4) Fonts

- The <font> tag specifies the font face, font size, and color of text.
- The <font> tag is **not supported in HTML5**.

```
<font color="red" size="2" face="Times Roman">
    This is the text of line one
</font>
<br/>
<font color="green" size="4" face="Arial">
    Line two contains this text
</font>
<br/>
<font color="#FF9933" size="6" face="Courier">
    The third line has this additional text
</font>
```



5) List

Ordered List

1. Block-A
2. Block-B
3. Block-C
4. Block-D

a) Block-A
b) Block-B
c) Block-C
d) Block-D

A. Block-A
B. Block-B
C. Block-C
D. Block-D

i. Block-A
ii. Block-B
iii. Block-C
iv. Block-D

I. Block-A
II. Block-B
III. Block-C
IV. Block-D

Unordered List

- Block-A
- Block-B
- Block-C
- Block-D

- Block-A
- Block-B
- Block-C
- Block-D

- Block-A
- Block-B
- Block-C
- Block-D

5.1) Ordered List (OL)

```
<ol>
    <li> Item one </li>
    <li> Item two </li>
    <ol type="I">
        <li> Sublist item one </li>
        <li> Sublist item two </li>
        <ol type="i">
            <li> Sub-sub list item one </li>
            <li> Sub-sub list item two </li>
        </ol>
    </ol>
</ol>
```

Types:

Type = 1 (default)
Type = a
Type = A
Type = I
Type = i

Output

1. Item one
2. Item two
    I. Sublist item one
    II. Sublist item two
        i. Sub-sub list item one
        ii. Sub-sub list item two

5.2) Unordered List (UL)

```
<ul>
    <li> One </li>
    <li> Two </li>
    <ul type="circle">
        <li> Three </li>
        <li> Four </li>
        <ul type="square">
            <li> Five </li>
            <li> Six </li>
        </ul>
    </ul>
</ul>
```

Types:

Type = disc (default)

Type = circle

Type = square

Output

- One
- Two
  - Three
  - Four
    - Five
    - Six

6) <a> Anchor Tag (Hyperlinks)

- The <a> tag defines a hyperlink, which is used to link from one page to another.
- An Anchor tag have 3 important attributes:
  - the **href** attribute (**h**ypertext **ref**erence) defines the target address of the document.
  - the **name** attribute of the anchor tag can be used to enable users to “jump” to a specific point on a page.
  - the **target** attribute specifies how the destination page or the target document should be opened. `target="_ blank"` is used for opening of the target page in a new tab.
- Link to an absolute URL:
  - Example, Darshan .
- Link to a relative URL:
  - Example, Home .
- Link to a section within a URL:
  - Example, Reference Section. .

7) Images

- The HTML <img> element embeds an image into the document.
- Syntax: `<img src= "pathToImage" />`
- Attributes:
  - the **src** attribute is required, and contains the path to the image we want to embed.
  - the **alt** attribute holds a text description of the image, which isn't mandatory but is incredibly useful for accessibility (screen readers read this description out to their users so they know what the image means). Alt text is also displayed on the page if the image can't be loaded for some reason: for example, network errors, content blocking etc…
  - the **width** & **height** attribute can be in units of pixels or percentage of page or frame.
  - the **align** attribute (currently deprecated) will aligns the image with its surrounding context (Use the float and/or vertical-align CSS properties instead of this attribute).

8) Table

```
<table border=1>
    <caption>Table Caption</caption>
    <tr>
        <th>Heading1</th>
        <th>Heading2</th>
    </tr>
    <tr>
        <td>Row1 Col1 Data</td>
        <td>Row1 Col2 Data</td>
    </tr>
    <tr>
        <td>Row2 Col1 Data</td>
        <td>Row2 Col2 Data</td>
    </tr>
</table>
```

<table>	table tag
<caption>	optional table title
<tr>	table row
<th>	table header
<td>	table data element

Output

Table Caption

Heading1	Heading2
Row1 Col1 Data	Row1 Col2 Data
Row2 Col1 Data	Row2 Col2 Data

HTML Forms

- HTML forms are used to create GUIs on Web pages
  - Usually the purpose is to ask the user for information
  - The information is then sent back to the server
- A form is an area that can contain form elements
  - The syntax is: <form ***parameters***> ***...form elements...*** </form>
  - Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
    - Other kinds of HTML tags can be mixed in with the form elements
  - A form usually contains a Submit button to send the information in the form elements to the server
  - The form's ***parameters*** tell browser how to send the information to the server (there are two different ways it could be sent)

The <form> Tag

- The <form ***parameters***> … </form> tag encloses form elements (and probably other HTML as well)
- The parameters to form tell what to do with the user input
  - action="***url***" (required)
    - Specifies where to send the data when the Submit button is clicked
  - method="**get**" (default)
    - Form data is sent as a URL with ?form_data info appended to the end
    - Can be used *only* if data is all ASCII and not more than 100 characters
  - method="**post**"
    - Form data is sent in the body of the URL request
    - Cannot be bookmarked by most browsers
  - target="***target***"
    - Tells where to open the page sent as a result of the request
    - ***target=*** _blank means open in a new window
    - ***target=*** _top means use the same window

Form Elements

- Input
- Select
- Textarea
- Button
- Label
- Fieldset
- Legend
- Etc…

Input Types (HTML4)

- text
- password
- checkbox
- radio
- submit
- button
- reset
- file

Input Types (HTML5)

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

HTML5 Form Validation

- Form validation is a “technical process where a web-form checks if the information provided by a user is correct.”
- The form will either alert the user that something is not in correct format and need to fix to proceed, or the form will be validated and the user will be able to continue with their process.
- Form can be validated both in Client-Side as well as Server-Side, it is recommended to validate the form in both the side.
- Form validation generally performs two functions.
  1. **Basic Validation**
     - Emptiness
     - Length Validation etc……
  2. **Data Format Validation**

     Secondly, the data that is entered must be checked for correct **form** and **value**.
     - Email Validation
     - Mobile Number Validation
     - Enrollment Number Validation etc….

HTML5 Form Validation (Cont…)

- We can use **required** attribute in order to stop user sending empty data to server.

```
1 <input type="text" name="txtName" required/>
```

- We can use **pattern** attribute in order to force some format on user before sending the data to server.

```
1 <input type="text" name="txtName" pattern="[0-9]{10}"/>
```

- We can use **title** attribute for custom error message.

```
1 <input  type="text" name="txtName"
                pattern="[0-9]{10}"
                title="Please enter 10 digit mobile number"
                required/>
```

META Tag

- Metadata is data (information) about data.
- The <meta> tag provides metadata about the HTML document.
- Metadata will not be displayed on the page.
- Meta elements are typically used to specify page description, keywords, author of the document, last modified and other metadata.
- The metadata can be used by search engines (keywords), browsers (how to display content or reload page) or other web services.

Meta Tag Attributes

Attribute	Value	Description
charset	`character_set`	Specifies the character encoding for the HTML document

name	`author`
	`description`
	`keywords`
	`robots`
	`expires`
		Specifies a name for the metadata

http-equiv	`content-type`
	`default-style`
	`refresh`
		Provides an HTTP header for the information/value of the content attribute

content	`text`	Gives the value associated with the http-equiv or name attribute
scheme	`format/URI` `USA/Europe`	Not supported in HTML5. Specifies a scheme to be used to interpret the value of the content attribute

Media Tags

- Video tag
- Audio tag

Video Tag

- The HTML <video> element is used to show a video on a web page.
- The controls attribute adds video controls, like play, pause, and volume.
- The width and height attributes are used to set width and height respectively.
- The autoplay attribute start a video automatically.
- The muted attribute will mute your video sound.
- The <source> element allows you to specify alternative video files in src attribute which the browser may choose from. The browser will use the first recognized format.
- The text written in between the <video> and </video> tags will display only if browser do not support the <video> element.

```
1  <video width="300" height="200" autoplay muted>
     <source src="video.mp4" type="video/mp4">
     <source src="video.ogg" type="video/ogg">
     The video tag is not supported in your browser.
   </video>
```

Audio Tag

- The HTML <audio> element is used to play an audio file on a web page.
- The controls attribute adds audio controls, like play, pause, and volume.
- The autoplay attribute start a audio automatically.
- The muted attribute will mute your audio sound.
- The <source> element allows you to specify alternative audio files in src attribute which the browser may choose from. The browser will use the first recognized format.
- The text written in between the <audio> and </audio> tags will display only if browser do not support the <audio> element.

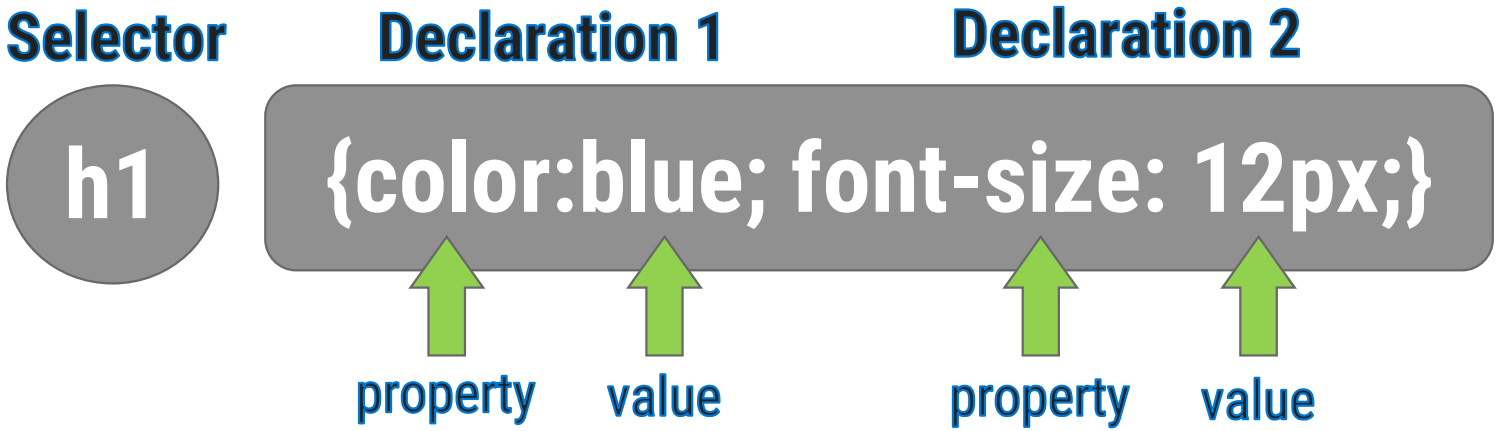
```
<audio controls autoplay muted>
  <source src="myaudio.ogg" type="audio/ogg">
  <source src="myaudio.mp3" type="audio/mpeg">
  The audio tag is not supported in your browser.
</audio>
```

What is CSS?

- **C**ascading **S**tyle **S**heets, fondly referred to as **CSS**, is a simple design language intended to **simplify** the process of making web pages **presentable**.
- CSS defines **layout of HTML** documents. For example, CSS covers Fonts, colors, margins, lines, height, width, background images, advanced positions and many other things.
- Importance of CSS :
  - CSS defines HOW HTML elements are to be displayed.
  - Styles are normally saved in external .css files.
  - External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file.
  - Advantages :
    - Improves Website Presentation
    - External CSS makes Updates Easier and Smoother
    - External CSS helps Web Pages Load Faster
  - Disadvantages :
    - Browser Dependent
    - Difficult to retrofit in old websites

Basic Syntax of CSS

- A CSS rule has two main parts: a **selector**, and one or more **declarations**



- The **selector** can be HTML element, id or class.
- Each **declaration** consists of a **property** and a **value**.
- The **property** is the style attribute you want to change. Each property has a **value**.

The “id” selector

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

HTML

```
<h1 id=“para1”>
        Hello Friends
</h1>

<h1>
        How are you
</h1>
```

CSS

```
#para1{
        color: blue;
}
```

Output

Hello Friends
How are you

The “class” selector

- The class selector is used to specify a style for a **group** of elements.
- The class selector uses the HTML class attribute, and is defined with a "**.**“ in css.

HTML

```
<h1 class=“myClass”>
        Hello Friends
</h1>
<h1>
        How are you
</h1>
<h1 class=“myClass”>
        How are you
</h1>
```

CSS

```
.myClass{
        color: blue;
}
```

Output

Hello Friends
How are you
How are you

Different ways to write CSS

- There are three ways of writing a style sheet:
  1. Inline Style
  2. Internal/Embedded Style sheet
  3. External Style Sheet

1) Inline Style

- It is possible to place CSS right in your HTML code, and this method of CSS usage is referred to as **inline css**.
- Inline CSS has the **highest priority** out of external, internal, and inline CSS.
- This means that you can **override styles** that are defined in external or internal by using inline CSS.
- If you want to add a style inside an HTML element all you have to do is specify the desired CSS properties with the **style** HTML attribute.
- Example:

HTML

```
<p style="background: blue; color: white;"> My Inline CSS </p>
```

2) Internal Style Sheet

- This type of CSS is only for **Single Web Page**.
- When using internal CSS, we must add a new tag, **<style>,** inside the **<head>** tag.
- The HTML code below contains an example of <style>'s usage.

HTML

```
<html><head>
        <style type="text/css">
                p{ color: red;}
        </style>
</head><body>
        <p>Your page's content!</p></body>
</html>
```

3) External Style Sheet

- When using CSS it is preferable to keep the **CSS separate from your HTML**.
- Placing CSS in a separate file allows the web designer to completely differentiate between content (HTML) and design (CSS).
- External CSS is a file that contains **only CSS** code and is saved with a **".css"** file extension.
- This CSS file is then referenced in your HTML using the **<link> instead of <style>**.

Demo.html

```
<html>
<head>
<link rel="stylesheet" type="text/css"
href="test.css">
</head>
<body>
        <p> Hello Friends </p>
        <p id="para1"> How are you? </p>
</body>
</html>
```

test.css

```
#para1{
                text-align: center;
}
p
{
                color : blue;
}
```

Output

Hello Friends

How are you?

3) External Style Sheet (Cont.)

- Advantages:
  - It keeps your website design and content separate.
  - It's much easier to reuse your CSS code if you have it in a separate file. Instead of typing the same CSS code on every web page you have, simply have many pages refer to a single CSS file with the "link" tag.
  - You can make drastic changes to your web pages with just a few changes in a single CSS file.

Assign Multiple Classes

- We can apply different class to same html element by giving space separated class names in the class attribute:

Demo.html

```
<html>
<head>
<link rel="stylesheet" type="text/css"
href="test.css">
</head>
<body>

<h1 class="class1 class2">
        How are you?
</h1>

</body>
</html>
```

test.css

```
. class1
{
        color : blue;
}
. class2
{
        text-align : center;
}
```

Output

How are you?

Multiple Selection

- We can apply same css to multiple selectors using **comma separated** selector list, for example

Demo.html

```
<html>
<head>
<link rel="stylesheet" type="text/css"
href="test.css">
</head>
<body>

<p> Hello Friends </p>
<h1> How are you? </h1>

</body>
</html>
```

test.css

```
p, h1
{
        color : blue;
}
```

Output

Hello Friends

How are you?

Multi-level Selection

- We can use hierarchical path to target html element by **space separated** element/class/id names, for example :

Demo.html

```
<html>
<head>
<link rel="stylesheet" type="text/css"
href="test.css">
</head>
<body>
<h1>Hello Friends…</h1>
<div>
        <h1>How are you?</h1>
</div>

</body>
</html>
```

test.css

```
div h1
{
        color : blue;
}
```

Output

Hello Friends…
How are you?

Background Property

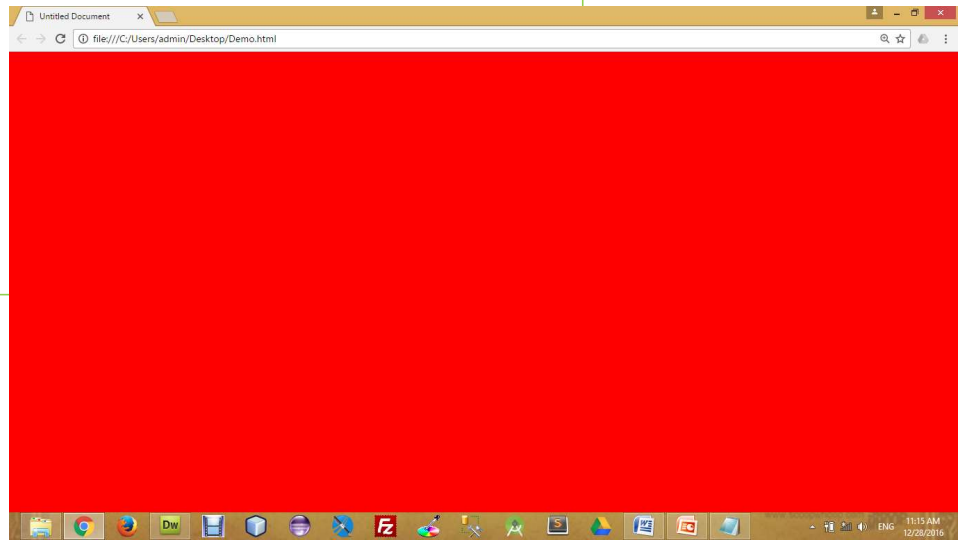
	Property Name
Background Color	(background-color)
Background Image	(background-image)
Background Image Repeat	(background-repeat)
Fixed Background Image	(background-attachment)
Background Image Positioning	(background-position)

Background Color

- The **background-color** property specifies the background color of an element.
- The background color of a page is defined in the body selector:
- Below is example of CSS backgrounds

test.css

```
body
{
        background-color : red;
        background-color : #FF0000;
        background-color : rgb(255,0,0);
}
```

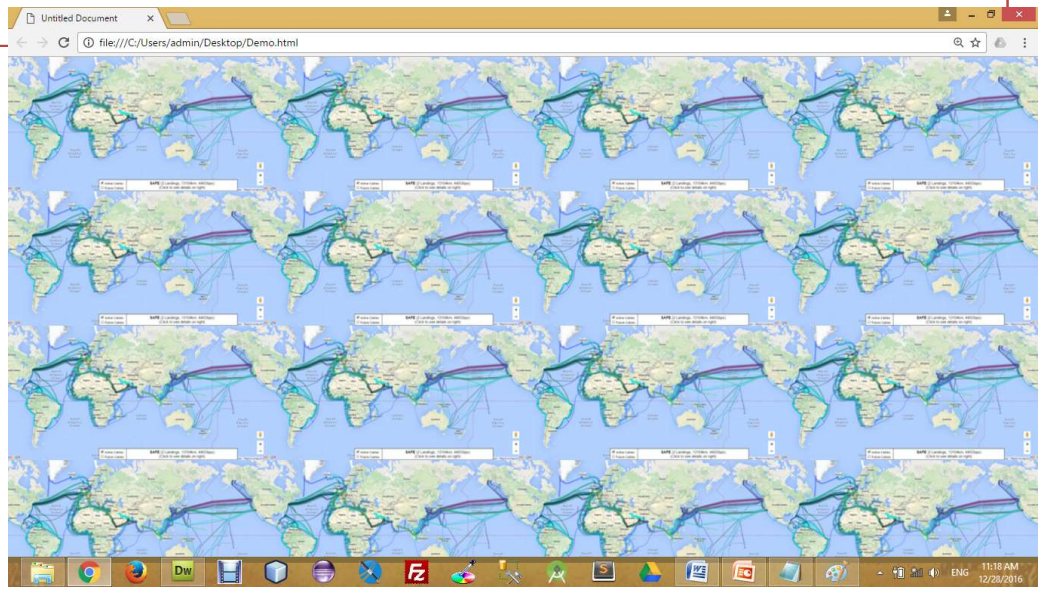


Background Image

- The **background-image** property specifies an image to use as the background of an element.
- For Example,

test.css

```
body
{
        background-image : url('pathToImage.jpg');
}
```

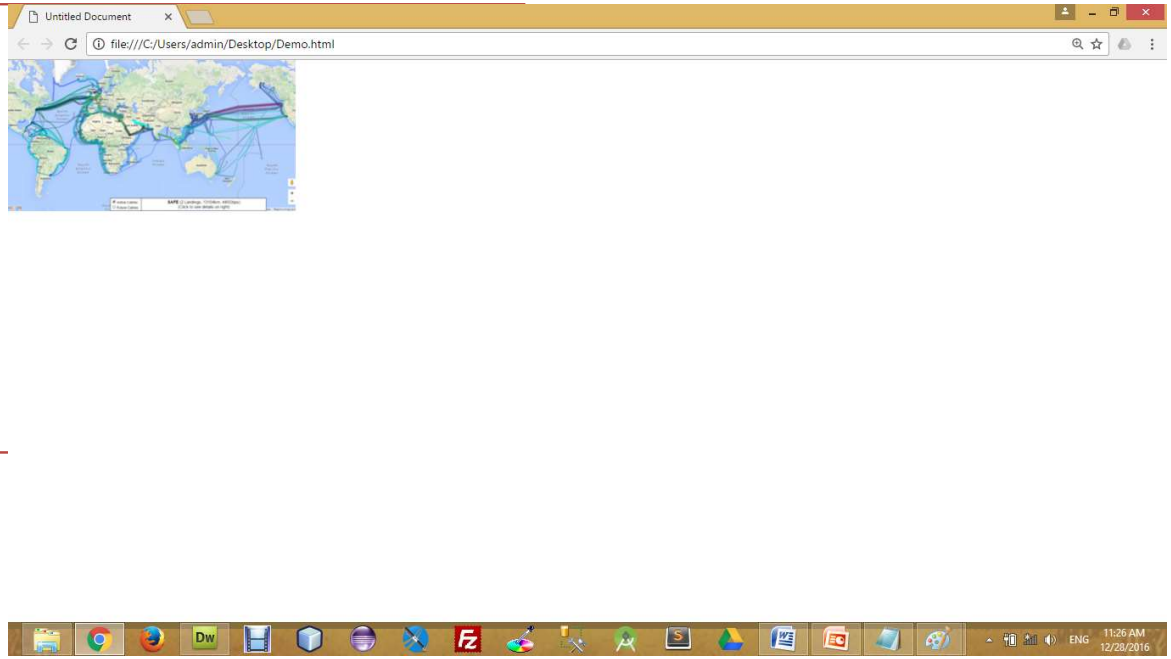


Background Image Repeat

- You can have a background image repeat vertically (y-axis), horizontally (x-axis), in both directions, or in neither direction.

test.css

```
body
{
        background-image : url('pathToImage.jpg');
        background-repeat : repeat;
         background-repeat : repeat-x;
        background-repeat : repeat-y;
        background-repeat : no-repeat;
}
```



Fixed Background Image

- The background-attachment property sets whether a background image is fixed or scrolls with the rest of the page.
- For Example,

test.css

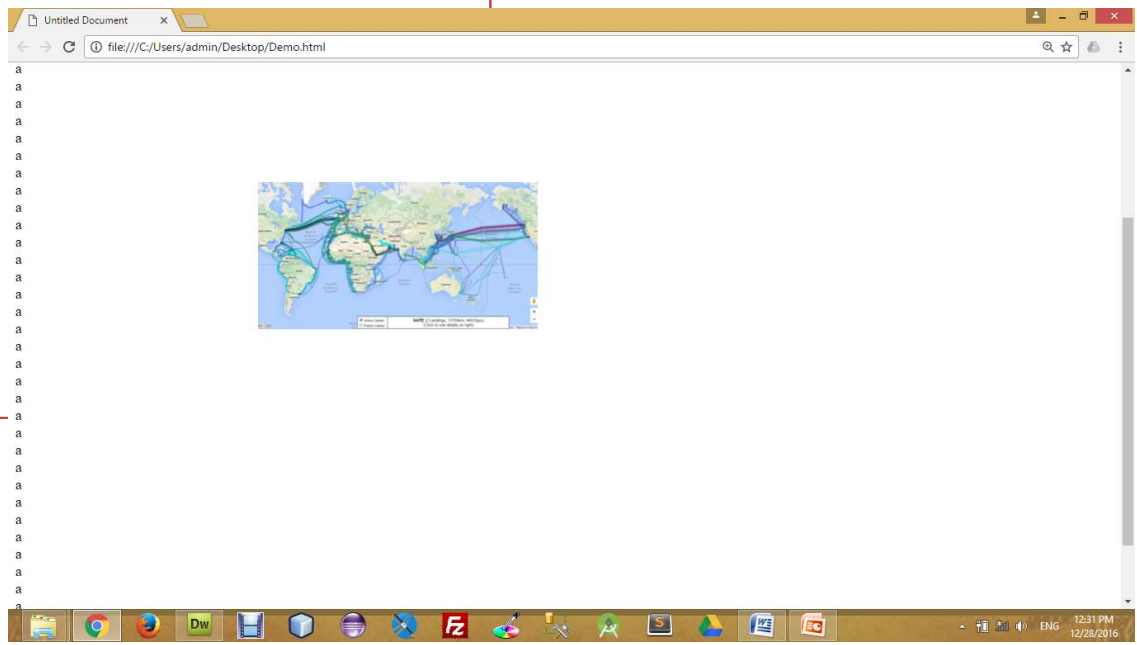
```
body
{
        background-image : url('pathToImage.jpg');
        background-repeat : no-repeat;
        background-attachment : fixed;
}
```

Background Image Positioning

- The **background-position** property sets the starting position of a background image.

test.css

```
body
{
        background-image : url('pathToImage.jpg');
        background-repeat : no-repeat;
        background-position: 20px 10px;
        background-position: 30%30%;
        background-position: top center;
}
```



CSS Font

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

	Property Name
1. Font Color	(color)
2. Font Family	(font-family)
3. Font Size	(font-size)
4. Font Style	(font-style)
5. Font Weight	(font-weight)
6. Font Variant	(font-variant)

CSS Font (Cont.)

- **Font Color**
  - Set the text-color for different elements

```
h4{
        color : red;
}
```

- **Font Family**
  - The font family of a text is set with the font-family property.

```
h4{
        font-family : sans-serif;
}
```

- **Font Size**
  - The font-size property sets the size of the text.
    - font-size : 120%
    - font-size : 10px;
    - font-size : x-large;

```
h4{
        font-size: 120%;
        font-size : 10px;
        font-size : small;
        font-size : smaller;
        font-size : x-small;
        font-size : xx-small;
        font-size : large;
        font-size : larger;
        font-size : x-large;
        font-size : xx-large;
        font-size : medium;
}
```

CSS Font (Cont.)

Font Style

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

```
h4{
        font-style: italic ;
}
```

Font Weight

- The font-weight property sets how thick or thin characters in text should be displayed.

```
h4{
        font-weight : 300;
        font-weight : bolder;
        font-weight : lighter;
}
```

Font Variant

- The font-variant property specifies whether or not a text should be displayed in a small-caps font.
  - font-variant : small-caps;

```
h4{
        font-variant: small-caps;

}
```

CSS Text Property

- While CSS Font covers most of the traditional ways to format your text, CSS Text allows you to control the spacing, decoration, and alignment of your text.

	Property Name
1. Text Decoration	(text-decoration)
2. Text Indent	(text-indent)
3. Text Align	(text-align)
4. Text Transform	(text-transform)
5. White Space	(white-space)
6. Word Spacing	(word-spacing)
7. Letter Spacing	(letter-spacing)
8. Line Height	(line-height)

CSS Text Property (Cont.)

- **Text Decoration**
  - The text-decoration property is used to set or remove decorations from text.
  - The text-decoration property is mostly used to remove underlines from links for design purposes.
- **Text Indent**
  - The text-indentation property is used to specify the indentation of the first line of a text.
- **Text Align**
  - The text-align property is used to set the horizontal alignment of a text.

```
h4{
        text-decoration : line-through;
        text-decoration : overline;
        text-decoration : underline;
        text-decoration : none;
}
```

```
h4{
                text-indent : 20px;
                text-indent : 30%;
}
```

```
h4{
                text-align : right;
                text-align : justify;
                text-align : left;
                text-align : center;
}
```

CSS Text Property (Cont.)

Text Transform

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

```
h4{
    text-transform : capitalize;
    text-transform : uppercase;
    text-transform : lowercase;
}
```

White Space

- The white-space attribute allows you to prevent text from wrapping until you place a break
 into your text.

```
h4{
        white-space : nowrap;
        white-space : normal;
        white-space : pre;
}
```

Word Spacing

- With the CSS attribute word-spacing you are able to specify the exact value of the spacing between your words. Word-spacing should be defined with exact values.

```
h4{
        word-spacing : 10px;
}
```

CSS Text Property (Cont.)

- **Letter Spacing**
  - With the CSS attribute letter-spacing you are able to specify the exact value of the spacing between your letters. Letter-spacing should be defined with exact values.

```
h4{
        letter-spacing : 3px;
}
```

- **Line Height**
  - The line-height attribute will set the height of the line in the page.

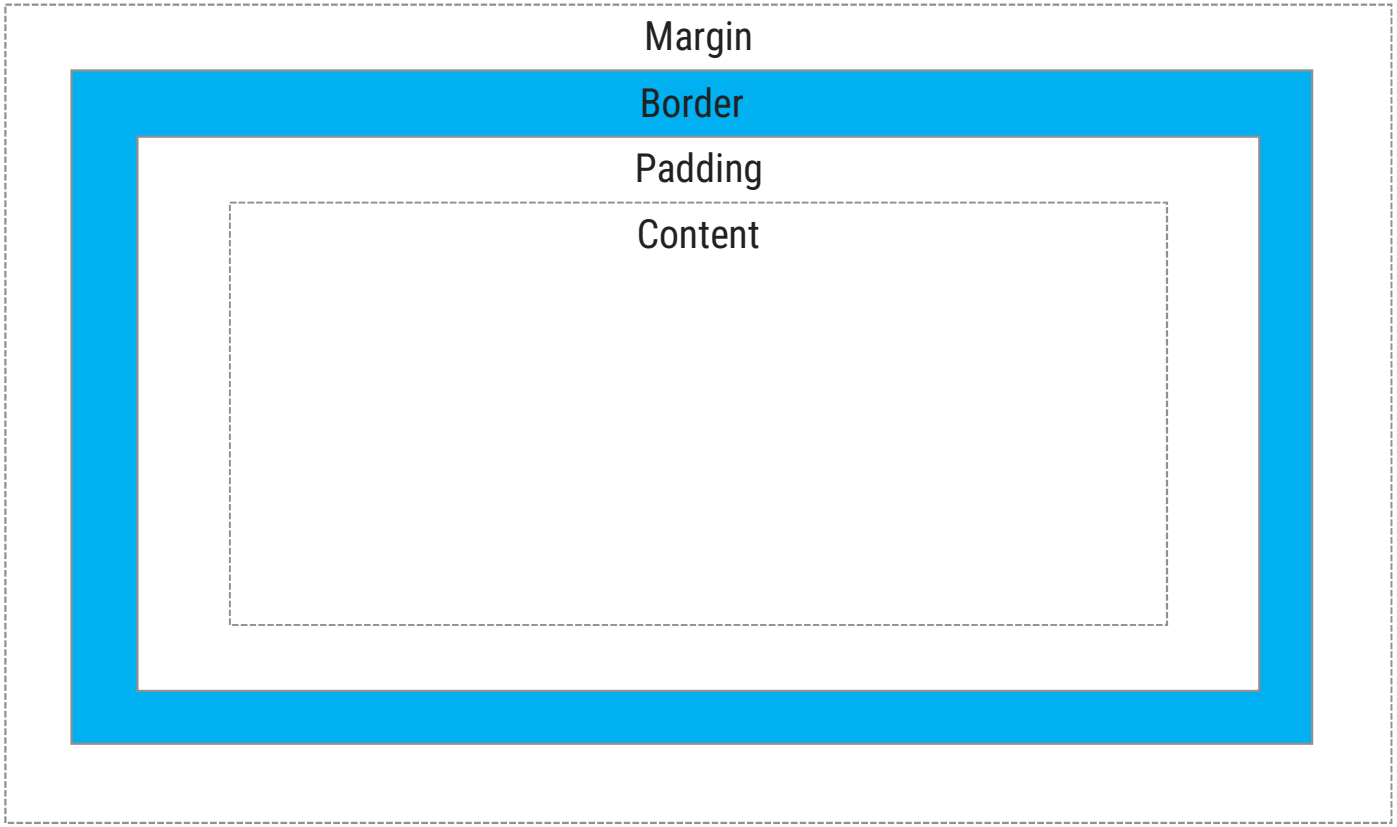
```
h4{
        line-height : 10px;
}
```

The Box Model

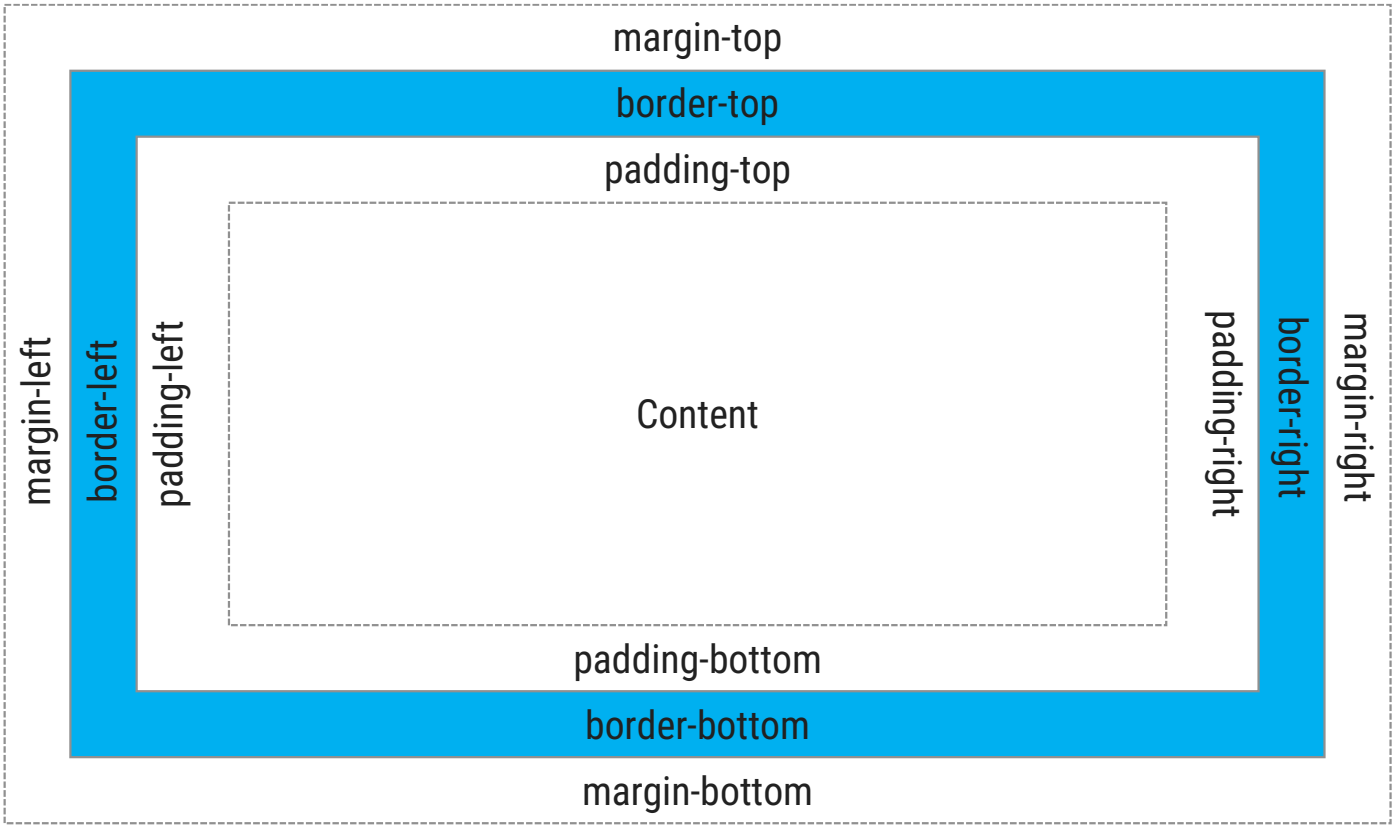
- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: **margins**, **borders**, **padding**, and the actual **content**.
- The box model allows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

The Box Model (Cont)

- The image below illustrates the box model:



The Box Model (Cont)



CSS Padding

- The CSS padding properties define the space between the element border and the element content.

```
h4{
        padding : 10px;
}
```

- The top, right, bottom, and left padding can be changed independently using separate properties.

```
h4{
        padding-top : 10px;
        padding-right : 20px;
        padding-bottom : 30 px;
        padding-left : 40 px;
}
```

- A shorthand padding property can also be used, to change all padding at once.

```
h4{
    padding : 10px 20px 30px 40px;
}
```

CSS Border

- The CSS border properties allow you to specify the style and color of an element's border.
- Border Style Types
  - The border-style property specifies what kind of border to display.
- Border Width
  - The border-width property is used to set the width of the border.
- Border Color
  - The border-color property is used to set the color of the border.
  - Border colors can be any color defined by RGB, hexadecimal, or key terms. Below is an example of each of these types.
- The top, right, bottom, and left border can be changed independently using separate properties.

```
h4{
        border : 1px solid red;
}
```

```
h4{
        border-style : solid;
        border-style : dotted;
        border-style : double;
}
```

```
h4{
        border-width : 7px;
}
```

```
h4{
        border-color : red;
}
```

```
h4{
        border-top : 1px solid red;
}
```

CSS Margin

- The CSS margin properties define the space around elements

```
h4{
        margin: 10px;
}
```

- The top, right, bottom, and left margin can be changed independently using separate properties.

```
h4{
        margin -top : 10px;
        margin -right : 20px;
        margin -bottom : 30 px;
        margin -left : 40 px;
}
```

- A shorthand margin property can also be used, to change all margins at once.

```
h4{
    margin : 10px 20px 30px 40px;
}
```

CSS Positioning

- Absolute Positioning
  - With absolute positioning, you define the exact pixel value where the specified HTML element will appear.
  - The point of origin is the top-left of the browser's viewable area, so be sure you are measuring from that point.
- Relative Positioning
  - Relative positioning changes the position of the HTML element relative to where it normally appears
- Fixed Positioning
  - The element is positioned relative to the browser window, in fixed position, element will be in the same place even we scroll the screen.
- Sticky Positioning
  - An element with position: sticky; is positioned based on the user's scroll position.
  - A sticky element toggles between relative and fixed, depending on the scroll position.

```
h1{
        position : absolute;
        left : 50px;
        top : 100px;
}
```

```
h1{
        position : relative;
        left : 50px;
        top : 100px;
}
```

```
h1{
        position : fixed;
        top : 50px;
        left : 100px;
}
```

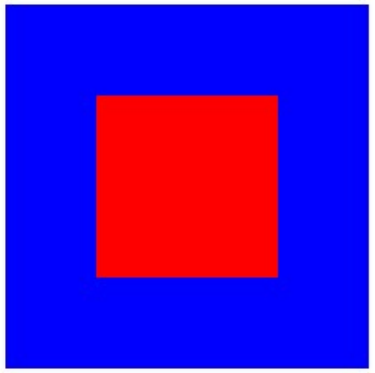
```
h1{
        position : sticky;
        top : 0px;
}
```

CSS Layers

- CSS allows you to control which item will appear on top with the use of layers.
- In CSS, each element is given a priority.
- If there are two overlapping CSS positioned elements, the element with the higher priority will appear on top of the other.
- To manually define a priority, set the z-index value. The larger the value, the higher the priority the element will have.

HTML

```
<div id="division1">
</div>
<div id="division2">
</div>
```

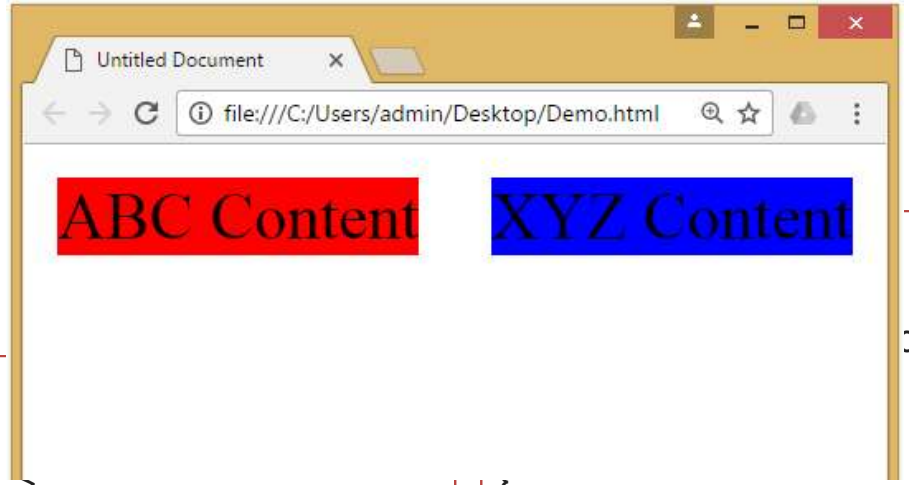


CSS

```
#division1{
        position : absolute;
        height : 100px;
        width : 100px;
        left : 100px;
        top : 150px;
        background-color : red;
        z-index : 5;
}
#division2{
        position : absolute;
        height : 200px;
        width : 200px;
        left : 50px;
        top : 100px;
        background-color : blue;
         z-index : 2;
```

CSS Float Property

- The CSS float property defines that an element should be taken out of the normal flow of the document and placed along the left or right side of its containing block.
- Text and inline elements will then wrap around this element.




```
<div id="division1" >
        ABC Content
</div>
<div id="division2">
        XYZ Content
</div>
```

```
#division1{
        background-color : red;
}
#division2{
        background-color : blue;
        float : right;
        width: 40%;
}
```

Introduction to CSS3

- CSS3 is the **latest standard** for CSS.
- CSS3 is completely backwards-compatible with earlier versions of CSS.
- CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.
- CSS3 Transitions are a presentational effect which allow property changes in CSS values, such as those that may be defined to occur on :hover or :focus, to occur smoothly over a specified duration – rather than happening instantaneously as is the normal behavior.
- Transition effects can be applied to a wide variety of CSS properties, including background-color, width, height, opacity, and many more.

Introduction to CSS3 (Cont)

- Some of the most important CSS3 modules are:
  - CSS Animations and Transitions
  - Calculating Values With calc()
  - Advanced Selectors
  - Generated Content and Counters
  - Gradients
  - Webfonts
  - Box Sizing
  - Border Images
  - Media Queries
  - Multiple Backgrounds
  - CSS Columns

Courtesy : http://tutorialzine.com/2013/10/12-awesome-css3-features-you-can-finally-use/

Animations

- CSS animations make it possible to animate transitions from one CSS style configuration to another.
- Animations consist of two components,
  - a style describing the CSS animation
  - a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.
- There are three key advantages to CSS animations over traditional script-driven animation techniques:
  - They're easy to use for simple animations; you can create them without even having to know JavaScript.
  - The animations run well, even under moderate system load. Simple animations can often perform poorly in JavaScript. The rendering engine can use frame-skipping and other techniques to keep the performance as smooth as possible.
  - Letting the browser control the animation sequence lets the browser optimize performance and efficiency by, for example, reducing the update frequency of animations running in tabs that aren't currently visible.

Configuring the animation

- To create a CSS animation sequence, you style the element you want to animate with the animation property or its sub-properties. This lets you configure the timing, duration, and other details of how the animation sequence should progress.
  - Note: This does not configure the actual appearance of the animation, which is done using the @keyframes
- The sub-properties of the animation property are:
  - **animation-name :** Specifies the name of the @keyframes at-rule describing the animation's keyframes.
  - **animation-duration :** Configures the length of time that an animation should take to complete one cycle.
  - **animation-timing-function :** Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves. (linear, ease, ease-in, ease-out etc…)
  - **animation-delay :** Configures the delay between the time the element is loaded and the beginning of the animation sequence.
  - **animation-iteration-count :** Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.
  - **animation-direction :** Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself. (normal, reverse, alternate etc…)
  - **animation-fill-mode :** Configures what values are applied by the animation before and after it is executing.
  - **animation-play-state :** Lets you pause and resume the animation sequence.

Defining the animation sequence using keyframes

- Once you've configured the animation's timing, you need to define the appearance of the animation. This is done by establishing two or more keyframes using the @keyframes at-rule.
- Each keyframe describes how the animated element should render at a given time during the animation sequence.
- Example

HTML

```
<p>
   Hello World
</p>
```

CSS

```
p {
    animation-duration: 3s;
    animation-name: slidein;
}

@keyframes slidein {
    from { /* or 0% */
        margin-left: 100%;
        width: 300%;
    }
    to { /* or 100% */
        margin-left: 0%;
        width: 100%;
    }
}
```

Specifying Intermediate steps

- We can specify intermediate steps using percentage of time in @keyframes, similar to the example given below.

HTML

```
<p>
   Hello World
</p>
```

CSS

```
p {
    animation-duration: 4s;
    animation-name: changeColors;
}

@keyframes changeColors {
    0%   {background-color: yellow;}
    25%  {background-color: blue;}
    50%  {background-color: green;}
    100% {background-color: red;}
}
```

Tooltip

- A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element.

HTML

```
<div class="tooltip">Hover here to see tooltip
  <span class="tooltiptext">Tooltip text</span>
</div>
```

CSS

```
.tooltip {
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black;
}
```

CSS

```
.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;
  position: absolute;
  z-index: 1;
}

.tooltip:hover .tooltiptext {
  visibility: visible;
}
```

Style Images

- We can use many properties with Images like,
  - opacity
  - border-radius
  - margin-left, margin-right to be auto and display to be block (to make it aligned center)
  - max-width to be 100% and height to be auto (to make it responsive)
  - filter
  - etc…

CSS Variables / Custom Properties

- Custom properties (sometimes referred to as CSS variables or cascading variables) are entities defined by CSS authors that contain specific values to be reused throughout a document.
- Advantages of using CSS variables are:
  - makes the code easier to read (more understandable)
  - makes it much easier to change.
- CSS variables can have a global or local scope.
  - Global variables can be accessed/used through the entire document.
    - To create a variable with global scope, declare it inside the **:root** selector. The :root selector matches the document's root element.
  - local variables can be used only inside the selector where it is declared.
    - To create a variable with local scope, declare it inside the selector that is going to use it.
- We can use **var()** function in order to access the variable set in the css.

CSS Variables (Example)

HTML

```
<p>
    Hello World
</p>
<div>
    How are you?
</div>
```

CSS

```
:root{
    --myFavColor : orange;
    --myNextFacColor : green;
}
p{
    background-color: var(--myFavColor);
    color: var(--myNextFacColor);
}
div{
    background-color: var(--myNextFacColor);
    color: var(--myFavColor);
}
```

Media Queries

- Media queries are a way to target browser by certain characteristics, features, and user preferences, then apply styles or run other code based on those things.
- Perhaps the most common media queries in the world are those that target particular viewport ranges and apply custom styles, which is the whole idea of responsive design.
- General syntax for the media queries in CSS is,

Syntax

```
@media [media-type] ([media-feature]) {
    /* Styles! */
}
```

- It consists of:
  - A **media type**, which tells the browser what kind of media this code is for (e.g. print, screen etc…).
  - A **media feature**, which is a rule, or test that must be passed for the contained CSS to be applied.
  - A set of CSS rules that will be applied if the test passes and the media type is correct.

Media Query (Cont.)

- Media Types:
  - all
  - print
  - screen
  - speech
- Media Features:
  - width & height
  - min-width, min-height, max-width & max-height
  - orientation
- The viewport meta tag:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

  - This is needed because by default, most mobile browsers lie about their viewport width.
  - Non-responsive sites commonly look really bad when rendered in a narrow viewport, so mobile browsers usually render the site with a viewport width wider than the real device width by default (usually 960 pixels), and then shrink the rendered result so that it fits in the display.

Media Query (Example)

CSS

```css
#div1{
    background-color: red;
    width: 50%;
    float: left;
}
#div2{
    background-color: yellow;
    width: 50%;
    float: left;
}
@media (max-width: 600px){
    #div1{
        width: 100%;
     }
    #div2{
        width: 100%;
    }
}
```

HTML

```html
<div id="div1">
    Hello
</div>

<div id="div2">
    World
</div>
```

Note: you have to add viewport meta tag in the head section

```
<meta  name="viewport" content=
"width=device-width, initial-scale=1.0">
```

Media Query (cont.)

- We can also use **and/or** logic in media query, for example
  - We can use **and** as a separator if we want two condition to satisfy before applying the CSS.

CSS

```
@media screen and (min-width: 600px) and (max-width: 900px) {
    body {
        color: blue;
    }
}
```

  - We can use , (comma) as a separator if we want anyone of the condition to satisfy before applying the CSS.

CSS

```
@media (min-width: 600px), (orientation: landscape) {
    body {
        color: blue;
    }
}
```

Wild Card Selectors

- Wildcard selector is used to select multiple elements simultaneously.
- It selects similar type of class name or attribute and apply CSS property.
- Some of the wild card selector are,
  - **[attribute*=”str”] Selector (e.g. [class*="str"])**
    - It will select all the elements with the given attribute **containing** the str.
  - **[attribute^=”str”] Selector (e.g. [class^="str"])**
    - It will select all the elements with the given attribute **starts** with the str.
  - **[attribute$=”str”] Selector (e.g. [class$="str"])**
    - It will select all the elements with the given attribute **ends** with the str.

Gradients

- CSS gradients let you display smooth transitions between two or more specified colors.
- CSS defines two types of gradients:
  - Linear Gradients (goes down/up/left/right/diagonally)
    - To create a linear gradient you must define at least two color stops.
    - Color stops are the colors you want to render smooth transitions among.
    - You can also set a starting point and a direction (or an angle) along with the gradient effect.
  - Radial Gradients (defined by their center)
    - The radial-gradient() function sets a radial gradient as the background image.
    - A radial gradient is defined by its center.
    - To create a radial gradient you must define at least two color stops.

Linear Gradients

- To create a linear gradient you must define at least two color stops.
- Color stops are the colors you want to render smooth transitions among.
- You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

- direction:
  - to bottom
  - to top
  - to right
  - to left
  - to bottom left
  - 180deg
  - Etc…

Linear Gradients (Example)

- Example:

```css
background-image: linear-gradient(red, yellow);
```

- Output:

Radial Gradients

- To create a linear gradient you must define at least two color stops.
- Color stops are the colors you want to render smooth transitions among.
- You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

```
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
```

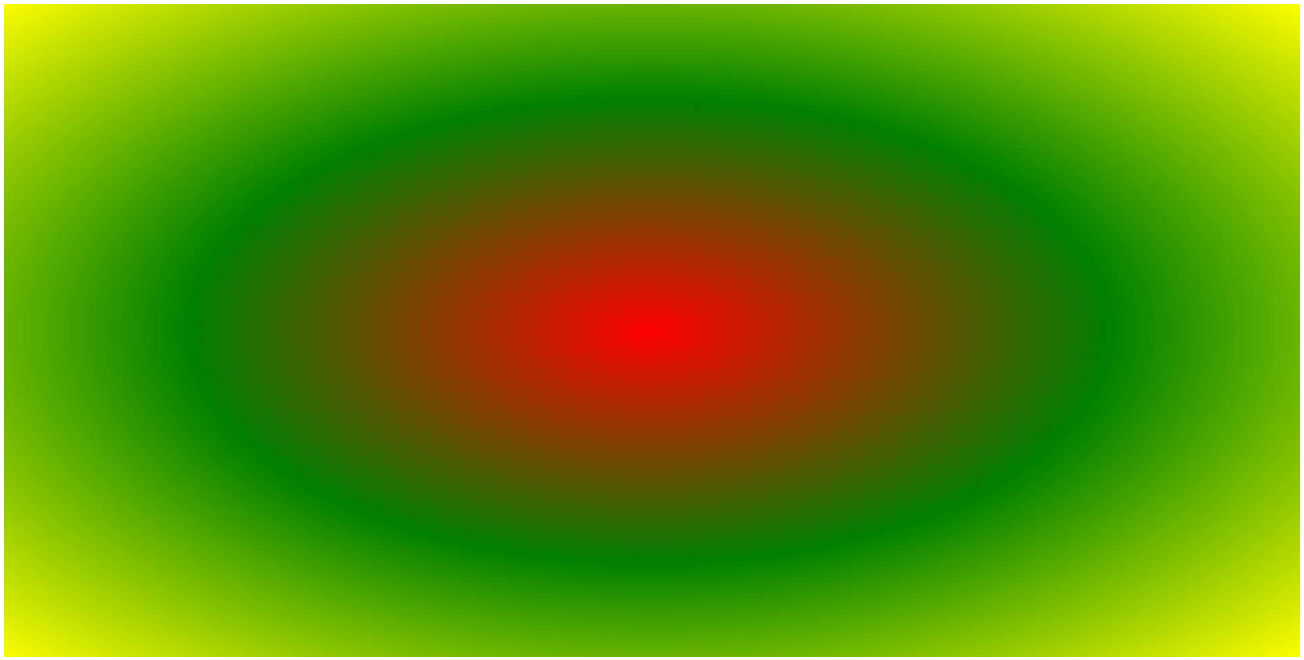
  - shape:
    - Ellipse (defalt)
    - Circle
  - size:
    - farthest-corner (default)
    - closest-corner
    - farthest-side
    - closest-side
  - position:
    - Center (default)
    - Etc…

Radial Gradients (Example)

- Example:

```css
background-image: radial-gradient(red, green, yellow);
```

- Output:



Pseudo Classes

- A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouse over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus
- Some important pseudo classes are:
  - active
  - disabled
  - first-child
  - nth-child()
  - focus
  - hover
  - visited

Syntax

```
selector:pseudo-class {
  property: value;
}
```

Pseudo Elements

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element
- pseudo elements are,
  - after
  - before
  - first-letter
  - first-line
  - selection

Syntax

```
selector::pseudo-element {
  property: value;
}
```

Bootstrap

- Bootstrap is Free **front-end framework** made of **HTML, CSS and JavaScript** Plugins (optional) for developing **Responsive Websites.**
- Responsive websites means websites which Automatically Adjust themselves to look good on all devices like mobile, desktop etc…
- **Why** to Use Bootstrap?
  - Easy to use
    - Anybody with just basic knowledge of HTML and CSS can start using Bootstrap
  - Responsive features
    - It's responsive CSS adjusts to phones, tablets, and desktops
  - Mobile-first approach
    - Mobile-first styles are part of the core framework
  - Browser compatibility
    - Compatible with all modern browsers (Chrome, Firefox, Internet Explorer, Safari, and Opera)
  - Free

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-04

Client Side Scripting using JavaScript

JavaScript

Prof. Vijay M Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
Vijay.shekhat@darshan.ac.in
9558045778

Outline HTML

1. Introduction
    I. Task Performed by Client side Scripts
    II. Pros & Cons of Client side Scripts
    III. Client side Scripts V/S Server side Scripts
2. Variables
3. Functions
4. Conditions
5. Loops
6. Arrays
7. Pop up boxes
8. External JavaScript
9. JavaScript Objects
10. DOM
11. DHTML

Introduction

- For a Web page, **HTML** supplies document **content and structure** while **CSS** provides **presentation styling**
- In addition, client-side scripts can **control browser actions** associated with a Web page.
- Client-side scripts are almost written in the **Javascript** language to control browser's actions.
- Client-side scripting can make Web pages more **dynamic** and more **responsive.**
- Tasks performed by client-side scripts are:
  - Checking **correctness** of user input
  - **Monitoring** user events and **specifying reactions**
  - **Replacing** and **updating** parts of a page
  - Changing the **style** and **position** of displayed elements **dynamically**
  - **Modifying** a page in **response** to **events**
  - Getting browser **information**
  - Making the Web page **different** depending on the browser and browser features
  - **Generating HTML** code for parts of the page

Pros & Cons of Client Side Scripting

- **Pros**
  - Allow for **more interactivity** by immediately responding to users' actions.
  - **Execute quickly** because they do not require a trip to the server.
  - The web **browser** uses its own **resources**, and **eases** the **burden** on the **server**.
  - It **saves** network **bandwidth**.
- **Cons**
  - **Code** is loaded in the browser so it will be **visible** to the client.
  - Code is **modifiable**.
  - **Local files** and **databases cannot** be **accessed**.
  - User is **able** to **disable** client side scripting

Client V/S Server Side Scripting

Server Side Scripting	Client Side Scripting

<script> tag

- The <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains **scripting statements, or** it points to an **external script** file through the **src** attribute.
- Example :

Code

```
<html>
 <head>
  <title>HTML script Tag</title>
 </head>
 <body>
  <script type="text/javascript">
   // Java Script Code Here
  </script>
 </body>
</html>
```

Code

```
<html>
 <head>
  <title>HTML script Tag</title>
 </head>
 <body>
  <script src="PathToJS">
  </script>
 </body>
</html>
```

Variables

- A variable can contain several types of value:
  - **String** : character wrapped in quotes e.g. “rajkot”
  - **Number** : a numeric value e.g. 156, 100, 1.2
  - **Boolean** : a value of true or false
  - **Null** : an empty variable
  - **Function** : a function name
  - **Object** : an object
- Attributes of Javascript variables :
  - It is a **case sensitive**. *(mynum and MyNum are different variables)*
  - It **cannot** contain **punctuation**, **space** or **start** with a **digit**
  - It **cannot** be a JavaScript **reserved** word

Strings

- A **string** can be defined as a **sequence of letters, digits, punctuation and so on**.
- A **string** in a JavaScript is **wrapped** with **single or double quotes**
- Strings can be **joined** together with the **+ operator**, which is called **concatenation**.

  For Example,

  mystring = “my college name is ” + “Darshan”;
- As string is an object type it also has some useful features.

  For Example,

  lenStr = mystring**.length**;

  Which returns the **length** of the **string** in **integer**
- Backslash escape character (\ ) is used for avoiding problem to use special character “”, ‘’, \ etc.
  - e.g. var text = “I am studding in \“Darshan\" University."; /* I am studding in “Darshan" University. */

String Handling in JavaScript

- String Concatenation
  - With “+” operator
    - e.g. “Darshan” + “ ” + “University”;
  - With “concat()” function
    - e.g.
    - var text1 = “Darshan";
    - var text2 = “University";
    - var text3 = “Rajkot”;
    - var result = text1.concat(“ ”, text2, “ ”, text3);
- String Lengths
  - e.g.
  - var name = “Darshan”;
  - var length = name.length;

String Handling in JavaScript (Cont.)

- Slicing and Extracting Parts of a String
  - e.g.
  - var name = “Darshan”;
  - var part = name.slice(0,3); /* output will be “Dar” */
- Changing Casing in String
  - toLowerCase()
  - toUpperCase()
    - e.g.
    - var name= “Darshan";
    - var result = name.toLowerCase();
- Other String functions are:
  - charAt(), charCodeAt()
  - startsWith(), endsWith()
  - includes()
  - indexOf(), lastIndexOf()
  - match()
  - repeat()
  - replace("", "")
  - search()
  - split()
  - substr(m, n), substring(m, n)
  - toString()
  - trim()

Numbers Handling in JavaScript

- Rounding off a number
  - e.g.
  - var x = 9.656;
  - x.toFixed(2); /* 9.66 */
- Convert JavaScript variables to numbers
  - e.g.
  - Number("10"); /* 10 */
  - Number("John"); /* NaN */
- “parseInt()” Method
  - e.g.
  - parseInt("10.33"); /* 10 */
  - parseInt("10 years"); /* 10 */
  - parseInt("years 10"); /* NaN */
- Other number functions are:
  - parseFloat()
  - valueOf()
  - toPrecision()
  - toExponential()

Arithmetic Operators in JavaScript

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement
**	Exponentiation (Added in ES2016)

Functions

- A JavaScript function is a **block of code** designed to perform a particular task.
- A JavaScript function is **executed** when "**something**" **invokes** it.
- A JavaScript function is defined with the **function** keyword, **followed by a name**, **followed by parentheses ()**.
- The **parentheses** may **include parameter** names **separated** by **commas**: (parameter1, parameter2, ...)
- The **code to be executed**, by the function, is placed inside **curly brackets**.
- Example :

Code

```
function myFunction(p1, p2) {
        return p1 * p2;
}
```

Functions (Cont.)

- When JavaScript **reaches a return** statement, the function will **stop executing**.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- The code inside the function will execute when "something" invokes (calls) the function:
  - When an **event occurs** (when a user clicks a button)
  - When it is invoked (**called**) from JavaScript code
  - **Automatically** (self invoked)

Conditions

- Simple if statement
- If else statement
- Else if ladder statement
- Nested if statement
- Switch statement

Loops

- While loop
- Do while loop
- For loop
- For in loop
- For of loop

- The break statement use to jumps out of a loop.
- The continue statement use to jumps over one iteration in the loop.

Arrays

- An **array** is a **collection of data**, each item in array has an index to access it.
- Ways to use array in JavaScript
  - var myArray = new Array();
    myArray[0] = “darshan”;
    myArray[1] = 222;
    myArray[2] = false;
  - var myArray = new Array(“darshan” , 123 , true);

Pop up Boxes

- Popup boxes can be used to raise an alert, or to get confirmation on any input or to have a kind of input from the users.
- JavaScript supports **three** types of popup boxes.
  - Alert box
  - Confirm box
  - Prompt box

Alert Box

- An **alert box** is used if you want to **make sure** information **comes through** to the **user**.
- When an alert box pops up, the user will **have to click "OK"** to proceed.
- It can be used to display the result of validation.

Code

```
<html>
   <head>
            <title>Alert Box</title>
   </head>
   <body>
            <script>
                     alert("Hello World");
            </script>
   </body>
</html>
```

Confirm Box

- A **confirm box** is used if you want the user to **accept something**.
- **When** a confirm box **pops up**, the user will have to **click** either "**OK**" or "**Cancel**" to proceed, If the user clicks "**OK**", the box **returns true**. If the user clicks "**Cancel**", the box **returns false**.
- Example :

Code

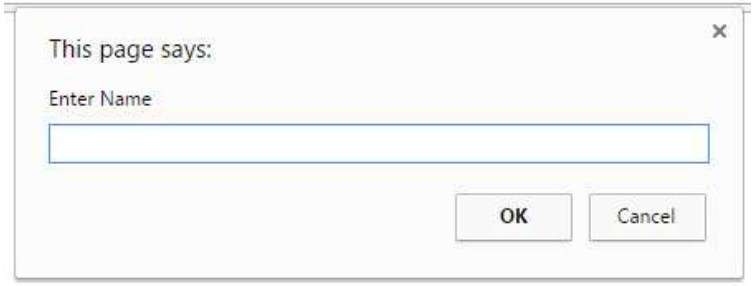
```
<script>
  var a = confirm("Are you sure??");
  if(a==true) {
            alert("User Accepted");
  }
  else  {
            alert("User Cancled");
  }
</script>
```

Prompt Box

- A **prompt box** is used if you want the user to **input a value**.
- **When** a prompt box **pops up**, user have to click either "**OK**" or "**Cancel**" to proceed, If the user clicks "**OK**" the box **returns the input value**, If the user clicks "**Cancel**" the box **returns null**.
- Example:

Code

```
<script>
  var a = prompt("Enter Name");
  alert("User Entered " + a);
</script>
```



External JavaScript

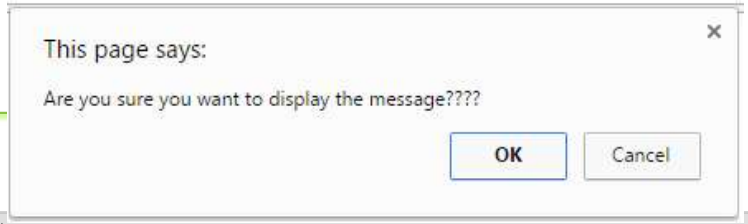
- We can create external JavaScript file and embed it in many html pages.
- It provides code reusability because single JavaScript file can be used in several html pages.
- An external JavaScript file must be saved by **.js** extension.
- To embed the External JavaScript File to HTML we can use ***script*** tag with ***src*** attribute in the head section to specify the path of JavaScript file.
- For Example :

message.js

```
function myAlert(msg) {
        if(confirm("Are you sure you want to display the message????")) {
                alert(msg);
        }
        else {
                alert("Message not Displayed as User Canceled Operation");
        }
}
```

myHtml.html

```
<html>
  <head>
    <script src="message.js"></script>
  </head>
  <body>
    <script> myAlert("Hello World"); </script>
  </body>
</html>
```



JavaScript Objects

- An object is just a special kind of data, with properties and methods.
- Accessing Object Properties
  - Properties are the values associated with an object.
  - The syntax for accessing the property of an object is below

    objectName.propertyName
  - This example uses the length property of the Javascript's inbuilt object(String) to find the length of a string:

    var message="Hello World!";
    var x=message.length;
- Accessing Object Methods
  - Methods are the actions that can be performed on objects.
  - You can call a method with the following syntax. *objectName.methodName()*
  - This example uses the toUpperCase method of the String object to convert string to upper case:

    var message="Hello World!";
    var x=message.toUpperCase();

JavaScript's inbuilt Objects

- JavaScript comes with some inbuilt objects which are,
  - String
  - Date
  - Array
  - Boolean
  - Math
  - RegExp

  etc….

Math Object in JavaScript

- The Math object allows you to perform mathematical tasks.
- The Math object includes several mathematical constants and methods.
- Math object has some properties which are,

Properties	Description
E	Returns Euler's number(approx.2.718)
LN2	Returns the natural logarithm of 2 (approx.0.693)
LN10	Returns the natural logarithm of 10 (approx.2.302)
LOG2E	Returns the base-2 logarithm of E (approx.1.442)
LOG10E	Returns the base-10 logarithm of E (approx.0.434)
PI	Returns PI(approx.3.14)
SQRT1_2	Returns square root of ½
SQRT2	Returns square root of 2

Math Methods

Method	Description
abs(x)	Returns the absolute value of x
sin(x)	Returns the sine of x (x is in radians)
cos(x)	Returns the cosine of x (x is in radians)
tan(x)	Returns the tan of x (x is in radians)
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value
atan2(x)	Returns arctangent of x
random()	Returns random floating number between 0 to 1

Method	Description
exp(x)	Returns the value of Ex
ceil(x)	Returns x, rounded upwards to the nearest integer
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm(base E) of x
round(x)	Rounds x to the nearest integer
pow(x,y)	Returns the value of x to the power of y
max(x,y,z,...,n)	Returns the number with the highest value
sqrt(x)	Returns the square root of x

User Defined Objects

- JavaScript allows you to create your own objects.
- The first step is to use the new operator.

  var myObj= new Object();
- This creates an empty object, This can then be used to start a new object that you can then give new properties and methods.
- In object- oriented programming such a new object is usually given a constructor to initialize values when it is first created.
- However, it is also possible to assign values when it is made with literal values.

example

```
person={
        firstname: "Darshan",
        lastname: "College",
        age: 50,
        eyecolor: "blue"
}
alert(person.firstname + " is " + person.age + " years old.");
```

User - Defined Objects (Cont.)

- A constructor is pre defined method that will initialize your object.
- To do this in JavaScript a function is used that is invoked through the *new* operator.
- Any properties inside the newly created object are assigned using *this* keyword, referring to the current object being created.

example

```
<script>
        function person(firstname, lastname, age){
                this.firstname = firstname;
                this.lastname = lastname;
                this. changeFirstName = function (name){ this.firstname = name };
        }
        var person1=new person("Narendra","Modi",50);
        person1.changeFirstName("NAMO");
        alert(person1.firstname + " "+ person1.lastname);
</script>
```

Document Object Model (DOM)

- The Document Object Model is a platform and language neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.
- The **window** object is the primary point from which most other objects come.
- From the current window object **access** and **control** can be given to most aspects of the **browser features** and the **HTML document**.
- When we write :

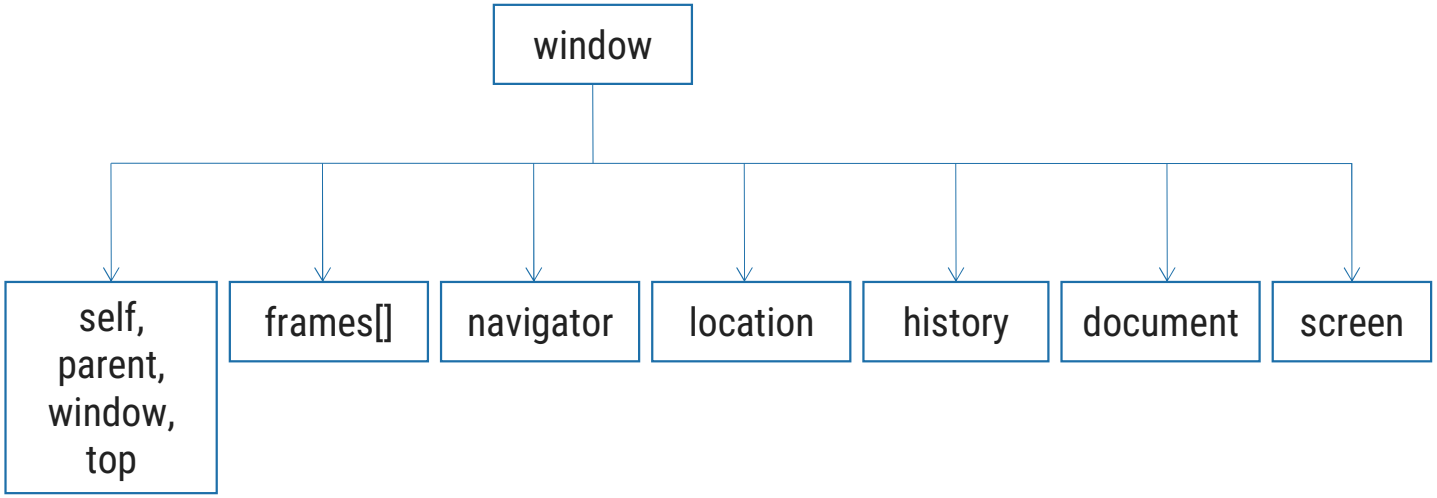
  document.write("Hello World");
- We are actually writing :

  window.document.write("Hello World");

  The **window** is just there by default

DOM (Cont)

- This **window** object represents the window or frame that displays the document and is the global object in client side programming for JavaScript.
- All the client side objects are connected to the window object.



Document Object Properties

Property	Description
anchors	Returns a collection of all the anchors in the document
applets	Returns a collection of all the applets in the document
body	Returns the body element of the document
cookie	Returns all name/value pairs of cookies in the document
domain	Returns the domain name of the server that loaded the document
forms	Returns a collection of all the forms in the document
images	Returns a collection of all the images in the document
links	Returns a collection of all the links in the document (CSSs)
referrer	Returns the URL of the document that loaded the current document
title	Sets or returns the title of the document
URL	Returns the full URL of the document

Note: Red color properties are deprecated and not supported in many browser versions

Document Object Methods

Method	Description
write()	Writes HTML expressions or JavaScript code to a document
writeln()	Same as write(), but adds a newline character after each statement
open()	Opens an output stream to collect the output from document.write() or document.writeln()
close()	Closes the output stream previously opened with document.open()
getElementById()	Accesses element with a specified id
getElementsByName()	Accesses all elements with a specified name
getElementsByTagName()	Accesses all elements with a specified tag name
setTimeout(), clearTimeout()	Set a time period for calling a function once; or cancel it.

getElementById()

- When we suppose to get the reference of the element from HTML in JavaScript using id specified in the HTML we can use this method.
- Example :

HTML

```
<html>
   <body>
      <input type="text" id="myText">
   </body>
</html>
```

JavaScript

```
<script>
   function myFunction()
   {
      var txt = document.getElementById("myText");
      alert(txt.value);
   }
</script>
```

getElementsByName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name specified in the HTML we can use this method.
- It will return the array of elements with the provided name.
- Example :

HTML

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

JavaScript

```
<script>
function myFunction()
{
  a=document.getElementsByName("myText")[0];
   alert(a.value);
}
</script>
```

getElementsByTagName()

- When we suppose to get the reference of the elements from HTML in JavaScript using name of the tag specified in the HTML we can use this method.
- It will return the array of elements with the provided tag name.
- Example :

HTML

```
<html>
   <body>
      <input type="text" name="uname">
      <input type="text" name="pword">
   </body>
</html>
```

JavaScript

```
<script>
function myFunction() {
  a=document.getElementsByTagName("input");
  alert(a[0].value);
  alert(a[1].value);
}
</script>
```

Forms using DOM

- We can access the elements of form in DOM quite easily using the name/id of the form.
- Example :

HTML

```
<html>
   <body>
     <form name="myForm">
       <input type="text" name="uname">
       <input type="text" name="pword">
       <input type="button" onClick="f()">
    </form>
   </body>
</html>
```

JS

```
function f()
{
   var a = document.forms["myForm"];
   var u = a.uname.value;
   var p = a.pword.value;
   if(u=="admin" && p=="123")
   {
      alert("valid");
   }
   else
   {
       alert("Invalid");
   }
}
```

Validation

- Validation is the process of **checking** data against a **standard** or **requirement**.
- Form validation normally used to occur at the server, after client entered necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server.

Validation (Cont.)

Form validation generally performs two functions.

1. **Basic Validation**
   - Emptiness
   - Confirm Password
   - Length Validation etc……
2. **Data Format Validation**

   Secondly, the data that is entered must be checked for correct **form** and **value**.
   - Email Validation
   - Mobile Number Validation
   - Enrollment Number Validation etc….

Validation using RegExp

- A regular expression is an object that describes a pattern of characters.
- Regular expressions are used to perform pattern-matching and "search-and-replace" functions on text.
- example:

```
var pattern = "^[\\w]+$";   // will allow only words in the string
var regex = new RegExp(pattern);
If(regex.test(testString)){
   //Valid
} else {
   //Invalid
}
```

RegExp (Cont.) (Metacharacters)

- To find **word** characters in the string we can use **`\w`**
  - We can also use [a-zA-Z0-9_] for the same
- To find **non-word** characters in the string we can use **`\W`**
- to find **digit** characters in the string we can use **`\d`**
  - We can also use [0-9] for the same
- To find **non-digit** characters in the string we can use **`\D`**
- We can use **`\n`** for **new line** and **`\t`** for tab

RegExp (Cont.) (Quantifiers)

Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X to Y n's
n{X,}	Matches any string that contains a sequence of at least X n's

Email Validation Using RegExp

JavaScript

```
<script>
        function checkMail()
        {
                var a = document.getElementById("myText").value;
                var pattern ="^[\\w-_\.]*[\\w-_\.]\@[\\w]\.+[\\w]+[\\w]$”;
                var regex = new RegExp(pattern);
                if(regex.test(a))
                {
                        alert("Valid");
                }
                else
                {
                        alert("Invalid");
                }
        }
</script>
```

DHTML – Combining HTML,CSS & JS

- DHTML, or Dynamic HTML, is a combination of HTML, JavaScript and CSS.
- The main problem with DHTML, which was introduced in the 4.0 series of browsers, is **compatibility**.
- The main focus generally when speaking of DHTML is animation and other such dynamic effects.
- We can obtain reference of any HTML or CSS element in JavaSCript using below 3 methods.
  1. document.getElementById("IdOfElement")
  2. document.getElementsByName("NameOfElement")
  3. document.getElementsByTagName("TagName")
- After obtaining the reference of the element you can change the attributes of the same using reference.attribute syntax
- For Example :

HTML Code

```
<img src="abc.jpg" id="myImg">
```

JS Code

```
<script>
  var a = document.getElementById('myImg');
  a.src  = "xyz.jpg";
</script>
```

DHTML (Cont) (Example)

JavaScript

```
<html>
     <body>
          <div id=“myDiv”>
       Red Alert !!!!!!
          </div>
          <script>
             var objDiv = document.getElementById(“myDiv”);
             var colors = [‘white’,’yellow’,’orange’,’red’];
             var nextColor = 0;
      setInterval(“objDiv.style.backgroundColor = colors[nextColor++%colors.length];”,500);
          </script>
     </body>
</html>
```

HTML Element Properties

Event	Description
className	Sets or returns the class attribute of an element
id	Sets or returns the id of an element
innerHTML	Sets or returns the HTML contents (+text) of an element
style	Sets or returns the style attribute of an element
tabIndex	Sets or returns the tab order of an element
title	Sets or returns the title attribute of an element
value	Sets or returns the value attribute of an element

Mouse Events

Event	Attribute	Description
click	onclick	The event occurs when the user clicks on an element
dblclick	ondblclick	The event occurs when the user double-clicks on an element
mousedown	onmousedown	The event occurs when a user presses a mouse button over an element
mousemove	onmousemove	The event occurs when a user moves the mouse pointer over an element
mouseover	onmouseover	The event occurs when a user mouse over an element
mouseout	onmouseout	The event occurs when a user moves the mouse pointer out of an element
mouseup	onmouseup	The event occurs when a user releases a mouse button over an element

Keyboard Events

Event	Attribute	Description
keydown	onkeydown	The event occurs when the user is pressing a key or holding down a key
keypress	onkeypress	The event occurs when the user is pressing a key or holding down a key
keyup	onkeyup	The event occurs when a keyboard key is released

Frame/Object Events

Event	Attribute	Description
abort	onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)
error	onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)
load	onload	The event occurs when a document, frameset, or <object> has been loaded
resize	onresize	The event occurs when a document view is resized
scroll	onscroll	The event occurs when a document view is scrolled
unload	onunload	The event occurs when a document is removed from a window or frame (for <body> and <frameset>)

Form Events

Event	Attribute	Description
blur	onblur	The event occurs when a form element loses focus
change	onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)
focus	onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)
reset	onreset	The event occurs when a form is reset
select	onselect	The event occurs when a user selects some text (for <input> and <textarea>)
submit	onsubmit	The event occurs when a form is submitted

Callbacks in Javascript

```
<script type="text/javascript">
        function add(a, b, d, nextFun){
                ans = a + b;
                nextFun(ans, d);
                return ans;
        }

        function mul(a, b){
                ans = a * b;
                return ans;
        }

        p1 = 10;
        p2 = 20;
        p3 = 30;

        //(P1 + P2) * P3
        ans = add(p1, p2, p3, mul);
        alert(ans);
 </script>
```

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-05

Server Side Scripting with PHP

Prof. Vijay M Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline

- ✓ Introduction to PHP
- ✓ Basics of PHP
- ✓ Variables
- ✓ Array
- ✓ Function
- ✓ Browser Control
- ✓ Browser Detection
- ✓ String Functions
- ✓ Form Processing
- ✓ File Uploadig
- ✓ File Handling
- ✓ Exception Handling
- ✓ Date and Time zone
- ✓ Cookie / Session

Introduction

- PHP is a scripting language that allows you to create dynamic Web pages
- You can embed PHP scripting within normal html coding
- PHP was designed primarily for the Web
- PHP includes a comprehensive set of database access functions
- High performance/ease of learning/low cost
- Open-source
  - Anyone may view, modify and redistribute source code
  - Supported freely by community
- Platform independent

Basics of PHP

- PHP files end with .php, you may see .php3 .phtml .php4 as well
- PHP code is contained within tags

  <?php ?> **or**

  Short-open: <? ?>
- HTML script tags: (This syntax is removed after PHP 7.0.0)

  <script language="php"> </script>
- Comments

  // for single line

  /* */ for multiline

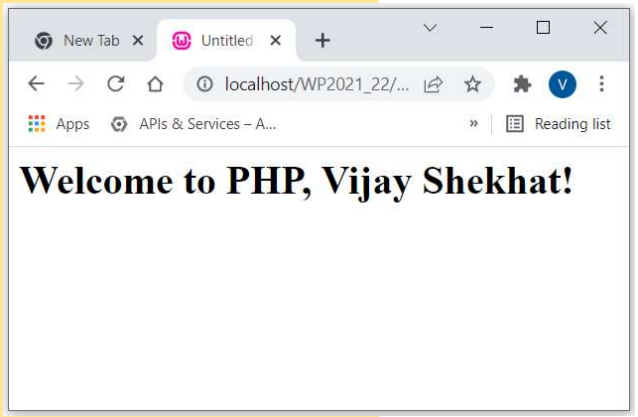
PHP Basic Example

```
7  <?php
8     $name = "Vijay Shekhat";    // declaration
9  ?>
10
11 <html xmlns = "http://www.w3.org/1999/xhtml">
12    <head>
13       <title>Untitled document</title>
14    </head>
15
16    <body style = "font-size: 2em">
17       <p>
18          <strong>
19
20             <!-- print variable name's value -->
21             Welcome to PHP, <?php print( "$name" ); ?>!
22          </strong>
23       </p>
24    </body>
25 </html>
```

Scripting delimiters

Declare variable $name

Single-line comment



Function `print` outputs the value of variable $name

Variables

- All variables begin with **$** and can contain letters, digits and underscore (variable name can not begin with digit)
- PHP variables are Case-sensitive
- Don’t need to declare variables
- The value of a variable is the value of its most recent assignment
- Variables have no specific type other than the type of their current value
- Can have variable variables $$variable **(not recommended)**

Example :

```
$a = “b”;
$b = “Vijay Shekhat”;
echo($$a);
```

Variables (Cont.)

- Variable names inside strings replaced by their value

  Example : $name = “Vijay Shekhat”;
  $str = “My name is $name”;

- Type conversions
  - settype function

```
<?php
    $b = 32; // integer
    settype($b, "string");
?>
```

  - Type casting

```
<?php
    echo (int)12.5;  // 12
?>
```

- Concatenation operator
  - . (period)

Data Type	Description
int, integer	Whole numbers (i.e., numbers without a decimal point).
float, double	Real numbers (i.e., numbers containing a decimal point).
string	Text enclosed in either single (') or double ("") quotes.
bool, boolean	True or false.
array	Group of elements.
object	Group of associated data and methods.
resource	An external data source.
NULL	No value.

Variables Scope

- Scope refers to where within a script or program a variable has meaning or a value
- Mostly script variables are available to you anywhere within your script.
- Note that variables inside functions are local to that function and a function cannot access script variables which are outside the function even if they are in the same file.
- The modifiers **global** and **static** allow function variables to be accessed outside the function or to hold their value between function calls respectively

Decision Making Statements

- Simple if statement.
- If else statement.
- Else if ladder statement.
- Nested if statement.
- Switch statement.

Loops

- While loop.
- Do while loop.
- For loop.
- Foreach loop.

PHP Arrays

- Array is a group of variable that can store multiple values under a single name.
- In PHP, there are three types of array.
  - Numeric Array

    These arrays can store numbers, strings and any object but their index will be represented by numbers. By default array index starts from zero.
  - Associative Array

    The associative arrays are very similar to numeric arrays in term of functionality but they are different in terms of their index. Associative array will have their index as string so that you can establish a strong association between key and values.
  - Multidimensional Array

    A multi-dimensional array each element in the main array can also be an array. And each element in the sub-array can be an array, and so on. Values in the multi-dimensional array are accessed using multiple index.

PHP Arrays (Example)

```
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8     <head>
9        <title>Array manipulation</title>
10    </head>
11
12    <body>
13       <?php
14
15          // create array first
16          print( "<strong>Creating the first array</strong>
17             <br />" );
18          $first[ 0 ] = "zero";
19          $first[ 1 ] =
20          $first[ 2 ] =
21          $first[]= "t
22
23          // print each element's index and value
24          for ( $i = 0; $i < count( $first ); $i++ )
25             print( "Element $i is $first[$i] <br />" );
```

Create the array `$first` by assigning a value to an array element.

Assign a value to the array, omitting the index. A
of the ar

Use a `for` loop to print out each element's index and value. Function `count` returns the total number of elements in the array.

PHP Arrays (Example) (cont.)

```
27          print( "<br /><strong>Creating t
28             </strong><br />" );
29
30          // call function array to create
31          $second = array( "zero", "one", "two", "three" );
32          for ( $i = 0; $i < count( $second ); $i++ )
33             print( "Element $i is $second[$i] <br />" );
34
35          print( "<br /><strong>Creating the third array
36             </strong><br />" );
37
38          // assign values to non-numerical indices
39          $third[ "ArtTic" ] = 2
40          $third[ "LunaTic" ] =
41          $third[ "GalAnt" ] = 23;
42
43          // iterate through the ar
44          // element's name and value
45          for ( reset( $third ); $element = key( $third );
46             next( $third ) )
47             print( "$eleme
```

Call function `array` to create an array that contains the arguments passed to it. Store the array in variable `$second`.

Assign values to non-numerical indices in array `$third`.

Function `reset` sets the internal pointer to the first element of the array.

Function key returns the index of the element

Function `next` moves the internal pointer to the next element.

PHP Arrays (Example) (cont.)

```
49          print( "<br /><strong>Creating the fourth array
50             </strong><br />" );
51
52          // call function array to create array fourth using
53          // string indices
54          $fourth = array(
55             "January"   => "first",   "February" => "second",
56             "March"     => "third",
57             "May"       => "fifth",
58             "July"      => "seventh
59             "September" => "ninth",
60             "November"  => "eleventh
61             );
62
63          // print each element's name and value
64          foreach ( $fourth as $element => $value )
65             print( "$element is the $value month <br />" );
66       ?>
67    </body>
68 </html>
```

Operator => is used in function array to assign each element a string index. The value to the left of the operator is the array index, and the value to the right is the element's value.

PHP Array Functions

- `count($array) / sizeof($array)`
  Count all elements in an array, or something in an object
- `array_shift($array)`
  Remove an item from the start of an array and shift all the numeric indexes
- `array_pop($array)`
  Remove an item from the end of an array and return the value of that element
- `array_unshift($array,"New Item")`
  Adds item at the beginning of an array
- `array_push($array,"New Item")`
  Adds item at the end of an array

PHP Array Functions (cont.)

- `sort($array [, $sort_flags])`
  - Array can be sorted using this command, which will order them from the lowest to highest
  - If there is a set of string stored in the array they will be sorted alphabetically.
  - The type of sort applied can be chosen with the second optional parameter `$sort_flags` which can be
    - `SORT_REGULAR` compare items normally (don't change type)
    - `SORT_NUMERIC` compare items numerically
    - `SORT_STRING` compare items as string
    - `SORT_LOCALE_STRING` compare items as string, based on the current locale
- `rsort($array [, $sort_flags])`
  - It will sort array in reverse order (i.e. from highest to lowest)
- `shuffle($array)`

  It will mix items in an array randomly.
- `array_merge($array1,$array2)`

  It will merge two arrays.
- `array_slice($array,$offset,$length)`

  returns the sequence of elements from the array $array as specified by the $offset and $length parameters.

PHP Functions

- A function is a piece of code which takes zero or more input in the form of parameter and does some processing and returns a value.
- Creating PHP function

  Begins with keyword function and then the space and then the name of the function then parentheses"()" and then code block "{}"

```
<?php
        function functionName()
        {
                //code to be executed;
        }
?>
```

Note: function name can start with a letter or underscore "_", but not a number!

PHP Functions (Cont.)

- Where to put the function implementation?

  In PHP a function could be defined before or after it is called.

```php
<?php
    function functionName()
    {
        //code to be executed;
    }

    functionName();
?>
```

Here function call is after implementation, which is valid

```php
<?php
    functionName();

    function functionName()
    {
?>
```

Here function call is before implementation, which is also valid

Browser Control

- PHP can control various features of a browser.
- This is important as often there is a need to reload the same page or redirecting the user to another page.
- Some of these features are accessed by controlling the information sent out in the HTTP header to the browser, this uses the header() command such as :

```
header("Location: index.php");
```

- We can also control the caching using same header() command

```
header("Cache-Control: no-cache");
```

Or can specify the content type like,

```
header("Content-Type: application/pdf");
```

Browser Detection

- The range of devices with browsers is increasing so it is becoming more important to know which browser and other details you are dealing with.
- The browser that the server is dealing can be identified using:

```
$browser_ID = $_SERVER['HTTP_USER_AGENT'];
```

- Typical response of the above code is follows:

  Mozilla/5.0 (**Windows NT 10.0**; Win64; **x64**) AppleWebKit/537.36 (KHTML, like Gecko) **Chrome**/56.0.2924.87

  ↪ which specifies that user is using **Chrome** browser and **windows 10** OS with **64 bit** architecture

PHP String Functions

- Most of the time in PHP we suppose to do manipulation of strings, wheatear it be input from the user, databases or files that have been written.
- String can be think as a array of characters, so it is possible to do something like this,

```
$mystring = “Welcome to Darshan College of Engineering”;
print ($mystring[11]) ; // which will print ‘D’
```

  - This uses an index as an offset from the beginning of the string starting at **0**
- Often, there are specific things that need to be done to a string, such as reversing, extracting part of it, finding a match to part or changing case etc..

PHP String Functions (Cont.)

String Function	Purpose

PHP String Functions (Cont.)

String Function	Purpose

Form Processing

- We can access form data using inbuilt PHP associative array.
  - `$_GET` => in case we have used **get** method in the form
  - `$_POST` => in case we have used **post** method in the form
  - `$_REQUEST` => in both the cases

- For example,

html

```
<form action="recive.php" method="get">
    <input type="text" name="UserName">
    <input type="submit">
</form>
```

recive.php

```
<?php
   $u = $_GET['UserName'];
   echo($u);
?>
```

File uploads

```
<!DOCTYPE html>
<html>
<body>
        <form action="con_upload.php" method="post" enctype="multipart/form-data">
                Select Profile Picture to upload: <input type="file" name="profilepic">
                <input type="submit" value="Upload Profile Picture" name="submit">
        </form>
</body>
</html>
```

```
// con_upload.php
<?php
move_uploaded_file($_FILES['profilepic']['tmp_name'],'./uploads/'. $_FILES['profilepic']['name']);
?>
```

Regular Expression

Function	Description
preg_match()	Returns 1 if the pattern was found in the string and 0 if not
preg_match_all()	Returns the number of times the pattern was found in the string, which may also be 0
preg_replace()	Returns a new string where matched patterns have been replaced with another string

```php
<?php
$str = "Darshan University";
$pattern = "/darshan/i";
echo preg_match($pattern, $str);
?>
//Output: 1
```

```php
<?php
$str = "Darshan College, Darshan University";
$pattern = "/darshan/i";
echo preg_match_all($pattern, $str);
?>
//Output: 2
```

```php
<?php
$str = "Darshan College";
$pattern = "/College/";
echo preg_replace($pattern, "University", $str);
?>
//Output: "Darshan University"
```

File Handling in PHP

- PHP has several functions for creating, reading, uploading, and editing files.
- fopen($filename, $mode) will return the handle to access file.
  - "r" (Read only. Starts at the beginning of the file)
  - "r+" (Read/Write. Starts at the beginning of the file)
  - "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist)
  - "w+" (Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist)
  - "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist)
  - "a+" (Read/Write. Preserves file content by writing to the end of the file)

File Handling in PHP (Cont.)

Function	Purpose

File Handling Example

text.txt

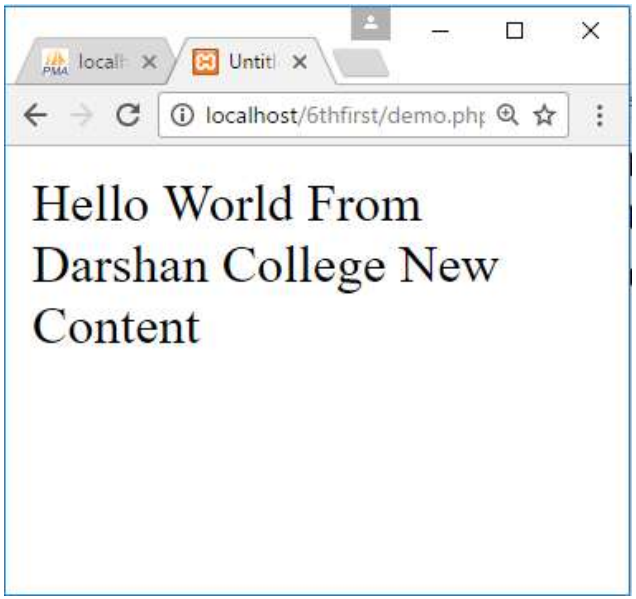
Hello World From Darshan College

Read File

```
<?php
	$file = fopen("text.txt","a+");
	$text = fread($file,filesize("text.txt"));
	echo($text);
?>
```

Write File

```
<?php
	fwrite($file," New Content");
	$file = fopen("text.txt","a+");
	$text = fread($file,filesize("text.txt"));
	echo($text);
?>
```



Exception Handling

```php
<?php
	try{
		echo ("<br>".myFunction(5, 3));
		echo ("<br>".myFunction(5, 0));
	}
	catch(Exception $e) {
		echo("<br>Message: ".$e->getMessage());
	}
	finally{
		echo("<br>This will always execute");
	}

	function myFunction($a, $n){
		if ($n == 0) {
			throw new Exception("Divide by zero exception", 2);
		}
		$ans = $a / $n;
		return $ans;
	}
?>
```

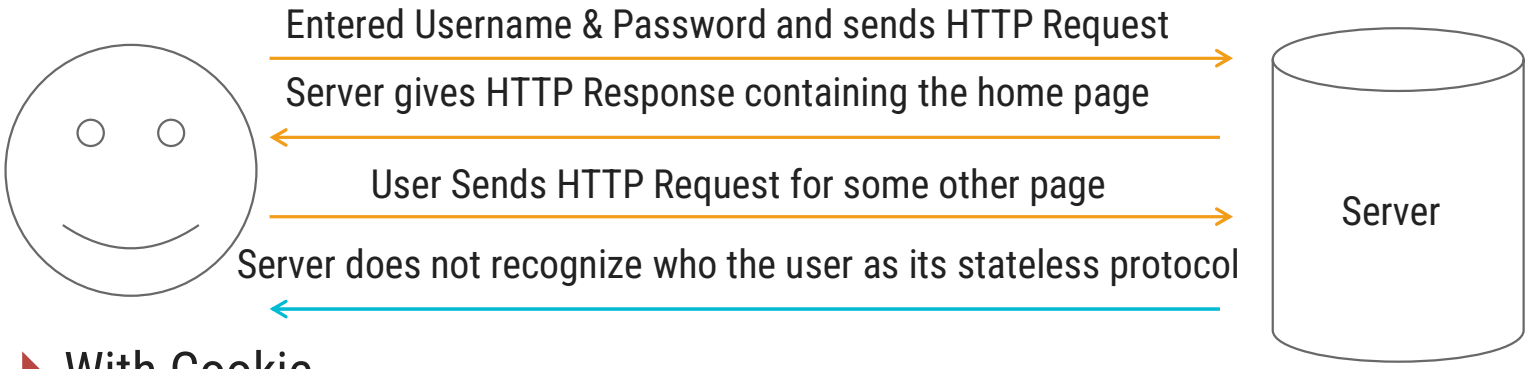
Date and Time zone

- Calculate Number of days left in your next birthday.

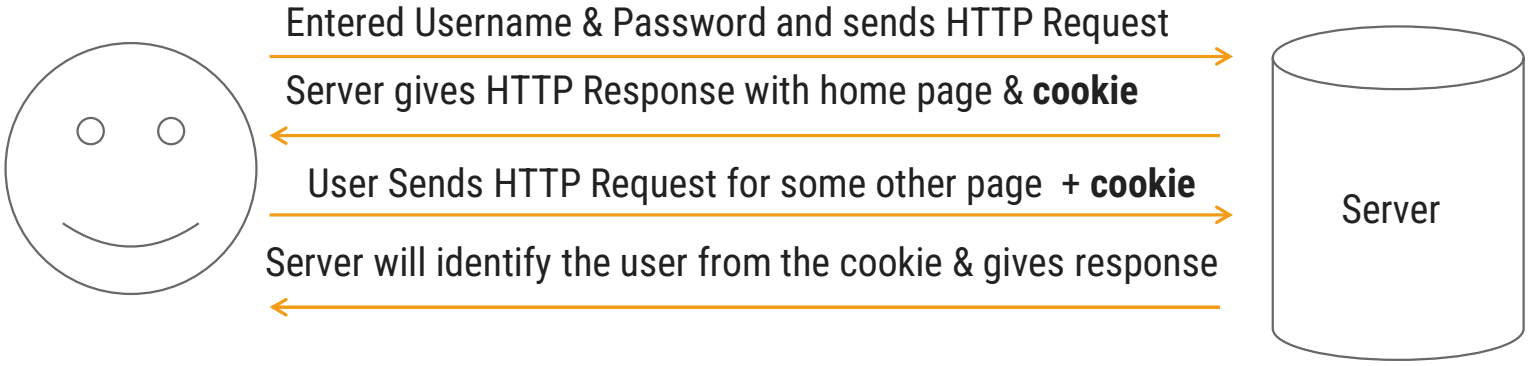
```php
<?php
	$dob = new DateTime("1990-02-29");
	$currentDate = new DateTime(date("Y-m-d"));

	$dobNew = date_create(date_format($dob,date("Y")."-m-d"));

	if($currentDate > $dobNew)
	{
		date_add($dobNew, date_interval_create_from_date_string('1 year'));
	}

	$dateDiff= date_diff($dobNew,$currentDate);

	echo("Days left in your next birthday: " . $dateDiff->days . "days");
?>
```

Cookies in PHP

- Without Cookie



- With Cookie



Cookies in PHP (Cont.)

- HTTP cookies are data which a server-side script sends to a web client to keep for a period of time.
- On every subsequent HTTP request, the web client automatically sends the cookies back to server (unless the cookie support is turned off).
- The cookies are embedded in the HTTP header (and therefore not visible to the users).
- **Shortcomings/disadvantages of using cookies to keep data**
  - User may turn off cookies support.
  - Users using the same browser share the cookies.
  - Limited number of cookies (~20) per server/domain and limited size (~4k bytes) per cookie
  - Client can temper with cookies

Cookies in PHP (Cont.)

- To set a cookie, call setcookie()
  - e.g., `setcookie('username', 'Vijay');`
- To delete a cookie (use setcookie() without a value)
  - e.g., `setcookie('username');`
- To retrieve a cookie, refer to $COOKIE
  - e.g. `$username = $_COOKIE['username'];`
- Note :
  - Cookies can only be set before any output is sent.
  - You cannot set and access a cookie in the same page. Cookies set in a page are available only in the future requests.

Cookies in PHP (Cont.)

setcookie(name, value, expiration, path, domain, secure)

- Expiration
  - Cookie expiration time in seconds
  - 0 ➔ The cookie is not to be stored persistently and will be deleted when the web client closes.
  - Negative value ➔ Request the web client to delete the cookie
  - e.g.: setcookie('username', 'Joe', time() + 1800); // Expire in 30 minutes
- Path
  - Sets the path to which the cookie applies. (Default is '/')
- Domain
  - The domain that the cookie is available.
- Secure
  - This can be set to 1 to specify that the cookie should only be sent by secure transmission using HTTPS otherwise set to 0 which mean cookie can be sent by regular HTTP.

Session in PHP

- PHP Session is a way to make data accessible across the various pages of an entire website.
- A session creates a file in a temporary directory on the server where registered session variables and their values are stored.
- The **location** of the temporary file is determined by a setting in the **php.ini** file called **session.save_path**.
- When a session is started following things happen
  - PHP first creates a unique identifier for that particular session which is a random string of 32 hexadecimal numbers such as 3c7foj34c3jj973hjkop2fc937e3443.
  - A cookie called **PHPSESSID** is automatically sent to the user's computer to store unique session identification string.
  - A file is automatically created on the server in the designated temporary directory and bears the name of the unique identifier prefixed by sess_, sess_3c7foj34c3jj973hjkop2fc937e3443.

Starting a PHP Session

- A PHP session is easily started by making a call to the `session_start()` function.
- This function first checks if a session is already started and if none is started then it starts one.
- It is recommended to put the call to `session_start()` at the beginning of the page.
- The following example starts a session then register a variable called counter that is incremented each time the page is visited during the session.

```
<?php
        session_start();
        if( isset( $_SESSION['counter'] ) ) {
                        $_SESSION['counter'] += 1;
        }else {
                        $_SESSION['counter'] = 1;
        }
        $msg = "You have visited this page ".
                $_SESSION['counter'];
        $msg .= "in this session.";
?>
    <html><head>
          <title>Setting up a PHP session</title>
       </head><body>
          <?php  echo ( $msg ); ?>
       </body></html>
```

Destroying a PHP Session

- A PHP session can be destroyed by `session_destroy()` function.
- This function does not need any argument and a single call can destroy all the session variables.

Logout.php

```
<?php
        session_destroy();
?>
```

- If you want to destroy a single session variable then you can use `unset()` function to unset a session variable.

Logout.php

```
<?php
        unset($_SESSION['counter']);
?>
```

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP) (3160713)

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Unit-06

Database programming with PHP and MySQL

Prof. Vijay M Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline

1. Basic MySQL Commands
2. Database connectivity
   - Connection to the Server
3. Creating a Database
4. Selecting a Database
5. Listing Database
6. Listing Table Names
7. Creating a Database Table
8. Inserting Data
9. Updating Data
10. Deleting Databases
11. Select Queries
12. Accessing the Result
13. Prepared Statements
14. Stored Procedure

Basic MySQL Commands

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

PHP functions for database connectivity

```php
<?php
	$con = mysqli_connect("localhost","username","password","dbname");

	// Check connection error
	if (mysqli_connect_errno()) {
		echo "Failed to connect to MySQL: " . mysqli_connect_error();
		exit();
	}

	//Close dataase connection
	mysqli_close($con);
?>
```

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Web Programming (WP) (3160713)

Dedicated Faculty, Committed Education

Darshan

Institute of Engineering & Technology

Unit-07

Advanced Web Programming Concepts

Prof. Vijay M Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778

Outline

1. Asynchronous Web Programming
2. AJAX, and Jquery
3. Web service and API development using PHP

Asynchronous Web Programming

- In a ***synchronous*** programming model, things happen **one at a time**. When you call a function that performs a long-running action, it returns **only when the action has finished** and it can return the result. This stops your program for the time the action takes.
- An ***asynchronous*** model allows **multiple things to happen at the same time**. When you start an action, your **program continues to run**. When the action finishes, the program is informed and gets access to the result.
- We can use asynchronous programming where our main program/thread is not dependent on the process we want to execute asynchronously.
- For example we want to fill the State dropdown/select based on selection of Country dropdown, we can call asynchronously to get the States of that Country.

AJAX

- AJAX stands for **Asynchronous JavaScript and XML.**
- AJAX is not a technology in itself, it describes a "new" approach to using a number of existing technologies together, including HTML or XHTML, CSS, JavaScript, DOM, XML, XSLT, and most importantly the **XMLHttpRequest** object.
- When these technologies are combined in the Ajax model, web applications are able to make quick, incremental updates to the user interface **without reloading the entire web page**.
- This makes the application faster and more responsive to user actions.
- Although **X** in Ajax stands for XML, JSON is used more than XML nowadays because of its many advantages such as being lighter and a part of JavaScript.

Client

1. Wait for the Event to Occur
2. Create an XMLHttpRequest object
3. Send HttpRequest over internet
4. Process the returned Response
5. Update the page content using DOM

Server

1. Wait for the HttpRequest
2. Process HttpRequest
3. Send Response to client

AJAX using JavaScript

- Lets continue with the Country/State dropdown example to implement AJAX using JavaScript.
- We will create a dropdown elements for the two, but only fill the data in Country dropdown.
- Then we will set change event listener on country dropdown element.
- We will then create Java Script function to handle the event, where we will create XMLHttpRequest object and send HttpRequest to the server.
- After receiving the response from the server we will update the state dropdown using DOM.

```
<select id="country">
<select id="country"
        onchange="fillState(this)">
    <option>Select Country</option>
    <option>INDIA</option>
    <option>Sri Lanka</option>
    <option>Nepal</option>
</select>
<select id="state">

</select>
```

```
function fillState(c){
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function() {
        if (this.readyState == 4 && this.status == 200) {
            var state = document.getElementById("state");
            state.innerHTML = this.responseText;
        }
    };
    xhttp.open("GET", "getState.php?c="+c.value, true);
    xhttp.send();
}
```

AJAX using JQuery

- Steps to implement AJAX in JQuery are similar to Java Script, just syntax will be different.

```
$(document).ready(
  function(){
    $("#country").change(
      function(){
        $.ajax(
          {
            url: "http://localhost:8088/6thDemo/getState.php,
            method: "POST",
            data: { c : $("#country").val() },
            success: function(result){
                $("#state").html(result);
            }
          }
        );
      }
    );
  }
);
```

Web service and API development using PHP

- The term **Web service** (**WS**) is either:
  - a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web, **or**
  - a server running on a computer device, listening for requests at a particular port over a network, serving web documents (HTML, JSON, XML, images).
- **API** is the acronym for **A**pplication **P**rogramming **I**nterface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

Frequently Asked Full Forms

- **HTTP** : Hypertext Transfer Protocol
- **DOM** : Document Object Model
- **PHP** : Hypertext Preprocessor
- **SOAP** : Simple Object Access Protocol
- **DTD** : Document Type Definition
- **XML** : Extensible Markup Language
- **XSL** : Extensible Stylesheet Language
- **WAP** : Wireless Application Protocol
- **WML** : Wireless Markup Language

Web Programming (WP)
GTU # 3160713

Dedicated Faculty, Committed Education
Darshan
Institute of Engineering & Technology

Thank You

Prof. Vijay M. Shekhat
Computer Engineering Department
Darshan Institute of Engineering & Technology, Rajkot
vijay.shekhat@darshan.ac.in
9558045778