
# Web Programming Course Module

**Course Title: Web Programming**
**Course Number: CoSc3101**
**Credit*: 7 ECTS***

# *Table of Contents*

# *Chapter 1: Introduction*

## 1.1 Course overview

This chapter deals with the Overview of Internet, website, webpage, types of websites (Static vs dynamic), webpage design tools, Web World Wide Web, Web Hosting and Domain Name Registration. To deliver these contents active learning methods such as brain storming, interactive lecture, group discussion and independent learning will be used and access students' achievement, continuous assessments such as quiz, test, class activities, assignments and others will be used.

## 1.2 The Internet and World Wide Web

The history of the Internet dates back to the Cold War days of the late 1950s. It evolved when the US Defense Force began to investigate a method of geographically dispersing their centralized computer system. It was believed that reducing reliance on one single route for transmission of data and using decentralized system, would provide a safer option for controlling their missiles. This idea was a safe guard to protect the flow of communications in the event of a major interruption. A 'fear' that in the event of a nuclear war, an enemy may destroy a link in the US chain of communications, was the precursor to the technology revolution!

Today, the internet is an international forum for exchanging information and ideas between millions of people worldwide, a rapidly growing information super-highway. In contrast, two decades prior to this, it was known mainly to those involved in the military or academia.

Q. Define the following basic terms Web Browser, Website, webpage, Home Page, Social Media or Social Networking Internet, Internet Access Provider or Internet Service Provider (ISP) – Interactive Forms.

### *1.2.1 World Wide Web:*

***What is the World Wide Web?***

- The World Wide Web (WWW) is most often called **the Web**.
- The Web is a network of computers **all over the world**.
- All the computers in the Web can **communicate with each other**.
- All the computers use a **communication standard called HTTP**.

***How does the WWW work?***

- Web information is stored in documents called **Web pages**.

- Web pages are files stored on computers called **Web servers**.
- Computers reading the Web pages are called **Web clients**.
- Web clients view the pages with a program called a **Web browser**.

Q. explains in detail world wide web, web client, web page, web server, HTTP.

## 1.3 Web Hosting and Domain Name Registration

Domain name registration and web hosting are two critical elements of running a website. It involves different steps like finding availability of domain, registration and hosting of a site. Web hosting and domain registration are frequently paired together. But it's important to understand exactly what they do. In this section we will see what web hosting and domain name registration is and their difference.

**What is web hosting?**

Web hosting is the place where all the files of your website live. Web hosting is a service that allows organizations and individuals to post a website or a web page onto the Internet. A web host, or web hosting service provider, is a business that provides the technologies and services needed for the website or webpage to be viewed in the Internet. Websites are hosted, or stored, on special computers called servers. When Internet users want to view your website, all they need to do is type your website address or domain (the domain name for the site has to be registered) into their browser. Their computer will then connect to your server and your webpages will be delivered to them through the browser.

**Domain Name Registration**

Domain name registration is the act of reserving a name on the Internet for a certain period. To get the address of a new website you have to register the domain name with a domain name registrar.

# *Chapter 2: Hyper Text Markup Language (HTML)*

This chapter deals with Web page Design and development, information Architecture & visualization, and Hyper Text Markup Language. To deliver these contents active learning methods such as brain storming, interactive lecture, group discussion and independent learning will be used and access students' achievement, continuous assessments such as quiz, test, class activities, assignments and others will be used.

## 2.1 Introduction to HTML

**HTML** (HyperText Markup Language) is the most basic building block of the Web. HTML is not a programming language rather it is a markup language that tells web browsers how to structure the web page. HTML consists of a series of elements, which you use to enclose, wrap or markup different parts of content to make it appear or act in a certain way.

HTML defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/ behavior (JavaScript).

HTML has been in use by the World-Wide Web (WWW) global information initiative since 1990. It is a simple markup language used to create hypertext documents that are platform independent. HTML markup can represent hypertext news, mail, documentations, and hyper media; menus of options; database query results;

This document covers html versions ranging from 2 up to 5

HTML is composed of several tags. HTML Document:-

- A text document containing markup tags
- The tags tell the browser how to display the document
- Should have an **.htm** or **.html** file name extension
- Can be created using a simple text editor

**An HTML document has the following basic structure:**

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

```
    <title>
       page title
    </title>
  </head>
  <body>
      content
  </body>
</html>
```

Q. Explain in detail the different versions of HTML?

## 2.2 HTML Tags

In HTML, a tag is used for creating an element. Tags are used to delimit the start and end of elements in the markup. Normal html elements have a start tag to indicate where they begin, and an end tag to indicate where they end. The start and end tags of certain normal elements can be omitted. An HTML tag is composed of the name of the element, surrounded by angle brackets (<>). An end tag also has a slash after the opening angle bracket, to distinguish it from the start tag.

E.g.

```
<p> paragraph </p>
```

You may have come across the term HTML elements in this module or some other document. You may have also seen HTML tags and HTML elements used interchangeably. But there is a difference between the two. HTML tags are just the opening and closing entities of an element. Whereas HTML Elements contain the opening tag, closing tag and content (content is optional for void elements).
There are six different kinds of HTML elements. Void elements, template elements, raw text elements, Escapable raw text elements, Foreign elements and Normal Elements

Q. Read and write a short essay about the different types of elements

### *2.2.1 Basic HTML Tags (Html, Head, Body, Title)*

**<html> … </html>**

- The root element of an html document;

- All other elements are contained in this element
- The HTML element delimits the beginning and the end of an HTML document
- Both start and end tags may be omitted (HTML5)

**<head>...</head>**

- Contains information which is not displayed in the browser display area and may contain other tags in it
- Both the start and end tags may be omitted and inferred from child elements (HTML5)
- The head element may contain other elements such as base, link, meta, style, script, title

**<title></title>**

- Web browsers usually display it in a window's title bar when the window is open, and (where applicable) in the task bar when the window is minimized.
- Sets the title of the web page to be displayed in the browser's title bar and is found within the <head> tag.

**<body></body>**

-The body tag contains the visible part of the web page and is displayed in the display area of the browser. Just like the html and head tag, both the start and end tags of the body tag may be omitted and inferred from child elements (HTML5). It contains several other tags and content in it.

**Attributes:**

HTML Attributes are special words used to control the behavior of an element. HTML Attributes are a modifier of an HTML element type. An attribute is added to an HTML start tag before the closing angle bracket separated by space from the elements name. We can include more than one attribute in an element, but one has to be space separated from the other.

Some of the common attributes used in the body element

- ✓ bgcolor="color"
- ✓ background="img url"
- ✓ text="text color"
- ✓ link="link color"
- ✓ alink="active link color"
- ✓ vlink="visited link color"……

adding attributes to body

```
<body bgcolor="red">
```

### *2.2.2 Meta tag*

Meta elements are tags used in HTML document to provide structured metadata about a web page. This element represents metadata that cannot be represented by other HTML meta-related elements, like <base>, <link>, <script>, <style> or <title>. Meta elements are part of a web pages head section. There can be more than one meta element with different attributes on the same web page. In general, meta elements convey hidden information about the document.

E.g.

```
<meta charset="utf-8">
```

### *2.2.3 HTML Comments*

Just like other markup or programming languages you can add comments to an HTML document to add additional information which will not be displayed to the browser or preventing an unwanted piece of code from being executed and displayed on the browser. HTML comments are not displayed in the browser, but they can help document your html source code.

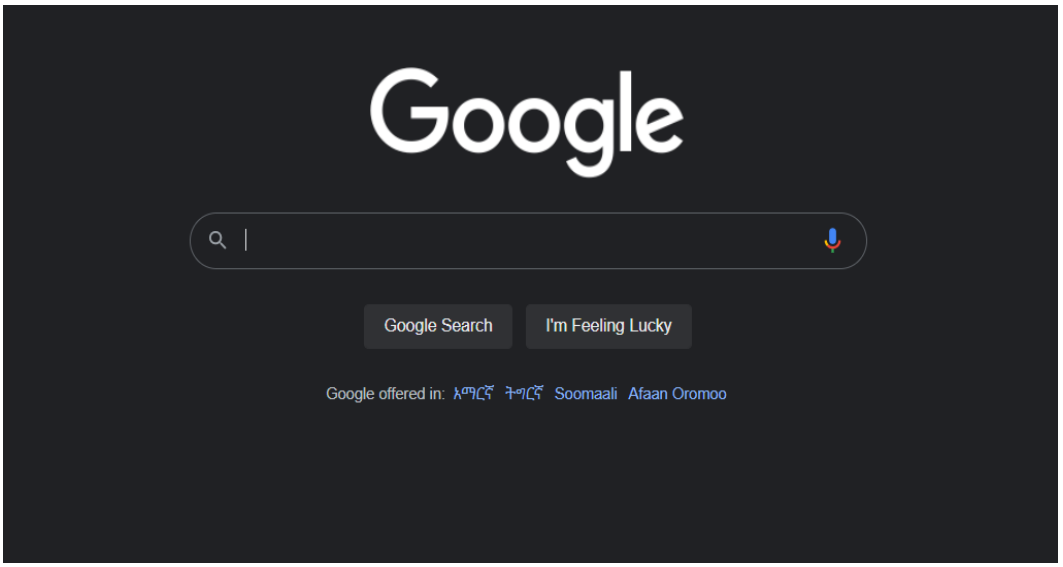
You can add comments to your html document using the following syntax:

```
<!-- Write your comments here -->
```

### *2.2.4 HTML Link*

Hyperlinks are what makes the web a web. Hyperlinks have been part of the web since the beginning. They allow us to link documents to other documents or resources, links to specific parts of documents, or make apps available at a web address. Almost any web content can be converted into a link so that when clicked goes to another web address (URL) or another part of the document. In this course you will see creating links to an external document as well as linking parts of the document.

Take a look at the following screenshot of Google's homepage




In the above screenshot, the Google homepage contains different links. Each link when clicked will take you to a different page. If you for example click on the link with a text Soomaali it will take you to a google page where the language is Somali

In HTML, a link is created by wrapping the text or other content inside an <a> tag (anchor tag) and using the href attribute, also known as Hypertext reference, or target that contains the web address.

Here is an example of a link to the google website

```
<a href="https://www.google.com">Google</a>.
```

You can also add supporting information to a link using the title attribute.

Another attribute that we can use on the anchor tag is the target attribute. The target attribute can be applied to a link to specify how the new page is opened.

Active Learning:

- First create a simple web page containing the basic html tags.
- Then add one or more paragraphs
- Change some of the content into links
- Preferably add title to each of the links you created

## *2.2.5 HTML Text Formatting tags*

In this section we will see the way HTML can be used to structure a page of text by adding headings, paragraphs, emphasizing words, creating lists and more

**Headings** –

Headings are predefined formats for text presentation and there are six heading formats defined in HTML:

<h1> up to <h6>
<h1> the largest font size
<h6> the smallest font size
Format :<h1>…</h1>
E.g., <h2>a text in heading two</h2>

One thing you should bear in mind is html out of the box comes with no styling. This has to do with the purpose of html, which is used to structure content in the browser. The styles you see when you apply different tags like headings, lists, links etc. comes by default with the browser and it is called user agent stylesheet. You can check this by going to the developer's tool of your preferred browser and selecting the styles tab. You will learn more about the different types of style sheets including user agent style sheet and how to style your web page in the next chapter

**Bold** - makes the text enclosed appear in bold.
Format: … or …E.g., a text in bold

**Italics -** makes a text appear in italics
Format: … or …. E.g., a text in italics
**Underline** - Makes a text appear underlined.
Format: <u>…</u> E.g., <u>underlined text</u>

Q. Write simple html code by using the html tags you have seen above?
**Paragraph: definition of paragraph**
Format: …
E.g., this is a paragraph of text. it has a new line before and after it.
The browser inserts a new line before and after the text in the paragraph tag.
**Attribute:**
align="alignment" {left, right, center, justify}

- **line break** - inserts a new line.

Format: <br>. E.g., Line one <br> line two <br> line three <br> line four

- **horizontal rule** - inserts a horizontal line

Format: <hr>

**Attributes:**

- ✓ width="width" {absolute: in pixels or relative: in %}
- ✓ noshade
- ✓ color="color" {browser dependent}

E.g. <hr width="75%" noshade color="#FF0000">

- **sub/sup** - define either a subscript or a superscript

Format: <sub>…</sub> ; <sup>…</sup>

E.g.

X<sub>1</sub><sup>2</sup> + 2X<sub>3</sub>

Q. Add paragraph by using paragraph tags in the above question?

## *2.2.6 HTML image inserting tag*

In the beginning, the web was just a text and it was boring. Fortunately, the ability to add images inside web pages was added.

### *2.2.6.1 IMG tag and its attribute*

To put an image to a web page, we use the <img> element. This is a void element (meaning that it has no text content or closing tag.) that requires a minimum of at least one attribute to be useful src.

The src attribute contains a path pointing to the image you want to embed in the page, which can be a relative or absolute URL, in the same way as href attribute values in <a> elements

```
<img src="dinosaur.jpg">
```

You can also embed an image using its absolute URL from

```
<img src="https://www.example.com/images/dinosaur.jpg">
```

Another attribute we frequently see in an img tag is alt. Its value is textual description of the image, which is displayed in a situation where the image cannot be seen or takes a long time to render.

```
<img src="images/dinosaur.jpg" alt="head and torso of a
dinosaur">
```

You can also use width and height attributes to specify the width and height of the image. Do not apply a width and height that is out of proportion, this will result in the image being very fuzzy. Which is not what your user desire.
You can also add title attribute to add supporting information to the user. The information inside title is display when the mouse is over the element like a tooltip

Supported image formats are: gif, jpg/jpeg, png

Q. Why would you need an alt text?

### *2.2.6.2 Inserting Image Map*

An image map is a list of coordinates relating to a specific image, created in order to hyperlink areas of the image to different destinations (as opposed to a normal image link in which the entire area of the image is links to a single destinations). The map HTML element is used with the area element to define an image map (a clickable link area).

The name attribute gives the map a name so that it can be referenced. The attribute must be present and must have a non-empty value with no-space characters

```
<map name="primary">
  <area shape="circle" coords="75,75,75" href="left.html">
  <area shape="circle" coords="275,75,75" href="right.html">
</map>
<img usemap="#primary" src="sample-image.jpg" alt="350 x 150 pic">
```

## *2.2.7 HTML Table*

A very common task in html is structuring tabular data. HTML has a number of elements and attributes just for this purpose. This section will explain how to create tables using the different elements provided by html.

A table is a structured data made up of rows and columns. To create table in html use the fowling elements table, tr, th, td, and captions

tr stands for table row, th stands for table header and td stands for table data

Following is an example of a simple table with two rows and one column, the first row contains the heading of the table the second row contains the data related to the header in the first row.

```
<table>
  <tr>
    <th>Name</th>
  </tr>
  <tr>
    <td>Abebe</td>
  </tr>
</table>
```

By default, HTML tables are border less. To add border to HTML table you can use the border attribute of the table elements

If you have a reason to merge columns or rows or the website you are building needs this functionality you can use the colspan and rowspan attribute to do so. These attributes are used to span two or more columns or rows respectively. Both colspan and rowspan takes unitless numbers that specifies the number of columns and rows you want to span

E.g.

```
<table>
   <caption align="center" valign="bottom">table 1.0</caption>
     <tr>
        <th>Column 1</th>
        <th>Column 2</th>
     </tr>
     <tr>
         <td>Cell 1</td>
         <td>Cell 2</td>
     </tr>
     <tr>
        <td>Cell 3</td>
        <td>Cell 4</td>
     </tr>
</table>
```

Q. Read in detail about the different table attributes, row attributes and data/heading attributes?

## *2.2.8 Ordered and Unordered List in HTML*

**Lists**

I. **Unordered Lists ( ul )** - define bulleted lists

Format:

```
<ul>
    <li>…</li>
    <li>…</li>…
</ul>
```

**Attribute:**

type="bullet type" { disc, circle, square}

E.g.

```
<ul type="square">
     <li>book</li>
     <li>marker</li>
     <li>chalk</li>
</ul>
```

II. **Ordered Lists (ol) -** element represents an ordered list of items — typically rendered as a numbered list.

Format:

```
<ol>
      <li>…</li>
      <li>…</li>…
</ol>
```

**Attribute:** type="number type" {1, i, I, a, A}

E.g. <ol type="i"> <li>book</li><li>marker</li><li>chalk</li></ol>

other attributes:

reversed: items will be numbered from high to low

start: an integer to start counting from for the list items

**III. Definition Lists (dl) -** define a list of term-description pairs

Format : <dl>

<dt>…</dt>
<dd>…</dd>

<dt>…</dt>
<dd>…</dd>
</dl>

E.g. <dl>
<dt>book</dt><dd>something that we read …</dd>
<dt>marker</dt><dd>something we write with …</dd>

</dl>

Q. Write a list of students by using order list and change the type to roman numbers

### *2.2.9 HTML Frames*

HTML frames allows us to present documents in multiple views, which may be independent windows or sub-windows. Multiple views offer designers a way to keep certain information visible, while other views are scrolled or replaced.

An HTML document that describes frame layout has a different markup than HTML document without frames. A standard document has one HEAD section and one BODY. A frameset document has a HEAD and a FRAMESET in place of BODY.

#### *2.2.9.1 Frame Set*

The FRAMESET section of a document specifies the layout of views in the main user agent window. In addition, the FRAMESET section can contain a NOFRAMES element to provide alternate content for user agents that do not support frames or are configured not to display frames.

Attributes:

Cols -> specifies how many columns are contained in the frameset and the size of each column. You can specify the width of each column in one of four ways: Pixels, Percentage, wildcard symbol (takes reminder of the window) and relative width of the browser window

Rows -> This attribute works just like the cols attribute and takes the same values, but it is used to specify the rows in the frameset

Border -> specifies the width of the border

#### *2.2.9.2 Internal Frame*

The Frame tag define the particular area within an HTML file where another HTML web page can be displayed. A <frame> tag is used with in a <frameset>, and it divides a webpage into multiple sections or frames, and each frame can contain different web pages.

Syntax:

```
< frame src = "URL" >
```

Example:

```
<!DOCTYPE html>
<html>
<head>
   <title>Frame tag</title>
</head>
 <frameset cols="25%,50%,25%">
   <frame src="frame1.html" >
   <frame src="frame2.html">
   <frame src="frame3.html">
 </frameset>
</html>
```

### *2.2.10 HTML Form and Form Controls*

The <form> HTML element represents a document section containing interactive controls for submitting information. An HTML form is a section of a document containing normal content, markup, special elements called controls (checkboxes, radio buttons, menus, etc.) and labels on those controls. HTML Forms are required, when we want to collect data from a user. Using forms, we can build a website with user registration. We take the information collected from the user and then post it to the backend application such as php.

Users interact with forms through named controls. There are various from controls available like text fields, text area, check box, drop-down menus etc.

Form attributes:

- action → Backend script ready to process the data collected
- method → method used to upload Data. The most frequently used
- target → Specify the target window or frame where the result of the script will be displayed. It takes values like _blank, _self, _parent etc.
- enctype → tells how the browser encodes data. Possible values
  1. application/x-www-form-urlencoded → most common and used from simple scenarios
  2. mutlipart/form-data → used to upload binary data

Form controls

1. Text inputs create using the <input> tag, an input control can take many forms using the type attribute. Possible values for type attribute
   - Text – most common used for text inputs
   - Password – for password fields
   - Checkbox
   - Radio

   Other attributes for <input> tag

   - Name, value, size, maxlength
2. Text area – created using the <textarea> tag
3. Select box → also called drop down box which provides option to list down various options in the form of drop-down list, from where a user can select one or more options. We can use <select> tag to create a select box and <option> tag nested inside the <select> tag to provide the list items in the drop down

```
<select name = "dropdown">
    <option value = "Web" selected>Web Programming</option>
    <option value = "Node">Node.js</option>
</select>
```

4. File upload box → this is created using the <input> tag but the type attribute has to be set to **file**
5. Button controls → we can control various types of buttons in HTML. We can create a clickable button by using the <input> tag and setting the type attribute to one of the following values
   - Submit, reset, button, image

### *2.2.11 Inserting Multimedia in HTML*

Sometimes we want to add or embed images or videos to our site. An image or a video will make a page appealing. We can embed an image using the <embed> tag and for those browsers that doesn't support the <embed> tag we can use the <noembed> tag. The most important attribute for the embed tag is src (it will provide where the image or video file is located)

```
<embed src = "video" width = "100%" height = "60" >
        <noembed>
              <img src = "image" alt = "Alternative Media" >
        </noembed>
</embed>
```

Other attributes

Align, autostart, loop, playcount,width, height

#### *2.2.11.1 Embed vs Video and Audio tags*

The <embed> tag allows us to embed a video, audio and gif files to our website, but the browser doesn't know what type of media file we are embedding a head of time. The <video> element allows you to embed a video very easily. A really simple example looks like this:

```
<video src="video location" controls>
  <p>fallback text </p>
</video
```

The src attribute specify the location of the video file. Contorls attribute adds a feature for the user to control the video or audio.
The <audio> element works just like the <video> element, with a few small differences.

```
<audio controls>
        <source src="audio source" type="type of audio like audio/mp3">
        <source src="audio source" type="type of audio like ">
        <p> if the browser doesn't support this feature it will fallback to this text </p>
</audio>
```

### *2.2.12 HTML Graphics*

Web graphics are visual representations used on a Web site to enhance or enable the representation of an idea or feeling, in order to reach the Web site user. Graphics may entertain, educate, or emotionally impact the user, and are crucial to strength of branding, clarity of illustration, and ease of use for interfaces.

Examples of graphics include maps, photographs, designs and patterns, family trees, diagrams and architectural or engineering blueprints.

Graphics are used for everything from enhancing the appearance of Web pages to serving as the presentation and user interaction layer for full-fledged Web Applications.

### *2.2.12.1 HTML Canvas*

The Canvas API provides a means for drawing graphics via JavaScript and the HTML <canvas> element. Among other things, it can be used for animation, game graphics, data visualization, photo manipulation, and real-time video processing. Canvas was initially introduced by Apple. Later it was adopted by other browsers.

If you want to create a 2D or 3D scene on a web page, you need to start with an HTML <canvas> element. This element is used to define the area on the page into which the image will be drawn.

E.g.

```
<canvas width="320" height="240"></canvas>
```

This will create a canvas on the page with a size of 320 by 240 pixels. If the browser doesn't support <canvas> we can add a suitable fallback text in between the opening and closing tag. You will have to give the canvas a class name or id so that it will be easily referenced in JavaScript.

You will use the following methods after you get the reference of the canvas in JavaScript

- Getcontext
- fillStyle
- fillRect
- etc.

Q After you are introduced to JavaScript, draw a rectangle using the methods given above

### *2.2.12.2 HTML SVG*

SVG defines vector-based graphics in XML format. The HTML <svg> element is a container for SVG graphics. The svg element is a container that defines a new coordinate system and viewport. It is used as the outermost element of SVG documents, but it can also be used to embed an SVG fragment inside an SVG or HTML document. SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

E.g.

```
<svg viewBox="0 0 300 100" xmlns="http://www.w3.org/2000/svg"
        stroke="red" fill="grey">
  <circle cx="50" cy="50" r="40" />
```

```
  <circle cx="150" cy="50" r="4" />
</svg>
```

# *Chapter 3: Cascading Style Sheets (CSS)*

## 3.1 CSS Basics

In the previous chapter we have seen how to structure a web page using HTML. Now it is time to add presentation to our sites, to do that we will be using a technology called CSS. CSS (Cascading Style Sheets) is the code that styles web content. Using CSS you can apply styles to make web pages look exactly how you want. This works because CSS is connected to the DOM(Document Object Model). CSS is not a programming language. It's not a markup language either. CSS is a style sheet language. CSS is what you use to selectively style HTML elements.

```
p {
  color: red;
}
```

### *3.1.1 Introduction to CSS*

CSS stands for Cascading Style Sheet. It is a stylesheet language used to describe the presentation of a document written in HTML. Styles define how to display HTML elements and are normally stored in style sheets. HTML tags were originally designed to define the content of a document. They were supposed to say “this is a paragraph”, “this is a table” by using tags like <p>, <table>…. The layout of the document was supposed to be taken care of by the browser, without any formatting tags.

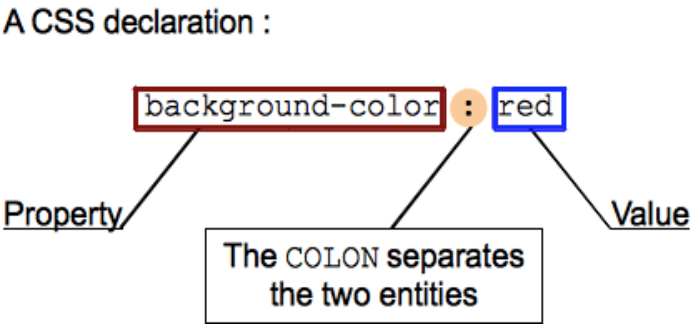
### *3.1.2 CSS Syntax (CSS Selectors and Declarations)*

The basic goal of the Cascading Stylesheet (CSS) language is to allow a browser engine to paint elements of the page with specific features, like colors, positioning, or decorations.
The building block of CSS syntax are:

- The **property** which is an identifier, that is a human-readable name, that defines which feature is considered.
- The **value** which describes how the feature must be handled by the engine. Each property has a set of valid values, defined by a formal grammar, as well as a semantic meaning, implemented by the browser engine.

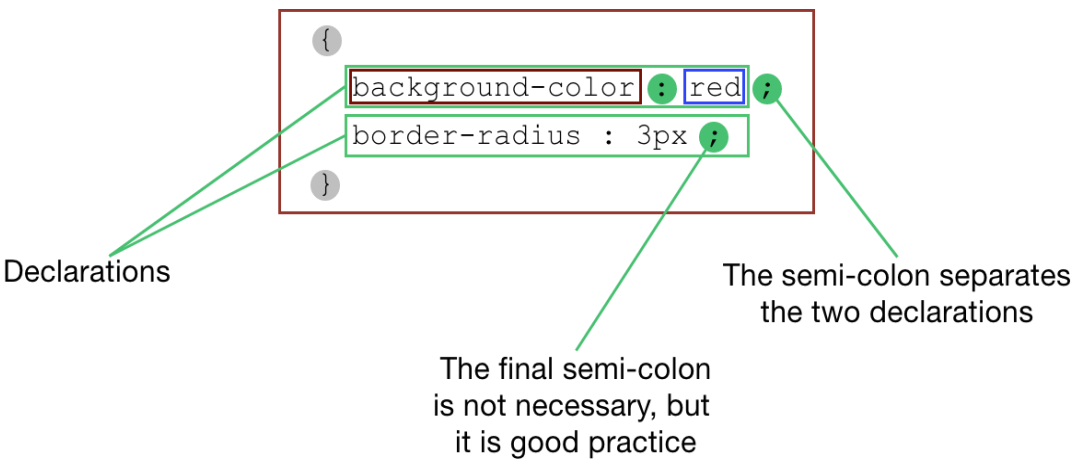
CSS Declarations:

Setting CSS properties to specific values is the core function of the CSS language. A property and value pair are called a **declaration**,




Declaration Block:

Declarations are grouped in blocks, that is in a structure delimited by an opening brace, '{' and a closing one, '}'



### *3.1.3 Attaching CSS with HTML (External , Embedded and Inline)*

Styles can be applied to documents in three distinct ways

1. **Inline stylesheet**

In HTML, style information can be specified for an individual element via the style attribute. The value of a style attribute is a *declaration block* without the curly braces.

E.g.

```
<p style="color: red; background:green;">Inline style</p>
```

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly when you need a style to be applied to a single occurrence of an element.

2. **Embedded stylesheet**

This type of stylesheet should be used when a single document has a unique style. You define internal styles in the head section by using the <style> tag.

e.g.

```
<head>
   <style>
        H1 {
             color:yellow;
            Text-align:left;
         }
      P{
          margin-left:20px;
        }
   </style>
</head>
```

3. **External style sheets**

An external style sheet is used when the style is applied to many pages. With an external style sheet, you can change the look of an entire website by changing one file (the style sheet). Each page should be linked to the style sheet using the <link> tag inside the head section.

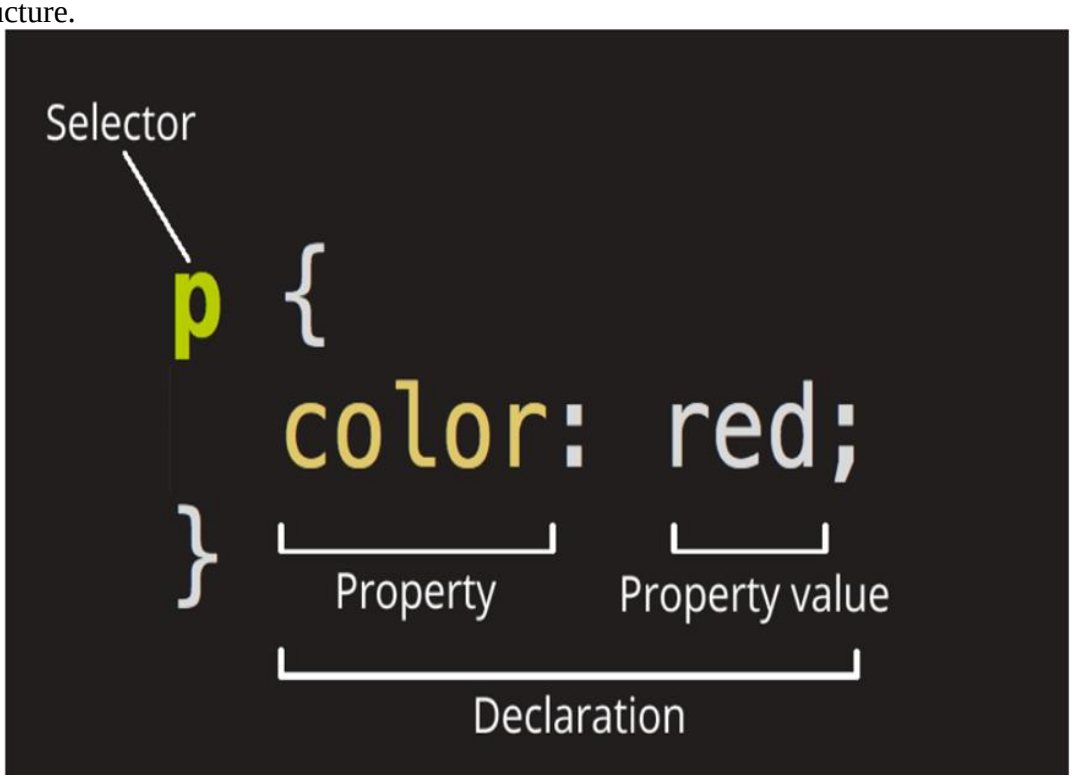
E.g.

```
<link rel="stylesheet" href="style.css">
```

The **rel** attributes names the relationship of the linked document to the current document. The **href** attribute specifies the URL of the linked document. The URL can be relative or absolute

## 3.2 Style Sheet Rules

A selector group and an associated declarations block, together, are called a ruleset, or often a rule. A Cascading Style Sheet (CSS) rule is a statement that defines the style of one or more elements in your web page. These rules follow a specific structure.




The whole structure is called a ruleset. Note the names of the individual parts:

**Selector** –

This is the HTML element name at the start of the ruleset. It defines the element(s) to be styled.

**Declaration**

This is a single rule like color: red. It specifies which of the element's properties you want to style.

**Property** – These are ways in which you can style an HTML element. In CSS, you choose which properties you want to affect in the rule.

**Value** – To the right of the property—after the colon—there is the property value. This chooses one out of many possible appearances for a given property.

## Selecting multiple elements

You can also select multiple elements and apply a single ruleset to all of them. Separate multiple selectors by commas. Example

```
p, li, h1 {
  color: red;
}
```

## Different types of selectors

There are many different types of selectors. The example above use element selectors, which select all elements of a given type

1. **Type selector**

   selects all HTML elements of the specified type**.** Type selectors are very common in css. Let's say you have the following HTML code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CSS Selectors</title>
</head>
    <body>
        <p>Type Selector</p>
    </body>
</html>
```

   if you want to change the text color of the paragraph you can use the name of the element p and set the color property to the color of your choice

```
p {
    color: red;
}
```

2. **Class selector** -> The element(s) on the page with the specified class. Multiple instances of the same class can appear on a page. We use the . (dot) symbol and the name of the class to use it as a selector in css
3. **Id selector** -> The element on the page with the specified ID. On a given HTML page, each id value should be unique. We use # (hashtag ) symbol and the name of the id as a selector
4. **Attribute Selector** -> The element(s) on the page with the specified attribute.
5. **Pseudo-class selector** -> The specified element(s), but only when in the specified state. (For example, when a cursor hovers over a link.)

Q. Read in detail the above ways of selectors it's definition, advantage, syntax and examples for each?

### *3.2.1 Style Inheritance*

In CSS, **inheritance** controls what happens when no value is specified for a property on an element. CSS uses the document tree for inheritance, in which an element applied to an element is inherited by its descendants.

CSS properties can be categorized in two types:

- **Inherited properties** - which by default are set to the computed value of the parent element
- **Non-inherited properties** -> which by default are set to initial value of the property. Some examples of non-inherited properties are padding, border, margin and background

When no value for an inherited property has been specified on an element, the element gets the computed value of that property on its parent element. Only the root element of the document gets the initial value given in the property's summary.

Assignment: list inherited and non inherited properties of CSS

### *3.2.2 Style Rules Precedence*

The style rule that is applied to an element may not be the one you intended. This has haunted many junior developers. At some point, you will be working on a project and you will find that the CSS you thought should be applied to an element is not working. This happens because of conflicting rules. Rules that are applied to an element is affected by the order of the rule that appeared in your stylesheet file or the order of stylesheets you have linked to your HTML file.

When the browser needs to resolve what styles to apply to a given HTML element, it uses a set of CSS precedence rules. You remember the first letter CSS which stands for Cascade, this is what the browser uses to resolve conflicting rules in your stylesheet. The other determining factor in selecting CSS rules is specificity. **Specificity** is the means by which browsers decide which CSS property values are the most relevant to an element and, therefore, will be applied.

Following are CSS precedence rules

1. !important after CSS properties.
2. Specificity of CSS rule selectors.
3. Sequence of declaration.

!important

If you need a certain CSS property to take precedence over all other CSS rules setting the same CSS property for the same HTML elements, you can add the instruction !important after the CSS property when you declare it. The !important instruction has the highest precedence of all precedence factors.

```
<style>
    div {
        font-size: 16px !important;
    }
    .text-size {
        font-size: 18px;
    }
  </style>

  <div class="text-size">
    !important
  </div>
```

The code above is trying to change the font-size of an element using different selectors. Normally, since class selectors has higher specificity than a CSS rule with an

element selector, it would take precedence over the first rule. But since the element selector has the word !important, it will take precedence over the class selector.

## Specificity

Specificity is how the browser decides which rule applies if multiple rules have different selectors, but could still apply to the same element. It is basically a measure of how specific a selector's selection will be. Specificity is a weight that is applied to a given CSS declaration, determined by the number of each selector type in the matching selector.

### Selector types

The following list of selector types increases by specificity:

1. Type Selectors (e.g., p) and pseudo-elements(e.g., ::before).
2. Class Selectors (e.g., .examples), attributes selectors (e.g., [type=”radio”] and pseudo-classes(e.g., :hover)
3. ID Selectors (e.g., #example)

Universal Selector (*), combinators (+, >, ~, || ) and negation pseudo-class (:not()) have no effect on specificity.

## The cascade

Stylesheets cascade — at a very simple level, this means that the order of CSS rules matters; when two rules apply that have equal specificity, the one that comes last in the CSS is the one that will be used.

```
p {
    color: brown;
}
p {
    color: green;
}
```

We have two rules for the same element p. The paragraph will be green as it comes last

## 3.3 Style Properties

### *3.3.1 Foreground and Background Properties*

The background shorthand CSS property sets all background style properties at once, such as color, image, origin and size, or repeat method. You have also specific background properties that you can apply like background-color, background-image etc.

To set the background-color of an element and provide the value of the color in different form.

1. Keyword values -> red, blue
2. Hexadecimal value -> #ffffff, #bbb
3. RGB value -> rgb(255,255,255), rgba(0,0,0,0.3) //transparent
4. HSL value -> hsl(50,33%,25%)
5. Special keyword values -> currentcolor, transparent
6. Global values -> inherit, initial, revert, unset

You can also set an image as a background color using the background-image property

Syntax:

background-image: url('anyname.jpg')

The color property sets the color of the foreground content of the selected elements

### *3.3.2 Font and Text Properties*

The font CSS shorthand property sets all the different properties of an element's font. Alternatively, it sets an element's font to a system font. The property is a shorthand for the following properties

- font-family
- font-size
- font-stretch
- font-style
- font-variant
- font-weight
- line-height

If font is specified as a shorthand for several font-related properties, then:

- it must include values for:
  <font-size>
  <font-family>

Example:

font: italic 1.2em "Fira Sans", serif;

## Text Properties:

Text transform: Allows you to set your font to be transformed. Values include:

- none: Prevents any transformation.
- uppercase: Transforms all text to capitals.
- lowercase: Transforms all text to lower case.
- capitalize: Transforms all words to have the first letter capitalized.

Text-decoration: Sets/unsets text decorations on fonts (you'll mainly use this to unset the default underline on links when styling them.) Available values are:

- none: Unsets any text decorations already present.
- underline: Underlines the text.
- overline: Gives the text an overline.
- line-through: Puts a strikethrough over the text.

The **text-align** property is used to control how text is aligned within its containing content box. Values

- left: Left-justifies the text.
- right: Right-justifies the text.
- center: Centers the text.
- justify: Makes the text spread out, varying the gaps in between the words so that all lines of text are the same width.

The **letter-spacing** and **word-spacing** properties allow you to set the spacing between letters and words in your text.

## Lists

In the previous chapter we have seen different types of lists. We have the two common types of lists => order and unorder lists. In this section we will see how we will go about styling lists.

Lists behave like any other text for the most part, but there are some CSS properties specific to lists that you need to know about.

To start with let's see the default styling of lists, which will surprise you when designing lists. Here are some of the common default list stylings

- ul and ol element have a default top and bottom margin of 16px(1em) and a padding left of 40px(2.5em)
- dl element has a top and bottom margin of 16px(1em), but no padding set
- dd elements have a margin-left of 40px(2.5em)

List specific styles

- list-style-types => Sets the type of bullets to use for the list, for example, square or circle bullets for an unordered list, or numbers, letters, or roman numerals for an ordered list.

- list-style-position => Sets whether the bullets, at the start of each item, appear inside or outside the lists.
- list-style-image => Allows you to use a custom image for the bullet, rather than a simple square or circle.

We can also use the short hand list-style property for the above properties. The values can be listed in any order, and you can use one, two, or all three (the default values used for the properties that are not included are disc, none, and outside).

### *3.3.3 CSS Box Model*

Everything in CSS has a box around it. In CSS we broadly have two types of boxes --- block boxes and inline boxes. An inline box has the following properties

- The box will not break onto a new line.
- The width and height properties will not apply.
- Vertical padding, margins, and borders will apply but will not cause other inline boxes to move away from the box.
- Horizontal padding, margins, and borders will apply and will cause other inline boxes to move away from the box

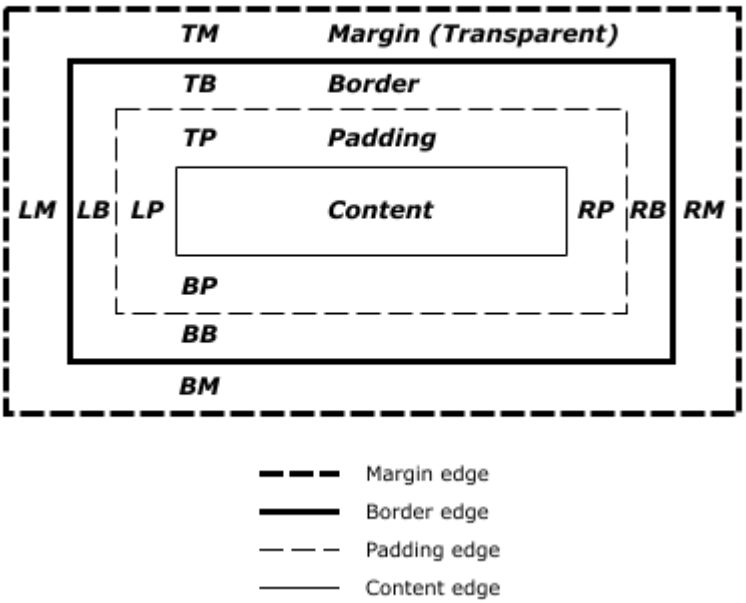
Whereas a block box has the following properties:

- The box will break onto a new line.
- The box will extend in the inline direction to fill the space available in its container. In most cases this means that the box will become as wide as its container, filling up 100% of the space available.
- The width and height properties are respected.
- Padding, margin and border will cause other elements to be pushed away from the box

The Box Model defines how the different parts of a box — margin, border, padding, and content — work together to create a box that you can see on a page.

Parts of a box

- Content box: The area where your content is displayed, which can be sized using properties like width and height.
- Padding box: The padding sits around the content as white space; its size can be controlled using padding and related properties.
- Border box: The border box wraps the content and any padding. Its size and style can be controlled using border and related properties.
- Margin box: The margin is the outermost layer, wrapping the content, padding, and border as whitespace between this box and other elements. Its size can be controlled using margin and related properties.




To set the padding of an element, we can use the padding short hand property. The padding property will set the four sides of the element padding-top, padding-right, padding-bottom and padding-left.

If there is only one component value, it applies to all sides. If there are two values, the top and bottom paddings are set to the first value and the right and left paddings are set to the second. If there are three values, the top is set to the first value, the left and right are set to the second, and the bottom is set to the third. If there are four values, they apply to the top, right, bottom, and left, respectively.

```
body { padding: 2em }        /* all padding set to 2em */
body { padding: 1em 2em }    /* top & bottom = 1em, right & left = 2em */
body { padding: 1em 2em 3em } /* top=1em, right=2em, bottom=3em, left=2em */
```

Q discuss about margin collapsing?

Display

The display CSS property sets whether an element is treated as a block or inline element and the layout used for its children, such as flow layout, grid or flex. Formally, the display property sets an element's inner and outer display types. The

outer type sets an element's participation in flow layout; the inner type sets the layout of children.

You can set the display property of an element using the following syntax:

display: value

possible values

- block, inline, inline-block, flex, inline-flex, grid, inline-grid, etc

### *3.3.4 Table Styling Properties*

Styling an HTML table isn't the most glamorous job in the world, but sometimes we all have to do it. The default HTML table is not that fancy, it doesn't even have a border separating the different rows and columns. In this section of the module, we will look at different CSS properties to style a table.

- **table-layout** => Normally, table columns tend to be sized according to how much content they contain, which produces some strange results. A **table-layout** value of fixed is generally a good idea to set on your table, as it makes the table behave a bit more predictably by default. With table-layout: fixed, you can size your columns according to the width of their headings, and then deal with their content as appropriate.
- **border-collapse =>** By default, when you set borders on table elements, they will all have spacing between them. To prevent that from happening we can use the border-collapse CSS property and set its value to collapse.
- We can set borders to using the **border** property, we can also apply the following properties depending on out preference => padding, font, text-align, text-shadow, color, background-color and others

### *3.3.5 More On Styling List (Creating Navigation bars)*

In this section we will see how to create navigation bar using lists. A navigation bar is a list links. To create a navigation bar we will use an ordered list and list items.

```
<ul>
   <li><a href="#">Home</a></li>
   <li><a href="#">News</a></li>
   <li><a href="#">Contact</a></li>
    <li><a href="#">About</a></li>
</ul>
```

If you run the above code and insepect it in your browser, in the style tab under the source tab of the browser's developer's tool you we will see the default styling of <ul>

tag. Lists by default have a list-style-type, margin and padding properties set by the user agent style sheet. Since we don't want the default styling lets reset it

Here is the code to do that

```
ul {
        list-style-type: none;
        margin: 0;
        padding: 0;
    }
```

Next set the display property of ul to flex. This will align the list items horizontally. If you preview the code in your browser the list items are cluttered together. This is not appealing and it doesn't look like the navigation bar we have come to know

In its current state the navigation bar looks like this

HomeNewsContactAbout

Now let's make it look like a navigation bar by adding a background color to the list like this

```
background-color: lightblue;
```

In web development spacing is everything. Look at the picture above there is no spacing between the list items. This made the items look ugly. You can solve this by adding padding and margin to the list items

```
li {
        padding: 10px 20px;
    }
```

This will set the top and bottom padding to 10px and the left and right padding to 20px. If you are confused about passing values to padding and margin, revisit CSS Box Model. Here is the finished styling of <li> tag, you can tweak the values to see how it affects the items

```
li {
      padding: 10px 20px;
      background-color: maroon;
      border-radius: 0.3rem;
      margin-left: 20px;
    }
```

Next lets style the anchor tags

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

Text decoration none will remove the underline.

You can play around with this code and add new styles like hover effect in your lab session.

### *3.3.6 Layout and Positioning Properties*

CSS page layout techniques allow us to take elements contained in a web page and control where they're positioned relative to the following factors: their default position in normal layout flow, the other elements around them, their parent container, and the main viewport/window.

There are different page layout techniques out there, the main layouts techniques are listed below. This module will cover Normal flow, Flexbox, and Grid

- Normal flow
- The display property
- Flexbox
- Grid
- Floats
- Positioning
- Table layout
- Multiple-column layout

Each techniques have its uses, advantages and disadvantage. You can use any one of the above techniques depending on what your website needs.

Normal Flow

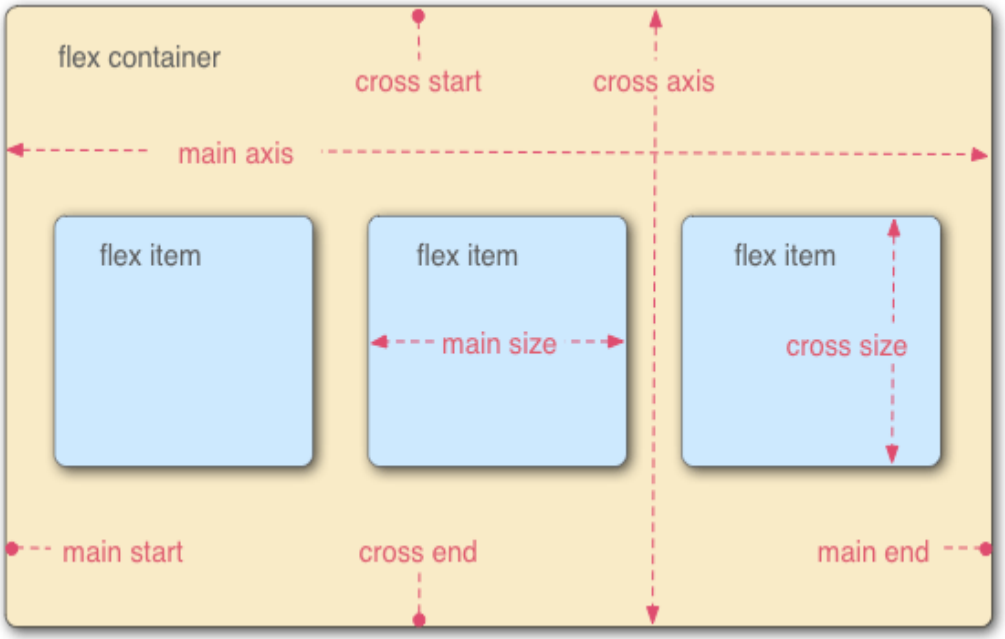
Normal flow is how the browser lays out HTML pages by default when you do nothing to control page layout. In this layout HTML elements are displayed in the exact order in which it appears.

## Flexbox

Flexbox is a one-dimensional layout method for arranging items in rows or columns. Items flex (expand) to fill additional space or shrink to fit into smaller spaces. There are two kinds of flex elements: the flex container and the flex items that are placed within the container. All the direct children of the flex container element are flex items.

To start with, we need to select which elements are to be laid out as flexible boxes. To do this, we set a special value of display on the parent element of the elements you want to affect.

When elements are laid out as flex items, they are laid out along two axes:




- The **main axis** is the axis running in the direction the flex items are laid out in
- The **cross axis** is the axis running perpendicular to the direction the flex items are laid out in.
- The parent element that has display: flex set on it is called the **flex container**.
- The items laid out as flexible boxes inside the flex container are called **flex items**

We can use flex-direction to specify which directions the main axis runs. The default is set to row. If we want to change the flex-items to appear in a column layout we should set the flex-direction to column.

## Grid

CSS Grid Layout excels at dividing a page into major regions or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives. Using Grid we can align items into rows and columns.

To use grid in your site, you have to set the wrapper(container) of the items display property to **grid or inline-grid**. Next we define the number of columns and rows using **grid-template-columns, grid-template-rows.** The values for the two propeties represent track-size and line-name

- track-size: can be a length, a percentage, or a fraction of the free space in the grid (using the fr unit)
- line-name: an arbitrary name of your choosing

Properties of the children

- grid-column → value start-line/end-line. It is a short hand for grid-column-start and grid-column-end
- grid-row → value start-line/end-line. It is a short hand for grid-row-start and grid-row-end

Q Read about grid-template-areas and grid-areas?

## Positioning

Positioning allows us to produce interesting results by overriding normal document flow. There are a number of different types of positioning that you can put into effect on HTML elements. To make a specific type of positioning active on an element, we use the **position** property.

## Static Positioning:

Static positioning is the default that every element gets. It just means "put the element into its normal position in the document flow — nothing special to see here."

```
div  {
     position: static;
}
```

## Relative Positioning

This is very similar to static positioning, except that once the positioned element has taken its place in the normal flow, you can then modify its final position, including making it overlap other elements on the page. **top, bottom, left, and right** are used alongside position to specify exactly where to move the positioned element to.

```
div {
     position: relative;
     top: 50%;
     left: 50%;
}
```

## Absolute Positioning

An absolutely positioned element no longer exists in the normal document flow. Instead, it sits on its own layer separate from everything else. This is very useful: it means that we can create isolated UI features that don't interfere with the layout of

other elements on the page. For example, popup information boxes, control menus, rollover panels, UI features that can be dragged and dropped anywhere on the page, and so on. In order for absolute positioning property to work as we want it, we have to set its container to position relative. If we don't do that it uses the documents body as its relative.

```
div {
     position: absolute;
     top: 0;
     left: 50%;
}
```

## 3.4 CSS Measuring Units

There are various numeric value types that you might find yourself using in CSS. The following are all classed as numeric:

Lengths

There are two types of lengths used in CSS — relative and absolute. It's important to know the difference in order to understand how big things will become.

Absolute units

Absolute length units are not relative to anything else, and are generally considered to always be the same size → cm,mm,in,pt,px

Relative units

Relative length units are relative to something else, perhaps the size of the parent element's font, or the size of the viewport.

- Em → font-size of the parent
- Rem → font size of the root element
- Vw → 1% viewport width
- Vh → 1% viewport height

Reading Assignment: Read and write about viewport

# *Chapter 4: Client-Side Scripting (JavaScript)*

## 4.1 Introduction to JavaScript

JavaScript (JS) is a lightweight, interpreted or compiled programming language. It is a single threaded, dynamic language, supporting object oriented, functional programming styles. Do not confuse JavaScript with the Java Programming language. The two-programming language have very different syntax, semantics, and use.

## 4.2 JavaScript Basic

JavaScript is a scripting or programming language that allows us to implement complex features on web pages. JavaScript will enable us to display timely content updates, interactive maps, 2D/3D animated graphics etc. JavaScript is one of the three layers of web technologies which lets to add functionality to a web page. The three layers of web technologies are stacked together nicely, HTML will let's structure the webpage, CSS deals with the presentation of the web page and JavaScript the functionality.'

What can JavaScript really do? It can

- Stores values inside variables
- Perform operations on pieces of texts
- Run a code in response to events
- etc.

We can use JavaScript in coordination with Application programming interfaces (APIs). APIs provide us extra information. They are ready made code building blocks that are used to simplify programming in JavaScript. There are two types of APIs

- Third party API → are not built into the browser. We have to grab the code from the web. Examples: Twitter API, Google maps API etc.
- Browser API → are built into the browser. Examples DOM (Document object model) which is covered later in this chapter, Geolocation API, Canvas and WebGL, Audio and Video elements

Q Discuss the difference between interpretated and compiled language

### *4.2.1 JavaScript Syntax*

JavaScript is added to an HTML document in a similar fashion as that of CSS. The main difference is we use <script> tag instead of <link> tag.

Here is the basic JavaScript syntax

```
<script>

  // JavaScript code goes here

</script>
```

- Internal JavaScript is one way of including JavaScript using the script tag inside our HTML document. We can use the above syntax to include it anywhere in our document. The best place to include a JavaScript file is before the end of body tag </body>.
- ```
  <script>
    console.log(“Hello world”)
  </script>
  ```

- We can also include JavaScript as an external file. We can use a similar syntax like the internal and add the src attribute. Here is how

  ```
  <script src="script.js" defer></script>
  ```

  The defer attribute (Boolean attribute) is set to indicate to a browser that the script is meant to be executed after the document has been parsed, but before firing DOMContentLoaded.
  Scripts with the defer attribute will prevent the DOMContentLoaded event from firing until the script has loaded and finished evaluating. This attribute must not be used if the src attribute is absent (i.e., for inline scripts), in this case it would have no effect.

  We can also use the async attribute. This is a Boolean attribute indicating that the browser should, if possible, load the script asynchronously. This attribute must not be used if the src attribute is absent (i.e., for inline scripts). If it is included in this case, it will have no effect. Browsers usually assume the worst-case scenario and load scripts synchronously, (i.e., async="false") during HTML parsing.

As a rule, only the simplest scripts are put into HTML. More complex ones reside in separate files.
The benefit of a separate file is that the browser will download it and store it in its cache.
One other important thing to know is if src is set, the script content is ignored. A single <script> tag can’t have both the src attribute and code inside.
  ```

- We can also come across an inline JavaScript like this

```
<button onclick="createParagraph()">Click me!</button>
```

## 4.3 JavaScript Comments

As with HTML and CSS, it is possible to write comments into a JavaScript code that will be ignored by the browser, and exist to provide instructions to fellow developers on how the code works. There are two types of JavaScript comments

- A single line comment written after a double forward slash (//)

```
// I am a single line comment
```

- A multiline comment is written between the strings (/* */)

```
/*
I am
a multi line comment
*/
```

### 4.4 Basic JavaScript Input Output

Accepting user input in JavaScript actually is not a difficult task. JavaScript has a function called prompt, which is used to accept input from the user. But for the result to be useful for us, it has to be stored in a variable. Here is a simple example

```
var name = prompt("Enter your name: ");

alert("Hello  " + name);
```

- Prompt will show a modal windows with a text message, an input field for the visitor, and the buttons OK/Cancel

- We can also use confirm to get information from a user:- the function confirm shows a modal window with a question and two buttons: OK and Cancel

  The result is true if OK is pressed and false otherwise

Generating an output in JavaScript can be done in different ways.

1. Console.log: outputs a string to the browser console.

2. Alert: shows a browser pop up with the information inside the parenthesis
3. The write() method

- The write() method writes HTML expressions or JavaScript code to a document.
- The write() method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML.

Usage:

document.write(expr)

## 4.5 JavaScript Data Types and Variables

Most of the time, a JavaScript application needs to work with information.

- An online shop – the information might include goods being sold and a shopping cart.
- A chat application – the information might include users, messages, and much more.

In this section we will see how to declare a variable, how to store a value into a variable and the different data types JavaScript supports.

### *4.5.1 JavaScript Data types*

A value in JavaScript is always of a certain type. In JavaScript, a variable can take one type at one moment and then another. This is called dynamic typing.

- Number → represents integers and floating points. There are many operations we can perform on a number multiplication *, division /, addition +, subtraction – and so on
- String → a string in JavaScript must be surrounded by quotes. We have three different quotes
    1. Single quotes --- ‘single’
    2. Double quotes --- ‘double’
    3. Backticks -- `hello`. In a standard US keyboard, you can find the backticks above the tab key before the number 1 key.
       Backticks allows to embed variables and expressions into a string by wrapping them with ${}

```
alert( `the result is ${1 + 2}` );
```

- Boolean → a boolean has two values true or false

- Objects → are used to store a key/value pair property. Objects are created using curly braces {}
- Arrays → are regular objects and used to store indexed values. Arrays are created by using square brackets []

### *4.5.2 Variable declaration in JavaScript*

Variables are used to store this information. A variable is a "named storage" for data. We can use variables to store goodies, visitors, and other data. We have three ways of creating variables in JavaScript

- Using the **let** keyword → the following statement declares a variable
  ```
  let message;
  ```
  now we can put values using the assignment operator (=)
- Using **var** keyword → we can also declare variable using the **var** keyword in a similar fashion as **let**. There is a subtle difference between declaring a variable using **var** and **let**. Variables declared with **var** are hoisted where as those declared with **let** are not. The other difference between var and let is variables declared with **let** has a block level scope unlike that of **var.** Variables declared with var have function level scope
- Using const keyword → to declare a constant variable we can use **const** instead of **var** and **let.** Variables declared with const can not be changed once they are declared.
  ```
  const PI = 3.14;
  ```

### *4.5.3 Data Type Conversion*

Type conversations are a very common occurrence in JavaScript. Operators and functions converts the values given to them to the right type. Mathematical operations convert value to numbers.

- String conversion → we can use String(value) function to convert a value to string
- Numeric Conversion → happens automatically in mathematical operations and expression. We can use the Number(value) function to explicitly convert a value to a number:
  ```
  let str = "123";

  let num = Number(str);
  ```
- Boolean conversation → happens automatically in logical operations. To explicitly convert them we can use Boolean(value) functions

## 4.6 Arithmetic and Logical Operators in JavaScript

Arithmetic operators are used to perform mathematical operations between numeric operands. We can perform different operations on numbers, like addition, subtraction multiplication. Here is the list of arithmetic operators we can use on numbers

| Operator | Name | Purpose | Example |
|---|---|---|---|
| + | Addition | Adds two numbers together. | 6 + 9 |
| - | Subtraction | Subtracts the right number from the left. | 20 - 15 |
| * | Multiplication | Multiplies two numbers together. | 3 * 7 |
| / | Division | Divides the left number by the right. | 10 / 5 |
| % | Remainder (sometimes called modulo) | Returns the remainder left over after you've divided the left number into a number of integer portions equal to the right number. | 8 % 3 (returns 2, as three goes into 8 twice, leaving 2 left over). |
| ** | Exponent | Raises a base number to the exponent power, that is, the base number multiplied by itself, exponent times. It was first Introduced in EcmaScript 2016. | 5 ** 2 (returns 25, which is the same as 5 * 5). |

Logical Operators:
In JavaScript, the logical operators are used to combine two or more conditions. JavaScript provides the following logical operators.

| Operator | Usage | Description |
|---|---|---|
| Logical AND ( && ) | expr1 && expr2 | Returns expr1 if it can be converted to false; otherwise, returns expr2. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false. |
| Logical OR ( \|\| ) | expr1 \|\| expr2 | Returns expr1 if it can be converted to true; otherwise, returns expr2. Thus, when used with Boolean values, \|\| returns true if either operand is true; if both are false, returns false. |
| Logical NOT ( ! ) | !expr | Returns false if its single operand that can be converted to true; otherwise, returns true. |

## 4.7 Control Structures (Conditional and Looping Statements)

The “if” statement:

The if(...) statement evaluates a condition in parentheses and, if the result is true, executes a block of code.

```
let age = 20;
if (age > 18) alert( 'You are right!' );
```

The “else” clause → if statements may contain an optional “else” block. It executes when the condition is false.

In an if statement we may have more than one condition to test. To do that we can use “else if” clause

Example:

```
let age = 20;

if (age < 18) {
        alert( 'Under...' );
} else if (age > 18) {
         alert( 'Over' );
} else {
         alert( '18!' );
}
```

Ternary Operators:

Ternary operator is represented by a question mark “?”. We can use ternary operator to perform conditional operations like if-else statement

```
let result = condition ? value1 : value2;
```

Loops

Loops are a way to repeat the same code multiple times.

- While loop → while loop has the following syntax

```
while (condition) {
   //"loop body"
}
```

- Do-while loop → do while loops are very similar to that of while loops, but in do while even if the condition is false there is always on execution

```
do{
    //"loop body"
} while (condition);
```

- For loop → syntax

```
for (begin; condition; step) {
 // ... loop body ...
}
```

## 4.8 Array in JavaScript

Arrays are used to store ordered collections, where we have a 1st, a 2nd, a 3rd element and so on. For example, we need that to store a list of something: students, courses, HTML elements etc.

There are two syntax for creating an array:

```
let arr = new Array();
let arr = [];
```

Array elements are numbered, starting from zero

```
let fruits = ["Apple", "Orange", "Plum"];
```

- the length property can be used to get the total counts of elements in the array
- Arrays can store elements of any type

Methods

- Push → appends an element to the end
- Pop → takes an element from the end.
- Shift → Extracts the first element of the array and returns it
- Unshift → Add the element to the beginning of the array:

## 4.9 JavaScript Functions

Functions are the main "building blocks" of the program. They allow the code to be called many times without repetition. To create a function, we can use a function declaration.

```
function showCSMessage() {
  alert( 'Hello CS Students!' );
}
```

To create a function, first we use the function keyword then the function name. There are times when we remove the function name. This type Functions without a function name are called anonyms functions.

After the function name we optionally put parameters in between the parenthesis, if we don't have parameters, we always have to put the empty parenthesis and then finally the block of code to be executed.

An arrow function expression is a compact alternative to a traditional function expression, but is limited and can't be used in all situations.

Syntax:

```
param => expression
```

Multiple params require parentheses.

```
(param1, paramN) => expression
```

## 4.10 JavaScript DOM (Document object Model)

The Document Object Model (DOM) is the data representation of the objects that comprise the structure and content of a document on the web. The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

### *4.10.1 Accessing HTML elements in JavaScript*

In the DOM Api, there are different methods to get access to HTML elements. We will list the available methods here, but only the most recent methods are covered here.

- getElementById(id) returns an element with that specific ID
- getElmentsByClassName(class) returns elements with that specific class
  getElementsByTagName(tag) returns elements with the given tag name
- querySelector(selector) → The Document method querySelector() returns the first Element within the document that matches the specified selector, or group of selectors. If no matches are found, null is returned.

```
   element = document.querySelector(selectors);
Accessing an element with the class of “myclass”
   var el = document.querySelector(".myclass");
A more Complex Selector
   var el = document.querySelector("div.myclass”)
```

- querySelectorAll(name) → The Document method querySelectorAll() returns a static (not live) NodeList representing a list of the document's elements that match the specified group of selectors.

### *4.10.2 CSS in JavaScript*

It is possible to change the style of an HTML element by using JavaScript. First access the element you want to change the style by using, the ways After accessing an element we change the style of the element by using the style attribute. In contrast to CSS, properties used in JavaScript has to be written in a camelCase format

Example

element.style.backgroundColor = “red”;

### *4.10.3 Events in JavaScript*

An event is a signal that something has happened. All DOM nodes generate such signals (but events are not limited to DOM).

Here’s a list of the most useful DOM events, just to take a look at:

- `click` – when the mouse clicks on an element (touchscreen devices generate it on a tap).
- `contextmenu` – when the mouse right-clicks on an element.
- `mouseover` / `mouseout` – when the mouse cursor comes over / leaves an element.
- `mousedown` / `mouseup` – when the mouse button is pressed / released over an element.
- `mousemove` – when the mouse is moved.

## Event Handlers

To react on events we can assign a *handler* – a function that runs in case of an event.
Handlers are a way to run JavaScript code in case of user actions.
There are several ways to assign a handler. Let’s see them, starting from the simplest one.
addEventListener
The fundamental problem of the aforementioned ways to assign handlers – we can’t assign multiple handlers to one event. To add an event in an element we can use addEventListener function.

```
element.addEventListener(event, handler, [options]);
```

event
Event name, e.g. "click".
handler
The handler function.
options
An additional optional object with properties:

- once: if true, then the listener is automatically removed after it triggers.
- capture: the phase where to handle the event, to be covered later in the chapter Bubbling and capturing. For historical reasons, options can also be false/true, that's the same as {capture: false/true}.
- passive: if true, then the handler will not call preventDefault(), we'll explain that later in Browser default actions.

To remove the handler, use removeEventListener:

```
element.removeEventListener(event, handler, [options]);
```

### *4.10.4 Handling Exception in JavaScript*

There's a syntax construct try...catch that allows us to "catch" errors so the script can, instead of dying, do something more reasonable.

The "try…catch" syntax

The try...catch construct has two main blocks: try, and then catch:

```
try {

  // code...

} catch (err) {

  // error handling

}
```

## 4.11 Form Processing using JavaScript

A web form consists of any number of input fields grouped in a <form> tag. HTML allows a number of different styles of fields, ranging from simple on/off checkboxes to drop-down menus and fields for text input.

In order to start working with forms with JavaScript we need to intercept the submit event on the form element:

*const form = document.querySelector('form')*

*form.addEventListener('submit', event => {*

*// submit event detected*

*})*

Now inside the submit event handler function we call the event.preventDefault() method to prevent the default behavior and avoid a form submit to reload the page, this gives us control. After we gain control over the submit event we can all kind of stuff, like form validation

Here is a simple but different way of validating a form using javascript

Form Validation

```
<html>
  <head>
    <title>Form Validation</title>
    <script type = "text/javascript">
      if( document.myForm.Name.value == "" ) {
        alert( "Please provide your name!" );
        document.myForm.Name.focus() ;
        return false;
      }
    </script>
  </head>

  <body>
    <form action = "" name = "myForm" onsubmit = "return(validate());">
```

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

## 4.12 JavaScript BOM (Browser Object Model)

The Browser Object Model (BOM) is a browser-specific convention referring to all the objects exposed by the web browser. The BOM allows JavaScript to "interact with" the browser.

### *4.12.1 JavaScript Window*

There are no official standards for the Browser Object Model (BOM).

Since modern browsers have implemented (almost) the same methods and properties for JavaScript interactivity, it is often referred to, as methods and properties of the BOM.

The window object is supported by all browsers. It represents the browser's window.

All global JavaScript objects, functions, and variables automatically become members of the window object.

Global variables are properties of the window object

Global functions are methods of the window object.

Even the document object (of the HTML DOM) is a property of the window object:

window.document.getElementById("header");

is the same as:

document.getElementById("header");

Window Size

Two properties can be used to determine the size of the browser window.

Both properties return the sizes in pixels:

window.innerHeight - the inner height of the browser window (in pixels)

window.innerWidth - the inner width of the browser window (in pixels)

The browser window (the browser viewport) is NOT including toolbars and scrollbars.

window.open() - open a new window
window.close() - close the current window
window.moveTo() - move the current window
window.resizeTo() - resize the current window
The Window interface represents a window containing a DOM document; the document property points to the DOM document loaded in that window.

A window for a given document can be obtained using the document.defaultView property.

A global variable, window, representing the window in which the script is running, is exposed to JavaScript code.

The Window interface is home to a variety of functions, namespaces, objects, and constructors which are not necessarily directly associated with the concept of a user interface window. However, the Window interface is a suitable place to include these items that need to be globally available. Many of these are documented in the JavaScript Reference and the DOM Reference.

In a tabbed browser, each tab is represented by its own Window object; the global window seen by JavaScript code running within a given tab always represents the tab in which the code is running. That said, even in a tabbed browser, some properties and methods still apply to the overall window that contains the tab, such as resizeTo() and innerHeight. Generally, anything that can't reasonably pertain to a tab

### *4.12.2 JavaScript Location*

The Location interface represents the location (URL) of the object it is linked to. Changes done on it are reflected on the object it relates to. Both the Document and Window interface have such a linked Location, accessible via Document.location and Window.location respectively. The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.

Window Location

The window.location object can be written without the window prefix.

Some examples:

window.location.href returns the href (URL) of the current page

window.location.hostname returns the domain name of the web host

window.location.pathname returns the path and filename of the current page

window.location.protocol returns the web protocol used (http: or https:)

window.location.assign() loads a new document

Window Location Href

The window.location.href property returns the URL of the current page.

### *4.12.3 JavaScript Cookies*

Cookies let you store user information in web pages.

What are Cookies?

Cookies are data, stored in small text files, on your computer.

When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user. Cookies were invented to solve the problem "how to remember information about the user":

When a user visits a web page, his/her name can be stored in a cookie.

Next time the user visits the page, the cookie "remembers" his/her name.

Cookies are saved in name-value pairs like:

When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.

None of the examples below will work if your browser has local cookies support turned off.

Create a Cookie with JavaScript

JavaScript can create, read, and delete cookies with the document.cookie property.

With JavaScript, a cookie can be created like this:

document.cookie = "username=John Doe";

You can also add an expiry date (in UTC time). By default, the cookie is deleted when the browser is closed:

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC";

With a path parameter, you can tell the browser what path the cookie belongs to. By default, the cookie belongs to the current page.

document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";

The Document property cookie lets you read and write cookies associated with the document. It serves as a getter and setter for the actual values of the cookies.

# *Chapter 5: Server-Side Scripting (PHP)*

## 5.1 Introduction to PHP

PHP is a server-side scripting language, which can be embedded in HTML or used as a standalone binary (although the former use is much more common). PHP stands for Hypertext Preprocessor. The product was originally named Personal Home Page Tools, and many people still think that is what the acronym stands for. But as it expanded in scope, a new and more appropriate name was selected by community vote. PHP is the language that you use to make the server generate dynamic output—output that is potentially different each time a browser re-quests a page.

Client-side scripting is the glamorous, eye-catching part of Web development. In contrast, server-side scripting is invisible to the user. Server-side Web scripting is mostly about connecting Web sites to back-end servers, such as databases. This enables the following types of two-way communication:

- Server to client: Web pages can be assembled from back end-server output.
- Client to server: Customer-entered information can be acted upon.

Common examples of client-to-server interaction are online forms with some drop-down lists (usually the ones that require you to click a button) that the script assembles dynamically on the server.

By default, PHP documents end with the extension *.php*. When a web server encounters this extension in a requested file, it automatically passes it to the PHP processor. At its very simplest, a PHP document will output only HTML.

## 5.2 Basic PHP Syntax

The most universally effective PHP syntax is:

```
<?php
//php script
?>
```

If you use this style, you can be positive that your tags will always be correctly interpreted. The way you use this tag is quite flexible. Some programmers enclose everything between this opening and closing tag. Others, however, choose to insert only wherever dynamic scripting is required, leaving the rest of the document in standard HTML.

Unless you have a very, very strong reason to prefer one of the other styles, use this

one. Some
or all of the other styles of PHP tag may be phased out in the future—only this one is certain to be safe.

**Short-open (SGML-style) tags**

Short or short-open tags look like this:

```
<?

//php script
?>
```

Those who escape into and out of HTML frequently in each script will be attracted by the prospect of fewer keystrokes; however, the price of shorter tags is pretty high. Additional adjustment in the php.ini file is required to enable PHP to recognize the tags. PHP code written with short-open tags is less portable because you can't be sure another machine will have enabled them in the php.ini file.

### *5.2.1 PHP Comments*

There are two ways in which you can add comments to your PHP code. The first turns a single line into a comment by preceding it with a pair of forward slashes, The "one-line" comment styles only comment to the end of the line or the current block of PHP code, whichever comes first. This means that HTML code after // ... ?> or # ... ?> WILL be printed: ?> breaks out of PHP mode and returns to HTML mode, and // or # cannot influence that.

```
<h1>This is an <?php # echo 'simple';?> example</h1>
```

When you need multiple-line comments, there's a second type of comment. PHP multi-line line comment begins with /* , and ends with */.

```
<?php

/* The following line of code will output the "Hello World!"
   message */

   echo "Hello World!";

?>
```

### *5.2.2 Predefined and User Variables in PHP*

Variables are used to store data, like string of text, numbers, etc. Variable values can change over the course of a script. Here're some important things to know about variables:

- In PHP, a variable does not need to be declared before adding a value to it. PHP automatically converts the variable to the correct data type, depending on its value.
- After declaring a variable, it can be reused throughout the code.
- The assignment operator (=) used to assign value to a variable.
- In php there are two types of variables Predefined and user defined

PHP provides a large number of predefined variables to any script which it runs. PHP provides an additional set of predefined arrays containing variables from the web server the environment, and user input. These new arrays are called superglobals −

All the following variables are automatically available in every scope.

- $GLOBALS → Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables.
- $_SERVER → This is an array containing information such as headers, paths, and script locations. The entries in this array are created by the web server. There is no guarantee that every web server will provide any of these. See next section for a complete list of all the SERVER variables.
- $_GET → An associative array of variables passed to the current script via the HTTP GET method.
- $_POST → An associative array of variables passed to the current script via the HTTP POST method.

In PHP user defined variable can be declared as:

```
$var_name = value;
```

Example

```
<?php
// Declaring variables
$txt = "Hello World!";
$number = 10;

// Displaying variables value
```

```
echo $hello;  // Output: Hello World!
echo $number; // Output: 10
?>
```

## 5.3 PHP Output Statements

The two most basic constructs for printing to output are echo and print. Their language status is somewhat confusing, because they are basic constructs of the PHP language, rather than being functions. As a result, they can be used either with parentheses or without them.

### Echo

The simplest use of echo is to print a string as argument.

For example:

echo “This will print in the user’s browser window.”;

Or equivalently:

echo(“This will print in the user’s browser window.”);

You can also give multiple arguments to the unparenthesized version of echo, separated by commas, as in:

echo “This will print in the “, “user‟s browser window.”;

The parenthesized version, however, will not accept multiple arguments:

echo (“This will produce a “, “PARSE ERROR!”);

### Print

The command print is very similar to echo, with two important differences:

- ✦ unlike echo, print can accept only one argument.
- ✦ unlike echo, print returns a value, which represents whether the print statement succeeded. The value returned by print will be 1 if the printing was successful and 0 if unsuccessful.

## 5.4 Data Types and Variables in PHP

PHP is a very loosely typed language. This means that variables do not have to be declared before they are used, and that PHP always converts variables to the *type* required by their context when they are accessed. Variables can store data of different types, and different data types can do different things. PHP supports the following data types: String, Integer, Float (floating point numbers - also called double), Boolean, Array, Object, NULL and Resource

**Variable:**

The main way to store information in the middle of a PHP program is by using a variable which is a way to name and hang on to any value that you want to use later. Here are the most important things to know about variables in PHP.

□ All variables in PHP are denoted with a leading dollar sign ($).

All variables in PHP start with a leading $ sign. After the initial $, variable names must be composed of letters (uppercase or lowercase), digits (0–9), and underscore characters (_). Furthermore, the first character after the $ may not be a number.

- The value of a variable is the value of its most recent assignment.
- Variables are assigned with the = operator, with the variable on the left-hand side and the expression to be evaluated on the right.
- Variables can, but do not need, to be declared before assignment.
- Variables have no intrinsic type other than the type of their current value.

In PHP, because types are associated with values rather than variables, no such declaration is necessary—the first step in using a variable is to assign it a value.

✓ Variables used before they are assigned have default values.

The following are valid variable declarations.

$pi=3.14;

$name=”kebede”

The following variable declarations are invalid in php.

Int x=3; / *no type declaration is necessary and the variable

Should start with the dollar sign.*/

Char $v=”c”; // no type declaration.

Example: the following program will add two numbers and display the result in the page

```
<HTML>
   <HEAD>
        <title>some title</title>
   </HEAD>
   <BODY>
      <?php
         $first=6;
         $second=5;
         $result=$first+$second;
         Echo $result;
```

```
        ?>
    </BODY>
</html>
```

## 5.5 Arithmetic and Logical Operators

Operators are used to perform operations on variables and values. This section will cover the two most common operators groups Arithmetic and Logical operators.

### Arithmetic Operators:

Arithmetic operators do what you would expect. They are used to perform mathematics. You can use them for the main four operations (plus, minus, multiply, and divide) as well as to find a modulus (the remainder after a division) and to increment or decrement a value.

### Logical Operators:

As a rule, if something has a TRUE or FALSE value, it can be input to a logical operator. A logical operator takes two true-or-false inputs and produces a true-or-false result. The PHP logical operators are sued to combine conditional statements.

## 5.6 Conditional Statements

PHP Conditional Statements are found in every programming language. Conditionals alter program flow; these allows the program to decide a particular path of execution based on a comparative condition. In Conditional statement, a condition is tested and then according to the result of the testing, i.e., either **TRUE** or **FALSE**, an action is performed. There are three types of non-looping conditionals: the if statement, the switch statement, and the ? operator

### The if Statement

- **if statement:** Executes a block of code if the given condition is true.
- **if-else statement:** Executes a block of code if a condition is true or another block of code if that condition is false.
- **if elseif else statement:** Used to test more than one condition with different blocks of code.
- **switch statement:** Alternative for If-elseif statement.

Syntax:

```
if (condition) {
    //code to be executed(condition is true);
} elseif (condition) {
   //code to be executed (elseif condition is true)
} else {
  //code to be executed
}
```

E.g.

Let's write a program that checks the time in your local machine and prints something depending on what the time its

```
<?php
$t = date("H");

if ($t < "10") {
 echo "Have a good morning!";
} elseif ($t < "20") {
 echo "Have a good day!";
} else {
 echo "Have a good night!";
}
?>
```

## Ternary Operators

The Ternary operators are a form of Shorthand Technique for the If-else statement. It uses a question mark (?) and a colon (:) with three operands: a condition which will be evaluated, an action for TRUE and an action for FALSE. Look at the syntax and things will get cleared

```
(condition) ? if TRUE execute this : otherwise execute this;
```

## Switch

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch (n) {
  case label1:
    code to be executed if n=label1;
    break;
  case label2:
    code to be executed if n=label2;
    break;
  case label3:
    code to be executed if n=label3;
    break;
    ...
  default:
    code to be executed if n is different from all labels;
}
```

## 5.7 Loop Statements in PHP

Loops are a way to repeat the same code multiple times. PHP supports four types of looping techniques;

1. for loop
2. while loop
3. do-while loop
4. foreach loop

1. **for loop**:
   This type of loops is used when the user knows in advance, how many times the block needs to execute. That is, the number of iterations is known beforehand. These types of loops are also known as entry-controlled loops. There are three main parameters to the code, namely the initialization, the test condition and the counter.
   Syntax:

```
for (initialization expression; test condition;
update          expression) {
       // code to be executed
   }
```

```
<?php
  for($y = 0; $y <= 10; $y++){
    echo "The value of Y is: $x <br>";
```

```
  }
?>
```

2. **while loop**: The while loop is also an entry control loop like for loops i.e., it first checks the condition at the start of the loop and if its true then it enters the loop and executes the block of statements, and goes on executing it as long as the condition holds true.

```
while (if the condition is true) {
    // code is executed
}
```

3. **do-while loop**: This is an exit control loop which means that it first enters the loop, executes the statements, and then checks the condition. Therefore, a statement is executed at least once on using the do…while loop. After executing once, the program is executed as long as the condition holds true.

```
do {
    //code is executed
} while (if condition is true);
```

```
<?php
  $y = 1;

  do {
    echo "The value of Y is: $y <br>";
    $y++;
  } while ($y <= 5);
?>
```

4. **foreach loop**: This loop is used to iterate over arrays. For every counter of loop, an array element is assigned and the next counter is shifted to the next element.

```
foreach(array_element as value) {
   //code to be executed
}
```

## 5.8 Arrays in PHP

An array in PHP is actually an ordered map. A map is a type that associates *values* to *keys*. This type is optimized for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary,

collection, stack, queue, and probably more. As array values can be other arrays, trees and multidimensional arrays are also possible.

**Syntax**

An array can be created using the array() language construct. It takes any number of comma-separated key => value pairs as arguments.

```
array(
    key  => value,
    key2 => value2,
    key3 => value3,
    ...
)
```

E.g.

```
<?php
   $array1 = array("Item 1", "Item 2", "Item 3", "Item 4");


   $array2 = array('item1' => "Item 1",
             'item2' => "Item 2",
             'item3' => "Item 3",
             'item4' => "Item 4");

?>
```

The use of => is similar to the regular = assignment operator, except that you are assigning a value to an *index* and not to a *variable*.

We can also create and populate an array like this,

```
<?php
  $sample[] = "sample 1";
  $sample [] = "sample  2";
  $sample [] = "sample  3";
  $sample [] = "sample 4";

 print_r($sample);
?>
```

In the example above, each time you assign a value to the array $sample, the first empty location within that array is used to store the value, and a pointer internal to PHP is incremented to point to the next free location, ready for future insertions.
We have also used the print_r function (which prints out the contents of a variable, array, or object) is used to verify that the array has been correctly populated.

We can also create arrays by adding the index of the item in between the square brackets. This will enable us to add items in unorderly fashion

```
<?php
  $sample[0] = "sample 1";
  $sample[1] = "sample  2";
  $sample[2] = "sample  3";
  $sample[3] = "sample 4";

 print_r($sample);
?>
```

## Iterating arrays

We can use any of the loop statements we have seen above to iterate through arrays. In the next example we will add items to an array and retrieve them

```
<?php
   $sample[] = "sample 1";
   $sample [] = "sample  2";
   $sample [] = "sample  3";
   $sample [] = "sample 4";

   for ($j = 0 ; $j < 4 ; ++$j)
       echo "$j: $sample [$j]<br>";
?>
```

The problem with the example above is that we don't always know the size of the array ahead of time. In this case we case use the count and sizeof php functions to get the number of items in the array instead of hard coding it. The modified version the example above is shown here

```
<?php
   $sample[] = "sample 1";
   $sample [] = "sample  2";
   $sample [] = "sample  3";
   $sample [] = "sample 4";

   for ($j = 0 ; $j < count($sample); ++$j)
       echo "$j: $sample [$j]<br>";
?>
```

## Associative Arrays

Associative arrays unlike normal arrays enable has to reference items in an array by name rather than by number. Adding items to associative array and retrieving them

```
<?php
  $sample[index1] = "sample 1";
```

```
  $sample[index2] = "sample 2";
  $sample[index3] = "sample 3";
  $sample[index4] = "sample 4";

 print_r($sample);
?>
```

This very powerful feature of PHP is often used when you are extracting information from XML and HTML. For example HTML parsers uses associative arrays whose name reflect the page's structure

### The foreach loop

We can use foreach loop to step through all the items in an array, one at a time. The process starts with the first item and ends with the last one, so you don't even have to know how many items there are in an array

```
<?php
 $array1 = array("Item 1", "Item 2", "Item 3", "Item 4");

   $j = 0;
   foreach($array1 as $item)
   {
       echo "$j: $item<br>";
       ++$j;
    }
?>
```

## 5.9 PHP Functions

A function is a set of statements that performs a particular function and—optionally—returns a value. Functions have many advantages

- Involve less typing
- Reduce syntax and other programming errors
- Decrease the loading time of program files

PHP comes with hundreds of ready-made functions. You have seen some of them like print, echo, count, sizeof

```
print("hello world")
```

The parenthesis tells php that you are referring to a function. Otherwise, PHP thinks you are referencing to a constant. Functions can take any number of arguments, including zero

### Declaring functions

Function definitions have the following form:

```
function function_name([parameter [, ...]])
{
    // Statements
}
```

That is, function definitions have four parts:

- A definition starts with the word function.
- A name follows, which must start with a letter or underscore, followed by any number of letters, numbers, or underscores.
- The parentheses are required.
- One or more parameters, separated by commas, are optional

Example:

```
function info($name,$age,$salary){
        echo "name:".$name."<br>";
          echo "age:".$age."<br>";
          echo "salary:".$salary."<br>";  }
```

Function names are case-insensitive. PHP functions can return a value including arrays

## 5.10 Form Processing using PHP

Handling a form is a multipart process. First a form is created, into which a user can enter the required details. This data is then sent to the web server, where it is interpreted,
often with some error checking. If the PHP code identifies one or more fields that require
reentering, the form may be redisplayed with an error message. When the code is satisfied with the accuracy of the input, it takes some action that usually involves the database.

**Form Validation:**

Form validation is done to ensure that the user has provided the relevant information. Basic validation can be done using HTML elements. For example, in the above script, the email address text box is having a type value as “email”, which prevents the user from entering the incorrect value for an email. Every form field in the above script is followed by a required attribute, which will intimate the user not to leave any field empty before submitting the form. PHP methods and arrays used in form processing are:

- **isset():** This function is used to determine whether the variable or a form control is having a value or not.
- **$_GET[]:** It is used the retrieve the information from the form control through the parameters sent in the URL. It takes the attribute given in the url as the parameter.
- **$_POST[]:** It is used the retrieve the information from the form control through the HTTP POST method. IT takes name attribute of corresponding form control as the parameter.
- **$_REQUEST[]:** It is used to retrieve an information while using a database.

```
<?php
if (isset($_POST['submit']))
{
   if ((!isset($_POST['firstname']))

    {
        $error = "*" . "Please fill all the required fields";
    }
    else
    {
        $firstname = $_POST['firstname'];
    }
}
?>
<html>

<head>
    <title>Simple Form Processing</title>
</head>

<body>
```

```
    <h1>Form Processing using PHP</h1>
    <fieldset>
        <form id="form1" method="post" action="form.php">
            <?php
                if (isset($_POST['submit']))
                {
                    if (isset($error))
                    {
                        echo "<p style='color:red;'>"
                        . $error . "</p>";
                    }
                }
             ?>

                FirstName:
                <input type="text" name="firstname"/>
                 <span style="color:red;">*</span>
                <br>
                <br>
                <input type="submit" value="Submit"
name="submit" />
        </form>
    </fieldset>
</body>
</html>
```

## 5.11 PHP File Upload

With PHP, it is easy to upload files to the server. A PHP script can be used with a HTML form to allow users to upload files to the server. Initially files are uploaded into a temporary directory and then relocated to a target destination by a PHP script. However, with ease comes danger, so always be careful when allowing file uploads!

**Configure The "php.ini" File**

First, ensure that PHP is configured to allow file uploads.

In your "php.ini" file, search for the file_uploads directive, and set it to On:

```
File_uploads = On
```

When you create the HTML for file upload, here are some rules you need to follow:

- Make sure that the form uses method="post"

- The form also needs the following attribute: enctype="multipart/form-data". It specifies which content-type to use when submitting the form.

- The type="file" attribute of the <input> tag shows the input field as a file-select control, with a "Browse" button next to the input control

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

PHP script explained:

- $target_dir = "uploads/" - specifies the directory where the file is going to be placed
- $target_file specifies the path of the file to be uploaded
- $uploadOk=1 is not used yet (will be used later)
- $imageFileType holds the file extension of the file (in lower case)
- Next, check if the image file is an actual image or a fake image

## 5.12 PHP Cookies and Session

A PHP session variable is used to store information about, or change settings for a user session.
Session variables hold information about one single user, and are available to all pages

in one application. A PHP session allows you to store user information on the server for later use (i.e., username, shopping items, etc.). However, session information is temporary and will be deleted after the user has left the website. If you need a permanent storage you may want to store the data in a database.

### Starting a PHP Session

Before you can store user information in your PHP session, you must first start up the session.

```
<?php session_start(); ?>
<html>
<body>…..
</body>
</html>
```

The code above will register the user's session with the server, allow you to start saving user information, and assign a UID for that user's session.

### Storing a Session Variable

The correct way to store and retrieve session variables is to use the PHP $_SESSION variable:

```
<?php
session_start();
// store session data
$_SESSION['views']=1;
?>
<html>
<body>
<?php
//retrieve session data
echo "Pageviews=". $_SESSION['views'];
?>
</body>
</html>
```

In the example below, we create a simple page-views counter. The isset() function checks if the "views" variable has already been set. If "views" has been set, we can increment our counter. If "views" doesn't exist, we create a "views" variable and set it to 1:

```
<?php
session_start();
if(isset($_SESSION['views']))
$_SESSION['views']=$_SESSION['views']+
1;
```

```
else

$_SESSION['views']=1;
echo "Views=". $_SESSION['views'];  ?>
```

### Destroying a Session

If you wish to delete some session data, you can use the unset() or the session_destroy() function. The unset() function is used to free the specified session variable:

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

You can also completely destroy the session by calling the session_destroy() function:

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

session_destroy() will reset your session and you will lose all your stored session data.

**What is a Cookie?**

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

How to Create a Cookie?

The setcookie() function is used to set a cookie.

**Note:** The setcookie() function must appear BEFORE the <html> tag.

Syntax

```
setcookie(name, value, expire, path, domain);
```

Example

In the example below, we will create a cookie named "user" and assign the value "Alex Porter" to it. We also specify that the cookie should expire after one hour:

```
<?php
setcookie("user", "Alex Porter", time()+3600);
?>
<html>
<body>
</body>
</html>
```

**Note:** The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use setrawcookie() instead).

How to Retrieve a Cookie Value?

The PHP $_COOKIE variable is used to retrieve a cookie value.

In the example below, we retrieve the value of the cookie named "user" and display it on a page:

```
<?php
 // Print a cookie
 echo $_COOKIE["user"];
 // A way to view all cookies
 print_r($_COOKIE);
?>
```

## 5.13 Database Programming using PHP

### *5.13.1 PHP File Overview on MySQL database*

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. It is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

- ***MySQL is a database management system.***

A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network.

- ***MySQL databases are relational.***

A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed.

- ***The MySQL Database Server is very fast, reliable, scalable, and easy to use.*** Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything.

## *5.13.2 Creating Database Connection in PHP*

### 1. Connecting to MySQL

The basic command to initiate a MySQL connection is $link=mysql_connect($hostname, $user, $password); if you‘re using variables, or $link=mysql_connect(„localhost“, „root“, „sesame“); if you‘re using literal strings. Here $Hostname – the ip address or hostname of the MySQL server. If the MySQL server and the php server are on the same machine use **localhost** or **127.0.0.1**, else use the ip address of the database server.
**$User** – the user name of the database if specified by one of the grant commands. If username and password is not given to the database (which is not recommended) use **root** as your username which is the default.
**$Password** – the password of the database if specified by one of the grant commands. The default password for any MySQL database is empty, so if password is not granted leave it null. **$link** – is a variable and is called the **link identifier**. You will use this variable later.

### 2. Selecting the database

After creating the connection identifier, the next thing, you will do is select a database among the databases you have created. To do this you will use the *mysql_select_db()* method.

Syntax:
$db=mysql_select_db(„dbname“,[link identifier]);
Here:

Dbname – the name of the database you are operating on.

Link identifier – is an optional link identifier variable you have created earlier in the mysql_connect function. Use this if there are more than one link identifiers in your page.

### *5.13.3 Sending Query to MySQL Database using PHP*

A database query from PHP is basically a MySQL command wrapped up in a tiny PHP function called mysql_query(). This is where you use the basic SQL workhorses of SELECT,
INSERT, UPDATE, and DELETE, CREATE or DROP a table (but not those to create or drop an entire database) can be used with this PHP function.

Syntax:
[$resultvar=] mysql_query("query",[link identifier]);

$resultvar – is an optional result variable. This is used when the query is a select statement.
Query – is any valid **insert, delete, update, select, create table** or **drop table** statement enclosed by quotes. It is a good habit to hold the query in a separate variable to add flexibility to your code.
Link identifier-is an optional link identifier

### *5.13.4 Processing Query Result.*

#### 4. Fetching data sets

The result of a query returned by the mysql_query() function is an interger showing the success or failure of the function not the desired data. So you have to use one of the mysql_fetch_..() functions to get the actual result. There are many functions to do so, but the most commonly used ones are- Mysql_fetch_row($result) – this function takes the result of the mysql_query() function as an argument and returns the data as an enumerated array.
Example: assume the student table has three attrubutes - **id,name and department**.

```
$query="select * from student";
$result=mysql_query($query);
     while($row=mysql_fetch_row($result)){
     echo "$row[0].$row[1].$row[2]";//counting
starts from 0 not 1
}
```

mysql_fetch_object($result) - this function takes the result of the mysql_query() function as an argument and returns the data as an object.
Example:

```
$query="select * from student";
```

```
$result=mysql_query($query);
While($row=mysql_fetch_object($result)){
      Echo "$row->id . $row->name . $row->department";

}
```

Mysql_fetch_array($result) - this function takes the result of the mysql_query() function as an argument and returns the data as an enumerated or associative array. This is the most widely used fetching function since results are displayed in association to their column values.

Example:

```
$query="select * from student";
$result=mysql_query($query);
While($row=mysql_fetch_array($result)){
      Echo "$row[„id"] . $row-[„name"] . $row[„department"]";
      //mysql_fetch_array() could be used with the enumerated type just
      //like the mysql_fetch_row function.
}
```

### 5. Closing the connection

After you have finished the mysql query the last thing you have to do is close the connection.

Syntax: MySQL-close([link identifier]);

## 5.14 Input-Output

PHP has several functions for creating, reading, uploading, and editing files. The fopen() function is used to open files in PHP.

Opening a file

The first parameter of this function contains the name of the file to be opened and the second parameter specifies in which mode the file should be opened: Example

```
<html>

<body>

<?php
```

```
$file=fopen("hello.txt","r");

?>

</body>

</html>
```

The file may be opened in one of the following modes:

| Modes | Description |
|---|---|
| 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 |

**Note:** If the fopen() function is unable to open the specified file, it returns 0 (false).

readfile() Function

The readfile() function reads a file and writes it to the output buffer.

Assume we have a text file called "hello.txt", stored on the server, that looks like this:

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

Closing a File:-The fclose() function is used to close an open file:

```
<?php

$file = fopen("test.txt","r");

//some code to be executed

fclose($file);?>

?>
```

## 5.15 PHP Date and Time

The typical PHP developer likely needs to be aware of the date and time functions available to them, such as when adding a date stamp to a database record entry or calculating the difference between two dates. PHP provides a DateTime class that can be used to handle date and time information at the same time. There is also a Date TimeZone class that works hand in hand with it.

Time zone management has become more prominent in recent years with the onset of web portals and social web communities like Facebook and Twitter.

There are a total of four interrelated classes for handling dates and times. The Date Time class handles dates themselves; the DateTimeZone class handles time zones; the DateTimeInterval class handles spans of time between two DateTime instances; and finally, the DatePeriod class handles traversal over regular intervals of dates and times.

The constructor of the DateTime class is naturally where it all starts. This method takes
two parameters, the timestamp and the time zone. For example:

```
$dt = new DateTime("2010-02-12 16:42:33", new DateTimeZone("America/Halifax"));
```

We can also do it like this,

```
$dtz = new DateTimeZone("America/Halifax");
$dt = new DateTime("2012-06-16 16:42:33", $dtz);
```

Now obviously we are assigning hardcoded values to these classes, and this type of

information may not always be available to your code or it may not be what you want. Alternatively, we can pick up the value of the time zone from the server and use that inside the DateTimeZone class.

```
$tz = ini_get('date.timezone');
$dtz = new DateTimeZone($tz);
$dt = new DateTime("2012-06-16 16:42:33", $dtz);
```

The diff() method of DateTime does what you might expect—it returns the difference between two dates. The return value of the method is an instance of the DateInterval class.

```
$tz = ini_get('date.timezone');
$dtz = new DateTimeZone($tz);

$past = new DateTime("2009-02-12 16:42:33", $dtz);
$current = new DateTime("now", $dtz);

// creates a new instance of DateInterval
$diff = $past->diff($current);
```

## 5.16 PHP Mathematical Functions

The math functions can handle values within the range of integer and float types.The PHP math functions are part of the PHP core. No installation is required to use these functions. Here are some PHP math functions, you can get the whole list from the internet

abs — Absolute value

acos — Arc cosine

acosh — Inverse hyperbolic cosine

asin — Arc sine

asinh — Inverse hyperbolic sine

atan2 — Arc tangent of two variables

atan — Arc tangent

atanh — Inverse hyperbolic tangent

base_convert — Convert a number between arbitrary bases

bindec — Binary to decimal

ceil — Round fractions up

cos — Cosine

Example

```
<?php
        echo abs(-4.2); // 4.2 (double/float)
        echo abs(5);    // 5 (integer)
        echo abs(-5);   // 5 (integer)
?>
```

## 5.17 PHP OOP

Object-oriented programming (OOP) opens the door to cleaner designs, easier maintenance, and greater code reuse. In php OOP we use the term class, which refers to a template for building objects. And an object is an instance of a class. The data associated with an object are called its properties. The functions associated with an object are called its methods. When you define a class, you define the names of its properties and give the code for its methods.

OOP Case

Let's assume we have a class named Fruit. A Fruit can have properties like name, color, weight, etc. We can define variables like $name, $color, and $weight to hold the values of these properties.

When the individual objects (apple, banana, etc.) are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.

### Creating Objects

To create an object of a given class, use the keyword new

```
$object = new Class;
```

Assuming that a Fruit class has been defined, here's how to create a Fruit object:

```
$apple = new Fruit;
```

Some classes permit you to pass arguments to the new call. The class's documentation should say whether it accepts arguments. If it does, you'll create objects like this:

```
$apple = new Fruit("apple", "20g");
```

## Accessing Properties and methods:

Once you have an object, you can use the -> notation to access methods and properties of the object:

```
$object->propertyname $object->methodname([arg, … ])
```

Methods act the same as functions (only specifically to the object in question), so they can take arguments and return a value. Within a class's definition, you can specify which methods and properties are publicly accessible and which are accessible only from within the class itself using the public and private access modifiers.

## Declaring a Class

A class definition includes the class name and the properties and methods of the class. Class names are case-insensitive and must conform to the rules for PHP identifiers. The class name stdClass is reserved. Here's the syntax for a class definition:

```
class classname [ extends baseclass ] [ implements
interfacename ,
[interfacename, ... ] ]
{
     [ use traitname, [ traitname, ... ]; ]
     [ visibility $property [ = value ]; ... ]

     [ function functionname (args) {
          // code
```

```
        }
        ...
    ]
}
```

## Declaring Methods

A method is a function defined inside a class. Although PHP imposes no special restrictions, most methods act only on data within the object in which the method resides.

Within a method, the $this variable contains a reference to the object on which the method was called. For instance, if you call $apple->getWeight(), inside the getWeight() method, $this holds the same value as $apple. Methods use the $this variable to access the properties of the current object and to call other methods on that object. Here's a simple class definition of the Fruit class that shows the $this variable in action:

```
class Fruit
{
    public $weight = '';
    function getWeight()
    {
        return $this->weight;
    }
    function setWeight($newWeight)
    {
        $this->weight = $newWeight;
    }
}
```

To declare a method as a static method, use the static keyword. Inside of static methods
the variable $this is not defined. If you declare a method using the final keyword, subclasses cannot override that method.

## Declaring Properties

Property declarations are optional and are simply a courtesy to whomever maintains your program. It's good PHP style to declare your properties, but you can add new

properties at any time. Here's a version of the Person class that has an undeclared $name property:

```
class Fruit
{
      public $weight = '';

      function setWeight($newWeight)
      {
            $this->weight = $newWeight;
      }
}
```

# *Chapter 6: Advanced JavaScript and XML (AJAX)*

## 6.1 Introduction to AJAX

AJAX is a web development technique for creating interactive web applications. AJAX stands for Asynchronous JavaScript and XML. It can send and receive information in various formats, including JSON, XML, HTML, and text files. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script. AJAX's most appealing characteristic is its "asynchronous" nature, which means it can communicate with the server, exchange data, and update the page without having to refresh the page.
The two major features of AJAX allow you to do the following:

- Make requests to the server without reloading the page
- Receive and work with data from the server

## 6.2 XMLHttpRequest Object

XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full-page refresh. This enables a Web page to update just part of a page without disrupting what the user is doing. Despite its name, XMLHttpRequest can be used to retrieve any type of data, not just XML.

```
var httpRequest = new XMLHttpRequest()
```

## 6.3 Sending Request to PHP server

XMLHttpRequest (XHR) is an API that can be used by JavaScript and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

In order to make an HTTP request to the server with JavaScript, you need an instance of an `XMLHttpRequest` Object. We make a request to the sever, by calling the open() and send() methods of the HTTP request object, like this:

```
httpRequest.open('GET', 'http://www.example.org/some.file', true);
httpRequest.send();
```

open() takes three parameters, the first is the HTTP request method and the second is the URL. The optional third parameter sets whether the request is asynchronous.

## 6.4 Handling Response from Server

Before making a request to the server we have to write a callback function which will handle the response. But Because Ajax calls are asynchronous, we can't be sure when the response will come, so we must write code to wait for the response and handle it when it arrives. That is what the callback function is used for

At this stage, we need to tell the XMLHttpRequest object which JavaScript function will handle the response, by setting the onreadystatechange property of the object and naming it after the function to call when the request changes state, like this:

```
httpRequest.onreadystatechange = nameOfTheFunction;
```

# *Chapter 7: Introduction to web development frameworks*

This chapter will introduce you to the some of the different frameworks available for the web.

**a. Bootstrap**

Now a days the website you build has to support all browser types and all sizes of screen (Desktop, Tablets, Phones etc..). To make that possible we use different frameworks. This part of the chapter will introduce you to the Bootstrap framework.

Bootstrap is a free front end framework for faster and easier web development. It is s an open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

Bootstrap solves many problems that we face when using HTML,CSS and vanilla JavaScript.

Why use Bootstrap?

1. It is easy to setup
2. Faster and Easier Web Development.
3. Has lots of component and a good grid system
4. It creates Platform-independent web pages.
5. It creates Responsive Web-pages.
6. It is designed to be responsive to mobile devices too.

How to use Bootstrap

To add bootstrap you can either use the bootstrapCDN or you can download it and use it locally. Here are the steps to add the bootstrapCDN to our project,

1. Copy-paste the stylesheet <link> into your <head> before all other stylesheets to load our CSS.

2. Many of Bootstrap components require the use of JavaScript to function. Specifically, they require jQuery, Popper.js, and boostrap JavaScript plugins.
   Follow this order when including the JS files

- JQuery

- Popper

- then boostrap

*Assignment: Download bootstrap( or bootstrapCDN) and build a simple card with navigation bar at the top*

**b. Node.js**

Node is an open source, cross platform runtime environment that allows developers to create server-side tools and applications in JavaScript. As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

Benefits of using Node

1. Performance
2. Uses plain old JavaScript
3. Thousands of reusable packages
4. Portable

To use Node, We first have to install Node in our system. You can follow the installation process from the official Node website.

After the installation you can start building a web server using Node. To create a node application follow the following steps

1. Create a folder to store your node files
2. Create a file that ends with .js (you can name anything you want) in the folder you create in step 1
3. Open the file in your favorite editor and put the following starter code

```
// code goes here

const http = require('http');

const port = 8000;

const server = http.createServer((req, res) => {

  res.statusCode = 200;
```

```
  res.setHeader('Content-Type', 'text/plain');

  res.end('Hello World');

});

server.listen(port, () => {

  console.log(`Server running at http://${hostname}:${port}/`);

});
```

4. Safe the file
5. Go to CMD and run this code from the directory you created – node filename.js

*Assignment: Write a simple node server that accepts a message from a client*

**c. Angular.js**

Angular is a development platform, built on TypeScript. As a platform, Angular includes:

1. A component-based framework for building scalable web applications
2. A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
3. A suite of developer tools to help you develop, build, test, and update your code

The fastest and recommended way to develop angular application is using angular CLI(Command line interface). The CLI makes a number of task easy. Here are some of the commands

| | |
|---|---|
| ng build | Compiles an Angular app into an output directory. |

| | |
|---|---|
| ng serve | Builds and serves your application, rebuilding on file changes. |
| ng generate | Generates or modifies files based on a schematic. |
| ng test | Runs unit tests on a given project. |
| ng e2e | Builds and serves an Angular application, then runs end-to-end tests. |

You can install angular cli using the following command

npm install -g @angular/cli

To create a new application use the ng new command,

ng new <name> --routing=false --style=css

Then use the ng server command to run the application

*Assignment: Write a simple hello world angular app using angular cli*

**d. React.js**

React.js is an open-source front-end JavaScript library for building user interfaces based on UI components. React builds encapsulated component that manage their own state, then compose them to complex UIs. To build for the web, developers use

React in tandem with ReacDOM. The goal of React is to minimize the bugs that occur when developers are building UIs.

React can also be use to create mobile application using React Native, you can read more about this and how to set it up from the official website.

We can create a React app using the create-react-app. Create React App is the best way to start building a new single-page application in React. It sets up your development environment so that you can use the latest JavaScript features, provides a nice developer experience, and optimizes your app for production.

React does not enforce strict rules around code conventions or file organization. React also utilizes features of modern JavaScript for many of its patterns. React allows us to include HTML codes into our JavaScript code by using JSX. JSX is a syntax extension of JavaScript. Here is an example of JSX syntax

const element = <h1>Hello, world!</h1>;

The above code doesn't work with normal JavaScript, you can check this by running the following code in the developers tool console.

## Initializing your app

create-react-app takes one argument: the name you'd like to give your app. create-react-app uses this name to make a new directory, then creates the necessary files inside it.

npx create-react-app <name>

use npm start to run the application (Make sure you cd to the the folder created by the create-react-app command)

Reading Assignment

1. Componnets
2. Props
3. State