Web Technology (PS02CMCA33)

Unit 1: Client-side Web Technologies – I

Introduction to HTTP and HTML5:

HTTP:

HTTP is a protocol which allows the fetching of resources, such as HTML documents. It is the foundation of any data exchange on the Web and it is a client-server protocol, which means requests are initiated by the recipient, usually the Web browser.

(HyperText Transfer Protocol) The communications protocol used to connect to Web servers on the Internet or on a local network (intranet). The primary **function** of **HTTP** is to establish a connection with the server and send HTML pages back to the user's browser.

It is stateless protocol.

OR

HTTP is the protocol used to transfer data over the web. It is part of the Internet protocol suite and defines commands and services used for transmitting webpage data. **HTTP** uses a server-client model. A client, for example, may be a home computer, laptop, or mobile device.

HTML5:

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and last major HTML version that is a World Wide Web Consortium recommendation. The current specification is known as the HTML Living Standard.

HTML5 is a programming language whose acronym stands for Hyper Text Markup Language. It is a system that allows the modification of the appearance of web pages, as well as making adjustments to their appearance. It also used to structure and present content for the web.

HTML5 is the latest version of HTML programming that allows better management of the web application or the website contents. While HTML doesn't allow support for

Video and Audio data in the programming language**, HTML5 allows** any kind of data to be incorporated into the program.

HTML5 may contain HTML, JavaScript, CSS.

URL format:

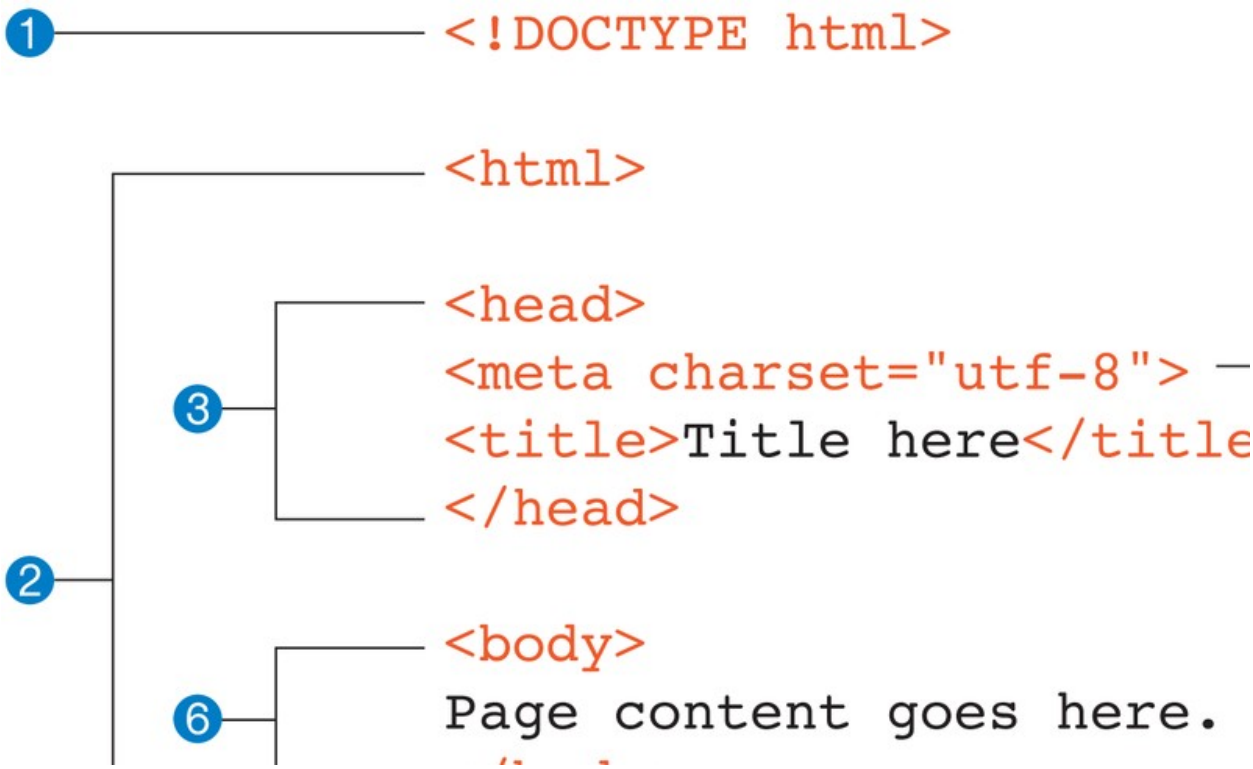
A URL is another word for a web address.

Web browsers request pages from web servers by using a URL.

A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.

- **scheme** - defines the **type** of Internet service (most common is **http or https**)
- **prefix** - defines a domain **prefix** (default for http is **www**)
- **domain** - defines the Internet **domain name** (like google.com)
- **port** - defines the **port number** at the host (default for http is **80**)
- **path** - defines a **path** at the server (If omitted: the root directory of the site)
- **filename** - defines the name of a document or resource

HTML5 document structure:



DOCTYPE - A basic HTML page starts with the Document Type Declaration or doctype. That is a way to inform the browser what type of document it is. The doctype is always the first item at the top of any HTML file. Then sections and subsections come, each possibly has its heading and subheading. These heading and sectioning elements help the reader to perceive the content meaning.

The <html> element
The <html> element follows the doctype information, which is used to inform the browser that this is an HTML document.

he <head> section
The next part is the <head> section. The <head> element contains metadata (document title, character set, styles, links, scripts), specific information about the web page that is not displayed to the user.

The <body> element
The <body> of a document contains the content of the document. The content may be presented by a user agent in different ways. E.g., the content can be text, images, links, colors, graphics, etc.

Headers:

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

Example:

A header for an <article>:

```
<article>
  <header>
    <h1>A heading here</h1>
    <p>Posted by John Doe</p>
    <p>Some additional information here</p>
  </header>
  <p>Lorem Ipsum dolor set amet....</p>
</article>
```

Body:

The <body> tag defines the document's body.

The <body> element contains all the contents of an HTML document, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

Example:

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

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

</html>
```

Declarations:

All HTML documents must start with a <!DOCTYPE> declaration.

The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.

Example:

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

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

</html>
```

HTML Elements:

The HTML **element** is everything from the start tag to the end tag:

<tagname>Content goes here...</tagname>

Examples of some HTML elements:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

Nested HTML Elements

HTML elements can be nested (this means that elements can contain other elements).

All HTML documents consist of nested HTML elements.

The following example contains four HTML elements (<html>, <body>, <h1> and <p>):

e.g,

```
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

HTML element ID:

The HTML id attribute is used to specify a unique id for an HTML element.

You cannot have more than one element with the same id in an HTML document.

Using The id Attribute

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.

The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

HTML name attribute:

The name attribute specifies a name for an HTML element.

This name attribute can be used to reference the element in a JavaScript.

HTML attributes:

HTML attributes provide additional information about HTML elements.

- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

e.g. href, src, etc.

HTML events:

HTML has the ability to let events trigger actions in a browser.

onsubmit

Execute a JavaScript when a form is submitted

e.g.

```
<html>

<body>

<script>

function myFunction()

{

  alert("The form was submitted");

}

</script>

<p>When you submit the form, a function is triggered which alerts some text.</p>
```

<form action="/action_page.php" onsubmit="myFunction()">

** Enter name: <input type="text" name="fname">**

** <input type="submit" value="Submit">**

</form>

</body>

</html>

onclick

Execute a JavaScript when a button is clicked.

<html>

<body>

<script>

function myFunction() {

  document.getElementById("demo").innerHTML = "Hello World";

}

</script>

<h1>The onclick Event</h1>

The onclick event is used to trigger a function when an element is clicked on.

Click the button to trigger a function that will output "Hello World" in a p element with id="demo".

```
<button onclick="myFunction()">Click me</button>
<p id="demo"></p>
</body>
</html>
```

ondblclick

Execute a JavaScript when a <p> element is double-clicked.

```
<html>
<body>

<p ondblclick="myFunction()">Double-click this paragraph to trigger a function.</p>

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

<script>
function myFunction()
{
  document.getElementById("demo").innerHTML = "Hello World";
}
</script>

</body>
</html>
```

HTML5 media

Multimedia on the web is sound, music, videos, movies, etc.

<video>

e.g.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

To start a video automatically, use the autoplay attribute.

<audio>

e.g.

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

To start an audio file automatically, use the autoplay attribute.

HTML forms:

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

The HTML <form> element is used to create an HTML form for user input:

```
<form>

.

.

.

form elements

.

.

.

</form>
```

The <form> element is a container for different types of input elements like, test fields, checkboxes, radio buttons, submit buttons, etc.

HTTP Verbs:

1. **GET:** It is **default method**. It is **less secure** to use because it displays user input values along with URL in address bar of web browser.

   Example:

   <form name=”test” method=”get” action=”test.php”>

   </form>

2. **POST:** It is not a default method. It is **secure** to use because it does not display user input values along with URL in address bar of web browser.

   Example:

   <form name=”test” method=”post” action=”test.php”>

   </form>

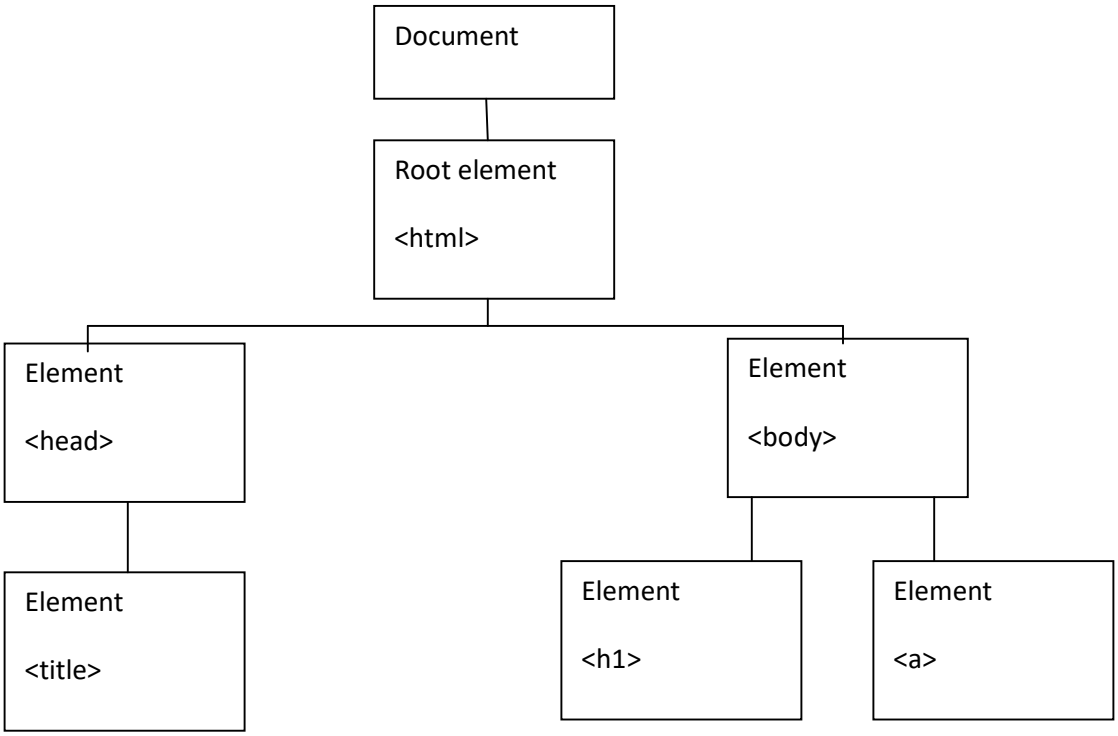
Introduction to the DOM (Document Object Model):

The HTML DOM is an Object Model for HTML.

It defines:

1. HTML elements as objects
2. Properties for all HTML elements
3. Methods for all HTML elements
4. Events for all HTML elements

DOM (basic hierarchy)



Introduction to CSS3:

The web today is almost unrecognizable from the early days of white pages with lists of blue links. Now, sites are designed with complex layouts, unique fonts, and customized color schemes. Cascading Style Sheets (CSS3) focus on how write CSS rules.

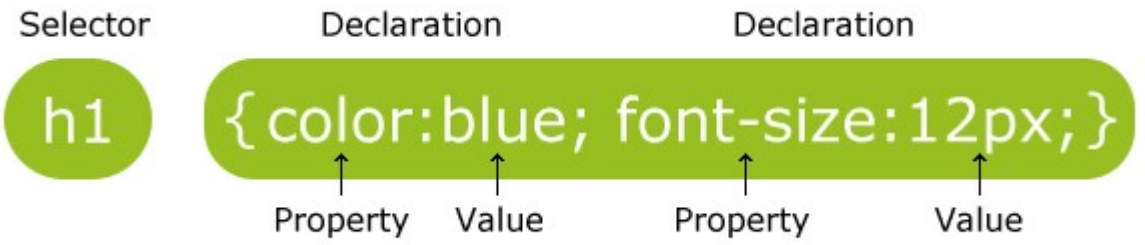
Using CSS once can implement the design by adding fonts, colors, and layouts.

CSS3 is the latest standard for CSS.

CSS3 is completely backwards-compatible with earlier versions of CSS.

CSS Syntax:

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

Different properties, values, units:

Once can set different properties as per CSS syntax.

e.g.

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

Specifying colors:

Example:

Set Background color:

```
<p style=”background-color:Tomato;”> abcd1 </p>
```

Set Text color:

```
<p style=”color:Blue;”> abcd2 </p>
```

Types of CSS:

1. External CSS
2. Internal CSS
3. Inline CSS

1. **External CSS:** One can create external CSS by adding separate .css file, and can specify different properties along with values.

   External styles are defined within the <link> element, inside the <head> section of an HTML page.

```
<head>
<link rel=”stylesheet” href=”test.css”>
</head>
<body>
        <p> This is test paragraph. </p>
</body>
```

test.css

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

2. **Internal CSS:**

Internal styles are defined within the <style> element, inside the <head> section of an HTML page.

```
<head>
<style>

p {

color:red;

text-align:center;

}


</style>
</head>


<body>

    <p> This is test paragraph. </p>

</body>
```

3. **Inline CSS:**

Inline styles are defined with style attribute, style attribute can contain any CSS property.

```
<body>

    <p style=”background-color:Tomato;”> abcd1 </p>

</body>
```

HTML control examples:

1. **Select option and Text area**

index.php

```
<h1>The select element</h1>

<p>The select element is used to create a drop-down list.</p>

<form action="test1.php">
  <label for="cars">Choose a car:</label>

  <select name="cars" id="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="opel">Opel</option>
    <option value="audi">Audi</option>
  </select>

  <br><br>

<textarea name="t1">

</textarea>

  <input type="submit" value="Submit">
</form>

<p>Click the "Submit" button .</p>
```

test1.php

```
echo $_GET['cars'];
echo $_GET['t1'];
```

2. **Checkboxes**

```
<h1>Show Checkboxes</h1>


<form action="action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>
```

3. **Radio button**

```
<form action="action_page.php">
  <p>Please select your gender:</p>
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label><br>
  <input type="radio" id="female" name="gender" value="female">
  <label for="female">Female</label><br>
  <input type="radio" id="other" name="gender" value="other">
  <label for="other">Other</label>
```

```
  <br>

  <p>Please select your age:</p>
  <input type="radio" id="age1" name="age" value="30">
  <label for="age1">0 - 30</label><br>
  <input type="radio" id="age2" name="age" value="60">
  <label for="age2">31 - 60</label><br>
  <input type="radio" id="age3" name="age" value="100">
  <label for="age3">61 - 100</label><br><br>
  <input type="submit" value="Submit">
</form>
```