INTERNET FUNDAMENTALS

SECTION-C (NOTES)

SEM-4TH

HTML

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

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

</body>
</html>
```

- The <!DOCTYPE html> declaration defines this document to be HTML5
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the document
- The <title> element specifies a title for the document

- The <body> element contains the visible page content
- The <h1> element defines a large heading
- The <p> element defines a paragraph

HTML Tags

HTML tags are element names surrounded by angle brackets:

<tagname>content goes here...</tagname>

- HTML tags normally come **in pairs** like <p> and </p>
- The first tag in a pair is the **start tag,** the second tag is the **end tag**
- The end tag is written like the start tag, but with a **forward slash** inserted before the tag name

Styling HTML with CSS

CSS stands for **C**ascading **S**tyle **S**heets.

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

CSS **saves a lot of work**. It can control the layout of multiple web pages all at once.

CSS can be added to HTML elements in 3 ways:

- **Inline** - by using the style attribute in HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using an external CSS file

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

This example sets the text color of the <h1> element to blue:

Example

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
      body {background-color: powderblue;}
      h1   {color: blue;}
      p    {color: red;}
</style>
  </head>
  <body>

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

</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

With an external style sheet, you can change the look of an entire web site, by changing one file!

To use an external style sheet, add a link to it in the <head> section of the HTML page:

Example

```
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" href="styles.css">
</head>
<body>

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

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

```
body {
    background-color: powderblue;
}
h1 {
    color: blue;
}
p {
    color: red;
}
```

CSS Fonts

The CSS **color** property defines the text color to be used.

The CSS **font-family** property defines the font to be used.

The CSS **font-size** property defines the text size to be used.

```
<!DOCTYPE html>
<html>
<head>
<style>
   h1 {
    color: blue;
    font-family: verdana;
    font-size: 300%;
         }
           p  {
                color: red;
               font-family: courier;
               font-size: 160%;
              }
              </style>
            </head>
        <body>

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

   </body>
   </html>
```

JavaScript is one of the **3 languages** all web developers **must** learn:

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is **getElementById()**.

This example uses the method to "find" an HTML element (with id="demo") and changes the element content (**innerHTML**) to "Hello JavaScript":

Example
```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

With HTML and JavaScript you can use single or double quotes:

Example
```
document.getElementById('demo').innerHTML = 'Hello JavaScript';
```

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

Example
```
document.getElementById("demo").style.fontSize = "25px";
```

JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the display style:

Example
```
document.getElementById("demo").style.display = "none";
```

JavaScript Operators

Example
Assign values to variables and add them together:

```
var x = 5;         // assign the value 5 to x
var y = 2;         // assign the value 2 to y
var z = x + y;     // assign the value 7 to z (x + y)
```

The **assignment** operator (=) assigns a value to a variable.

The **addition** operator (+) adds numbers:

Adding

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

The **multiplication** operator (*) multiplies numbers.

Multiplying

```
var x = 5;
var y = 2;
var z = x * y;
```

Arithmetic operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

Example

```
txt1 = "John";
txt2 = "Doe";
txt3 = txt1 + " " + txt2;
```

The result of txt3 will be:

```
John Doe
```

The += assignment operator can also be used to add (concatenate) strings:

Example
txt1 = "What a very ";
txt1 += "nice day";

The result of txt1 will be:

What a very nice day

Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

Example
x = 5 + 5;
y = "5" + 5;
z = "Hello" + 5;

The result of *x*, *y*, and *z* will be:

10
55
Hello5

JavaScript Comparison Operators

Operator	**Description**
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	**Description**
&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JavaScript Data Types

JavaScript variables can hold many **data types**: numbers, strings, objects and more:

```
var length = 16;                              // Number
var lastName = "Johnson";                     // String
var x = {firstName:"John", lastName:"Doe"};   // Object
```

JavaScript Types are Dynamic.

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
var x;              // Now x is undefined
var x = 5;          // Now x is a Number
var x = "John";     // Now x is a String
```

JavaScript Strings

A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

Example

```
var carName = "Volvo XC60";   // Using double quotes
var carName = 'Volvo XC60';   // Using single quotes
```

JavaScript Numbers

JavaScript has only one type of numbers.

Numbers can be written with, or without decimals:

Example

```
var x1 = 34.00;     // Written with decimals
var x2 = 34;        // Written without decimals
```

JavaScript Booleans

Booleans can only have two values: true or false.

Example

```
var x = true;
var y = false;
```

Booleans are often used in conditional testing.

JavaScript Arrays

JavaScript arrays are written with square brackets.

Array items are separated by commas.

The following code declares (creates) an array called cars, containing three items (car names):

Example

```
var cars = ["Saab", "Volvo", "BMW"];
```

JavaScript Objects

JavaScript objects are written with curly braces.

Object properties are written as name:value pairs, separated by commas.

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

JavaScript Forms

JavaScript Form Validation

HTML form validation can be done by JavaScript.

JavaScript Example

```
function validateForm() {
    var x = document.forms["myForm"]["fname"].value;
    if (x == "") {
        alert("Name must be filled out");
        return false;
    }
}
```

The function can be called when the form is submitted:

HTML Form Example

```
<form name="myForm" action="/action_page_post.php" onsubmit="return validateForm()"
method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

Data Validation

Data validation is the process of ensuring that user input is clean, correct, and useful.

Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct user input.

Validation can be defined by many different methods, and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.

Client side validation is performed by a web browser, before input is sent to a web server.

HTML Constraint Validation

HTML5 introduced a new HTML validation concept called **constraint validation**.

HTML constraint validation is based on:

- Constraint validation **HTML Input Attributes**
- Constraint validation **CSS Pseudo Selectors**
- Constraint validation **DOM Properties and Methods**

The checkValidity() Method

```
<input id="id1" type="number" min="100" max="300" required>
<button onclick="myFunction()">OK</button>
```

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

<script>
function myFunction() {
   var inpObj = document.getElementById("id1");
   if (inpObj.checkValidity() == false) {
      document.getElementById("demo").innerHTML = inpObj.validationMessage;
   }
}
</script>
```

The rangeOverflow Property

```
<input id="id1" type="number" max="100">
<button onclick="myFunction()">OK</button>

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

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

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

Use POST requests when:

- A cached file is not an option (update a file or database on the server).
- Sending a large amount of data to the server (POST has no size limitations).
- Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

GET Requests

A simple GET request:

Example
```
xhttp.open("GET", "demo_get.asp", true);
xhttp.send();
```

```
xhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);
xhttp.send();
```

If you want to send information with the GET method, add the information to the URL:

Example
```
xhttp.open("GET", "demo_get2.asp?fname=Henry&lname=Ford", true);
xhttp.send();
```

A simple POST request:

Example
```
xhttp.open("POST", "demo_post.asp", true);
xhttp.send();
```

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

Example
```
xhttp.open("POST", "ajax_test.asp", true);
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
xhttp.send("fname=Henry&lname=Ford");
```

The url - A File On a Server

The url parameter of the open() method, is an address to a file on a server:

```
xhttp.open("GET", "ajax_test.asp", true);
```

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

Example
```
document.getElementById("demo").innerHTML = xhttp.responseText;
```

The responseXML Property

The XML HttpRequest object has an in-built XML parser.

The **responseXML** property returns the server response as an XML DOM object.

Using this property you can parse the response as an XML DOM object:

Example
Request the file cd_catalog.xml and parse the response:

```
xmlDoc = xhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
  txt += x[i].childNodes[0].nodeValue + "<br>";
  }
document.getElementById("demo").innerHTML = txt;
xhttp.open("GET", "cd_catalog.xml", true);
xhttp.send();
```

The getAllResponseHeaders() Method

The **getAllResponseHeaders()** method returns all header information from the server response.

Example

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getAllResponseHeaders();
  }
};
```

The getResponseHeader() Method

The **getResponseHeader()** method returns specific header information from the server response.

Example

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
    document.getElementById("demo").innerHTML =
    this.getResponseHeader("Last-Modified");
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

XML BASICS:

XML stands for eXtensible Markup Language.

XML was designed to store and transport data.

XML was designed to be both human- and machine-readable.

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>
   Two of our famous Belgian Waffles with plenty of real maple syrup
   </description>
    <calories>650</calories>
</food>
<food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>
    Light Belgian waffles covered with strawberries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>Berry-Berry Belgian Waffles</name>
    <price>$8.95</price>
    <description>
    Belgian waffles covered with assorted fresh berries and whipped cream
    </description>
    <calories>900</calories>
</food>
<food>
    <name>French Toast</name>
    <price>$4.50</price>
    <description>
    Thick slices made from our homemade sourdough bread
    </description>
    <calories>600</calories>
</food>
<food>
    <name>Homestyle Breakfast</name>
    <price>$6.95</price>
    <description>
    Two eggs, bacon or sausage, toast, and our ever-popular hash browns
    </description>
    <calories>950</calories>
</food>
</breakfast_menu>
```

How Can XML be Used?

XML is used in many aspects of web development.

XML is often used to separate data from presentation.

XML Separates Data from Presentation

XML does not carry any information about how to be displayed.

The same XML data can be used in many different presentation scenarios.

Because of this, with XML, there is a full separation between data and presentation.

XML is Often a Complement to HTML

In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

XML Separates Data from HTML

When displaying data in HTML, you should not have to edit the HTML file when the data changes.

With XML, the data can be stored in separate XML files.

XML Documents Must Have a Root Element

XML documents must contain one **root** element that is the **parent** of all other elements:

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

Web - Server Types

Every Website sits on a computer known as a Web server. This server is always connected to the internet. Every Web server that is connected to the Internet is given a unique address made up of a series of four numbers between 0 and 255 separated by periods. For example, 68.178.157.132 or 68.122.35.127.

When you register a web address, also known as a domain name, such as tutorialspoint.com you have to specify the IP address of the Web server that will host the site. You can load up with Dedicated Servers that can support your web-based operations.

There are four leading web servers − Apache, IIS, lighttpd and Jagsaw.

Apache HTTP Server

This is the most popular web server in the world developed by the Apache Software Foundation. Apache web server is an open source software and can be installed on almost all operating systems including Linux, Unix, Windows, FreeBSD, Mac OS X and more. About 60% of the web server machines run the Apache Web Server.

You can have Apache with tomcat module to have JSP and J2EE related support.

Internet Information Services

The Internet Information Server (IIS) is a high performance Web Server from Microsoft. This web server runs on Windows NT/2000 and 2003 platforms (and may be on upcoming new Windows version also). IIS comes bundled with Windows NT/2000 and 2003; Because IIS is tightly integrated with the operating system so it is relatively easy to administer it.

Sun Java System Web Server

This web server from Sun Microsystems is suited for medium and large websites. Though the server is free it is not open source. It however, runs on Windows, Linux and Unix platforms. The Sun Java System web server supports various languages, scripts and technologies required for Web 2.0 such as JSP, Java Servlets, PHP, Perl, Python, Ruby on Rails, ASP and Coldfusion etc.

Jigsaw Server

Jigsaw (W3C's Server) comes from the World Wide Web Consortium. It is open source and free and can run on various platforms like Linux, Unix, Windows, Mac OS X Free BSD etc. Jigsaw has been written in Java and can run CGI scripts and PHP programs.

PERSONAL WEB SERVER:

Microsoft **Personal Web Server** (**PWS**) is a scaled-down **web server** software for Windows operating systems. It has fewer features than Microsoft's Internet Information Services (IIS) and its functions have been superseded by IIS and Visual Studio.

Microsoft Personal Web Server (**PWS**) is a scaled-down web server software for Windows operating systems. It has fewer features than Microsoft's Internet Information Services (IIS) and its functions have been superseded by IIS and Visual Studio. Microsoft officially supports PWS on Windows 95, Windows 98, Windows 98 SE, and Windows NT 4.0. Prior to the release of Windows 2000, PWS was available as a free download as well as included on the Windows distribution CDs. PWS 4.0 was the last version and it can be found on the Windows 98 CD and the Windows NT 4.0 Option Pack.

Personal Web Server was originally created by Vermeer Technologies, the same company which created Microsoft FrontPage, before they were acquired by Microsoft. It was installed by FrontPage versions 1.1 to 98 as well.

NT Workstation 4.0 shipped with **Peer Web Services**, which was based on IIS 2.0 and 3.0. With IIS 4.0, this was renamed to Personal Web Server to be consistent with the name used in 95/98.

Since Windows 2000, PWS was renamed to the same IIS name used in server versions of Windows as a standard Windows component. Windows Me and Windows XP Home Edition support neither PWS nor IIS, although PWS can be installed on Windows Me. In other editions of Windows XP, IIS is included as standard.

Before Microsoft Visual Studio 2005, PWS was useful in developing web applications on the localhost before deploying to a production web server. The IDE of Visual Studio 2005 (and later versions) now contains a built-in lightweight web server for such development purposes.