**UNIT-1**

**HTML, CSS & JAVA SCRIPT**

HTML is the standard markup language for creating Web pages.

**What is HTML?**

HTML stands for Hyper Text Markup Language

HTML is the standard markup language for creating Web pages

HTML describes the structure of a Web page

HTML consists of a series of elements

HTML elements tell the browser how to display the content

HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

**A Simple HTML Document**

<html>

<head>

<title>Page Title</title>

</head>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

**HTML Element**

An HTML element is defined by a start tag, some content, and an end tag:

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

Example: <h1>My First Heading</h1>

**HTML Attributes**

All HTML elements can have attributes.Attributes provide additional information about elements

Example: <a href="https://www.kmit.in">KMIT Web Site</a>

**Web Page Designing using HTML**

**Headings tag**: HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

**Paragraph tag**: The HTML <p> element defines a paragraph.

**Horizontal Ruler**: The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

**Style Attribute**: Setting the style of an HTML element, can be done with the style attribute.

<h1 style="color:blue;">This is a heading</h1>

**HTML Formatting Elements:**

<strong> - Important text

<em> - Emphasized text

<mark> - Marked text

<small> - Smaller text

<del> - Deleted text

<ins> - Inserted text

<sub> - Subscript text

<sup> - Superscript text

**HTML Comment Tags:** <!-- Write your comments here -->

**Hyperlinks**: You can click on a link and jump to another document.

Example: <a href="url">link text</a>

**Images**: Images can improve the design and the appearance of a web page.

Example: <img src="pic\_trulli.jpg" alt="Italian Trulli">

**Tables**: HTML tables allow web developers to arrange data into rows and columns. The <table> tag defines an HTML table. Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.

Example

<table style="width:100%">

<tr> <th>Firstname</th>

<th>Lastname</th>

<th>Age</th>

</tr>

</table>

**Lists**: HTML lists allow web developers to group a set of related items in lists.

Unordered List: starts with the <ul> tag. Each list item starts with the <li> tag.

**Example**

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

**Ordered List**: An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

**Example**

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

**Iframes**: An HTML iframe is used to display a web page within a web page.

Example:

<iframe src="demo\_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>

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

The <input> Element:

<input **type="text"**>: Displays a single-line text input field

<input **type="radio"**>: Displays a radio button (for selecting one of many choices)

<input **type="checkbox"**>: Displays a checkbox (for selecting zero or more of many choices)

<input **type="submit"**>: Displays a submit button (for submitting the form)

<input **type="button"**>: Displays a button.

**Select Element**: defines a drop-down list:

Example:

<select id="cars" name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option>

</select>

TextArea Element: The <textarea> element defines a multi-line input field (a text area):

Example:

<textarea name="message" rows="10" cols="30">

The cat was playing in the garden.

</textarea>

**HTML-5 CANVAS:**

The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.

The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Example

Creating the rectangular canvas area and adding JavaScript code to draw a line on it.

<script>

var c = document.getElementById("myCanvas");

var ctx = c.getContext("2d");

ctx.moveTo(0, 0);

ctx.lineTo(200, 100);

ctx.stroke();

</script>

**CSS (Cascading Style Sheets)**

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more.

CSS can be added to HTML documents in 3 ways:

**1. Inline** - by using the style attribute inside HTML elements

Example: <h1 style="color:blue;">A Blue Heading</h1>

**2. Internal** - by using a <style> element in the <head> section

<head><style>

p {color: red;}

</style></head>

<body>

<h1>This is a heading</h1>

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

</body>

**3. External Styles** - by using a <link> element to link to an external CSS file

s**tyles.css**

body { background-color: blue;}

h1 { color: blue;}

p { font-size: 30px;}

test.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>

**JAVA SCRIPT**

**JavaScript(LiveScript)**: JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

**Client-Side JavaScript**:

1. Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

2. It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

3. The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

4. The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

5. JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly. server side scripting.

**Datatypes**:

Numbers, String, Boolean, null, undefined and object.

**Variables**:

JavaScript variables are containers for storing data values. Declared with the var or let keyword.

Example:

var x = 5; var y = 6; var z = x + y;

Variables declared by var keyword are scoped to the immediate function body (hence the function scope) while let variables are scoped to the immediate enclosing block denoted by { } (hence the block scope).

**Rules for Java Script Variable Names**:

While naming your variables in JavaScript, keep the following rules in mind.

1. You should not use any of the JavaScript reserved keywords as a variable name.

2. JavaScript variable names should not start with a numeral (0-9). They must begin with a letter or an underscore character. For example, 123test is an invalid variable name but \_123test is a valid one.

3. JavaScript variable names are case-sensitive. For example, Name and name are two different variables.

**Operators:**  
1. Arithmetic Operators: **+, -, \*, /, %, ++, --**

2. Comparison Operators: **= =,** !=**, >, <, >=, <=**

3. Logical Operators: **&&** (Logical AND)**, ||** (Logical OR)**, !** (Logical NOT)

4. Bitwise Operators: **&** (Bitwise AND)**, |** (BitWise OR)**, ^** (Bitwise XOR)**, ~** (Bitwise Not)**, <<** (Left Shift)**, >>** (Right Shift) **and >>>** (Right shift with Zero)

5. Assignment Operators: **=** (Simple Assignment )**, += , −=,\*= , /= and %=**

6. Conditional Operator (? :)

7. typeof Operator: The typeof operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

**Expressions**

Any unit of code that can be evaluated to a value is an expression.

Since expressions produce values, they can appear anywhere in a program where JavaScript expects a value such as the arguments of a function invocation.

Example: 10+15

**Literals**:

JavaScript Literals are the fixed value that cannot be changed, you do not need to specify any type of keyword to write literals.

Literals are often used to initialize variables in programming, names of variables are string literals.

A JavaScript Literal can be a numeric, string, floating-point value, a boolean value or even an object.

In simple words, any value is literal, if you write a string "Studytonight" is a literal, any number like 7007 is a literal, etc.

**if statement**:

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

**Syntax:**

if (expression) {

Statement(s) to be executed if expression is true

}

**Switch Statement:**

The objective of a switch statement is to give an expression to evaluate and several different statements to execute based on the value of the expression. The interpreter checks each case against the value of the expression until a match is found. If nothing matches, a default condition will be used.

**Syntax**

switch (expression) {

case Label1: statement(s)

break;

case Label2: statement(s)

break;

default: statement(s)

}

**Iterative Statements:**

while Loop:

to execute a statement or code block repeatedly as long as an expression is true

**Syntax:**

while (expression) {

Statement(s) to be executed if expression is true

}

**do...while Loop**

**Syntax:**

do {

Statement(s) to be executed;

} while (expression);

**for Loop**

for (initialization; test condition; iteration statement) {

Statement(s) to be executed if test condition is true

}

**Java Script Functions:**

A function is a group of reusable code which can be called anywhere in your program.

This eliminates the need of writing the same code again and again.

It helps programmers in writing modular codes.

Functions allow a programmer to divide a big program into a number of small and manageable functions.

**Example:**

<html>

<head>

<script type = "text/javascript">

function sayHello(name, age) {

document.write (name + " is " + age + " years old.");

}

</script>

</head>

<body>

<script> sayHello('Shiva', 40)"</script>

</body>

</html>

**Convert temperature from Celsius to fahrenheit:**

<!DOCTYPE html>

<html>

<title>Celsius to Fahrenheit Temperature Converter</title>

<body>

<h2>Temperature Converter</h2>

<p>Type a value in the Celsius field to convert the value to Fahrenheit:</p>

<p>

<label>Celsius</label>

<input id="inputCelsius" type="number" placeholder="Celsius" oninput="temperatureConverter(this.value)" onchange="temperatureConverter(this.value)">

</p>

<p>Fahrenheit: <span id="outputFahrenheit"></span></p>

<script>

function temperatureConverter(valNum) {

valNum = parseFloat(valNum);

document.getElementById("outputFahrenheit").innerHTML=(valNum\*1.8)+32;

}

</script>

</body>

</html>

Output:

Temperature Converter

Type a value in the Celsius field to convert the value to Fahrenheit:

Celsius :1

Fahrenheit: 33.8

**Java Scripts Events**:

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

Example for Events: webpage page loads,user clicks a button, pressing any key, closing a window, resizing a window, etc.

**Common Events**

**1. onchange**: An HTML element has been changed

**2. onclick**: The user clicks an HTML element

**3. onmouseover**: The user moves the mouse over an HTML element

**4. onmouseout**: The user moves the mouse away from an HTML element

**5. onkeydown**: The user pushes a keyboard key

**6. onload**: The browser has finished loading the page

**onclick Event Example:**

<html> <head>

<script type = "text/javascript">

function sayHello() {

alert("Hello World")

}

</script> </head>

<body>

<form>

<input type = "button" onclick = "sayHello()" value = "Say Hello" />

</form> </body> </html>

**onsubmit Event Example:**

<html> <head>

<script type = "text/javascript">

function validation() { // all validation goes here }

</script> </head>

<body>

<form method = "POST" action = "ex1.jsp" onsubmit = "return validate()">

.......

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

</form>

</body> </html>

**onmouseover and onmouseout Events Example**:

<html> <head>

<script type = "text/javascript">

function over() { document.write ("Mouse Over"); }

function out() { document.write ("Mouse Out"); }

</script> </head>

<body>

<div onmouseover = "over()" onmouseout = "out()">

<h2> This is inside the division </h2>

</div>

</body> </html>

**JavaScript Objects**

**String Object**:

The String object lets you work with a series of characters; it wraps Javascript's string primitive data type with a number of helper methods.

**String Methods**

Sr.No. Method & Description

1.charAt(): Returns the character at the specified index.

2. concat(): Combines the text of two strings and returns a new string.

3. indexOf(): Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.

4. match(): Used to match a regular expression against a string.

5. search(): Executes the search for a match between a regular expression and a specified string.

6. split(): Splits a String object into an array of strings by separating the string into substrings.

7. substr(): Returns the characters in a string beginning at the specified location through the specified number of characters.

8. substring(): Returns the characters in a string between two indexes into the string.

9. toLowerCase(): Returns the calling string value converted to lower case.

10. toUpperCase():Returns the calling string value converted to uppercase.

**Array Object**

The Array object lets you store multiple values in a single variable.

It stores a fixed-size sequential collection of elements of the same type.

An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

Example: var fruits = new Array( "apple", "orange", "mango" );

**Array Methods:**

1. concat(): Returns a new array comprised of this array joined with other array(s) and/or value(s).

2. indexOf(): Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.

3. pop(): Removes the last element from an array and returns that element.

4. push(): Adds one or more elements to the end of an array and returns the new length of the array. array into a string.

5. reverse(): Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.

6. shift(): Removes the first element from an array and returns that element.

7. slice(): Extracts a section of an array and returns a new array.

8. sort(): Sorts the elements of an array

9. unshift(): Adds one or more elements to the front of an array and returns the new length of the array.

**Math Object:**

The math object provides you properties and methods for mathematical constants and functions. Unlike other global objects, Math is not a constructor.

All the properties and methods of Math are static and can be called by using Math as an object without creating it.

**Math Methods**

1. abs(): Returns the absolute value of a number.

2. ceil(): Returns the smallest integer greater than or equal to a number.

3. exp(): Returns EN, where N is the argument, and E is Euler's constant, the base of the natural logarithm.

4. floor(): Returns the largest integer less than or equal to a number.

5. log(): Returns the natural logarithm (base E) of a number.

6. max(): Returns the largest of zero or more numbers.

7. min(): Returns the smallest of zero or more numbers.

8. pow(): Returns base to the exponent power, that is, base exponent.

9. random(): Returns a pseudo-random number between 0 and 1.

10. round(): Returns the value of a number rounded to the nearest integer.

11. sin(): Returns the sine of a number.

12. tan(): Returns the tangent of a number.

13. sqrt(): Returns the square root of a number.

14. cos(): Returns the cosine of a number.

**Date Object:**

The Date object is a datatype built into the JavaScript language.

Date objects are created with the new Date( ).

**Date Methods**

1.Date(): Returns today's date and time

2. getDate(): Returns the day of the month for the specified date according to local time.

3. getDay(): Returns the day of the week for the specified date according to local time.

4. getFullYear(): Returns the year of the specified date according to local time.

5. getHours(): Returns the hour in the specified date according to local time.

6. getMilliseconds(): Returns the milliseconds in the specified date according to local time.

7. getMinutes(): Returns the minutes in the specified date according to local time.

8. getMonth(): Returns the month in the specified date according to local time.

9. getSeconds(): Returns the seconds in the specified date according to local time.

**Browser Object Model**

The Browser Object Model (BOM) allows JavaScript to "talk to" the browser.



**Window Object**:

top level object in the object hierarchy.

Represents entire browser window.

**Properties of Window Object**

**1. document**: contains info about the current document displyed in the window.

**2. history**: contains the urls visited so far by users in this window.

**3. location**: an object that contains info about the current url

**4. frames**: array containing all frames in the current window

**5. length**: the number of frames in the window

**6. navigator**: an object that contains info about browser application

**7. screen**: refers to the screen object associated with the window.

**8. ScreenLeft, ScreenRight**: x,y coordinates respectively of the window relative to users monitor screen. These properties are specific to the internet explorer.

**9. self**: this object is synonym for the current window.

**10. parent**: a reference to the parent window whose frameset contains the current frame.

**11. status**: it contains the message of the status bar of the windows.

**Methods of window object**

**1. alert(msg)**: displays the message in alert window

2. prompt(msg,default value): used to accept the data from the user.

**3. close()**: close the browser window on which it is invoked.

**4. moveBy(dx,dy)**: changes the position of the specified window by a specifed number of pixels.

It adds dx to screenLeft and dy to screenTop.

**5. moveTo(x,y)**: this method sets the (left,top) corner of the specified window to the specified coordinates x and y.

**6. resizeBy(dx,dy)**: resizes the context window by the specified amount in pixels in the horizontal an vertical directions.

**7. resizeTo(x,y)**: resizes the context window to the specifed pixel vaues.

**document object**:

Represents the display area of the web page

**Methods of document Object**

1. document.write()

2. document.close()

**properties of document Object**

**1. bgcolor**: indicates the background color of the document

**2. fgcolor:** indicates foreground color(color of text in the document)

**3. anchors[]:** an array containg references to all named anchors

**4. forms[]:** an array of the form objects contained in the current document

**5. images[]:** an array containing references to all images object in the current document

**6. domain:** indicates the domain name of the server from which the document originated

**7. link[]**: an array containg referenes to all the links

**8. title:** the title of the document as defined between <title></title>

**9. url**: the url of the current document object

**Form Validations**

**validation for username**

var u=form1.text\_username.value;

if (u.length==0) alert('user name cant be empty');

else if(u.length<6) alert('user name should be atleast 6 characters');

else {

var re1=/^\w+$/;

if(!u.match(re1)) alert('invalid user name'); }

**validation for contact number**

var c=form1.text\_contactno.value;

if (c.length==0) alert('plese enter the contact number');

else if(c.length!=10) alert('phone no should be exactly 10 digits');

else { var re3=/^[\d]{10}$/;

if (!c.match(re3)) alert('invalid phone number'); }

**validation for mail id**

var e=form1.text\_mailid.value;

if (e.length==0) alert('plese enter your mail id');

else {

// example : shiva@kmit.in then /^\w+@\w+\.\w{2,3}$/

// example : shiva@yahoo.co.in then /^\w+@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

// example : shiva-rama-krishna@yahoo.co.in

var re4 = /^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$/;

if (!e.match(re4))

alert('invalid mail id');

}