Paper Code: CSCS351 # WEB TECHNOLOGY

MODULE – IV Introduction to Java script - script tag, interactive data, DOM, A simple document, Add a form, Add a text input element, Add a button element, properties, methods and event handlers. Scripts and HTML.

Introduction to JavaScript

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

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.

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.

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.

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.

Example:

```
window.alert("Hello World!");
```

We can execute this program by referencing a file containing it from a script element within an HTML document.

Advantages of JavaScript

The merits of using JavaScript are −

Less server interaction − You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

Immediate feedback to the visitors − They don't have to wait for a page reload to see if they have forgotten to enter something.

Increased interactivity − You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

Richer interfaces − You can use JavaScript to include such items as drag-and-drop components and sliders to give a Rich Interface to your site visitors.

Limitations of JavaScript

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features −

Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.

JavaScript cannot be used for networking applications because there is no such support available.

JavaScript doesn't have any multi-threading or multiprocessor capabilities.

Variables and Data Types

As previously mentioned, variables do not have data types in JavaScript. However, every variable has a value, and every value belongs to one of six JavaScript data types. Every numeric value is of type Number, string values are of type String, the literals true and false represent the two Boolean values, the literal null represents the one value of type Null, and every object is of type Object.

Data Types

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

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

The Concept of Data Types

In programming, data types are an important concept. To be able to operate on variables, it is important to know something about the type.

Statements

There are different kinds of statement is present in java script.

Example

Assignment Statement

Let a=20

Output statement

Document.write("Welcome")

Alert("Hai")

Looping Statements

For loop, while loop

OPERATORS

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic on numbers:

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

Assignment operator =

```
let x = 10;
x += 5;
```

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 literal or constant

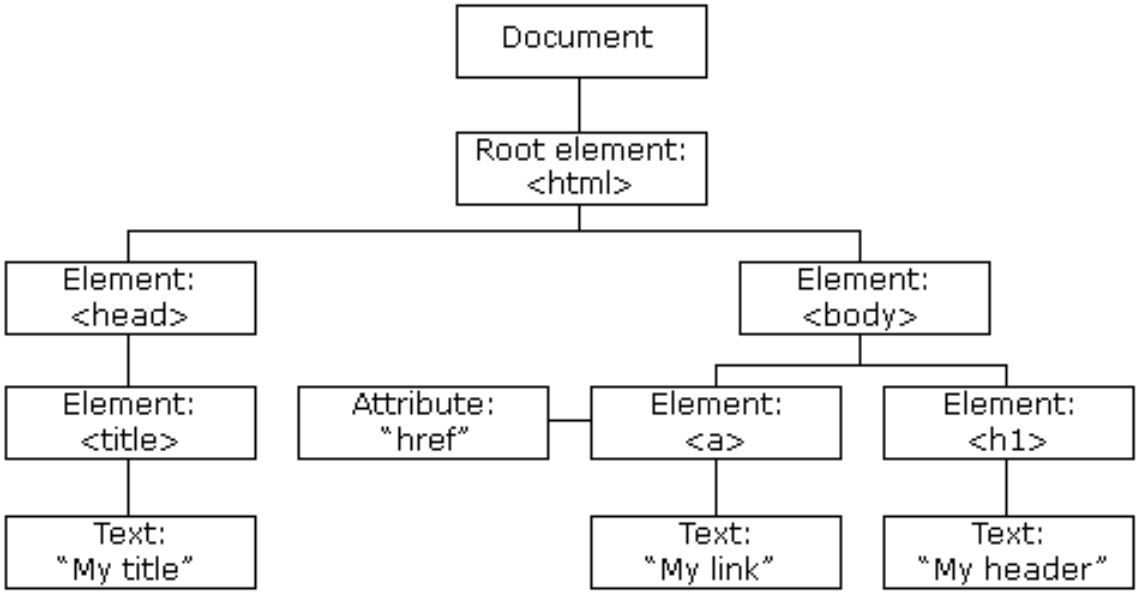
JavaScript Literals are constant values that can be assigned to the variables that are called literals or constants. JavaScript Literals are syntactic representations for different types of data like numeric, string, Boolean, array, etc data. ... The literal "john" represents, the value john for the variable name.

The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



HTML Forms

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

he <form> Element

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, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

All the different form elements are covered in this chapter: HTML Form Elements.

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.

Here are some examples:

Type	Description
<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 clickable button

Text Fields

The <input type="text"> defines a single-line input field for text input.

Example

A form with input fields for text:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

Finding HTML Elements

When you want to access HTML elements with JavaScript, you have to find the elements first.

There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name
- Finding HTML elements by CSS selectors
- Finding HTML elements by HTML object collections

```
var myElement = document.getElementById("intro");
var x = document.getElementsByTagName("p");
var y = document.getElementsByName("sam");
```

Event Handling

An HTML event can be something the browser does, or something a user does.

Here are some examples of HTML events:
An HTML web page has finished loading
An HTML input field was changed
An HTML button was clicked
Often, when events happen, you may want to do something.
JavaScript lets you execute code when events are detected.

HTML allows event handler attributes, with JavaScript code, to be added to HTML elements.

Example of onclick event

```
<html>
<body>
<p>Please input a number between 1 and 10:</p>
<input id="numb">
<button type="button" onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>
function myFunction() {
  var x, msg;
  x = document.getElementById("numb").value;
  if (isNaN(x) || x < 1 || x > 10) {
    msg= "Input not valid";
  } else {
    msg = "Input OK";
  }
  document.getElementById("demo").innerHTML = msg;
}
</script>
</body>
</html>
```

Javascript Example

A 45
B 78
ADD
C 123

```
<!-- Page6.htm  Multiplication Table-->
<html>
<script>
function multable()
{
  var n,i,z;
  n=parseInt(document.getElementById("t1").value);
  for(i=1;i<=20;i++)
  {
    z=i*n;
```

```
    document.write(i+" x "+n+" = "+z+"<br>");

  }
}
</script>
<body>
<h2>Multiplicaton Table</h2>
A<input type=text id=t1><br>
<input type=button value="OK" onclick="multable()"><br>
</body>
</html>
```

Multiplicaton Table

A 7

OK

1 x 7 = 7
2 x 7 = 14
3 x 7 = 21
4 x 7 = 28
5 x 7 = 35
6 x 7 = 42
7 x 7 = 49
8 x 7 = 56
9 x 7 = 63
10 x 7 = 70
11 x 7 = 77
12 x 7 = 84
13 x 7 = 91
14 x 7 = 98
15 x 7 = 105
16 x 7 = 112
17 x 7 = 119
18 x 7 = 126
19 x 7 = 133
20 x 7 = 140