JavaScript

HTML - displays the content on webpage.

CSS - Styles the webpage.

JavaScript - Validates & takes action.

(1) Client Side gives valid & invalid Data.

(2) Server Side validates & take necessary action.

(3) If the valid data is entered server side ask to database for required data.

(4) If the invalid data is entered, sever side takes necessary action.

Without JavaScript all the validation take place in server side which increase the load of the server, because of which the site may be crashed. JavaScript is a frontend technology, used to validate the data, also helps to reduce the server's load. Because of JavaScript all the validation and action take place in client side only.

JavaScript:

(1) We are sending only valid data to server side.

(2) Using JavaScript the load of server will get reduce.

(3) Validation and action we are performing at client side only.

Linking:

Internal JavaScript is used <script> </script> inside <head> </head>

->External JavaScript file is saved using ".js" extension.

To link external js file, the following syntax will be helpful:

<head> <script src = "Demo.js"> </script> </head>

Linking

External Cascading Style Sheet is saved using ".css" extension.

To link external css file, the following syntax will be helpful:

<head> <link href = "style.css" rel = "stylesheet"> </head>

In JavaScript webpage is known as document.

In JavaScript we can code in scripting, procedural and object paradigm style.

To comment:

// for single line comment

/\* for multiline comment \*/

Variables and datatypes in JavaScript

JavaScript is dynamically typed programming language.

In JavaScript datatypes will get assign to the variables during runtime by language interpreter.

(1) Number: To store the real and integer numbers.

(2) String: To store the character or strings. We can fetch the character using index position.

(3) Boolean: To store true or false.

(4) Object: To refer the objects.

(5) Undefined: datatypes to those variables which is not initialise but defined using var or let keyword

(6) Array:

Variable Declaration Keywords

In js, datatypes are not required to declare variables, but we have to use keywords (var, let and const) as per requirements, which describe the scope of variables.

var keyword: Whenever we declare any variable using keyword "var", then that variable does not have scope only inside the block. Re-declaration of any variable using var keyword inside same block is allowed.

let keyword: We can't re-declare variable using "let" keyword in a same block i.e., if it is declared globally then another time it can't be re-declared anywhere globally but it can be re-declared once inside the different blocks.

Variables declared using let can't be accessible outside the block's scope.

const keyword: We can't re-declare variable using "const" keyword in same block i.e., if it id declared globally then another time it can't be re-declared anywhere globally but it can be re-declared once inside the different blocks.

Redefinition is not allowed for the variables that is declared "const". Const keyword makes it constant.

Variable declared using const can't be accessible outside the block's scope.

Non defined variable can't be declared using const keyword.

console.log() is used to see the error in the console tab present inside inspection option of the browser.

typeof() is used to check the datatype of variable.

Semicolon (;) is option in JavaScript.

Typecasting: is a mechanism, which is used to convert one datatype into another type.

Number(), String(), Boolean(), Array() and Object() are the methods which are used in typecasting.

(1) Number can be casted into Number, String, Boolean and Object but not to undefined and Array.

(2) String can be casted into Number, String, Boolean, Object, Array but not to undefined. If String contains real or integer value only not other character then it works perfectly otherwise it stores NaN.

\*\*\* NaN stands for Not a Number.

(3) Boolean can be casted into Number, String, Boolean, Object, Array but not to undefined.

(4) Array can be casted into Number, String, Boolean, Object and Array but not to undefined. If array contains only single element as real or integer then it works perfectly at the time of typecasting from array to number otherwise it stores NaN.

(5) Undefined can be casted into Number, String, Boolean, Object and Array also.

Array in JavaScript:

Array in JS can store homogeneous & heterogeneous type of data. We can access data from array using index.

Declaration of Array

var n = [];

var n = new Array();

Declare and Define Array

var n = [1, "Akhilesh", true];

var n = new Array(1, "Akhilesh", true);

Fetching data from array

document.write(n); Display all the elements

console.log(n); Same functionality

console.log(n[2]); fetch the element present at second index

Add element into array

n[index] = element;

n.push(element); at last index and return the new length of array

n.unshift(element); append at the 0th index of an array

Delete element from array

n.pop(); removes the last element from an array

n.shift(); removes elements from the 0th index of an array

splice() is used to delete also to add element

splice(start\_index, no\_of\_element); deletes no of elements from specified position

splice(start-index, no\_of\_element, elements\_seperating\_using\_comma); it first deletes no of elements from specified position then from the specified position it will start to add the element and if the element already present at last of array then it will shift them to new indexes.

splice() return the elements which has been deleted from the array.

Operators

1. Arithmetic Operator: + for addition, - for subtraction, \* for multiplication, / for division, % for remainder

2. Relation Operator: < for less than, > for greater than, == for equal to, <= for less than equal to, >= for greater than equal to, != not equal to

3. Logical Operator: && for AND, || for OR

4. Ternary Operator: (condition) ? Exp1: Exp2;

Variable\_name = (condition) ? Exp1: Exp2;

5. String Operator: + for concatenation

Object Creation

In JS we can create an object using key: value pair inside the curly braces separated by using comma. Creating an Object using key: value pair, this form of creating an object is called as “JSON format”. JSON stands for JavaScript Object Notation.

Example: var dog = {‘name’: “puppy”, ‘age’: 22}

Adding the key: value pair in an object

dog[‘bread’] = ‘sheru’;

Accessing the values by using key

console.log(dog[‘age’]);

console.log(dog.name);

Looping Control Construct

1. for-loop: Used to iterate over an iterable objects.
2. for-of-loop: Used to iterate over an iterable objects.
3. for-in-loop: Used to iterate over the non-iterable objects.

for-loop

var a = [10, 20, 30, 40];

for(var i=0; i<a.length; i++)

{

console.log(a[i]);

}

for-of-loop

var a = [10, 20, 30, 40];

for(var i of b)

{

console.log(i);

}

for-in-loop

var fan = { ‘brand’: ‘bajaj’, ‘wings’: 4, ‘color’: ‘white’}

for(var k in fan)

{

console.log(k); //key

console.log(fan.k); //value  
}

console.log(fan); // to get complete key: value pair

Functions in JS

Types of function

1. No input and No output
2. Input and No output
3. No input but output
4. Input with Output

Three ways to create function in JS:

1. Function as Declaration
2. Function as Expression
3. Arrow Function

Function as Declaration

Syntax:

function fun\_name()

{

// body of function

}

function\_name() //calling

Example of type 1 function

var a = 10;

var b = 20;

function addition(a, b)

{

var c = a + b;

console.log(c);

}

addition(a, b)

Example of type 2 function

function addition()

{

var a = 10;

var b = 20;

var c = a + b;

console.log(c);

}

addition()

Example of type 3 function

function addition()

{

var a = 10;

var b = 20;

var c = a + b;

return c;

}

var c = addition()

console.log(c);

Example of type 4 Function

var a = 10;

var b = 20;

function addition(a, b)

{

return a + b;

}

var c = addition(a, b);

console.log(c);

Function as Declaration

Syntax

function\_name = function()

{

//body of function

}

function\_name() //calling

Example

addition = function(a, b)

{

return a + b;

}

console.log(addition(100, 399));

Arrow Function:

Syntax:

function\_name = () =>

{

//body of function

}

function\_name() //calling

Example

addition = (a, b)=>

{

return a + b;

}

console.log(addition(100, 399));

Pop Up Message

Here we use inbuilt method to create dialog box or popup windows in JavaScript

1. alert(“message…”) is used to give warning message. We will get one option – “OK” in popup window. Return undefined.
2. confirm(“message…”) is used to display confirmation message. We will get two option – “OK” and “Cancel”. Return true for Ok and false for Cancel.
3. prompt(“message…”) is used to take input from the user in the popup message. Return the input entered in text box.

Example of calculator using prompt()

choice = prompt("Enter your choice \n 1:: Add \n 2:: Subtract \n 3:: Multiply \n 4:: Divide");

choice = Number(choice); // Converting string into number

num1 = Number(prompt("Number1: "));

num2 = Number(prompt("Number2: "));

switch(choice)

{

    case 1:

        console.log(num1+num2);

        break;

    case 2:

        console.log(num1-num2);

        break;

    case 3:

        console.log(num1\*num2);

        break;

    case 4:

        console.log(num1/num2);

        break

}

Events

1. Onclick
2. OnMouseover
3. Onchange
4. Onmouseout

Example of onclick attribute

! !DOCTYPE html>

<html>

    <head>

        <title> JavaScript </title>

        <script>

            function fun()

            {

                console.log(alert("Alert..."));

            }

        </script>

    </head>

    <body>

        <p>Learning Events in html...</p>

        <form>

            <label> Name: </label>

            <input type="text"/>

            <input type="submit" onclick="fun()"/>

        </form>

    </body>

</html>

As we clicked to submit, it call the fun() defined inside validation.js file

Example of onmouseover attribute:

<input type="submit" onmouseover="fun()"/>

As we mouse cursor move over submit, it call the fun() defined inside validation.js file

Example of onmouseout attribute:

<input type="submit" onmouseout="fun()"/>

As we mouse cursor move outside the submit, it call the fun() defined inside validation.js file

Sending and Receiving data from html element to JavaScript

Example of onchange attribute:

<!DOCTYPE html>

<html>

    <head>

        <title> JavaScript </title>

        <script>

            function fun1()

            {

                // taking input from the html element to js

                var res = document.getElementById("one").value

                // showing the input, it would not work for console.log()

                document.write("I like " + res);

                // showing and sending would give error

                // sending from js to html

                document.getElementById("two").innerHTML="I like "+res

            }

        </script>

    </head>

    <body>

        <p>Learning Events in html...</p>

        <form>

            <select id="one" onchange="fun1()">

                <option> Red </option>

                <option> Blue </option>

                <option> Green </option>

            </select>

            <h4 id="two" style="color: blueviolet; font-family: cursive;"> </h4>

        </form>

    </body>

</html>

Regular Expression:

<html>

    <head>

        <title>Regular Expression</title>

        <script>

            var fun=function()

            {

                var str = "Technology";

                // Write regular expression

                // for the character "t" in the given string

                var res = str.search("T")

                console.log(res)    //0

                // replace with character c

                // will replace only first occurance

                var res2 = str.replace("o", "C")

                console.log(res2)

                // ignore the case use i quantifier

                // globally changes use g quantifer

                // pass the pattern inside // forward slash pair

                //instead of ""

                var res3 = str.replace(/t/ig,"c")

                console.log(res3)

            }

        </script>

    </head>

    <body>

        <h1 id = "one" onmouseover="fun()"> Technologies </h1>

    </body>

</html>

<!DOCTYPE html>

<html>

    <head>

        <title> Regular Expression </title>

    </head>

    <body>

        <h1 id = "one"></h1>

        <script>

            var reg = function()

            {

                var str = "Technologiesol"

                var res = str.match(/ol\*/g) //ol,o,ol

                //match the substring as well as first character of the sub string

                var res = str.match(/ol+/g) // ol,ol

                //mathces the exact substring

                document.getElementById("one").innerHTML=res

            }

            reg();

        </script>

    </body>

</html>

Taking action against valid and invalid phone number

<html>

    <head>

        <title> Validation </title>

        <script>

            function validate()

            {

                var res = document.getElementById("Phone").value

                var bool = /^[7-9]{1}[0-9]{9}$/.test(res)

                if(res.length == 0)

                {

                    document.getElementById("msg").innerHTML="Field is empty"

                    document.getElementById("Phone").style.border="5px solid red"

                    return false // since by default any function return

                    // which is called using onsubmit, onchange etc. attributes

                    // because of which action gets executed so we set it as false

                }

                else if(res.length != 10)

                {

                    document.getElementById("msg").innerHTML="Only 10 digit valid number"

                    document.getElementById("Phone").style.border="5px solid red"

                    return false

                }

                else if(bool == false)

                {

                    document.getElementById("msg").innerHTML="Only 10 digit valid number"

                    document.getElementById("Phone").style.border="5px solid red"

                    return false

                }

            }

        </script>

    </head>

    <body>

        <form onsubmit="return validate()" action="kodnest.com">

            <label> Phone </label>

            <input type="text" id="Phone">

            <span style="color: red" id="msg"> \* </span>

            <input type="Submit">

        </form>

        <!-- Since document.getElementByID().value was not

        working properly when we have linked it head tag

        so we have used this tag here by which we are getting

        desired output and input-->

        <!--

            <script src = "validate.js"> </script>

        -->

    </body>

</html>

* isNaN() is a method to check whether the given string has only number or not.

Classes

* In JavaScript objects can be created using class or object-oriented style
* To create a constructor inside the class “constructor()” is used.
* Execution of the code starts from such statement which is completely outside the class
* Instance variables or object variables, are created inside the constructor.
* Keyword new is used to create an object.
* JavaScript supports few of the object-oriented concepts like class, object and inheritance.

Example

<html>

    <head>

        <title>

            Class

        </title>

        <script>

            class Fan

            {

                constructor(name, wings)

                {

                    this.name = name

                    this.wings = wings

                }

                rotate()

                {

                    console.log("Fan is rotating")

                }

            }

            var f1 = new Fan("Bajaj", 3)

            console.log(f1)

            f1.rotate()

        </script>

    </head>

    <body>

    </body>

</html>

Another way of creating object

<!DOCTYPE html>

<html>

    <head>

        <title> OOPs: Object of function</title>

    </head>

    <body>

        <script>

            var fname

            var lname

            var fun = function(fname, lname)

            {

                this.fname = fname;

                this.lname = lname;

            }

            var person = new fun("Akash", "Pandey")

            console.log(person) //fun {fname: "Akash", name: "Pandey"}

            console.log(typeof(person)) //object

            console.log(person.fname + " " + person.lname) // Akash Pandey

        </script>

    </body>

</html>

Inheritance

Acquiring property from the base class. Example of Inheritance:

<html>

    <head>

        <title>

            Class

        </title>

        <script>

            class Parent

            {

                constructor(work)

                {

                    this.work = work;

                }

                parent\_working()

                {

                    return "Parent is working"

                }

            }

            class Child extends Parent

            {

                constructor(a, b)

                {

                    super(a)

                    this.b=b

                }

                child\_working()

                {

                    return this.parent\_working() + " " + "Child is working"

                }

            }

            var obj = new Child(1, 3)

            console.log(obj)    // Child {work: 1, b: 3}

            console.log(obj.child\_working()); //Parent is working Child is working

        </script>

    </head>

    <body>

    </body>

</html>

Exception handling:

<!DOCTYPE html>

<html>

    <head>

        <title>

            Exception Handling

        </title>

        <script>

            var fun=()=>

            {

                var strvalue = "NO\_EXCEPTION\_OCCURED"

                try

                {

                    var str = strvalue.toLower()

                    alert(str)

                }

                catch(ex)

                {

                    alert("Exception occured")

                }

                finally

                {

                    alert("Inside the finally block")

                }

            }

            fun()

        </script>

    </head>

    <body>

    </body>

</html>

Variable Hoisting

<!DOCTYPE html>

<html>

    <head>

        <title>

            Variable Hoisting

        </title>

        <script>

            /\* Without use strict

            b = 10

            console.log(b) //10 \*/

            // Variable hoisting fives strict mode to use

            // so that without declaration of variable

            // we can not initialise it using var, let or const

            "use strict"

            var b = 10

            console.log(b) // 10

            a = 10

            console.log(a) // a is not defined

        </script>

    </head>

    <body></body>

</html>

Browser Object Model (BOM)

It is used to manipulate browser with the help of different browser model properties and methods using JavaScript

How to get height and width of browser…

<!DOCTYPE html>

<html>

    <head>

        <title> Browser Object Model</title>

    </head>

    <body>

        <h3 id="one" style="color: blue"></h3>

        <h3 id="two" style="color: blue"></h3>

        <button onclick="win\_open()">OPEN</button>

        <button onclick="win\_close()">CLOSE</button>

        <!--

            Linking of external script of defining of internal

            script should be at the end of body tag so it is

            able to detect the browser's dimension otherwise not

        -->

        <script>

            document.getElementById("one").innerHTML="Height of Browser " + screen.height // inbuilt property

            document.getElementById("two").innerHTML="Width of Browser " + screen.width // inbuilt property

            var win\_open=function()

            {

                window.open("https://www.google.com") // inbuilt method

            }   // to open the input in new tab

            var win\_close=function()

            {   // Scripts may close only the windows that were opened by them.

                window.close()

            }

        </script>

    </body>

</html>

Document Object Model (DOM)

Document Object Model: HTML DOM defines a standard way for accessing and manipulating html document. It does not specify any relationship between objects.

1. <head>
   1. <style>
   2. <script>
   3. Other external packages
2. <body>
   1. <form>
      1. <select>
         1. <option>
      2. <checkbox>
      3. <input>
      4. Etc.
   2. <table>
      1. <td>
      2. <tr>
      3. <th>
   3. <img>
   4. Etc.

Animation in JS

<!DOCTYPE html>

<html>

    <head>

        <title> Animation in JS</title>

        <style>

            #box{

                border: 15px groove green;

                height: 400px;

                width: 400px;

                background-color: aqua;

            }

            #circle{

                border: 5px solid red;

                height: 50px;

                width: 50px;

                background-color: brown;

                border-radius: 50px;

                position: relative;

            }

        </style>

    </head>

    <body>

        <div id="box">

            <div id="circle"></div>

        </div>

        <br> <br>

        <button onclick="animation()">Click Me</button>

        <script>

            function animation()

            {

                var circle = document.getElementById("circle") // not value rather element

                var circle\_px = 0

                var ani\_start = setInterval(fun, 10) // setInterval() is inbuilt method

                // which accept a function and duration

                function fun()

                {

                    if(circle\_px==340)

                    {

                        clearInterval(ani\_start)

                    }

                    else

                    {

                        circle\_px++

                        circle.style.top = circle\_px+"px"

                        circle.style.left = circle\_px+"px"

                    }

                }

            }

        </script>

        <!--

            The setInterval() method calls a function at specified interval (in ms).

            The setInterval() method continues calling the function unitl clearInterval() is called.

            or the window is closed

            clearInterval() will stop the setInterval()

        -->

    </body>

</html>

Cookies: Cookies is an amount of information stored in the browser during the time of browsing.

<!DOCTYPE html>

<html>

    <head>

        <title> Cookies </title>

    </head>

    <body>

        <button onclick="getcookie()">GetCookie</button>

    </body>

    <script>

        function setcookie()

        {

            document.cookie = "All the best ..."

        }

        function getcookie()

        {

            if(document.cookie==0)

            {

                alert("cookie is not present")

            }

            else

            {

                alert("Yes... Cookie is present")

            }

        }

    </script>

</html>

MediaQuery

When the pixel (px) are increasing or decreasing based on different systems the media query will adjust the styling properties.

<!DOCTYPE html>

<html>

    <head>

        <title> Media Query </title>

        <style>

            h1{ color: blue;}

            @media(max-width: 200px)

            {

            h1{color: red} /\*On getting 200px or less width\*/

            }

        </style>

    </head>

    <body>

        <h1>Welcome to my webpage</h1>

    </body>

</html>