SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | Approved by AICTE

CUSTOMER INTERFACE DESIGN AND DEVELOPMENT

COURSE INCHARGE: E.BRUMANCIA

Dr.R.M.GOMATHI

COURSE CODE: SITA1502

COURSE OUTCOMES

On completion of the course, student will be able to

CO1 - Able to work with XML technologies.

CO2 - Design web page to perform form validation using client-side scripting language.

CO3 - Implement new technologies such as Angular JS & jQuery.

CO4 - Develop web applications using server-side scripting language.

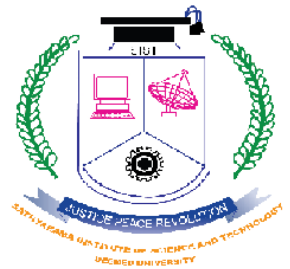
CO5 - Understand the differences between usability and user experience.

CO6 - Effectively select and utilize design thinking processes and UX/UI tools.

UNIT 2 CLIENT SIDE SCRIPTING

- Java Script
  - Advantages
  - Data types
  - Variables
  - Operators
  - Control statements
  - Functions
  - Objects and arrays
  - Windows and frames – Forms.
- AJAX
  - XMLHttp Request (XHR)
  - Create Object
  - Request
  - Response
  - Ready state.

What is Java Script?



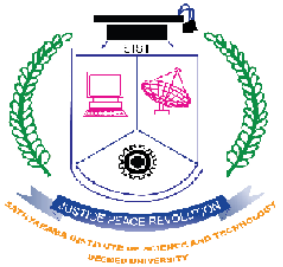
- JavaScript is a dynamic computer programming language.
- It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.
- It is an interpreted programming language with object-oriented capabilities.

History

- JavaScript was first known as LiveScript.
- Netscape changed its name to JavaScript.
- JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript.

- JavaScript is a lightweight, interpreted programming language.
- Designed for creating network-centric applications.
- Complementary to and integrated with Java.
- Complementary to and integrated with HTML.
- Open and cross-platform.

Client-Side JavaScript

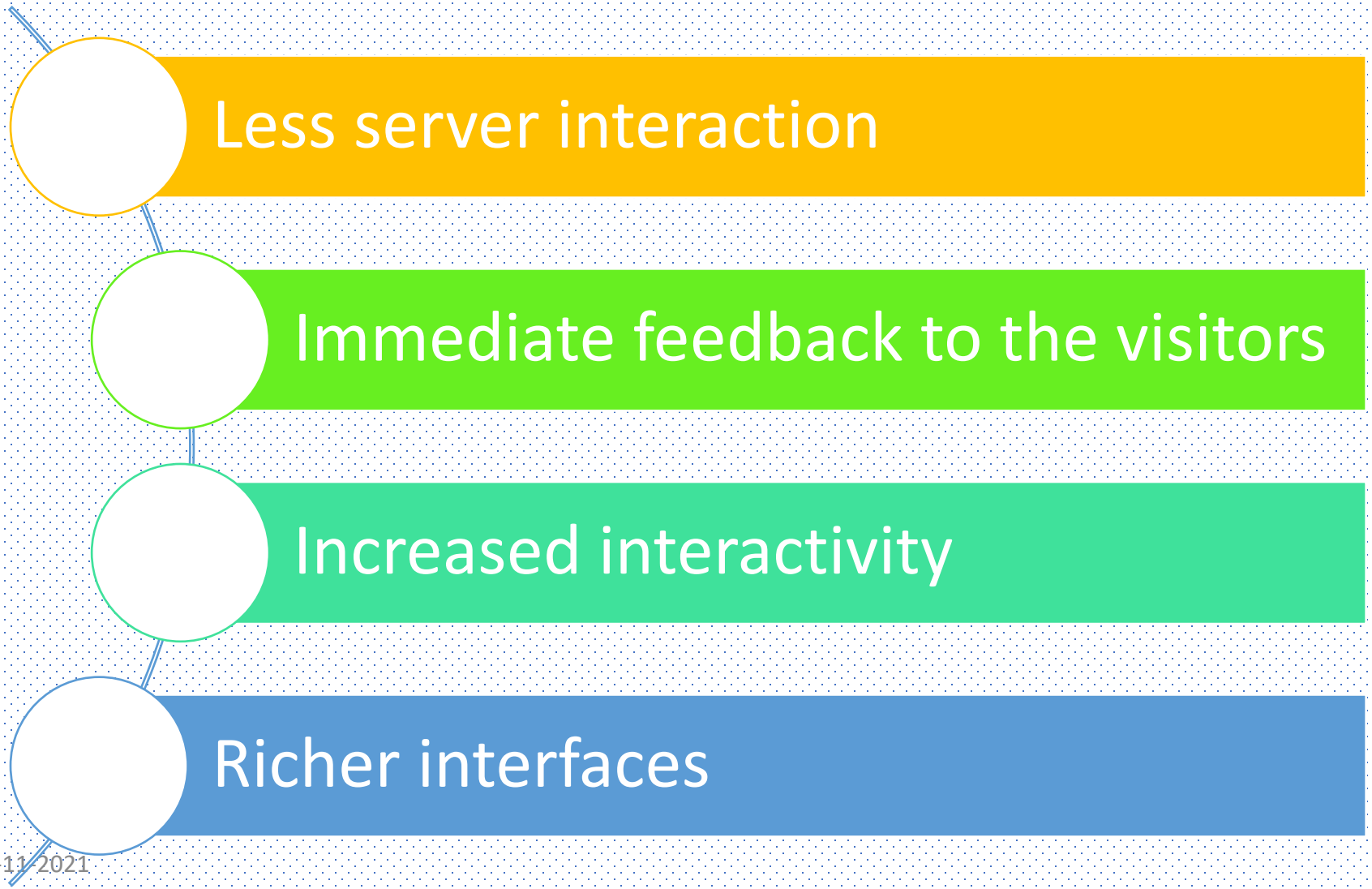


- Client-side JavaScript is the most common form of the language.
- The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.
- It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.
- The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts.

Example :

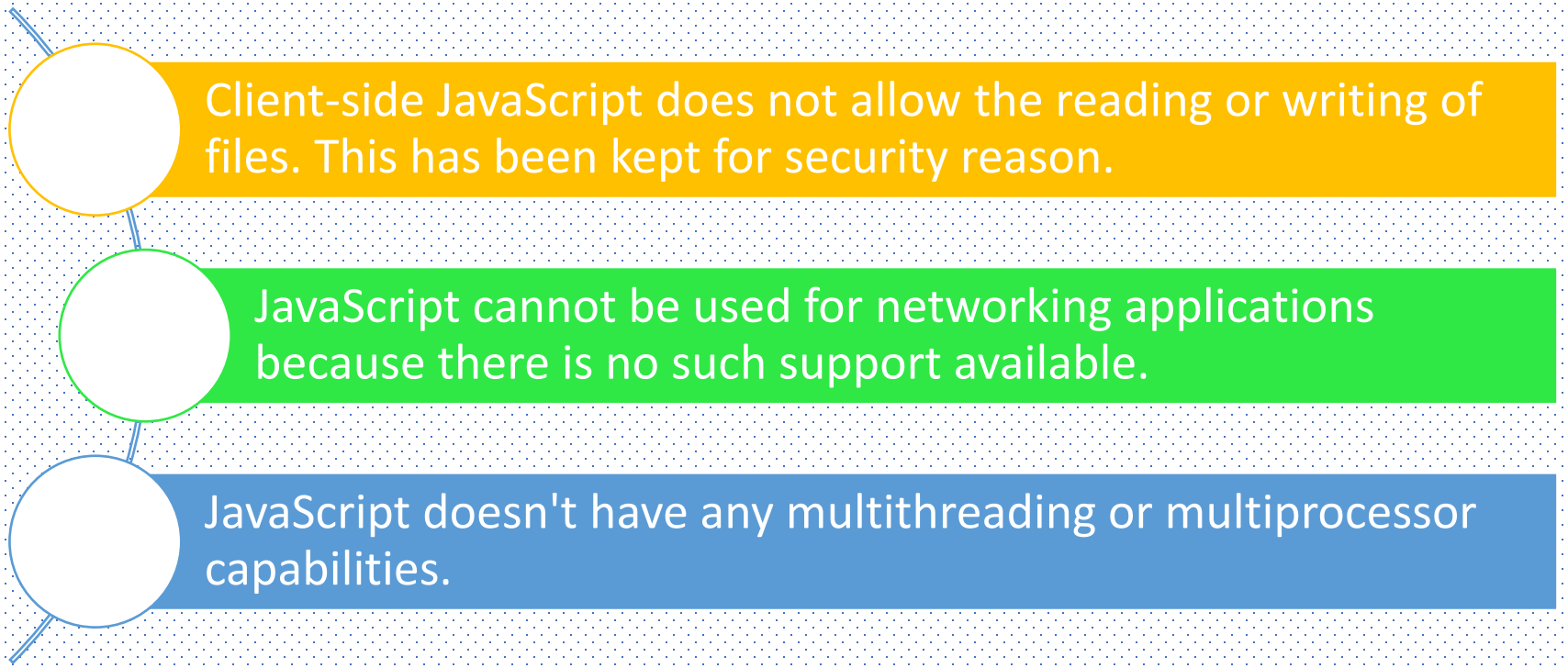
Registration form validation

Advantages of JavaScript



Limitations of JavaScript

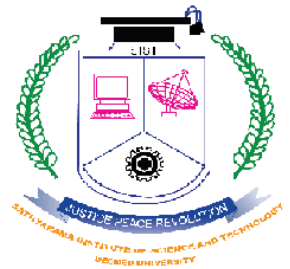
- Cannot treat JavaScript as a full-fledged programming language.



JavaScript Development Tools

- **Microsoft FrontPage:**
  - It provides web developers with a number of JavaScript tools to assist in the creation of interactive websites.
- **Macromedia Dreamweaver MX:**
  - Macromedia Dreamweaver MX is a very popular HTML and JavaScript editor in the professional web development crowd.
  - It provides several handy prebuilt JavaScript components, integrates well with databases, and conforms to new standards such as XHTML and XML.
- **Macromedia HomeSite 5:**
  - HomeSite 5 is a well-liked HTML and JavaScript editor from Macromedia that can be used to manage personal websites effectively.

Using JavaScript in your HTML



```
<script language="javascript" type="text/javascript">
        JavaScript code
</script>
```

The script tag takes two important attributes:

- **Language:** This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type:** This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

Your First JavaScriptCode

```
<html>
        <head>
                <title>Hello World in JavaScript</title>
        </head>
        <body>
                <script language=“javaScript” type="text/javascript">
                        document.write("Hello World!");
                </script>
        </body>
</html
```

- Whitespace and Line Breaks.
- Semicolons are Optional
- Case Sensitivity
- Comments in JavaScript

Where to Put your Scripts

- You can have any number of scripts
- Scripts can be placed in the HEAD or in the BODY – In the HEAD, scripts are run before the page is displayed – In the BODY, scripts are run as the page is displayed
- In the HEAD is the right place to define functions and variables that are used by scripts within the BODY

- JavaScript in <head>…</head> Section
- JavaScript in <body>…</body> Section
- JavaScript in <body> and <head> Sections

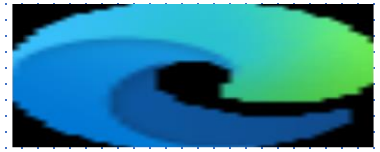
JavaScript in <head>...</head> Section

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
 alert("Hello World")
}
//-->
</script>
</head>
<body>
Click here for the result
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

ex3.html

JavaScript in <body>…</body> Section

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<p>This is web page body </p>
</body>
</html>
```



ex4.html

JavaScript in <body> and <head> Sections

```
<html>
<head>
<script type="text/javascript">
<!--
function sayHello() {
 alert("Hello World")
}
//-->
</script>
</head>
<body>
<script type="text/javascript">
<!--
document.write("Hello World")
//-->
</script>
<input type="button" onclick="sayHello()" value="Say Hello" />
</body>
</html>
```

ex5.html

JavaScript in External File

```
<html>
<head>
<script type="text/javascript" src="ex6.js" ></script>
</head>
<body>
.....
</body>
</html>
```

ec6.html

ex6.js

```
function sayHello() {
 alert("Hello World")
}
```

JavaScript Datatypes

JavaScript allows you to work with three **primitive data types**:

- Numbers, e.g., 123, 120.50 etc.
- Strings of text, e.g. "This text string" etc.
- Boolean, e.g. true or false.

JavaScript also defines two **trivial data types**

null and undefined, each of which defines only a single value.

In addition to these primitive data types

JavaScript supports a composite data type known as **object**.

JavaScript Variables

Variables are declared with the var keyword.

```
<script type="text/javascript">
<!--
var money;
var name;
//-->
</script>
```

JavaScript Variable Scope

Global Variables: A global variable has global scope which means it can be defined anywhere in your JavaScript code.

Local Variables: A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<script type="text/javascript">
<!--
var myVar = "global"; // Declare a global variable
function checkscope( ) {
 var myVar = "local"; // Declare a local variable
 document.write(myVar);
}
//-->
</script>
```

JavaScript Variable Names

- You should not use any of the JavaScript reserved keywords as a variable name. These keywords are mentioned in the next section. For example, break or boolean variable names are not valid.
- 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.
- JavaScript variable names are case-sensitive. For example, Name and name are two different variables

JavaScript Reserved Words

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

OPERATORS

What is an Operator?

Type of Operator

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Arithmetic Operators

- + (Addition)
- -(Subtraction)
- *(Multiplication)
- /(Division)
- %(Modulus)
- ++ (Increment)
- -- (Decrement)

arithmetic.html

Comparison Operators

- == (Equal)
- != (Not Equal)
- > (Greater than)
- < (Less than)
- >= (Greater than or Equal to)
- <= (Less than or Equal to)

comparison.html

Logical Operators

- && (Logical AND)
- || (Logical OR)
- ! (Logical NOT)

logical.html

Bitwise Operators

- & (Bitwise AND)
- | (BitWise OR)
- ^ (Bitwise XOR)
- ~ (Bitwise Not)
- << (Left Shift)
- >> (Right Shift)
- >>> (Right shift with Zero)

Bitwise.html

Assignment Operators

- = (Simple Assignment)
- += (Add and Assignment)
- -= (Subtract and Assignment)
- *= (Multiply and Assignment)
- /= (Divide and Assignment)
- %= (Modules and Assignment)

assign.html

Miscellaneous Operators

- Conditional Operator (? :)
- Typeof Operator

Control Statements

- JavaScript supports the following forms of if..else statement:
  - if statement
  - if...else statement
  - if...else if... Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var age = 20;
if( age > 18 )
{
 document.write("<b>Qualifies for driving</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

If-else Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var age = 15;
if( age > 18 ){
 document.write("<b>Qualifies for driving</b>");
}else{
 document.write("<b>Does not qualify for driving</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

if-else.html

Else-if Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var book = "maths";
if( book == "history" ){
 document.write("<b>History Book</b>");
}else if( book == "maths" ){
 document.write("<b>Maths Book</b>");
}else if( book == "economics" ){
 document.write("<b>Economics Book</b>");
}else{
 document.write("<b>Unknown Book</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
```

SWITCH-CASE

```
<html>
<body>
<script type="text/javascript">
<!--
var grade='A';
document.write("Entering switch block<br />");
switch (grade)
{
 case 'A': document.write("Good job<br />");
break;
 case 'B': document.write("Pretty good<br />");
 break;
 case 'C': document.write("Passed<br />");
 break;
 case 'D': document.write("Not so good<br />");
 break;
 case 'F': document.write("Failed<br />");
 break;
 default: document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

whileLoop

```
<html>
<body>
<script type="text/javascript">
<!--
var count = 0;
document.write("Starting Loop ");
while (count < 10){
 document.write("Current Count : " + count + "<br />");
 count++;
}
document.write("Loop stopped!");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

Do-while loop

```
<html>
<body>
<script type="text/javascript">
<!--
var count = 0;
document.write("Starting Loop" + "<br />");
do{
 document.write("Current Count : " + count + "<br />");
 count++;
}while (count < 5);
document.write ("Loop stopped!");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

do-while.html

for loop

```
<html>
<body>
<script type="text/javascript">
<!--
var count;
document.write("Starting Loop" + "<br />");
for(count = 0; count < 10; count++){
 document.write("Current Count : " + count );
 document.write("<br />");
}
document.write("Loop stopped!");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

The break Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var x = 1;
document.write("Entering the loop<br /> ");
while (x < 20)
{
 if (x == 5){
 break; // breaks out of loop completely
 }
 x = x + 1;
 document.write( x + "<br />");
}
document.write("Exiting the loop!<br /> ");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

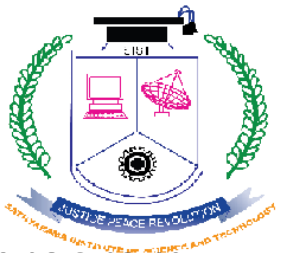
break.html

The Continue Statement

```
<html>
<body>
<script type="text/javascript">
<!--
var x = 1;
document.write("Entering the loop<br /> ");
while (x < 10)
{
 x = x + 1;
 if (x == 5){
 continue;
 }
 document.write( x + "<br />");
}
document.write("Exiting the loop!<br /> ");
//-->
</script>
<p>Set the variable to different value and then try...</p>
</body>
</html>
```

continue.html

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.

Syntax

```
<script type="text/javascript">
<!--
function functionname(parameter-list)
{
 statements
}
//-->
</script>
```

A function in JavaScript is by using the function keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

Calling a Function & Function Parameters

- Refer ex3.html


```
<html>
<head>
<script type="text/javascript">
function display(name, age)
{
 document.write (name + " is " + age + " years old.");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="display('Zara', 7)" value="Display">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Nested Function

```
<html>
<head>
<script type="text/javascript">
<!--
function hypotenuse(a, b) {
 function square(x) { return x*x; }

 return Math.sqrt(square(a) + square(b));
}
function secondFunction(){
 var result;
 result = hypotenuse(1,2);
 document.write ( result );
}
//-->
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

nested_function.html

Function () Constructor


```
<script type="text/javascript">
<!--
var variablename = new Function(Arg1, Arg2..., "Function Body");
//-->
</script>
```

- The Function() constructor is not passed any argument that specifies a name for the function it creates.
- The unnamed functions created with the Function() constructor are called anonymous functions.

```
<html>
<head>
<script type="text/javascript">
<!--
var func = new Function("x", "y", "return x*y;");
function secondFunction(){
 var result;
 result = func(10,20);
 document.write ( result );
}
//-->
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<input type="button" onclick="secondFunction()" value="Call Function">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

function_cons.html

Form validation

```
<html>
<head>
<script type="text/javascript">
<!-->
function validateform()
{
        var name=document.myform.name.value;
        var password=document.myform.password.value;

        if (name==null || name=="")
        {
                alert("Name can't be blank");
                return false;
        }
        else if(password.length<6)
        {
                alert("Password must be at least 6 characters long.");
                 return false;
         }
}
//-->
</script>
</head>
<body>

        <form name="myform" method="post" action="ex3.html" onsubmit="return validateform()" >
                Name: <input type="text" name="name"><br/>
                Password: <input type="password" name="password"><br/>
                <input type="submit" value="register">
        </form>

</body>
</html>
```

OBJECTS

- JavaScript is an Object Oriented Programming (OOP) language.
  - Encapsulation
  - Aggregation
  - Inheritance
  - Polymorphism

JavaScript Objects

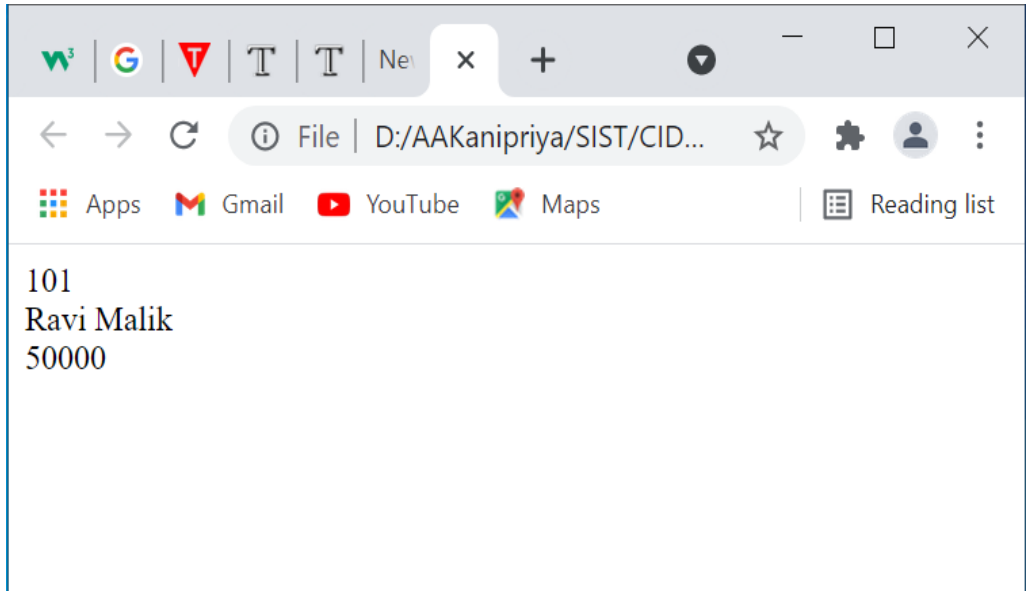
- A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.
- JavaScript is an object-based language. Everything is an object in JavaScript.
- JavaScript is template based not class based. Here, we don't create class to get the object. But, we direct create objects.
- The syntax of creating object directly is given below:

```
var objectname=new Object();
```

- Here, **new keyword** is used to create object.

```
<html>
<body>
<script>
var emp=new Object();
emp.id=101;
emp.name="Ravi Malik";
emp.salary=50000;
document.write(emp.id+"
   "+emp.name+" "+emp.salary);
</script>
</body>
</html>
```

Output



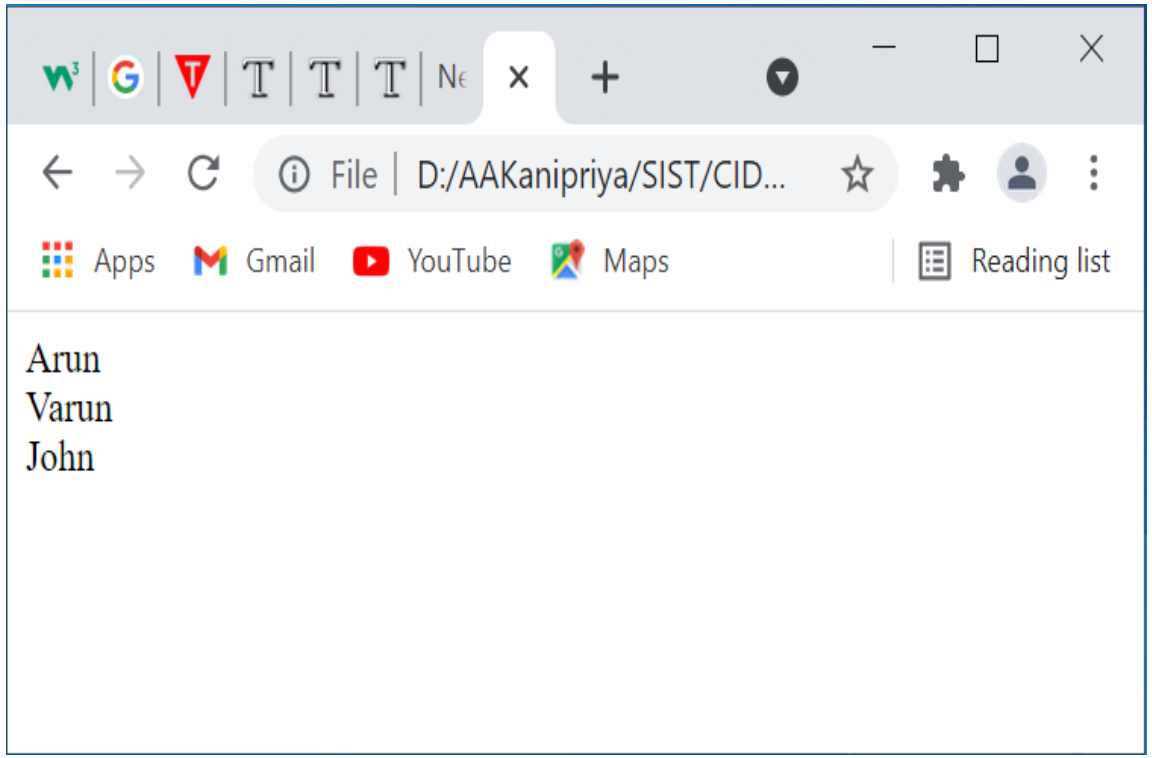
JavaScript Array

- **JavaScript array** is an object that represents a collection of similar type of elements.
- The syntax of creating array directly is given below:

var arrayname=new Array();

```
<html>
<body>
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
</body>
</html>
```

Output



Array Methods

- concat()
- Returns a new array comprised of this array joined with other array(s) and/or value(s).
- indexOf()
- Returns the first (least) index of an element within the array equal to the specified value, or -1 if none is found.
- join()
- Joins all elements of an array into a string.
- pop()
- Removes the last element from an array and returns that element.
- push()
- Adds one or more elements to the end of an array and returns the new length of the array.

- reverse()
- Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
- slice()
- Extracts a section of an array and returns a new array.
- sort()
- Sorts the elements of an array

Program Using concat()

```
<html>
  <head>
    <title>JavaScript Array concat Method</title>
  </head>

  <body>
    <script type = "text/javascript">
      var alpha = ["a", "b", "c"];
      var numeric = [1, 2, 3];
      var alphaNumeric = alpha.concat(numeric);
      document.write("alphaNumeric : " + alphaNumeric );
    </script>
  </body>
</html>
```

Output

alphaNumeric : a,b,c,1,2,3

JavaScript - Array pop() Method

```
<html>
  <head>
    <title>JavaScript Array pop Method</title>
  </head>

  <body>
    <script type = "text/javascript">
      var numbers = [1, 4, 9];

      var element = numbers.pop();
      document.write("element is : " + element );

      var element = numbers.pop();
      document.write("<br />element is : " + element );
    </script>
  </body>
</html>
```

Output

element is : 9
element is : 4

JavaScript - Array sort() Method

```
<html>
  <head>
    <title>JavaScript Array sort Method</title>
  </head>

  <body>
    <script type = "text/javascript">
      var arr = new Array("orange", "mango", "banana",
    "sugar");
      var sorted = arr.sort();
      document.write("Returned string is : " + sorted );
    </script>
  </body>
</html>
```

Output

Returned array is :
banana,mango,orange,sugar

Date Object

- Date is a data type.
- Date object manipulates date and time.
- Date() constructor takes no arguments.
- Date object allows you to get and set the year, month, day, hour, minute, second and millisecond fields.
- **Syntax:**

```
var variable_name = new Date();
```

Date Methods

- Date(): Returns current date and time.
- getDate(): Returns the day of the month.
- getDay(): Returns the day of the week.
- getFullYear(): Returns the year.
- getHours(): Returns the hour.
- getMinutes(): Returns the minutes.
- getSeconds(): Returns the seconds.
- getMilliseconds(): Returns the milliseconds.

```
<html>
    <body>
    <center>
        <h2>Date Methods</h2>
        <script type="text/javascript">
            var d = new Date();
            document.write("<b>Locale String:</b> " + d.toLocaleString()+"<br>");
            document.write("<b>Hours:</b> " + d.getHours()+"<br>");
            document.write("<b>Day:</b> " + d.getDay()+"<br>");
            document.write("<b>Month:</b> " + d.getMonth()+"<br>");
            document.write("<b>FullYear:</b> " + d.getFullYear()+"<br>");
            document.write("<b>Minutes:</b> " + d.getMinutes()+"<br>");
        </script>
    </center>
    </body>
</html>
```

Output

Date Methods

Locale String: 7/17/2021, 10:28:57 PM
Hours: 22
Day: 6
Month: 6
FullYear: 2021
Minutes: 28

JavaScript String

The **JavaScript string** is an object that represents a sequence of characters.

String Methods

Here is a list of the methods available in String object along with their description.

- **charAt()**

Returns the character at the specified index.

- **concat()**

Combines the text of two strings and returns a new string.

- **indexOf()**

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

- **slice()**

Extracts a section of a string and returns a new string.

- **substr()**

Returns the characters in a string beginning at the specified location through the specified number of characters.

- **toLowerCase()**

Returns the calling string value converted to lower case.

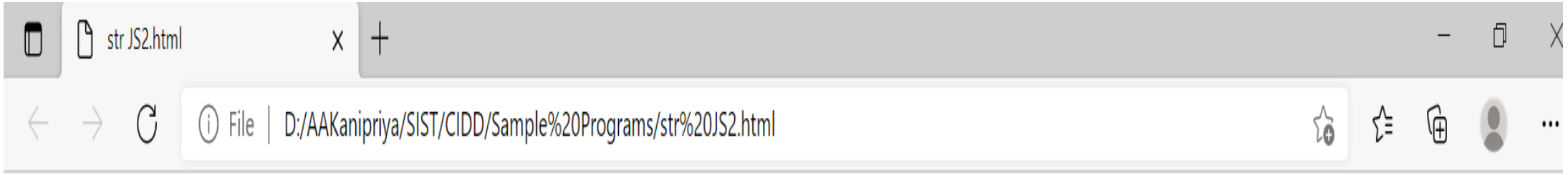
- **toUpperCase()**

Returns the calling string value converted to uppercase.

```
<!DOCTYPE html>
<html>
<body>
<script>
var s1="javascript ";
var s2="concat example";
var s3=s1.concat(s2);
document.write("String Concat:"+s3+"</br>");
var str="javascript";
document.write("CharAt:"+str.charAt(2)+"</br>");
document.write("Substring:"+str.substr(2,4)+"</br>");
var s1="javascript from sample programs";
var n=s1.indexOf("from");
document.write("Index Value is:"+n+"</br>");
```

```
var s1="JavaScript toLowerCase Example";
var s2=s1.toLowerCase();
document.write("Lowercase Letters:"+s2+"</br>");
var s1="abcdefgh";
var s2=s1.slice(2,5);
document.write("Sliced String:"+s2);
</script>
</body>
</html>
```

Output



String Concat:javascript concat example
CharAt:v
Substring:vasc
Index Value is:11
Lowercase Letters:javascript tolowercase example
Sliced String:cde

JavaScript Math

- The **JavaScript math** object provides several constants and methods to perform mathematical operation. Unlike date object, it doesn't have constructors.
- abs()

  Returns the absolute value of a number.
- ceil()

  Returns the smallest integer greater than or equal to a number.
- floor()

  Returns the largest integer less than or equal to a number.
- max()

  Returns the largest of zero or more numbers.

- min()

  Returns the smallest of zero or more numbers.

- pow()

  Returns base to the exponent power, that is, base exponent.

- sqrt()

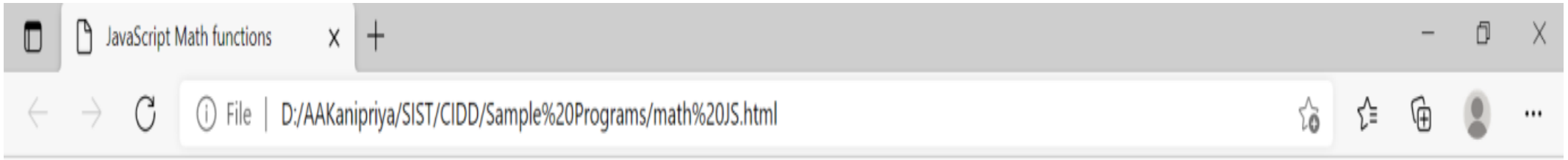
  Returns the square root of a number.

- tan()

  Returns the tangent of a number.

```
<html>
  <head>
    <title>JavaScript Math functions </title>
  </head>

  <body>
    <script type = "text/javascript">
      var value = Math.max(10, 20, -1, 100);
      document.write("Max Value : " + value );
   var value = Math.min(10, 20, -1, 100);
      document.write("<br />Min Value : " +
   value );
var value = Math.pow(7, 2);
      document.write("<br />Square value : " + value );
         var value = Math.sqrt( 81 );
      document.write("<br />Square root Value : " + value
);
         var value = Math.tan( 45 );
      document.write("<br />tan Value : " + value );
        </script>
</body>
</html>
```

Output



Max Value : 100
Min Value : -1
Square value : 49
Square root Value : 9
tan Value : 1.6197751905438615

JavaScript Number Object

- The **JavaScript number** object *enables you to represent a numeric value*. It may be integer or floating-point.
- JavaScript number object follows IEEE standard to represent the floating-point numbers.
- By the help of Number() constructor, you can create number object in JavaScript. For example:

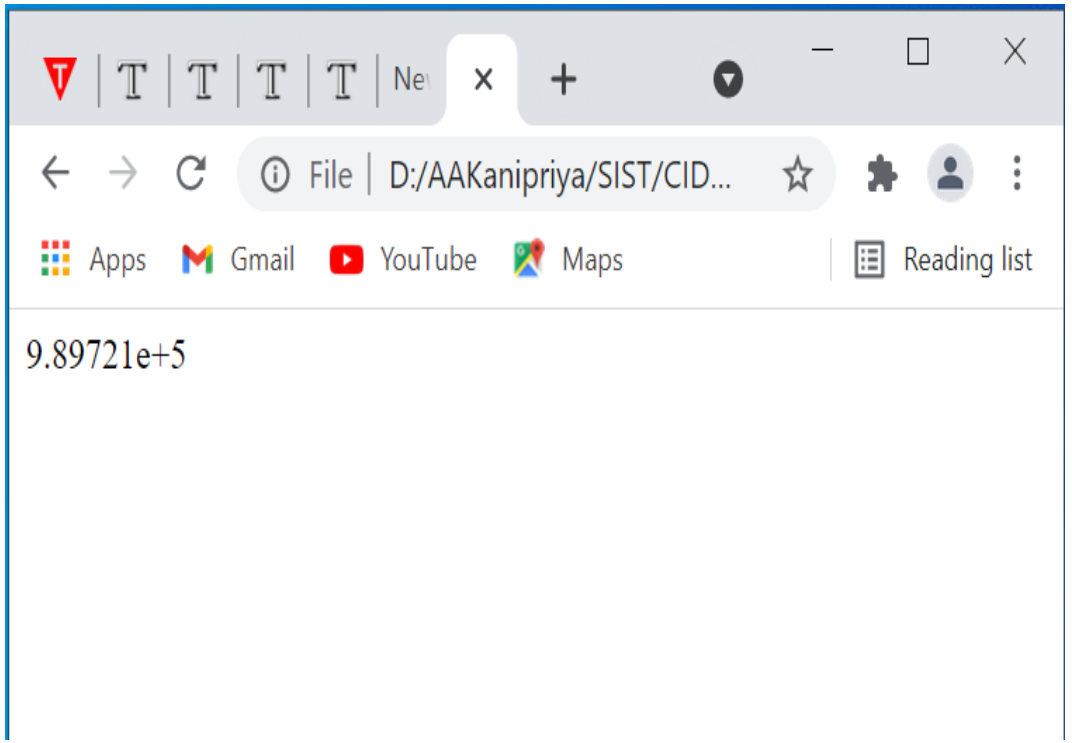
```
var n=new Number(value);
```

Number Methods

Sr.No.	Method & Description
1	toExponential() Forces a number to display in exponential notation, even if the number is in the range in which JavaScript normally uses standard notation.
2	toFixed() Formats a number with a specific number of digits to the right of the decimal.
3	toLocaleString() Returns a string value version of the current number in a format that may vary according to a browser's local settings.
4	toPrecision() Defines how many total digits (including digits to the left and right of the decimal) to display of a number.
5	toString() Returns the string representation of the number's value.
6	valueOf() Returns the number's value.

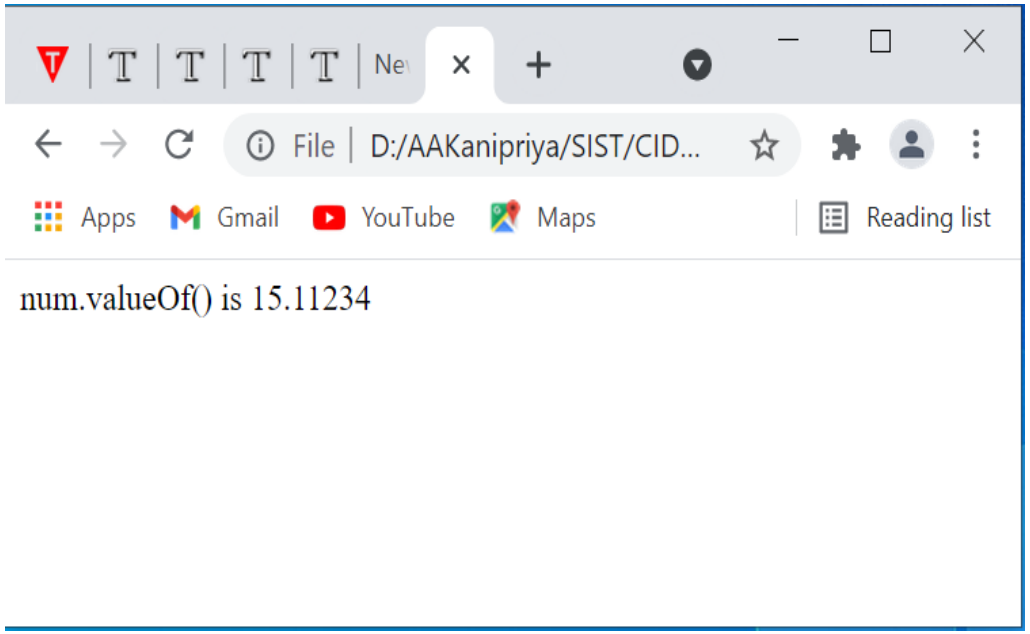
```
<!DOCTYPE html>
<html>
<body>

<script>
var a=989721;
document.writeln(a.toExponential());
</script>

</body>
</html>
```

Output



```
<html>
  <head>
    <title>JavaScript valueOf() Method </title>
  </head>

  <body>
    <script type = "text/javascript">
      var num = new Number(15.11234);
      document.write("num.valueOf() is " + num.valueOf());
    </script>
  </body>
</html>
```

Output



The Window Object

- The window object is supported by all browsers. It represents the browser's window.
- All global JavaScript objects, functions, and variables automatically become members of the window object.
- Global variables are properties of the window object.
- Global functions are methods of the window object.
- Even the document object (of the HTML DOM) is a property of the window object:

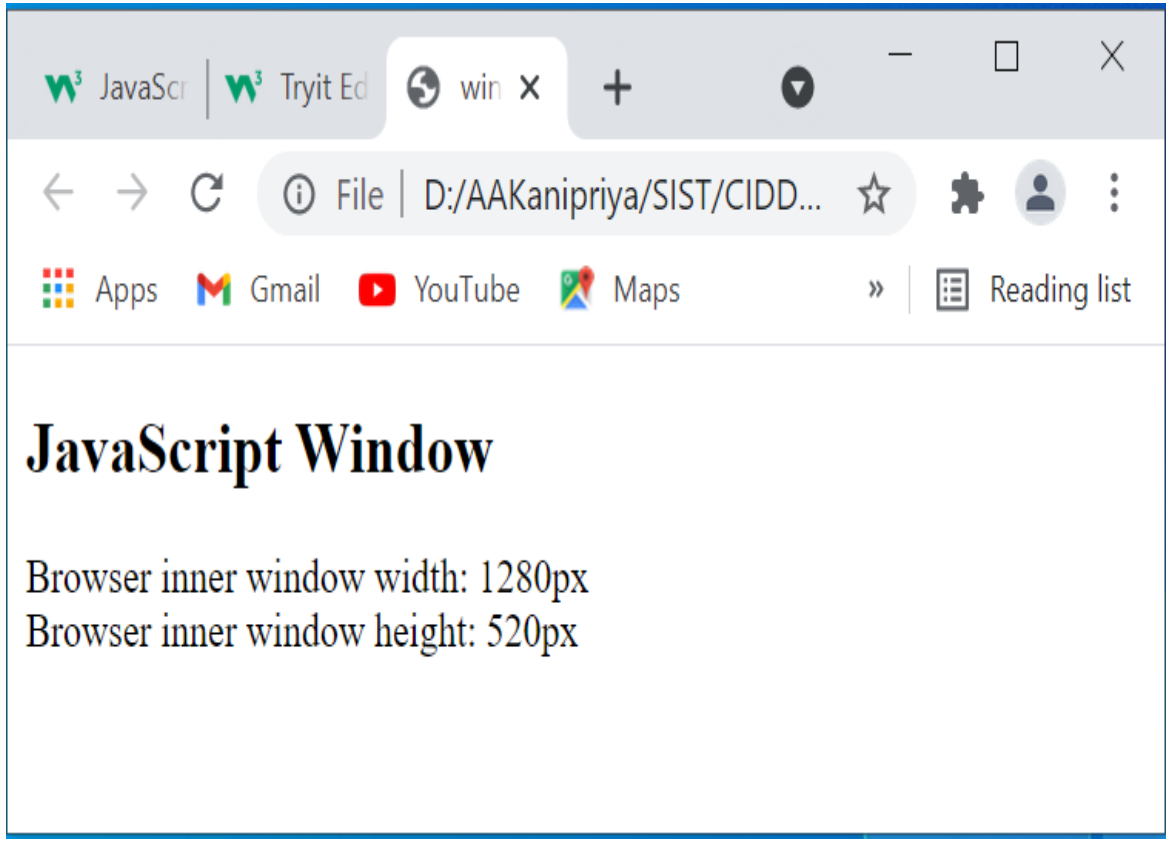
```
document.getElementById("header");
```

Window Size

- Two properties can be used to determine the size of the browser window.
- Both properties return the sizes in pixels:
- window.innerHeight - the inner height of the browser window (in pixels)
- window.innerWidth - the inner width of the browser window (in pixels)

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Window</h2>

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

<script>
document.getElementById("demo").innerHTML =
"Browser inner window width: " + window.innerWidth +
    "px<br>" +
"Browser inner window height: " + window.innerHeight +
    "px";
</script>

</body>
</html>
```

Output



Location Object

- Location object is a part of the window object.
- It is accessed through the '**window.location**' property.
- It contains the information about the current URL.

Property	Description
hash	It returns the anchor portion of a URL.
host	It returns the hostname and port of a URL.
hostname	It returns the hostname of a URL.
href	It returns the entire URL.
pathname	It returns the path name of a URL.
port	It returns the port number the server uses for a URL.
protocol	It returns the protocol of a URL.
search	It returns the query portion of a URL.

Location Object Methods

Method	Description
assign()	It loads a new document.
reload()	It reloads the current document.
replace()	It replaces the current document with a new one.

Simple Program on Location Object

```
<html>
    <body>
    <script type="text/javascript">
        document.write("<b>Path Name:
</b>"+location.pathname+"<br><br
>");
        document.write("<b>Href:
</b>"+location.href+"<br><br>");
        document.write("<b>Protocol:
</b>"+location.protcol+"<br><br>"
);
    </script>
    </body>
</html>
```

Output

Path Name: /webprj/ZYHerDfFj/index.html

Href: https://www.onlinegdb.com/webprj/ZYHerDfFj/index.html

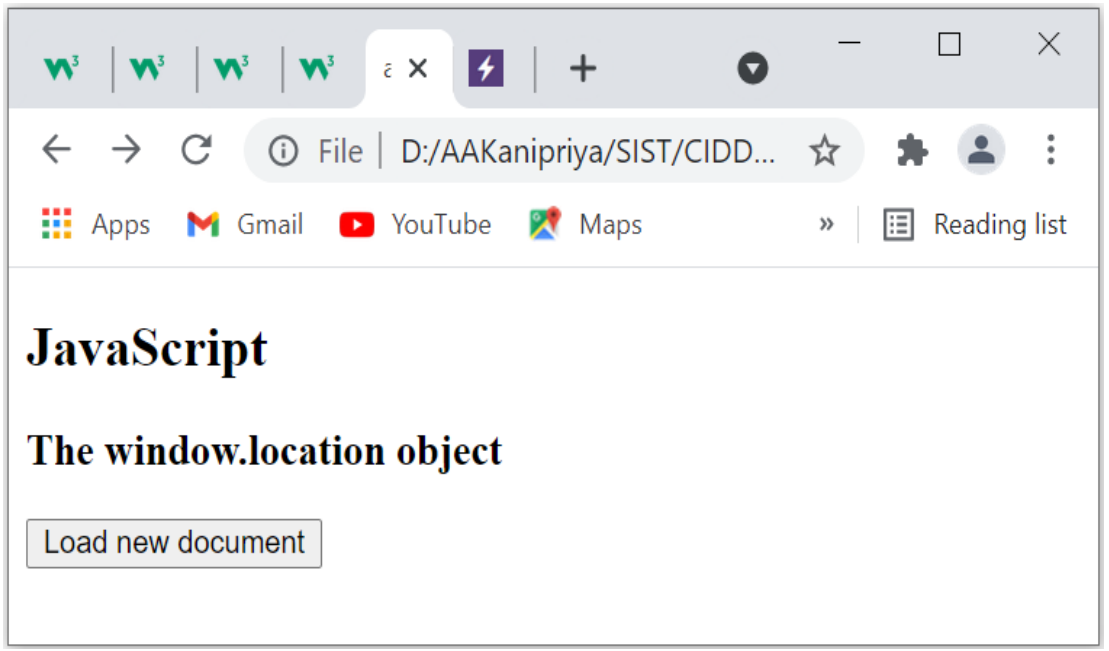
Protocol: undefined

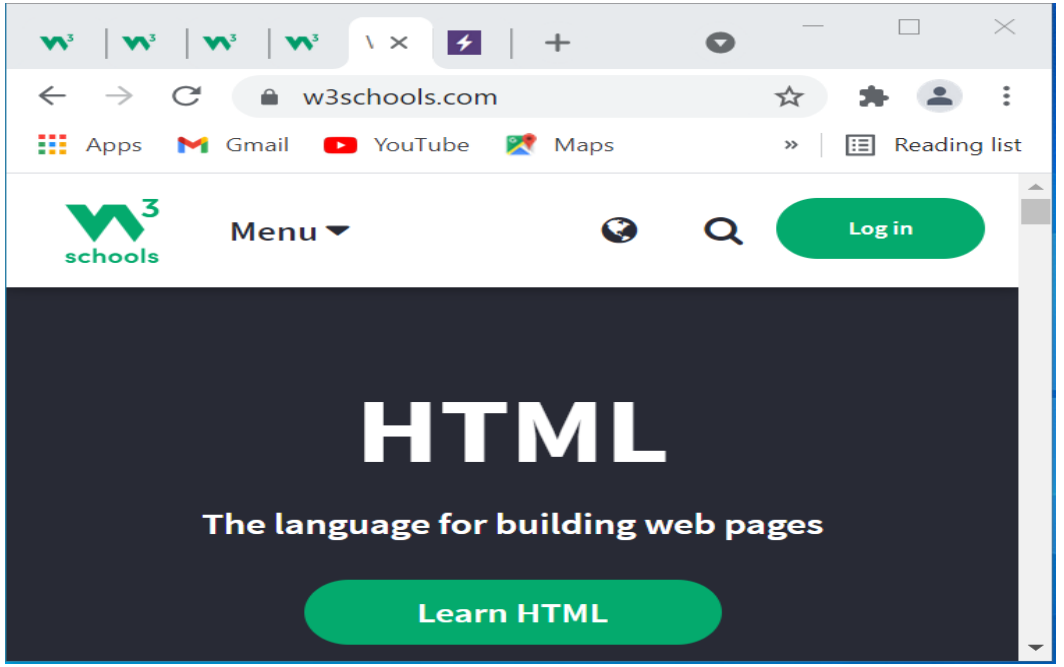
Window Location Assign

The window.location.assign() method loads a new document.

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript</h2>

<h3>The window.location object</h3>

<input type="button" value="Load new document" onclick="newDoc()">

<script>
function newDoc() {
  window.location.assign("https://www.w3schools.com")
}
</script>

</body>
</html>
```

Output


File | D:/AAKanipriya/SIST/CIDD...
Apps
Gmail
YouTube
Maps
Reading list
JavaScript
The window.location object
Load new document


w3schools.com
Apps
Gmail
YouTube
Maps
Reading list
Menu
Log in
HTML
The language for building web pages
Learn HTML

JavaScript Popup Boxes

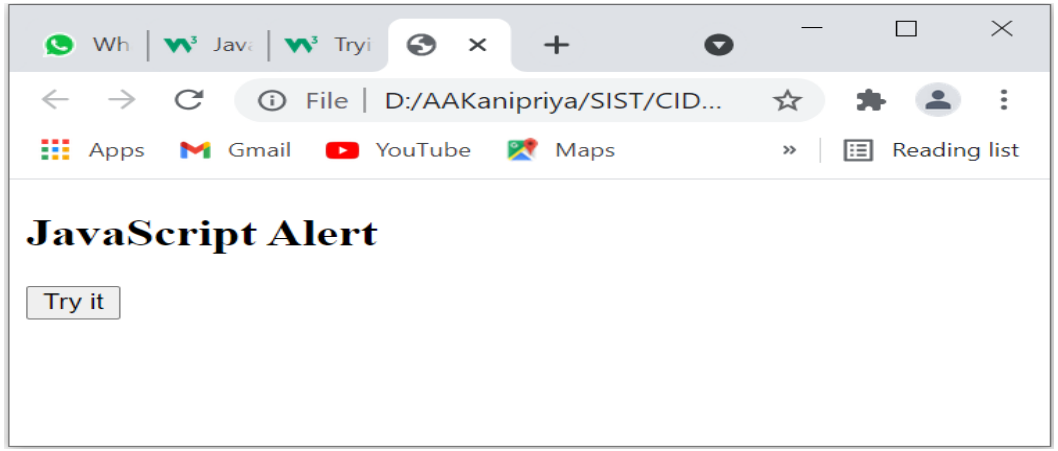
- JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

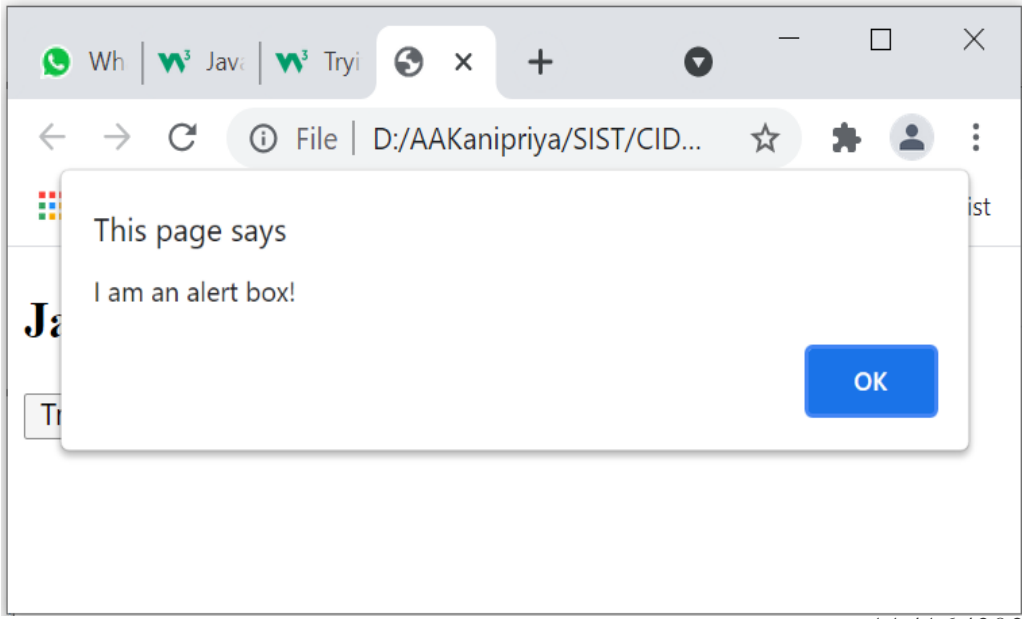
Alert Box

- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- Syntax: **window.alert("*sometext*");**

Sample program for alert box

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Alert</h2>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  window.alert("I am an alert box!");
}
</script>

</body>
</html>
```





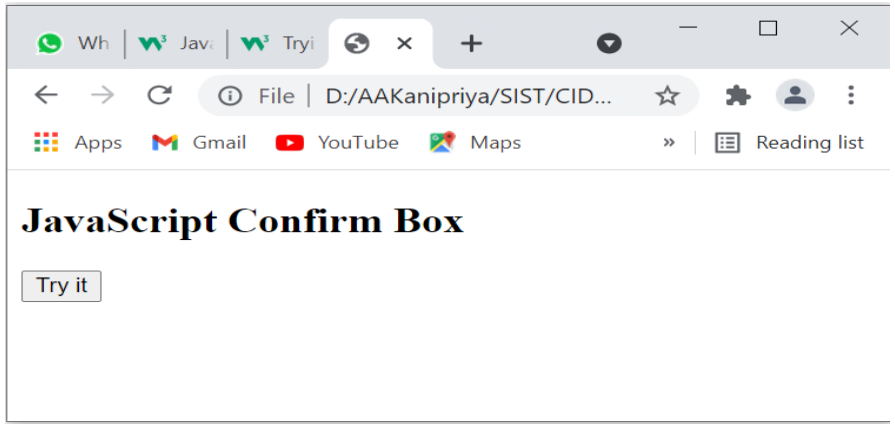
Confirm Box

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
- Syntax: **window.confirm("*sometext*");**

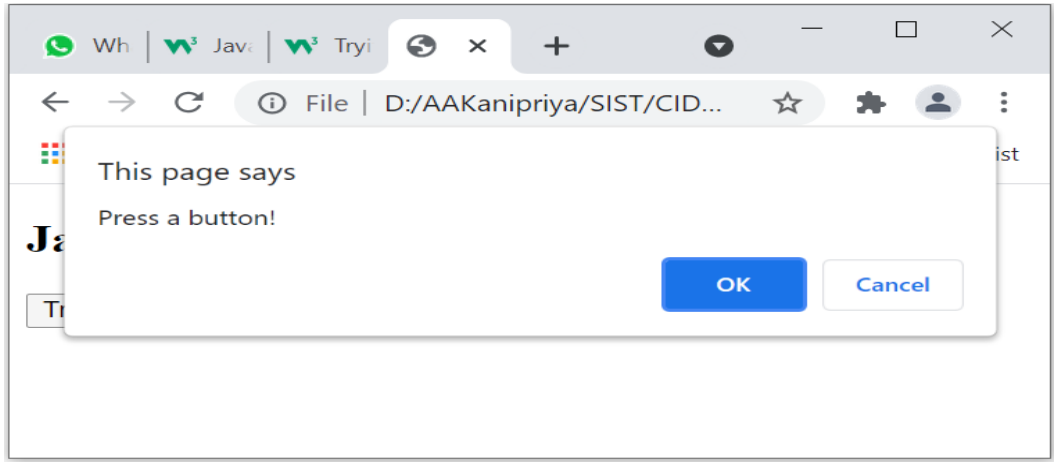
Sample program for confirm box

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Confirm Box</h2>
<button onclick="myFunction()">Try
   it</button>
<p id="demo"></p>
<script>
function myFunction() {
  var txt;
if (window.confirm("Press a button!")) {
    txt = "You pressed OK!";
  } else {
    txt = "You pressed Cancel!";
  }
  document.getElementById("demo").innerHTML =
txt;
}
</script>
</body>
</html>
```

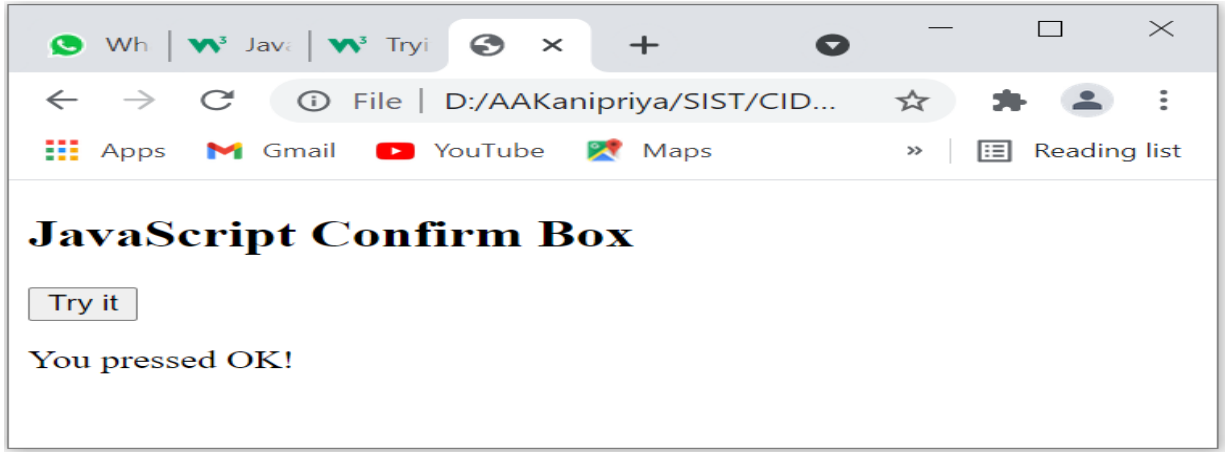
Initial Output window



Output window after clicking on Try it button



Output window after clicking on OK button

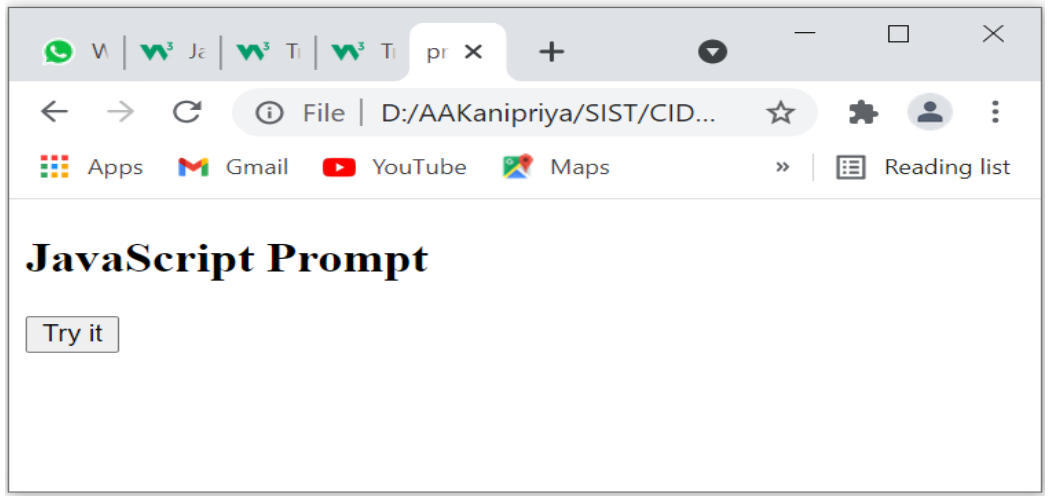


Prompt Box

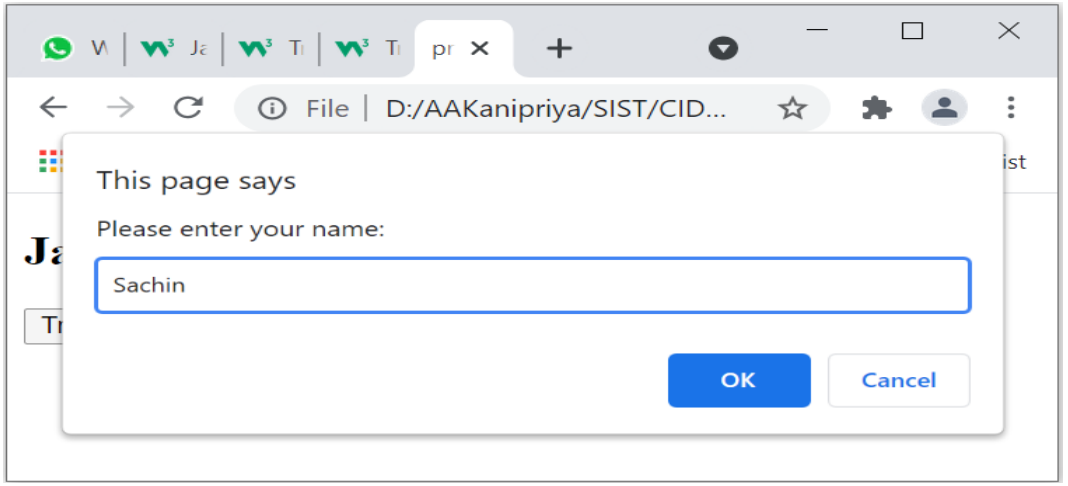
- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- Syntax: window.prompt("*sometext*","*defaultText*");

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Prompt</h2>
<button onclick="myFunction()">Try
   it</button>
<p id="demo"></p>
<script>
function myFunction() {
 var text;
 var person = prompt("Please enter your name:");
 if (person == null || person == "") {
   text = "User cancelled the prompt.";
 } else {
   text = "Hello " + person + "! How are you today?";
 }
 document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

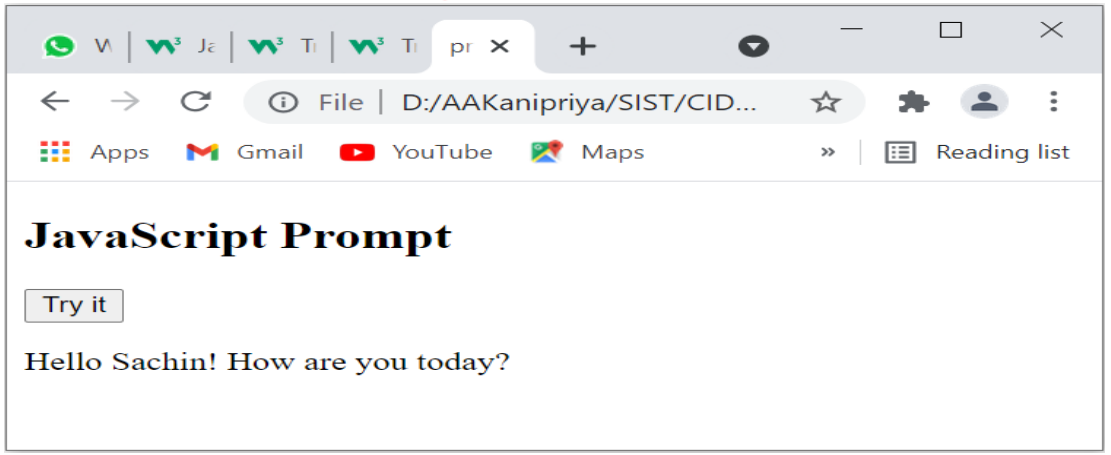
1) Initial Output



2) Output window after clicking Try it button



3) Output window after clicking OK button



Frames

- Frames are used to divide your browser window into multiple sections where each section can load a separate HTML document.
- A collection of frames in the browser window is known as a frameset.
- The window is divided into frames in a similar way the tables are organized: into rows and columns.
- The HTML <iframe> tag specifies an inline frame.
- An inline frame is used to embed another document within the current HTML document.
- Syntax

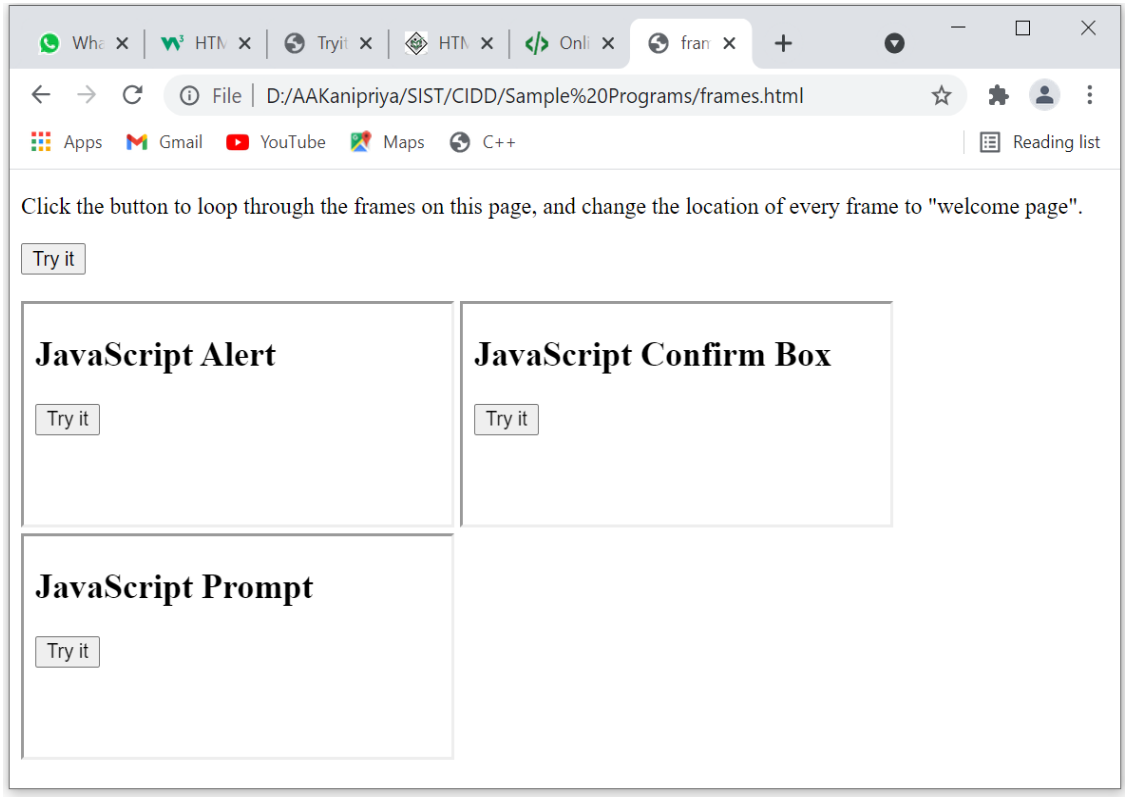
```
<iframe src="url" title="description"></iframe>
```

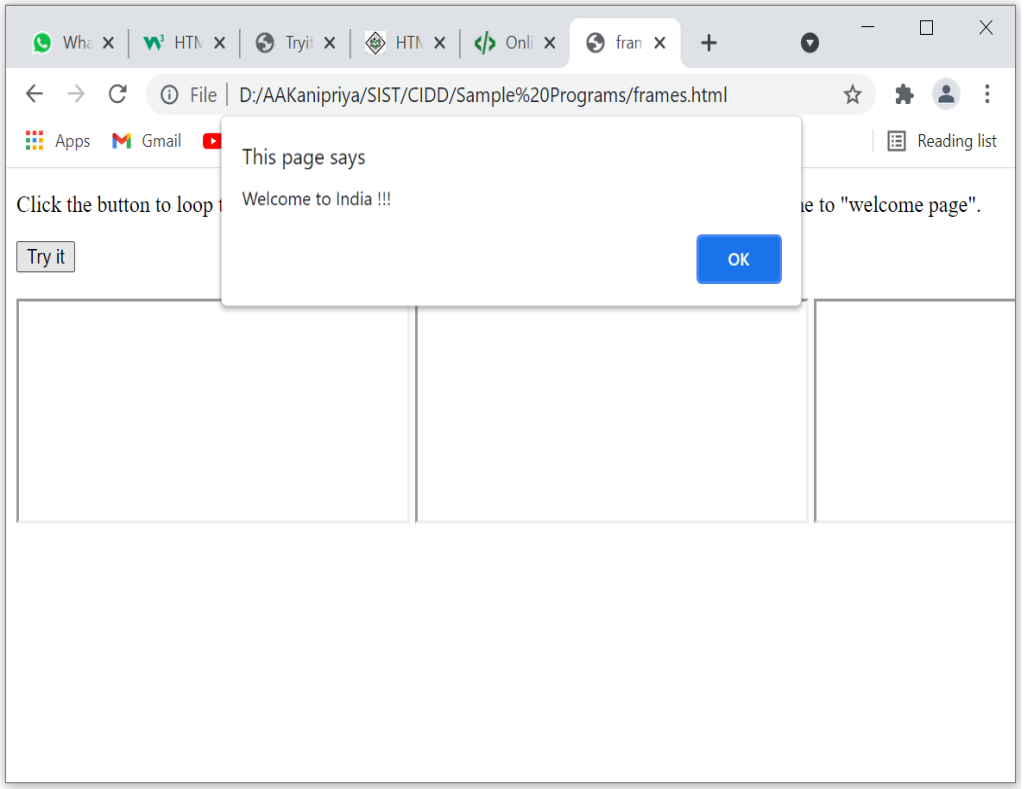
Sample Program for Frame Object

```
<!DOCTYPE html>
<html>
<body>

<p>Click the button to loop through the frames
   on this page, and change the location of every
   frame to "welcome page".</p>

<button onclick="myFunction()">Try
   it</button>
<br><br>

<iframe src="alert.html"></iframe>
<iframe src="confirm.html"></iframe>
<iframe src="prompt.html"></iframe>
```

```
<script>
function myFunction()
{
  var frames = window.frames;
  var i;

  for (i = 0; i < frames.length; i++)
{
    frames[i].location = "welcome.html";
  }
}
</script>

</body>
</html>
```

Output


File | D:/AAKanipriya/SIST/CIDD/Sample%20Programs/frames.html
Apps Gmail YouTube Maps C++ Reading list
Click the button to loop through the frames on this page, and change the location of every frame to "welcome page".
Try it
JavaScript Alert
Try it
JavaScript Confirm Box
Try it
JavaScript Prompt
Try it


File | D:/AAKanipriya/SIST/CIDD/Sample%20Programs/frames.html
Apps Gmail Reading list
This page says
Welcome to India !!!
OK
Try it

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.
- The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

Form Attributes

The Action Attribute

- The action attribute defines the action to be performed when the form is submitted.
- Usually, the form data is sent to a file on the server when the user clicks on the submit button.

The Target Attribute

- The target attribute specifies where to display the response that is received after submitting the form.

- The target attribute can have one of the following values:

Value	Description
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

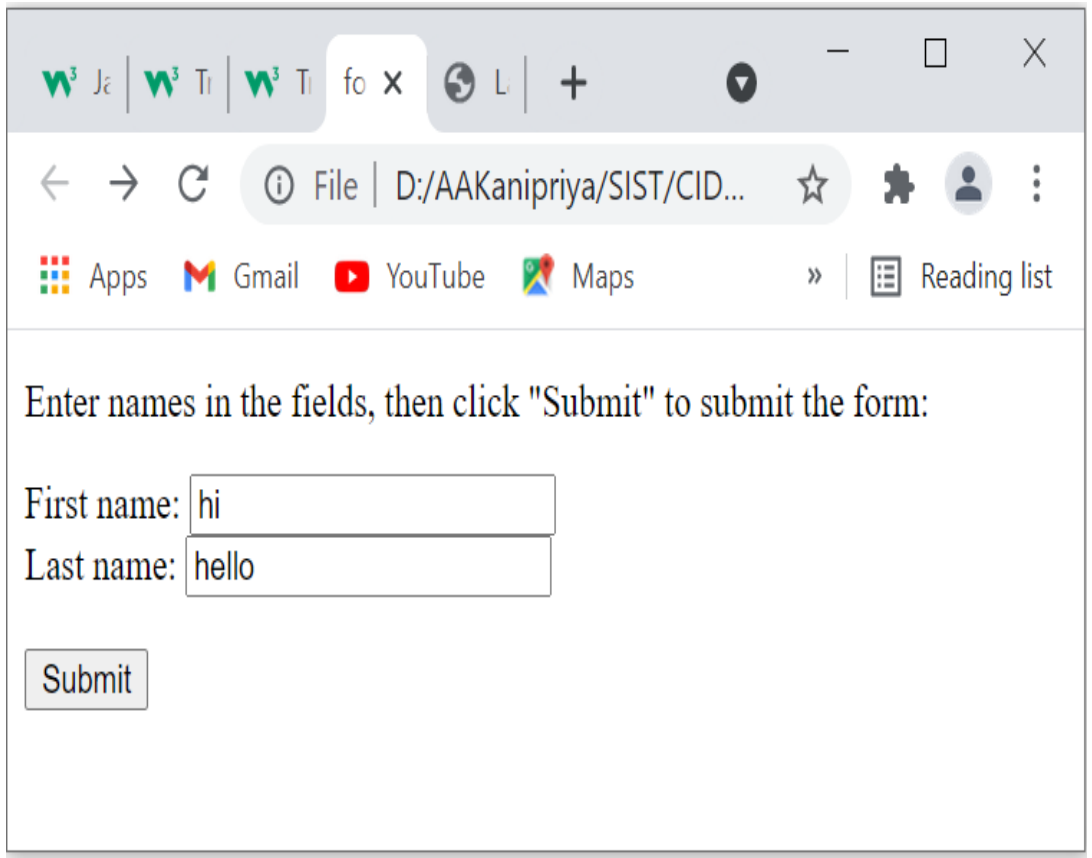
The Method Attribute

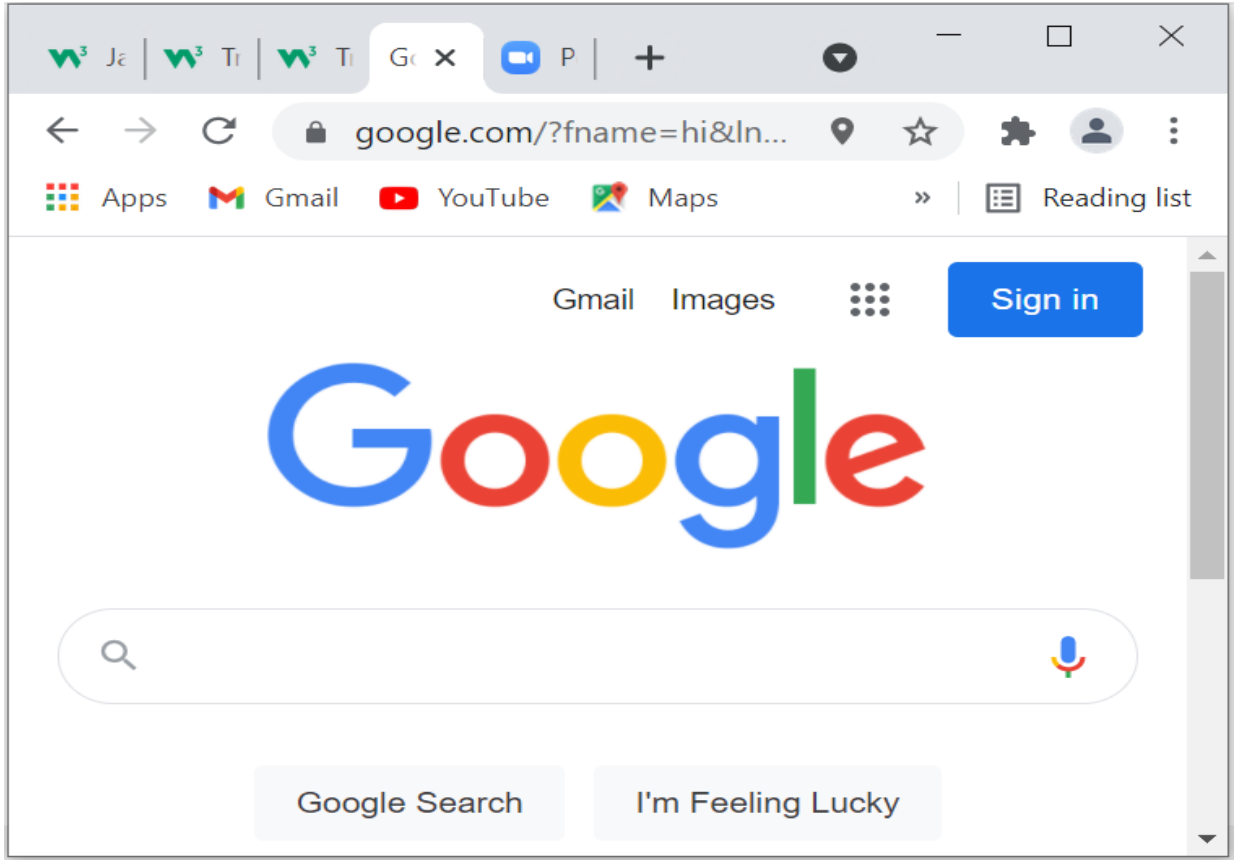
- The method attribute specifies the HTTP method to be used when submitting the form data.
- The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
- The default HTTP method when submitting form data is GET.
- After clicking on the submit, the form values will be visible in the address bar of the new browser tab when the form uses get method.
- Whereas the form values will not be visible in the address bar of the new browser tab when the form uses the post method.

Sample Program-1 using Form Object

```
<!DOCTYPE html>
<html>
<body>

<p>Enter names in the fields, then click "Submit" to submit the form:</p>

<form id="frm1" action="http://www.google.com">
 First name: <input type="text" name="fname"><br>
 Last name: <input type="text" name="lname"><br><br>
 <input type="button" onclick="myFunction()" value="Submit">
</form>

<script>
function myFunction() {
 document.getElementById("frm1").submit();
}
</script>

</body>
</html>
```

Output

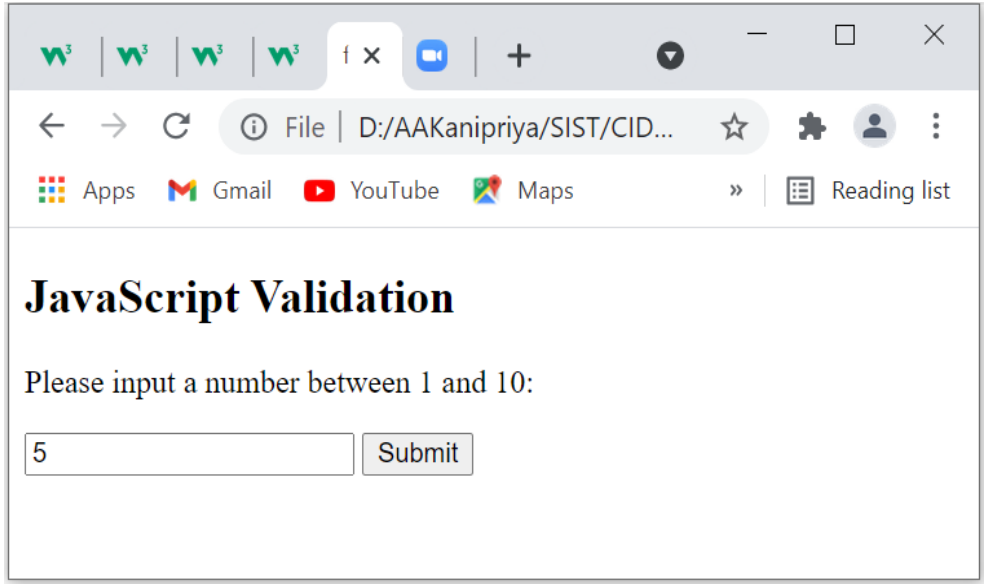

File | D:/AAKanipriya/SIST/CID...
Apps
Gmail
YouTube
Maps
Reading list
Enter names in the fields, then click "Submit" to submit the form:
First name: hi
Last name: hello
Submit

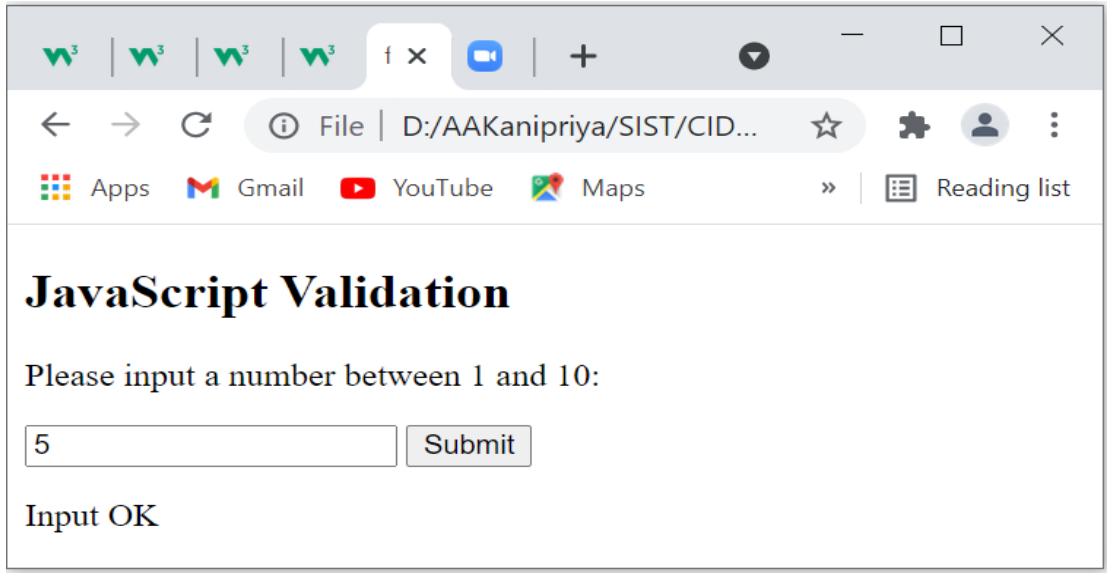

google.com/?fname=hi&ln...
Apps
Gmail
YouTube
Maps
Reading list
Gmail
Images
Sign in
Google
Google Search
I'm Feeling Lucky

Sample program-2

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Validation</h2>
<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() {
  // Get the value of the input field with id="numb"
let x = document.getElementById("numb").value;
 // If x is Not a Number or less than one or greater than 10
 let text;
 if (isNaN(x) || x < 1 || x > 10) {
   text = "Input not valid";
 } else {
   text = "Input OK";
 }
 document.getElementById("demo").innerHTML = text;
}
</script>

</body>
</html>
```

Output

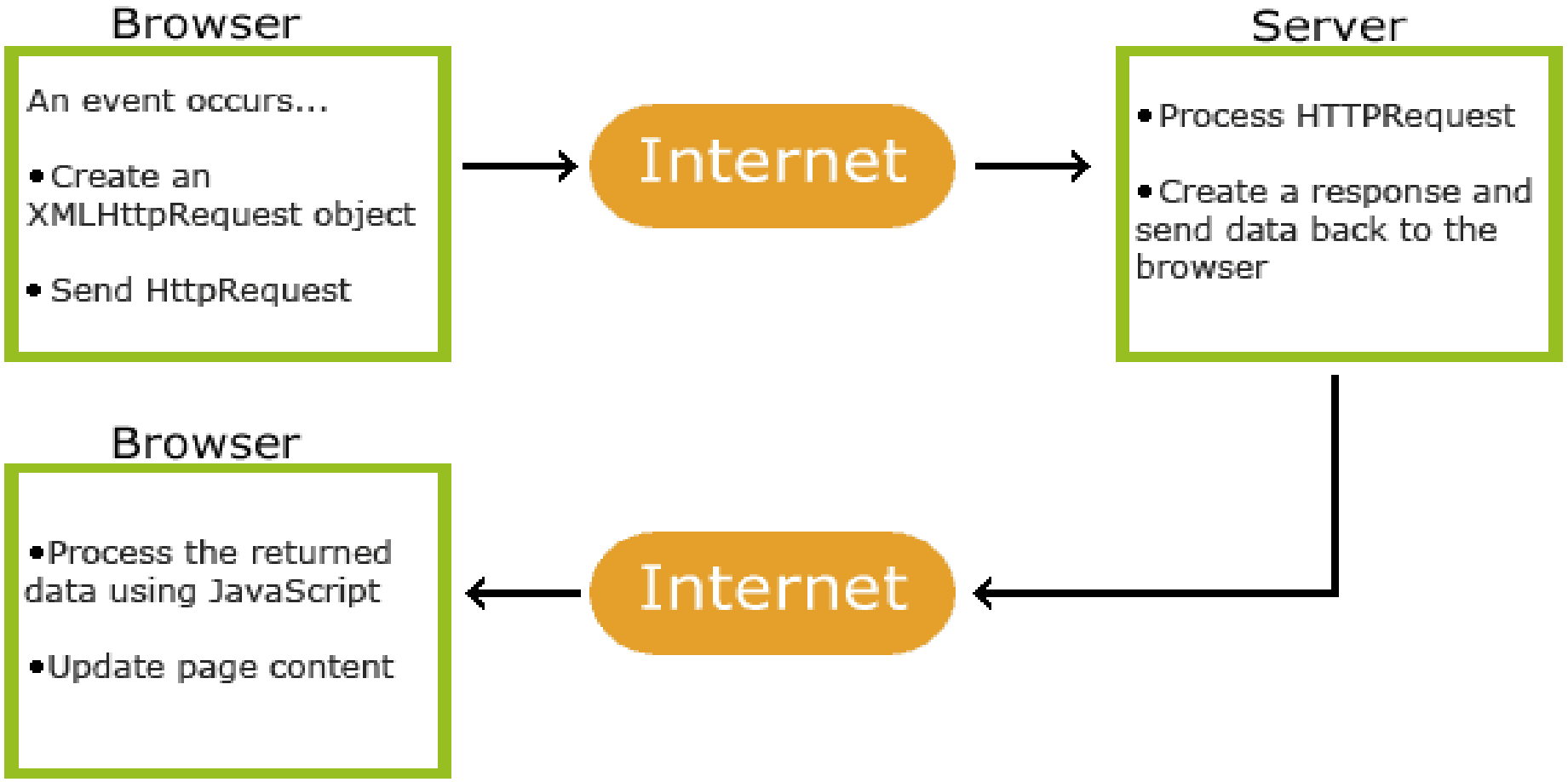

File | D:/AAKanipriya/SIST/CID...
Apps
Gmail
YouTube
Maps
Reading list
JavaScript Validation
Please input a number between 1 and 10:
5
Submit


File | D:/AAKanipriya/SIST/CID...
Apps
Gmail
YouTube
Maps
Reading list
JavaScript Validation
Please input a number between 1 and 10:
5
Submit
Input OK

AJAX

- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

How AJAX Works



1. An event occurs in a web page (the page is loaded, a button is clicked)
2. An XMLHttpRequest object is created by JavaScript
3. The XMLHttpRequest object sends a request to a web server
4. The server processes the request
5. The server sends a response back to the web page
6. The response is read by JavaScript
7. Proper action (like page update) is performed by JavaScript

The XMLHttpRequest Object

- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes.
- This means that it is possible to update parts of a web page, without reloading the whole page.
- Syntax for creating an XMLHttpRequest object:

```
var xhttp = new XMLHttpRequest();
```

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

AJAX Example

```
<!DOCTYPE html>
<html>
<body>
 <div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button"
   onclick="loadDoc()">Change
   Content</button>
</div>
 <script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function()
{
 if (this.readyState == 4 && this.status == 200)
{
document.getElementById("demo").innerHTML =
     this.responseText;
   }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```

Output

The XMLHttpRequest Object

Change Content

The output will look like the following after the change content button has been clicked.

AJAX

AJAX is not a programming language.

AJAX is a technique for accessing web servers from a web page.

AJAX stands for Asynchronous JavaScript And XML.

Server Response Properties

Property	Description
responseText	get the response data as a string
responseXML	get the response data as XML data

Server Response Methods

Method	Description
getResponseHeader()	Returns specific header information from the server resource
getAllResponseHeaders()	Returns all the header information from the server resource

The responseText Property

The responseText property returns the server response as a JavaScript string

The responseXML Property

- The XMLHttpRequest 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

```
<!DOCTYPE html>
<html>
<body>

<h2>The XMLHttpRequest Object</h2>
<p id="demo"></p>

<script>
const xhttp = new XMLHttpRequest();
xhttp.onload = function() {
  const xmlDoc = this.responseXML;
  const x = xmlDoc.getElementsByTagName("ARTIST");
  let txt = "";
 for (let i = 0; i < x.length; i++) {
    txt = txt + x[i].childNodes[0].nodeValue + "<br>";
  }
  document.getElementById("demo").innerHTML = txt;
}
xhttp.open("GET", "cd_catalog.xml");
xhttp.send();
</script>

</body>
</html>
```

Output

The XMLHttpRequest Object

Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore
Eros Ramazzotti
Bee Gees
Dr.Hook
Rod Stewart
Andrea Bocelli
Percy Sledge
Savage Rose
Many
Kenny Rogers
Will Smith
Van Morrison
Jorn Hoel
Cat Stevens
Sam Brown

Thank You