

# UNIT III CLIENT SIDE TECHNOLOGIES

- XML
  - Document Type Definition
  - XML Schema
  - Document Object Model
  - Presenting XML Using XML Parsers: DOM and SAX
- JavaScript Fundamentals
  - Evolution of AJAX
  - AJAX Framework
  - Web applications with AJAX
  - AJAX with PHP
  - AJAX with Databases

Presented by,  
B.Vijayalakshmi  
Computer Centre  
MIT Campus  
Anna University

# JavaScript

- A **script** is a small piece of program that can add interactivity to website
- JavaScript is an open source & most popular **client side scripting language** supported by all browsers.
- It is *an **object-based scripting language***.
- JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- JavaScript is also being used widely in game development and Mobile application development.
- was developed by Brendan Eich in 1995, which appeared in Netscape, a popular browser of that time.
- The language was initially called LiveScript and was later renamed JavaScript.

# Java Script Cont'd

- **JavaScript and Java are very much unrelated.**
- **Java is a very complex programming language whereas JavaScript is only a scripting language.**
- The syntax of JavaScript is mostly influenced by the programming language C.
- JavaScript has its own syntax and programming style.
- A semicolon at the end of every statement (is Optional)
- JavaScript interpreter **ignores the tabs, spaces, and newlines** that appear in the script, except for strings and regular expressions.
- JavaScript supports dynamic typing, we do not have to provide the data type while declaring a variable. In JavaScript, we just have to use var or let keyword before the variable name to declare a variable without worrying about its type.
- JavaScript is a **case sensitive** language

# Javascript Features

- Light Weight Scripting language
- Dynamic Typing
- Object-oriented programming support
- Functional Style
- Platform Independent
- Prototype-based
- Interpreted Language
- Async Processing
- Client-Side Validation
- More control in the browser

# JavaScript in HTML

- **JavaScript is the default scripting language for most of the browsers.**
- The JavaScript code can be inserted in HTML file by using the HTML `<script>` tag.
- When an HTML document is loaded with the `<script>` tag in the web browser, the browser processes the content enclosed inside the script tag as JavaScript code.
- JavaScript code can be stored in a separate file and then include it wherever it's needed (**External JavaScript**), or its functionality can be defined inside HTML document itself (**Internal Script**).
- In html, java script code must be inserted between `<script>` and `</script>` tags either in head or body section or as separate javascript file.

# Script tag Attributes and its Uses

Attributes	Values	Description
async	<ul style="list-style-type: none"><li>•true</li><li>•false</li></ul>	It specifies whether the script should be executed asynchronously or not.
type	<ul style="list-style-type: none"><li>•text/ECMAScript</li><li>•text/javascript</li><li>•application/ECMAScript</li><li>•application/javascript</li><li>•text/VBScript</li></ul>	It specifies the Multipurpose Internet Mail Extensions (MIME) type of script. Generally, <b>text/javascript</b> value is used.
charset	charset	It specifies the type of character encoding used in the script
defer	<ul style="list-style-type: none"><li>•true</li><li>•false</li></ul>	It specifies whether the browser continues parsing the web page or not.
src	URL	It specifies the uniform source locator(URL) of a file that contains the script.

## Example:

1. `<script src="JS_FILE_URL" type="..."></script>`
2. `<script type="text/javascript">`  
    `// JS code here`  
    `</script>`

# How to Run JavaScript?

- Being a scripting language, **JavaScript cannot run on its own.**
- **In fact, the browser is responsible for running JavaScript code.**
- When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it is up to the browser to execute it.
- The main advantage of JavaScript is that **all modern web browsers support** JavaScript.
- Also, JavaScript **runs on any operating system** including Windows, Linux or Mac.

# Output in JavaScript

- We can get JavaScript Output in four simple and different ways on a webpage
  - **Using innerHTML property**
    - lets us write into an HTML element
    - To do that first we need to **provide a specific Id to the HTML element** that we want to access by the JavaScript code.
  - **Using document.write() method**
    - using this method we can directly write output to the HTML page
  - **Using Alert Box**
    - send alert messages to notify something to the user
  - **By logging on the Console**
    - lets you create console logs which can be seen in the **browser's developers' tools(Console)** for debugging purposes
- We can use them according to the application requirement.

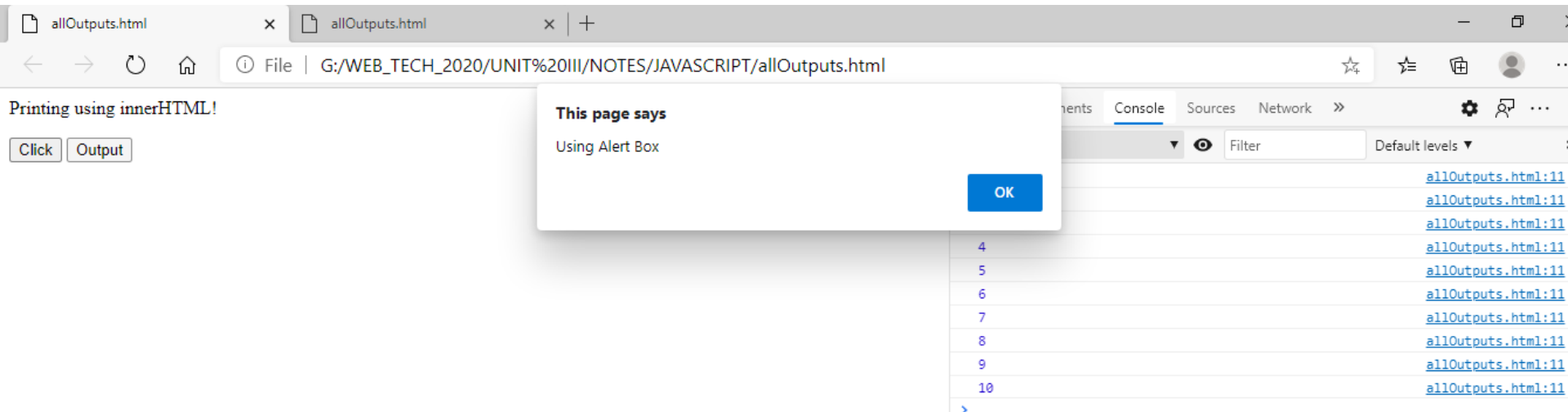


# Various outputs in java script

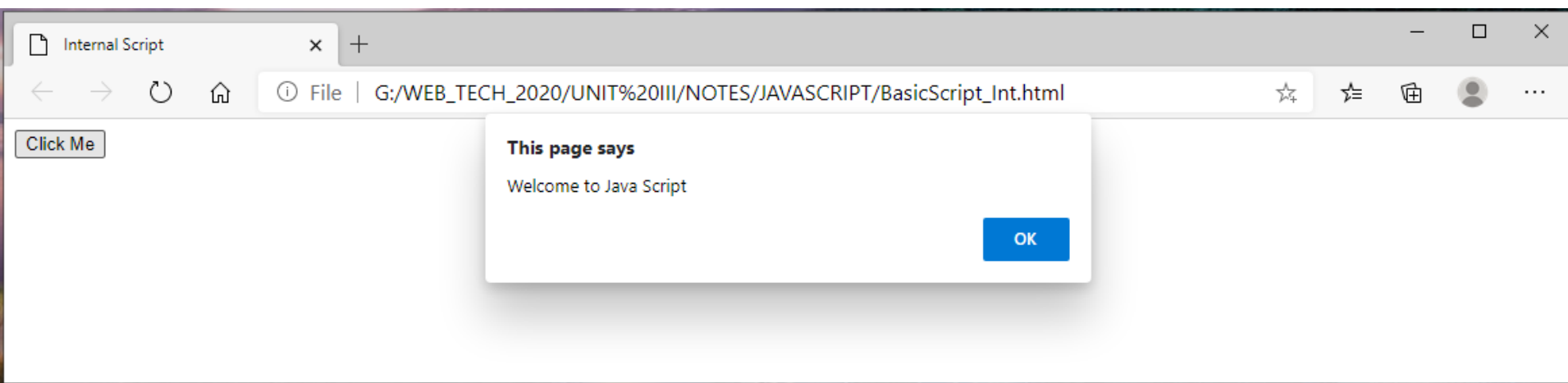
## Example

```
<html>
  <body>
    <script language="javascript" type="text/javascript">
      function output()
      {
        alert("Using Alert Box");
        document.write("Writing Using document.write");
      }
      var i;
      for(i=1;i<=10;i++)
        console.log(i)
    </script>
    <p id="para1">Good Morning</p>
    <button type="button" onclick='document.getElementById("para1").innerHTML =
    "Printing using innerHTML!">Click</button>
    <input type="button" onclick="output()" value="Output"/>
  </body>
</html>
```

# Various outputs in java script Cont'd



# Internal Script Example



```
<html>

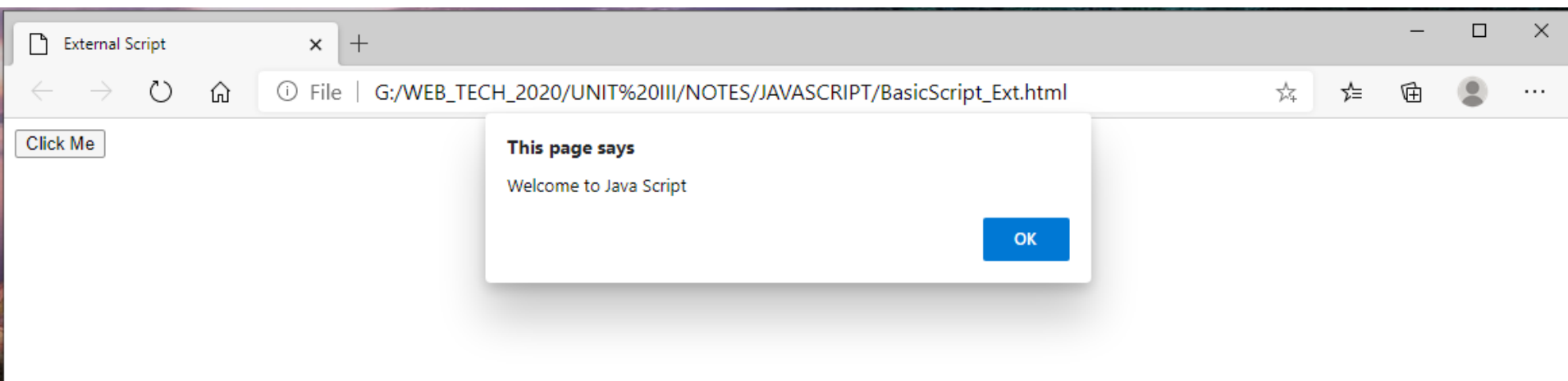
  <head>
    <title> Internal Script</title>

    <script type = "text/JavaScript">
      function Welcome() {
        alert("Welcome to Java Script");
      }
    </script>
  </head>

  <body>
    <input type = "button" onclick = "Welcome()" name = "ok" value = "Click Me" />
  </body>

</html>
```

# External Script Example



```
<html>

  <head>
    <title>External Script</title>
    <script src = "ext_script.js" type = "text/javascript"/></script>
  </head>

  <body>
    <input type = "button" onclick = "Welcome()" name = "ok" value = "Click Me" />
  </body>

</html>
```

**ext\_script.js**

```
function Welcome() {
  alert("Welcome to Java Script");
}
```

# JavaScript functionalities

- Java script Can Change HTML Content
- Java script Can Change HTML Attributes
- Java script Can Change HTML Styles (CSS)
- Java script Can Hide HTML Elements
- Java script Can Show HTML Elements

# JavaScript functionalities

## Example

sc1.html

File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/sc1.html


### Welcome to java script

**JavaScript can change HTML attributes**

Good Morning

Click Reset

**JavaScript changes the src (source) attribute of an image**



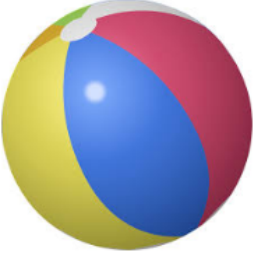
Turn on the light Turn off the light

**JavaScript can change the style of an HTML element**

Have a Great day

Change Font Change Color Change Style Reset

**JavaScript can hide HTML elements & can unhide hidden elements**



Hide Unhide

Type here to search

20:50  
26-09-2020

```
<html>
  <body>
    <script language="javascript" type="text/javascript">
      function reset()
      {
        document.getElementById("para2").style.color='black';
        document.getElementById("para2").style.fontSize='16px';
        document.getElementById("para2").style.fontStyle = 'normal';
      }
    </script>

    <h1 align="center">Welcome to java script</h1>

    <h3>JavaScript can change HTML attributes</h3>

    <p id="para1">Good Morning</p>

    <button type="button" onclick='document.getElementById("para1").innerHTML = "Welcome Friends!"'>Click</button>
    <button type="button" onclick='document.getElementById("para1").innerHTML = "Good Morning"'>Reset</button>
    <br>

    <h3>JavaScript changes the src (source) attribute of an image</h3>

    <button onclick="document.getElementById('Img').src='bulb_on.png'">Turn on the light</button>

    <button onclick="document.getElementById('Img').src='bulb_off.png'">Turn off the light</button>

    <br>

    <h3>JavaScript can change the style of an HTML element</h3>

    <p id="para2">Have a Great day</p>

    <button type="button" onclick="document.getElementById('para2').style.fontSize='35px'">Change Font</button>
    <button type="button" onclick="document.getElementById('para2').style.color='red'">Change Color</button>
    <button type="button" onclick="document.getElementById('para2').style.fontStyle = 'italic'">Change Style</button>
    <button type="button" onclick="reset()">Reset</button>

    <br>
    <h3>JavaScript can hide HTML elements & can unhide hidden elements</h3>
    </img>
    <button type="button" onclick="document.getElementById('Img1').style.display='none'">Hide</button>
    <button type="button" onclick="document.getElementById('Img1').style.display='block'">Unhide</button>
  </body>
</html>
```

# Advantages of Java Script

- Less server interaction
- Immediate feedback to the visitors
- Increased interactivity
- Richer interfaces

# Limitations of Java Script

- Client-side script does not allow the reading (or) writing of files (security reason)
- It cannot be used for networking applications because there is no such support available
- It does not have any multithreading (or) multiprocessor capabilities



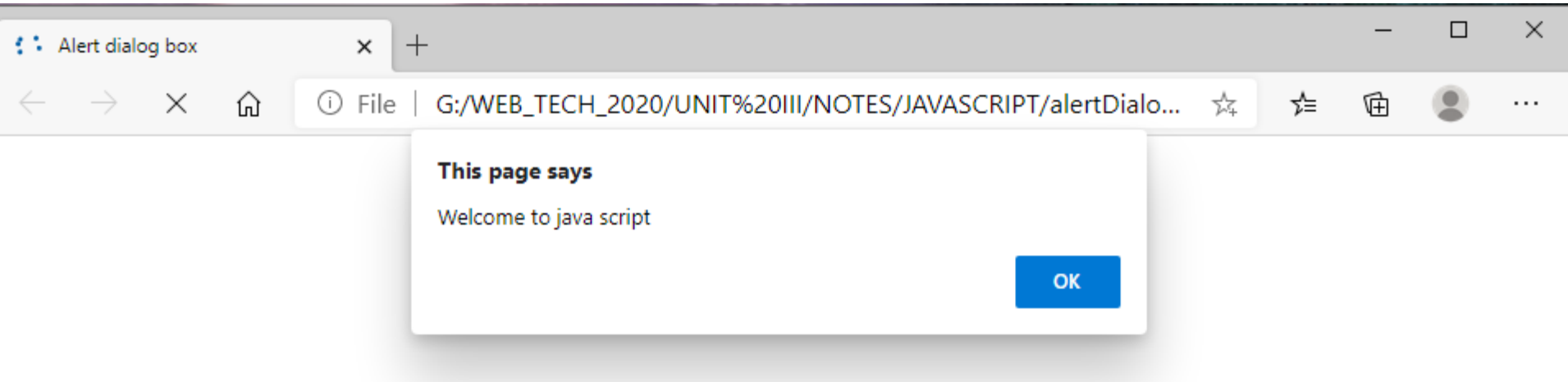
# Dialog Boxes

- Java script provides the ability to pickup user input (or) display small amounts of text to the user by using dialog boxes.
- These dialog boxes appear as separate windows & their content depends on the information provided by the user.
- This content is independent of the text in the HTML page containing java script
  - Alert Dialog
  - Prompt Dialog Box
  - Confirm Dialog Box

# Alert Dialog box

[**alert("Message")**] - used to display a cautionary message (or) display some info.]

## Example



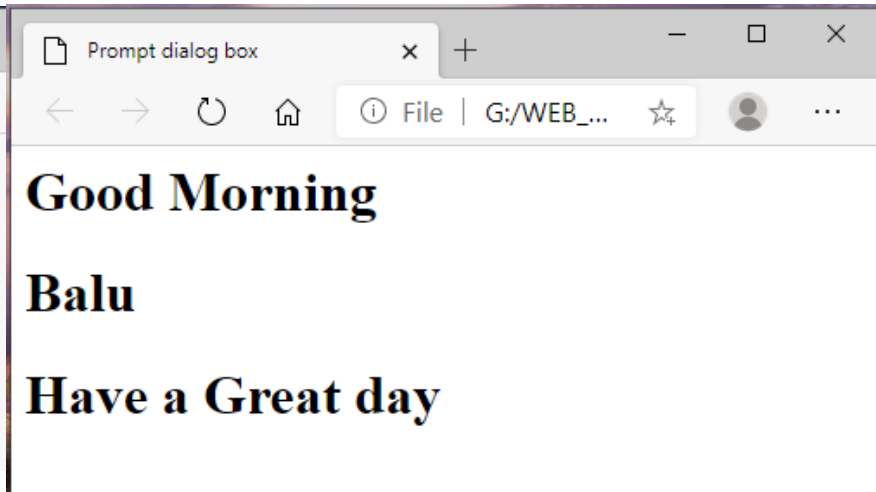
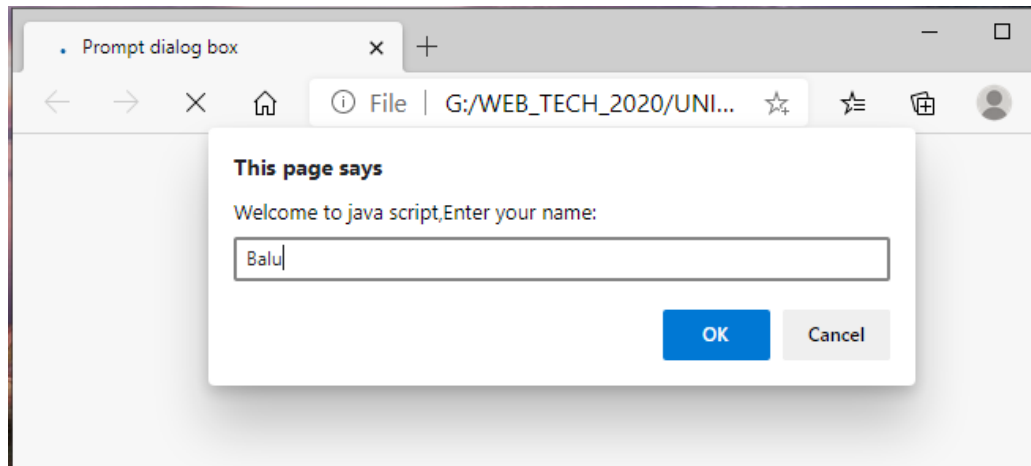
```
<html>
  <head><title>Alert dialog box</title></head>
  <body>
    <script language="javascript" type="text/javascript">
      alert("Welcome to java script")
      document.write("Welcome")
      document.write('</img>')
    </script>
  </body>
</html>
```



# Prompt Dialog Box

`prompt("message", "default value")` - Instantiates the prompt dialog box which displays a specified field. It also provides a single data entry field which accepts user input.

## Example



```
<html>
  <head>
    <title>Prompt dialog box</title>

  <head>
  <body>
    <script language="javascript" type="text/javascript">
      var name="";
      name=prompt("Welcome to java script,Enter your name:", "Name");
      document.write( <h1>Good Morning</h1><h1> );
      document.write(name)
      document.write('</h1><h1>Have a Great day</h1>');
    </script>
  </body>
</html>
```

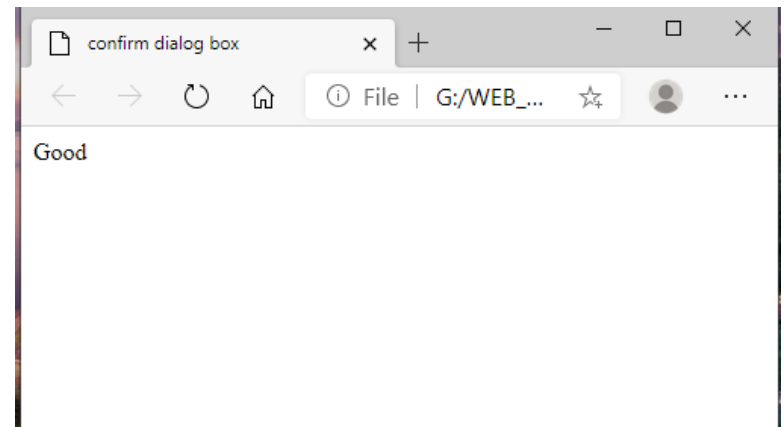
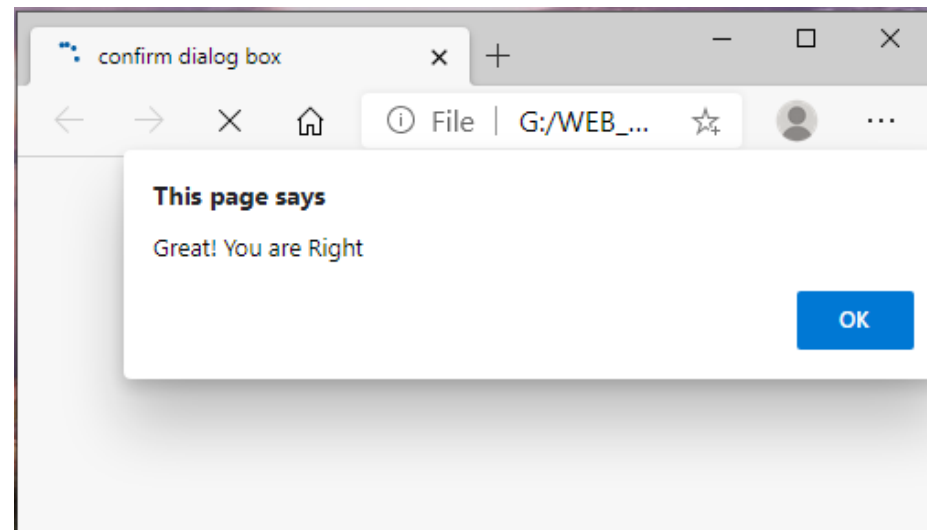
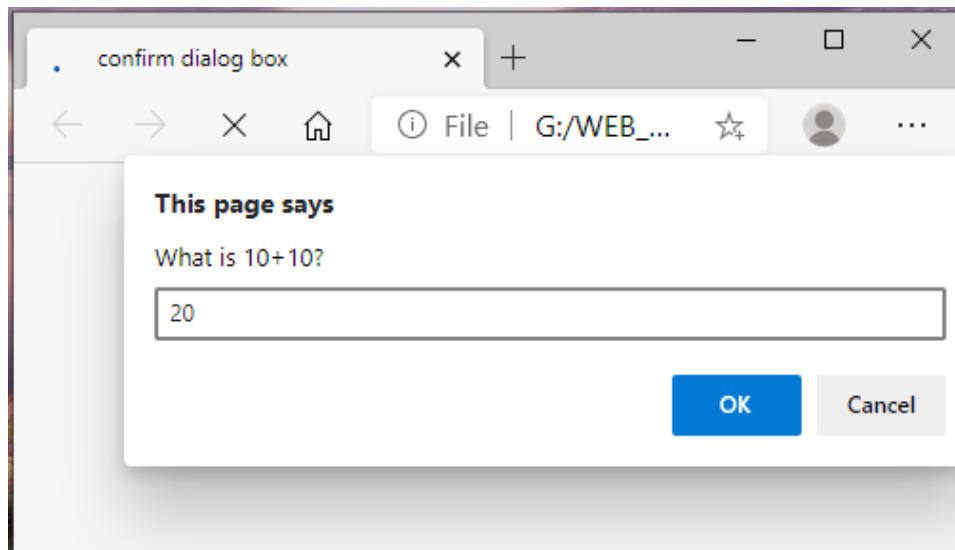
# Confirm Dialog Box

`confirm("message")` - This box serves as a technique for confirming user action.

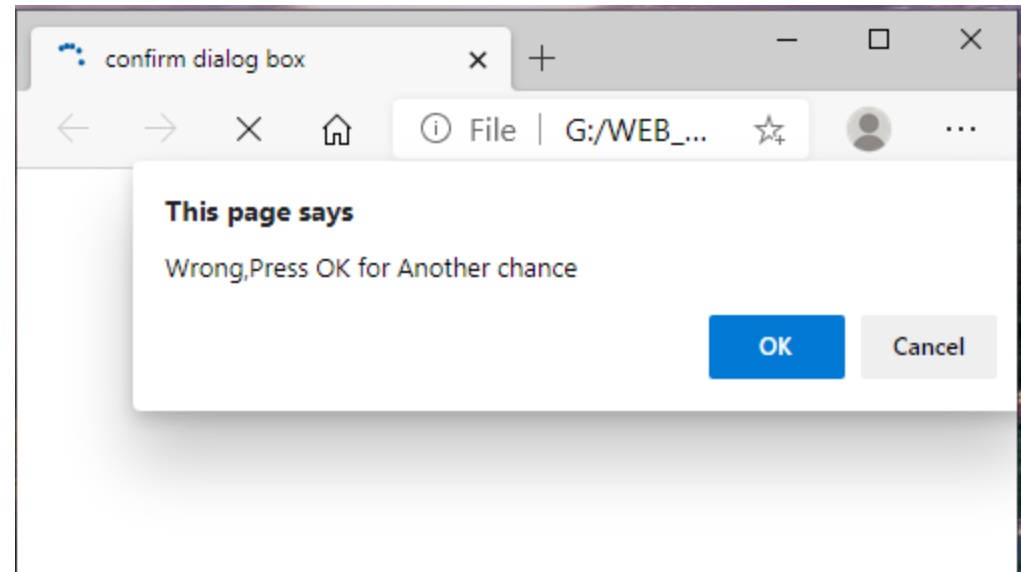
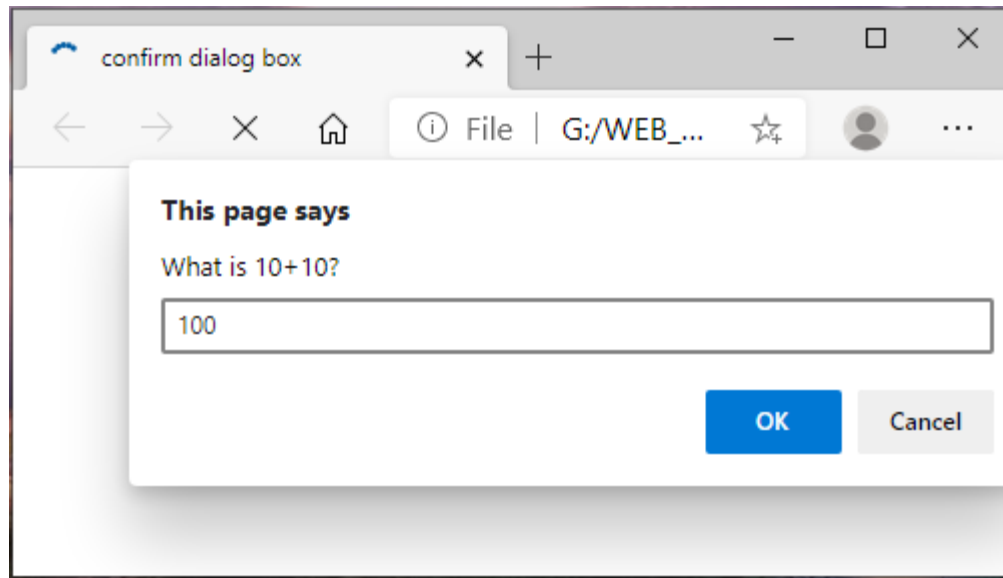
OK – pass True

CANCEL – Pass False

## Example



# Confirm Dialog Box Cont'd

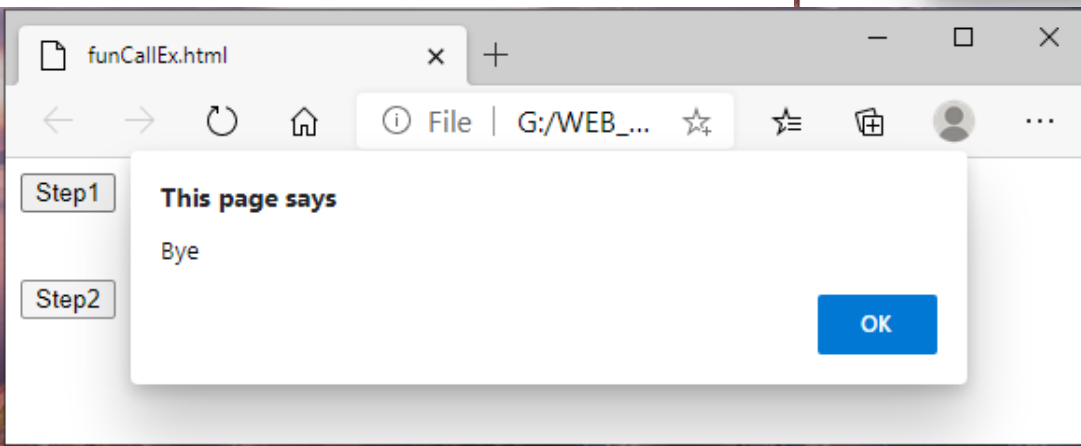
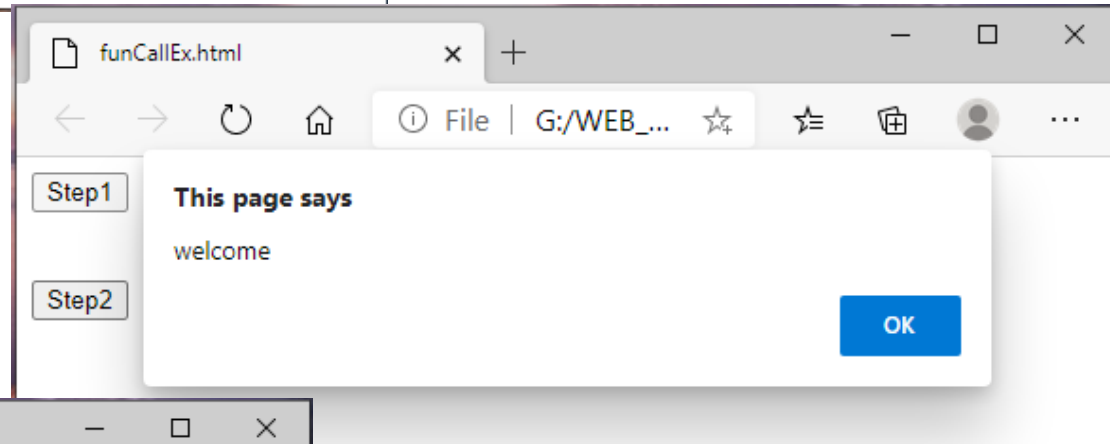
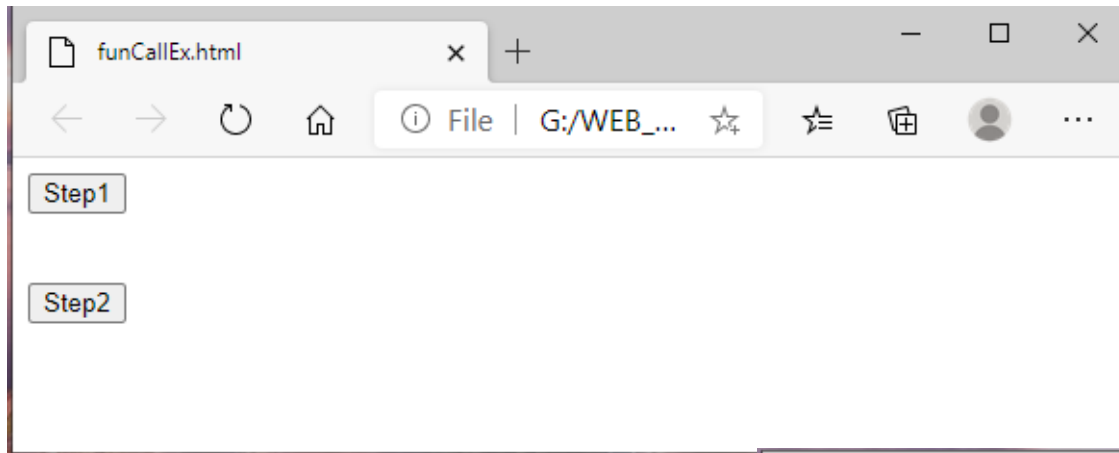


# Confirm Dialog Box Cont'd

```
<html>
  <head>
    <title>confirm dialog box</title>
    <script language="javascript">
      var question="What is 10+10?";
      var answer=20;
      var correct="Good";
      var incorrect="Bye";
      var response=prompt(question,"0");
      for(ct=0;ct<3;ct++)
      {
        if(response!=answer)
        {
          confirm("Wrong,Press OK for Another chance");
          response=prompt(question,"0");
          if(ct==1)
          {
            alert("Better Luck Next Time");
          }
        }
        else
        {
          alert("Great! You are Right");
          ct=3;
        }
      }
      var output=(response==answer)?correct:incorrect
      document.write(output);
    </script>
  </head>
</html>
```

# Function call in Java Script

Example



# Function call in Java Script Cont'd

```
<html>
  <head>
    <script language="javascript">
      function welcome()
      {

        alert("welcome");
      }
      function bye()
      {

        alert("Bye");
      }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="Step1" onclick="welcome()"/>
      <br><br><br>
      <input type="button" value="Step2" onclick="bye()"/>
    </form>
  </body>
</html>
```



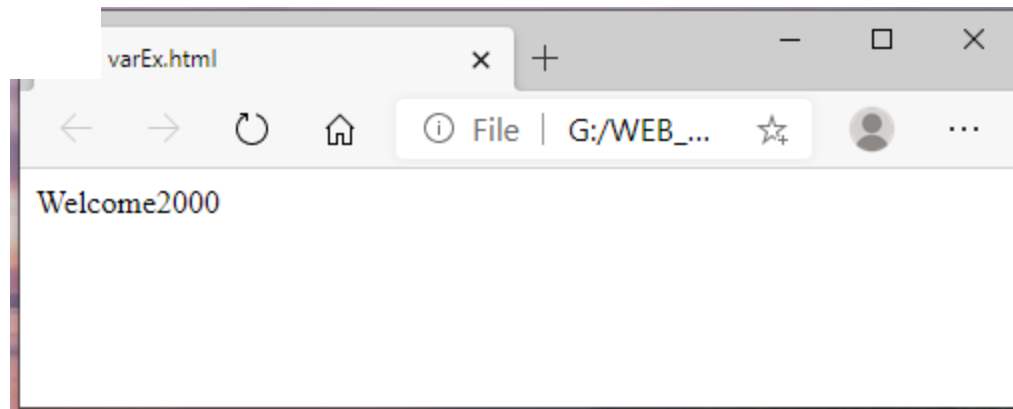
# JavaScript Variables

- JavaScript Variable is an object(or a named memory space) that is **used to store a value that can be used during program execution**. A variable in JavaScript, just like in other programming languages, has a **name**, a **value**, and a **memory address**.
- We can use either var or let keyword to define variables.
- ***Rules for naming:***
  - Variable names cannot contain spaces.
  - The first letter of the variable can be [a-z, A-Z], dollar sign (\$) or underscore(\_), after the first letter of name any digit [0-9] can be used.
  - Variable names are case sensitive. For example: var a and var A both are different.
  - We can not use reserved words as the name of the variables in JavaScript.
- **Types of Variables**
  - Local Variable
    - a variable that is **declared inside a code block or a function body or inside a loop body** and it has scope within the code block or the function.
  - Global Variable
    - a variable that is **declared anywhere inside the script** and **has scope for the complete script execution**.

# JavaScript let vs var Keyword

- The let and var, both keywords are used to declare variables, but the main difference is the scope of the declared variables.
- A variable declared inside a block using var is accessible outside of the block as it has a global scope but a variable declared using the let keyword has a local scope.

```
<html>
<head>
  <script type = "text/JavaScript">
    {
      let amount = 2500;    // block Scope
      var withdraw = 2000;  // global scope
    }
    document.write("Welcome");
    document.write(withdraw) // accessible
    document.write(amount)  // not accessible
  </script>
</head>
</html>
```



# JavaScript Data Types

- **Data types are used to identify the type of data that is stored inside a variable** during the script execution.
- Since **java script is dynamically typed** we do not have to specify the data type of the variable while declaring it.
- **It allows storing different types of values to a variable during programming.**

```
var n = 15; // n is a number  
n = "Good"; // n is string here
```

- JavaScript broadly supports three types of Data types, they are:
  - Primitive Type
  - Reference Type
  - Special Data Type

# JavaScript Primitive Data Type

- It can be classified further into the following types:
- **String Data Type**
  - a character or sequence of characters **inside single or double quotes**  

```
var str1 = "Apple"; // string using double quotes
```

```
var str2 = 'Grape fruit'; // string using single quotes.
```
- **Boolean Data Type**
  - It can have two values, either **true** or **false**.
- **Number Data Type**
  - numbers can be **with or without decimal points** and can have negative and positive values.
  - also represents some special values like **Infinity, -Infinity and Nan**
  - When a positive number is divided by zero (it's a popular case of runtime error), in JavaScript it's represented as Infinity. Similarly, when a negative number is divided by zero we will get -Infinity.
  - Nan means **Not a number**, if we try to perform any operation between a numeric value and a non-numeric value like a string we will get this as output.

# JavaScript Composite Data types

- These data types **can hold collections of values and more complex entities.**
- It is further divided into,
  - Object data type
  - Array data type
  - Function data type
- **Object data type:**
  - It is used to store the collection of data.
  - Object's properties are written as **key:value** pairs, which are **separated by commas** and **enclosed within curly braces {}**.
  - The **key (name)** must always be a string, but the **value** can be of any data type.

```
var name = { }; // It will create an empty object.  
var stu = {firstname="Balu", id=111,dept="ECE"};
```

# JavaScript Composite Data types Cont'd

- **Array data Type:**

- It is written inside a pair of square brackets [] and is used to store multiple values of the same datatype be it strings, numbers etc.
- The items in a JavaScript array are also written in a comma-separated manner.
- Each element in the array gets a numeric position, known as its **index**.
- The array index starts from **0** or we can say that **array indexes are zero-based**, so that the first array element is arr[0] and not arr[1].

// Creating an Array

```
var fruits=["Apple","Banana","Orange",Grapes"];
```

# JavaScript Composite Data types Cont'd

- **Function data Type:**

- In JavaScript **functions act as a data type which can be assigned to a variable.**
- JavaScript Function is nothing but **a set of statement inside a code block** which is used to perform a specific operation and **this data type is of callable in nature.**
- Since functions are objects, so it is possible to assign them to a variable.
- Functions can be stored in variables, objects, and arrays.
- Functions can be passed as arguments to other functions too and can be returned from other functions as well.

```
var welcome = function()  
{  
    return "Welcome to Java Script!";  
}
```

# JavaScript Special Data types

- JavaScript also has some special data types

- **Undefined Data type**

- When a variable is just declared and is not assigned any value, it has **undefined** as its value.
- It is a valid data type in JavaScript and it can have only one value which is **undefined**.
- We can even use this value while doing some comparison

```
var x; // Undefined  
alert(x == undefined); // returns true
```

- **Null Data type**

- It is used to represent no value.
- It is not similar to undefined, and neither it is similar to empty value or zero value.
- The Null data type means, the variable has been defined but it contains no value.
- The Null data type can have only one value, which is **null**.

```
var x = null;  
alert(x); // Output will be null
```



# Java Script Comment

- Single-line Comment
- Multi-line Comment

**<script>**

// It is single line comment

/\* It is multi line comment.

It will not be displayed \*/

document.write("JavaScript Comment");

**</script>**

# Java Script Operators

- Operators are symbols that are used to perform operations on operands
- There are following types of operators in JavaScript.
  - Arithmetic Operators
  - Comparison (Relational) Operators
  - Bitwise Operators
  - Logical Operators
  - Assignment Operators
  - Special Operators

# Java Script Arithmetic Operators

- Arithmetic operators are used to perform arithmetic on numbers

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

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

```
var txt1 = 5  
var txt2 = 5;  
alert(txt1+txt2);
```

Output:10

```
var txt1 = "Good"  
var txt2 = "Bye";  
alert(txt1+txt2);
```

Output:GoodBye

```
var txt1 = "Good"  
var txt2 = 5;  
alert(txt1+txt2);
```

Output:Good5

# JavaScript Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

Operator	Description	Example
==	Is equal to (perform type conversion before testing for equality)	15==17 → false 15==15 → true 15=="15" → true
===	equal value and equal type (do not perform type conversion before testing for equality)	15===17 → false 15===15 → true 15=== "15" → false
!=	Not equal to (perform type conversion before testing for equality)	15!=17 → true
!==	Not equal value and equal type (do not perform type conversion before testing for equality)	15!==15 → false 15 !== "15" → true 15 !== 18 → true
>	Greater than	15>13 → true
>=	Greater than or equal to	15>=10 → true
<	Less than	15<10 → false
<=	Less than or equal to	15<=10 → false

# JavaScript Logical Operators

Operator	Description	Example
&&	Logical AND	(15==15 && 20==33) → false
	Logical OR	(15==15    20==33) → True
!	Logical Not	!(10==20) → true

# JavaScript Bitwise Operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shifts left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shifts right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off
>>>	Zero fill right shift	Shifts right by pushing zeros in from the left, and let the rightmost bits fall off

# JavaScript Assignment Operators

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

# JavaScript Special Operators

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

# JavaScript If-else

- The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*.
- There are three forms of if statement in JavaScript.

- **If Statement**

```
if(expression){  
  //content to be evaluated  
}
```

- **If else statement**

```
if(expression){  
  //content to be evaluated if condition is true  
}  
else{  
  //content to be evaluated if condition is false  
}
```

- **if else if statement**

```
if(expression1){  
  //content to be evaluated if expression1 is true  
}  
else if(expression2){  
  //content to be evaluated if expression2 is true  
}  
else{  
  //content to be evaluated if no expression is true  
}
```



# JavaScript Switch

- The **JavaScript switch statement** is used *to execute one code from multiple expressions*. It is just like else if statement that we have learned in previous page. But it is convenient than *if..else..if* because it can be used with numbers, characters etc.

```
switch(expression)
```

```
{
```

```
case value1:
```

```
code to be executed;
```

```
break;
```

```
case value2:
```

```
code to be executed;
```

```
break;
```

```
.....
```

```
default:
```

```
code to be executed if above values are not matched;
```

```
}
```

# JavaScript Loops

- It is used *to iterate the piece of code*
- There are four types of loops in JavaScript.

## ➤ **for loop**

```
for (initialization; condition; increment)
{
    code to be executed
}
```

## ➤ **while loop**

```
while (condition)
{
    code to be executed
}
```

## ➤ **do-while loop**

```
do{
    code to be executed
}while (condition);
```

# JavaScript Loops Cont'd

## ➤ for-in loop

→ loops through the properties of an object

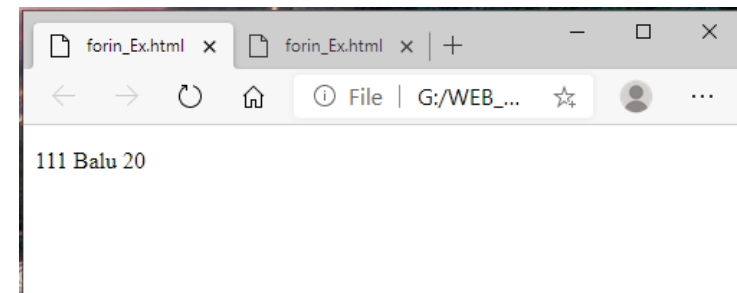
→ It automatically iterates over the fields of an object and the loop stops when all the fields are iterated.

### Syntax:

```
for(key in object)
{
    // code statements
}
```

### Example

```
<html>
<body>
    <p id="para"></p>
    <script>
        var txt = "";
        var student = {id:111,name:"Balu",age:20};
        var x;
        for (x in student)
        {
            txt += student[x] + " ";
        }
        document.getElementById("para").innerHTML = txt;
    </script>
</body>
</html>
```



# JavaScript Loops Cont'd

## ➤ for...of Loop

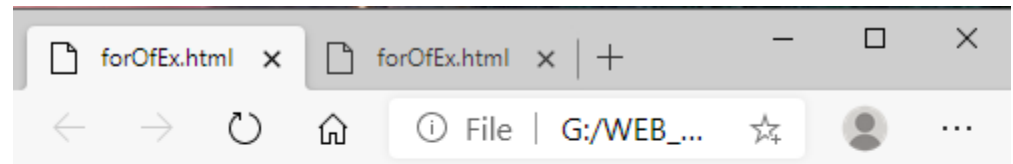
- This loop lets us to loop through the values of an iterable object like an array, strings and more.

### Syntax:

```
for(variable of iterable)
{
    //code here
}
```

## Example

```
<html>
  <body>
    <script>
      var fruits =['Apple','Banana','Grapes','Orange'];
      var x;
      for (x of fruits)
        document.write(x+"<br>");
    </script>
  </body>
</html>
```



Apple  
Banana  
Grapes  
Orange

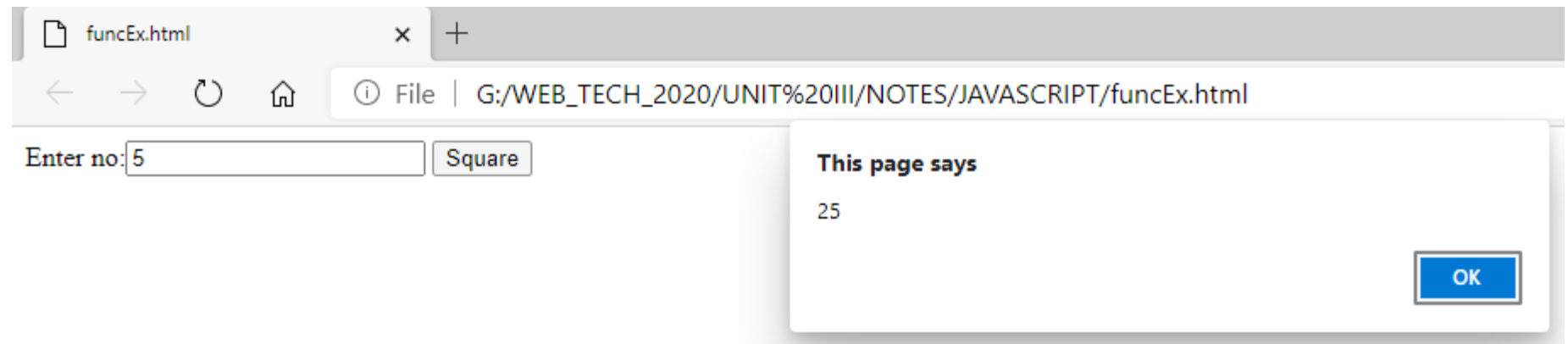
# JavaScript Functions

- A JavaScript function is a **block of code designed to perform a particular task.**
- A JavaScript function is executed when "something" invokes it (calls it).
- **Syntax:**

```
function functionName([arg1, arg2, ...argN])  
{  
  //code to be executed  
}
```

# JavaScript Functions Cont'd

## Example



```
<html>
  <head>

    <script language="javascript" type="text/javascript">
      function getSquare(n)
      {
        alert(n*n);
      }
    </script>
  </head>
  <body>
    <form>
      Enter no:<input type="text" id="textbox" value=""/>
      <input type="button" value="Square" onclick="getSquare((document.getElementById('textbox').value))"/>
    </form>
  </body>
</html>
```

# JavaScript Events

- The change in the state of an object is known as an **Event**.
- In JavaScript, events refer to the actions that are detected by a web browser whenever it detects any user movement on the browser screen.
- These are the **building blocks of an interactive webpage**.
- In html, there are various events which represents that some activity is performed by the user or by the browser.
- When java script code is included in HTML, js react over these events and allow the execution.
- This process of reacting over the events is called **Event Handling**.
- Thus, js handles the HTML events via **Event Handlers**.
- **For example**, when a user clicks over the browser, add js code, which will execute the task to be performed on the event.

# JavaScript Events Cont'd

- **Mouse events**

Event Performed	Event Handler	Description
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.



# JavaScript Events Cont'd

## Keyboard events:

Event Performed	Event Handler	Description
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

## Form events:

Event Performed	Event Handler	Description
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

# JavaScript Events Cont'd

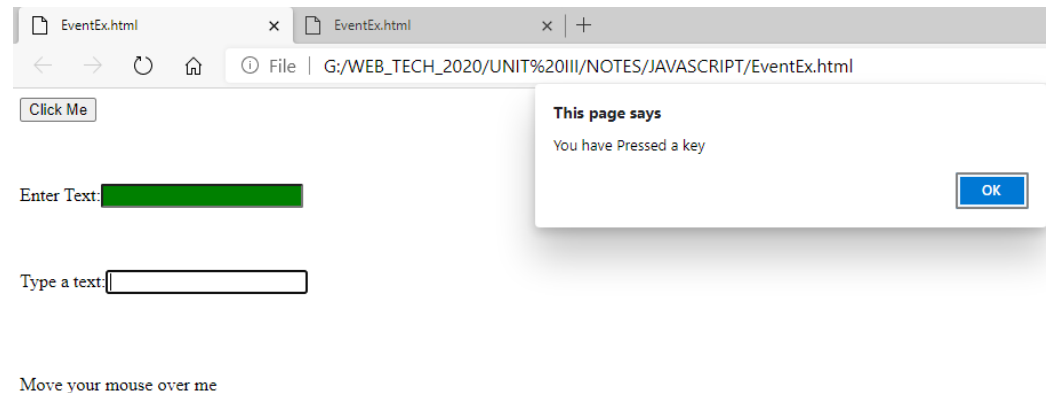
## Window/Document events

Event Performed	Event Handler	Description
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser

# JavaScript Events Example

```
<html>
<body>
<input type="button" onclick="clিকেvent()" value="Click Me"/>
<br><br><br><br>
Enter Text:<input type="text" id="input1" onfocus="focusevent()"/>
<br><br><br><br>
Type a text:<input type="text" id="input2" onkeydown="keydownevent()"/>
<br><br><br><br>
<p onmouseover="mouseoverevent()"> Move your mouse over me</p>
<script language="Javascript" type="text/Javascript">
```

```
function clickevent()
{
    alert("You have Clicked the mouse");
}
function focusevent()
{
    document.getElementById("input1").style.background=" green";
}
function keydownevent()
{
    document.getElementById("input2");
    alert("You have Pressed a key");
}
function mouseoverevent()
{
    alert("You have Moved your mouse");
}
</script>
</body>
</html>
```



# JavaScript Objects

- JavaScript object is an entity having state and behavior (properties and method).
- 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 directly create objects.**

## Creating Objects in JavaScript

- There are 3 ways to create objects.
  - By object literal
  - By creating instance of Object directly (using new keyword)
  - By using an object constructor (using new keyword)

# JavaScript Objects Cont'd

- **JavaScript Object by object literal**

**Syntax :**

object={property1:value1,property2:value2.....propertyN:valueN}

<script>

stu={id:111,name:"Balu",Avg:89}

document.write(stu.id+" "+ stu.name+" "+ stu. Avg);

</script>

- **By creating instance of Object**

**Syntax:** var objectname=new Object();

<script>

var stu =new Object();

stu.id=111;

stu.name=" Balu ";

stu. Avg =89;

document.write(stu.id+" "+ stu.name+" "+ stu. Avg);

</script>

# JavaScript Objects Cont'd

- **By using an Object constructor**

- Here, we need to create function with arguments.
- Each argument value can be assigned in the current object by using this keyword.
- The **this keyword** refers to the current object.

**<script>**

```
function stu(id,name,Avg)
```

```
{
```

```
    this.id=id;
```

```
    this.name=name;
```

```
    this.Avg=Avg;
```

```
}
```

```
s=new stu(111,"Balu",89);
```

```
document.write(s.id+" "+s.name+" "+s.Avg);
```

**</script>**

# JavaScript Built-in Objects

- JavaScript has rich set of built-in objects that can be used to deal with various types of collections.
- Some commonly used built-in types,
  - String
  - RegExp
  - Boolean
  - Number
  - Array
  - Math
  - Date

# JavaScript Array

- **JavaScript array** is an object that represents a collection of similar type of elements.
- There are 3 ways to construct array in JavaScript
  - By array literal
  - By creating instance of Array directly (using new keyword)
  - By using an Array constructor (using new keyword)

- **JavaScript array literal**

- **Syntax** : `var arrayname=[value1,value2.....valueN];`

```
<script>
```

```
var stu=[ "Arun","Balu","Vijay"];
```

```
for (i=0;i<stu.length;i++)
```

```
{
```

```
document.write(emp[i] + "<br/>");
```

```
}
```

```
</script>
```



# JavaScript Array Cont'd

- **JavaScript Array directly (new keyword)**

- **Syntax** : `var arrayname=new Array();`

- Here, **new keyword** is used to create instance of array.

`<script>`

`var i;`

`var stu = new Array();`

`stu[0] = "Arun";`

`stu[1] = "Balu";`

`stu[2] = "Vijay";`

`for (i=0;i< stu.length;i++)`

`{`

`document.write(emp[i] + "<br>");`

`}`

`</script>`

# JavaScript Array Cont'd

- **JavaScript array constructor (new keyword)**
- Here, we need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<script>
```

```
var stu=new Array("Ajay","Balu","Vijay");
```

```
for (i=0;i<stu.length;i++)
```

```
{
```

```
document.write(stu[i] + "<br>");
```

```
}
```

```
</script>
```

# JavaScript String Object

- The **JavaScript string** is an object that represents a sequence of characters.
- There are 2 ways to create string in JavaScript
  - By string literal → `var stringname="string value";`
  - By string object (using new keyword)  
→ `var stringname=new String("string literal");`
- **String Methods**
  - `charAt(index)`
  - `concat(str)`
  - `indexOf(str)`
  - `lastIndexOf(str)`
  - `toLowerCase()`
  - `toUpperCase()`
  - `slice(beginIndex, endIndex)`
  - `trim()`

# String Object Methods Example

## Example

```
<html>
  <body>
    <script>
      var str1="Welcome"
      document.write(str1.charAt(1)+"<br>");
      var str2="Friends"
      var str3=str1.concat(str2);
      document.write(str1+"<br>");
      document.write(str3+"<br>");
      var str4="Today is a wonderful day";
      document.write(str4.indexOf('day')+"<br>");
      document.write(str4.lastIndexOf('day')+"<br>");
      document.write(str1.toLowerCase()+"<br>");
      document.write(str1.toUpperCase()+"<br>");
      document.write(str1.slice(2,4)+"<br>");
      str1="      Friends      "
      document.write(str1+"<br>")
      document.write(str1.trim()+"<br>");
      document.write(str4.split(" ")+"<br>");
    </script>
  </body>
</html>
```

## Output

```
e
Welcome
WelcomeFriends
2
21
welcome
WELCOME
lc
Friends
Friends
Today,is,a,wonderful,day
```

# JavaScript Date Object

- The **JavaScript date** object can be used to get year, month and day.
- Constructor
  - Date()
  - Date(milliseconds)
  - Date(dateString)
  - Date(year, month, day, hours, minutes, seconds, milliseconds)
- Date Object Methods

Method	Description
Date.now()	Returns the number of milliseconds(Unix timestamp) for the current date and time.
Date.parse()	Parses a string value of date and returns the number of milliseconds since 1 January 1970, 00:00:00 UTC, with leap seconds ignored.
Date.UTC()	This function takes 7 parameters with numeric values but creates the date in UTC timezone.
setDate()	sets the date of the month that ranges from 1 to 31.

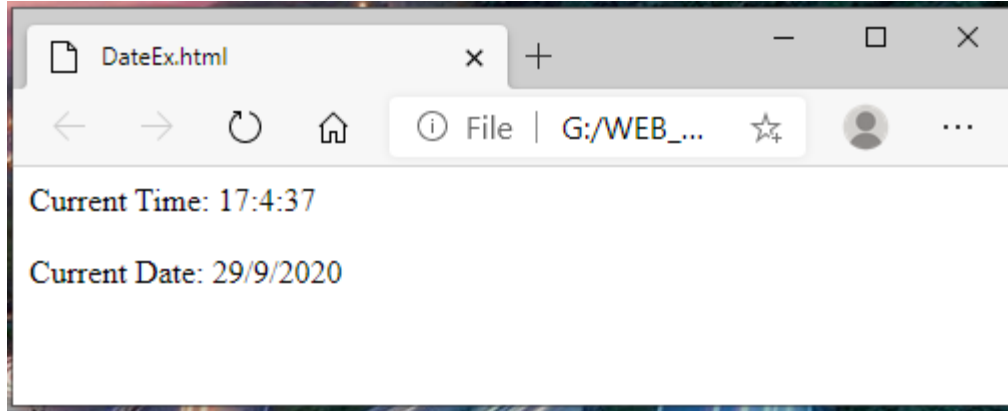
# Date Object Methods

Method	Description
setHours()	set hours that range from 0 to 23
getSeconds()	returns the seconds that range from 0 to 59
toDateStr()	converts a date value into a string
valueOf()	returns the primitive value of Date object.
setMinutes()	set minutes that range from 0 to 59.
setMonth()	set numerical equivalence of month range from 0 to 11
setMilliseconds()	set the milliseconds that range from 0 to 999
toTimeString()	converts time into a string.
getDate()	returns the day of the month from 1 - 31

# Date Object Methods

Method	Description
<code>getDay()</code>	returns the day of the week from 0 - 6
<code>getFullYear()</code>	returns the year (4 digits for 4-digit years) of the specified date
<code>getMinutes()</code>	returns the minutes (0–59) in the specified date
<code>getMonth()</code>	returns the month (0–11) in the specified date
<code>getHours()</code>	returns the hours that range from 0 to 23.

# Date Object Methods Example



**Example**

```
<html>
<body>
Current Time: <span id="txt1"></span>
<br><br>
Current Date: <span id="txt2"></span>
<script>
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
var day=today.getDate();
var month=today.getMonth()+1;
var year=today.getFullYear();
document.getElementById('txt1').innerHTML=h+":"+m+":"+s;
document.getElementById('txt2').innerHTML=day+"/"+month+"/"+year;
</script>
</body>
</html>
```



# JavaScript Math Object

- It allows to perform mathematical tasks on numbers.

Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns the value of x rounded up to its nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of $E^x$
floor(x)	Returns the value of x rounded down to its nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Returns the value of x rounded to its nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle

# Math Object Methods Example

## Example

```
<html>
  <body>
    <script type="text/javascript">
      document.write("Floor :"+Math.floor(15.8)+"<br/>");
      document.write("Log :"+Math.log(15.8)+"<br/>");
      document.write("Max :"+Math.max(15.8,10,26,77,16)+"<br/>");
      document.write("Min :"+Math.min(13,78,76,89,5,17.8)+"<br/>");
      document.write("pow :"+Math.pow(5,2)+"<br/>");
      document.write("Random :"+Math.random()+"<br/>");
      document.write("Round :"+Math.round(17.7)+"<br/>");
      document.write("Sin:"+Math.sin(45)+"<br/>");
      document.write("Sqrt :"+Math.sqrt(49)+"<br/>");
      document.write("Tan:"+Math.tan(45)+"<br/>");
    </script>
  </body>
</html>
```

## Output

Floor :15  
Log :2.760009940032921  
Max :77  
Min :5  
pow :25  
Random :0.4903510683514902  
Round :18  
Sin:0.8509035245341184  
Sqrt :7  
Tan:1.6197751905438615

# JavaScript Number Object

- A Number object can be created using the Number() constructor.

```
let n = new Number(SOME_NUMERIC_VALUE);
```

```
let x = new Number(456.89);
```

- **Methods**

Method name	Description
Number.isNaN()	Static method; used to check whether the given value is NaN or not.
Number.isFinite()	Static method; used to check whether the given value is a finite number or not.
Number.isInteger()	Static method; used to check whether the given value is an integer number or not.
Number.isSafeInteger()	Static method; used to check if the given value is a safe value or not, i.e. between $2^{53} - 1$ to $-(2^{53} - 1)$
Number.parseFloat(string)	Static method; used to convert a string to a floating-point number.

# Number Object Methods

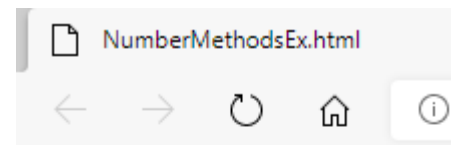
Method name	Description
<code>Number.parseInt(string, [radix])</code>	Static method; used to convert a string to an integer. Here radix represents the base in mathematical numeral systems.
<code>valueOf()</code>	Returns the primitive value of the Number object.
<code>toString()</code>	Returns a String value of number object.
<code>toFixed(x)</code>	Rounds up a number to x digits after the decimal.
<code>toPrecision(x)</code>	Rounds up a number to a length of x digits.
<code>toExponential(x)</code>	Converts a number into an Exponential notation.

# Number Methods

## Example

```
<html>
  <body>
    <script type="text/javascript">
      let n = new Number(123.67);
      document.write(typeof n + "<br/>");
      let N = n.valueOf();
      document.write(N + "<br/>");
      document.write(typeof N + "<br/>");
      let num = new Number('18.907456');
      document.write(num.toExponential() + "<br/>");
      document.write(num.toFixed() + "<br/>");
      document.write(num.toPrecision(5) + "<br/>");
      var s=num.toString();
      document.write(s + "<br/>");
      document.write(typeof s + "<br/>");
    </script>
  </body>
</html>
```

## Output



```
object
123.67
number
1.8907456e+1
19
18.907
18.907456
string
```

# JavaScript Boolean Object

- It is a member of global objects and a wrapper class. It is used to create a boolean object which holds **true** or **false** value, depending upon the value used while creating the Boolean object.
- To create an instance of the Boolean object, we use the new keyword with the Boolean object constructor

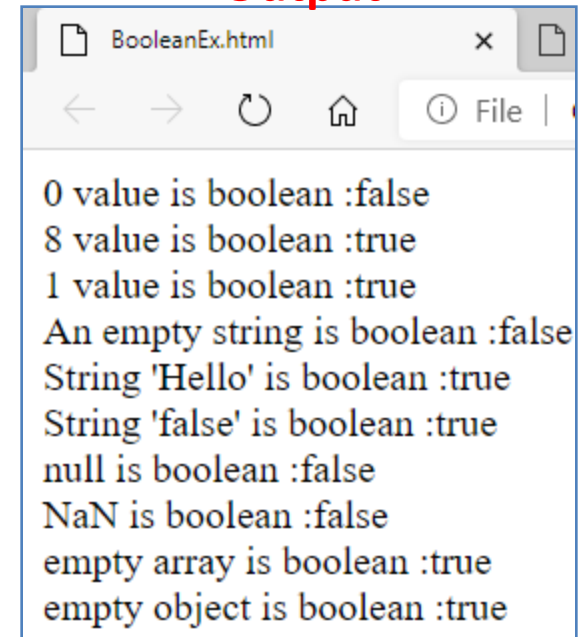
```
let b = new Boolean(SOME_VALUE);  
let boolObj = new Boolean(true);
```
- **Methods:**
  - **toString():** converts the boolean value into a string and returns the string.
  - **valueOf():** returns the primitive value of a Boolean object.

# Boolean Object Example

## Example

## Output

```
<html>
  <body>
    <script type="text/javascript">
      let bool1 = new Boolean(0);
      let bool2 = new Boolean(8);
      let bool3 = new Boolean(2);
      let bool4 = new Boolean("");
      let bool5 = new Boolean("Hello");
      let bool6 = new Boolean('false');
      let bool7 = new Boolean(null);
      let bool8 = new Boolean(NaN);
      let bool9 = new Boolean([]);
      let bool10 = new Boolean({});
      document.write("0 value is boolean :" + bool1+"<br>");
      document.write("8 value is boolean :" + bool2+"<br>");
      document.write("1 value is boolean :" + bool3+"<br>");
      document.write("An empty string is boolean :" + bool4+"<br>");
      document.write("String 'Hello' is boolean :" + bool5+"<br>");
      document.write("String 'false' is boolean :" + bool6+"<br>");
      document.write("null is boolean :" + bool7+"<br>");
      document.write("NaN is boolean :" + bool8+"<br>");
      document.write("empty array is boolean :" + bool9+"<br>");
      document.write("empty object is boolean :" + bool10+"<br>");
    </script>
  </body>
</html>
```



# JavaScript RegExp Object

- A regular expression is a sequence of characters that forms a **search pattern**.
- When we search for data in a text, we can use this search pattern to describe what we are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of **text search** and **text replace** operations.
- **It mainly helps us in validation.**
- We can use the JavaScript RegExp object to validate an e-mail entered in a form, or maybe check if a Pincode is all numeric, or a password has alphanumeric and special characters, etc.
- **Syntax**

*/pattern/modifiers;*



# JavaScript RegExp Object Cont'd

- Modifiers → are used to perform case-insensitive and global searches

Modifier	Description
g	Perform a global match (find all matches rather than stopping after the first match)
i	Perform case-insensitive matching
m	Perform multiline matching

- Brackets → are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets
[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x y)	Find any of the alternatives specified

# JavaScript RegExp Object Cont'd

- **Metacharacters** → are characters with a special meaning

Metacharacter	Description
\d	Find a digit
\s	Find a whitespace character
\b	Find a match at the beginning of a word like this: \bWORD, or at the end of a word like this: WORD\b
\uxxxx	Find the Unicode character specified by the hexadecimal number xxxx
.	Find a single character, except newline or line terminator
\w	Find a word character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd

# JavaScript RegExp Object Cont'd

• **Quantifiers** → define quantities

Quantifier	Description
<code>n+</code>	Matches any string that contains at least one <i>n</i>
<code>n*</code>	Matches any string that contains zero or more occurrences of <i>n</i>
<code>n?</code>	Matches any string that contains zero or one occurrences of <i>n</i>
<code>n{X}</code>	Matches any string that contains a sequence of <i>X</i> <i>n</i> 's
<code>n{X,Y}</code>	Matches any string that contains a sequence of <i>X</i> to <i>Y</i> <i>n</i> 's
<code>n{X,}</code>	Matches any string that contains a sequence of at least <i>X</i> <i>n</i> 's
<code>n\$</code>	Matches any string with <i>n</i> at the end of it
<code>^n</code>	Matches any string with <i>n</i> at the beginning of it
<code>?=n</code>	Matches any string that is followed by a specific string <i>n</i>
<code>?!n</code>	Matches any string that is not followed by a specific string <i>n</i>

# RegExp Object Methods

Method	Description
<code>exec()</code>	Tests for a match in a string. Returns the first match
<code>test()</code>	Tests for a match in a string. Returns true or false
<code>toString()</code>	Returns the string value of the regular expression

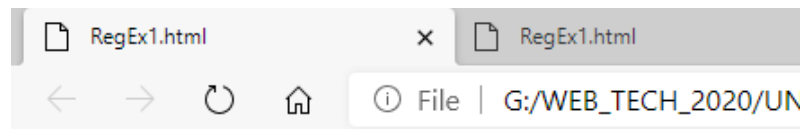
# RegExp Object Methods Example

## Example

```
<html>
<body>
    <p>Search for an "a" in the given paragraph:</p>
    <p id="p1">HAve A great day!</p>
    <p id="p2"></p>
    <p id="p3"></p>
    <p id="p4"></p>
    <script>
text = document.getElementById("p1").innerHTML;
document.getElementById("p2").innerHTML = /a/.test(text);

var x = /a/.exec(text);
document.getElementById("p3").innerHTML = "Found " + x[0] + " in position " + x.index + " in the text: " + x.input;

var x = /a/i.exec(text);
document.getElementById("p4").innerHTML = "Found " + x[0] + " in position " + x.index + " in the text: " + x.input;
    </script>
</body>
</html>
```



## Output

Search for an "a" in the given paragraph:

HAve A great day!

true

Found a in position 10 in the text: HAve A great day!

Found A in position 1 in the text: HAve A great day!

# JavaScript Form Validation

- It is important to validate the form submitted by the user because it can have inappropriate values.
- Form validation normally used to occur at the server, after the client had entered all the necessary data and then pressed the Submit button.
- If the data entered by a client was incorrect or was simply missing, the server would have to send all the data back to the client and request that the form be resubmitted with correct information.
- This was really a lengthy process which used to put a lot of burden on the server.
- JavaScript provides a way to validate form's data on the client's computer before sending it to the web server so processing will be fast than server-side validation. So, most of the web developers prefer JavaScript form validation.

# JavaScript Form Validation Cont'd

- Through JavaScript, we can validate name, password, email, date, mobile number etc fields
- Form validation generally performs two functions.
  - **Basic Validation**
    - First of all, the form must be checked to make sure all the mandatory fields are filled in.
    - It would require just a loop through each field in the form and check for data.
  - **Data Format Validation**
    - Secondly, the data that is entered must be checked for correct form and value.
    - Our code must include appropriate logic to test correctness of data.

# Automatic HTML Form Validation

- HTML form validation can be performed automatically by the browser
- Some **<input>** elements newly introduced in **HTML 5** have special attributes to help the browser to know how to **validate** its data automatically.

Attribute	Description
disabled	Specifies that the <b>Input</b> element should be disabled
max	Specifies the maximum value of an <b>Input</b> element
min	Specifies the minimum value of an <b>Input</b> element
pattern	Specifies the value pattern of an <b>Input</b> element
required	Specifies that the <b>Input</b> field requires an element
type	Specifies the type of an <b>Input</b> element

**Note:** Automatic HTML form validation does not work in Internet Explorer 9 or earlier.



# Automatic HTML Form Validation

autoValidationEx.html x +

File | G:/WEB\_TECH\_2020/

Enter your Name:

Enter your ID:

Enter your Mark:

Enter Password:

Submit

! Please fill out this field.

autoValidationEx.html x +

File | G:/WEB\_TECH\_2020/

Enter your Name:

Enter your ID:

Enter your Mark(0-100):

Enter Password:

Submit

! Please match the requested format.  
10 digit number

autoValidationEx.html x +

File | G:/WEB\_TECH\_2020/UNIT

Enter your Name:

Enter your ID:

Enter your Mark(0-100):

Enter Password:

Submit

! Value must be less than or equal to 100.

autoValidationEx.html x +

File | G:/WEB\_TECH\_2020/UNIT

Enter your Name:

Enter your ID:

Enter your Mark(0-100):

Enter Password:

Submit

! Please match the requested format.  
Invalid password!

# Automatic HTML Form Validation

## Example

```
<html>
  <body>
    <form name="myform" method="post" action="test.jsp" >
      Enter your Name: <input type="text" name="name" required><br/>
      Enter your ID: <input type="text" name = "id" pattern="[0-9]{10}" required title="10 digit number"/><br/>
      Enter your Mark(0-100): <input type="number" name="mark" min="0" max="100" required><br/>
      Enter Password: <input type="password" name = "password" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
                      title="Invalid password!" required/><br/>
      <!--Password must contain 8 or more characters that are of at least one number,
                      and one uppercase and lowercase letter-->
      <input type="submit" value="Submit">
    </form>
  </body>
</html>
```

# Data Validation

- Data validation is the process of ensuring that user input is clean, correct, and useful.
- Typical validation tasks are,
  - has the user filled in all required fields?
  - has the user entered a valid date?
  - has the user entered text in a numeric field?
- Most often, the purpose of data validation is to ensure correct user input.
- Validation can be defined by many different methods, and deployed in many different ways.
  - **Server side validation** is performed by a web server, after input has been sent to the server.
  - **Client side validation** is performed by a web browser, before input is sent to a web server.

# Data Validation Example1

ValidationEx1.html x +

← → ↻ 🏠 ⓘ File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/ValidationEx1.html

Name:

Password:

Re Enter Password:

Login

**This page says**

Name can't be blank

OK

ValidationEx1.html x +

← → ↻ 🏠 ⓘ File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/ValidationEx1.html

Name:

Password:

Re Enter Password:

Login

**This page says**

Password must be at least 6 characters long.

OK

ValidationEx1.html x +

← → ↻ 🏠 ⓘ File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/ValidationEx1.html

Name:

Password:

Re Enter Password:

Login

**This page says**

password must be same!

OK

## Data Validation Example1

```
<html>
  <head>
    <script>
      function validateform()
      {
        var name=document.myform.name.value;
        var password=document.myform.password.value;
        var nameErr = pswErr = rpswErr=true;
        if (name==null || name=="")
        {
          alert("Name can't be blank");
        }
        else
          nameErr=false;
        if(password.length<6)
        {
          alert("Password must be at least 6 characters long.");
        }
        else
          pswErr=false;
      }
    </script>
  </head>
</html>
```

```
var secondpassword=document.myform.passwordr.value;
if(password==secondpassword)
{
    rpswErr=false;
}
else
{
    alert("password must be same!");
}
if((nameErr || pswErr || rpswErr ) == true)
{
    return false;
}
else
    return true;
}
```

```
</script>
```

```
</head>
```

```
<body>
```

```
<form name="myform" method="post" action="test.jsp" onsubmit="return validateform()" >
```

```
Name: <input type="text" name="name"><br/>
```

```
Password: <input type="password" name="password"><br/>
```

```
Re Enter Password: <input type="password" name="passwordr"><br/>
```

```
<input type="submit" value="Login">
```

```
</form>
```

```
</body>
```

```
</html>
```

# Data Validation Cont'd

## Example2

ValidationEx3.html x +

← → ↻ 🏠 ⓘ File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/ValidationEx3.html

Name:

Email ID:

Login

**This page says**  
Enter valid name

OK

ValidationEx3.html x +

← → ↻ 🏠 ⓘ File | G:/WEB\_TECH\_2020/UNIT%20III/NOTES/JAVASCRIPT/ValidationEx3.html

Name:

Email ID:

Login

**This page says**  
Invalid Email ID

OK

```
<html>
  <head>
    <script>
      function validateform()
      {
        var name=document.myform.name.value;
        var email=document.myform.email.value;
        var nameErr = emailErr = true;
        var re1 = /^[a-zA-Z\s]+$/;
        let result1=re1.test(name);
        if (result1)
            nameErr = false;
        else
        {
            alert("Enter valid name");
        }
      }
    </script>
  </head>
</html>
```



```
let re2 = /^\\w+@[a-zA-Z_]+?\\. [a-zA-Z]{2,3}$/
let result2 = re2.test(email);
if(result2)
    emailErr = false;
else
{
    alert("Invalid Email ID");
}
if((nameErr || emailErr ) == true)
{
    return false;
}
else
    return true;
}
</script>
</head>
```

```
<body>
    <form name="myform" method="post" action="test.jsp"
onsubmit="return validateform()" >
    Name: <input type="text" name="name" ><br/>
    Email ID: <input type="text" name="email" ><br/>
    <input type="submit" value="Login">
    </form>
</body>
</html>
```