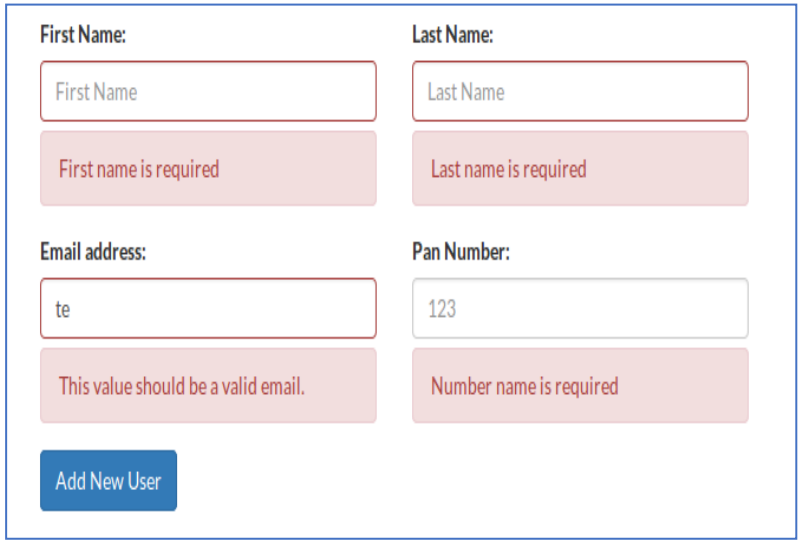
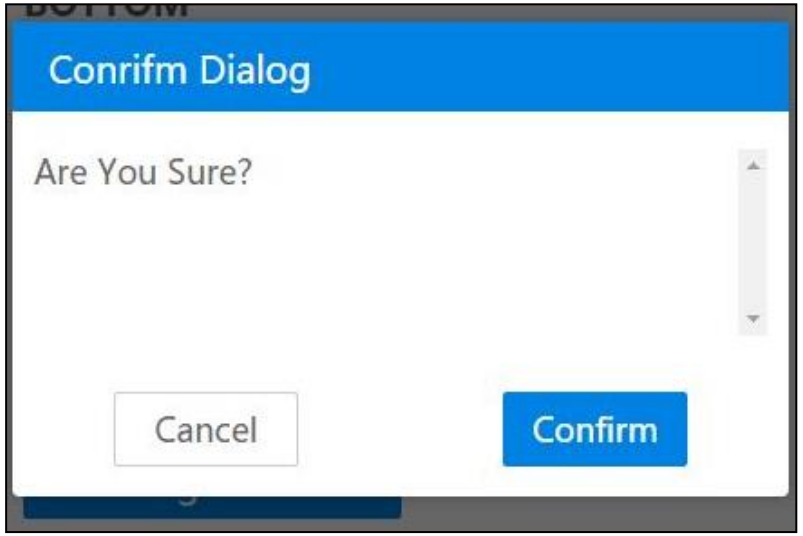
Introduction to Java-script

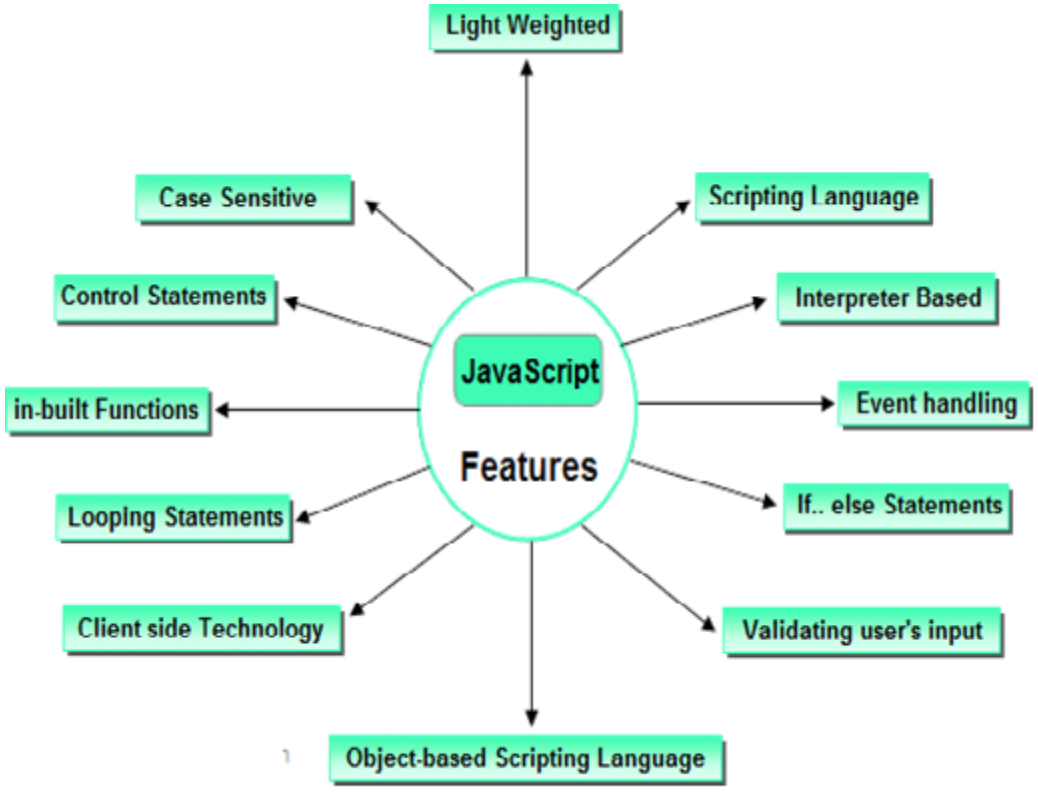
- JavaScript is most often used for client side validation.(i.e. form data)
- JavaScript is an **object based scripting language** designed to add interactivity (ex. pop-up alert, menu, windows, etc) to HTML pages
- A JavaScript is usually embedded directly into HTML pages.
- A JavaScript consists of lines of executable computer code.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)





Features of JavaScript

- Used to **validate form data** at client side.
- **Adds functionality** to a webpage
- It is **light weight programming language** , because Java script is return with very simple syntax.
- It is **interpreted language** , because script code can be executed without preliminary compilation.
- It Handle events like **onSubmit, onLoad, onClick, onMouseOver & etc** .
- An important part of JavaScript is the ability to **create new functions within scripts.**



Creating a java-script

Html **<script>** tag is used to script code inside the html page.

```
< script >
   JavaScript statements...
</ script >
```

A document can have multiple script tags, and each can enclose any number of JavaScript statements

The script is containing 2 attributes . They are

1) language attribute:-

It represents name of **scripting language** such as JavaScript, VbScript.

```
<script language=javaScript>
```

2) type attribute: - It indicates MIME (multi purpose internet mail extension) type of scripting code. It sets to an alpha-numeric MIME type of code.

```
<script type=text/javaScript>
```

Location of script or placing the script:

- Script code can be placed in both head & body section of html page.

Script in head section

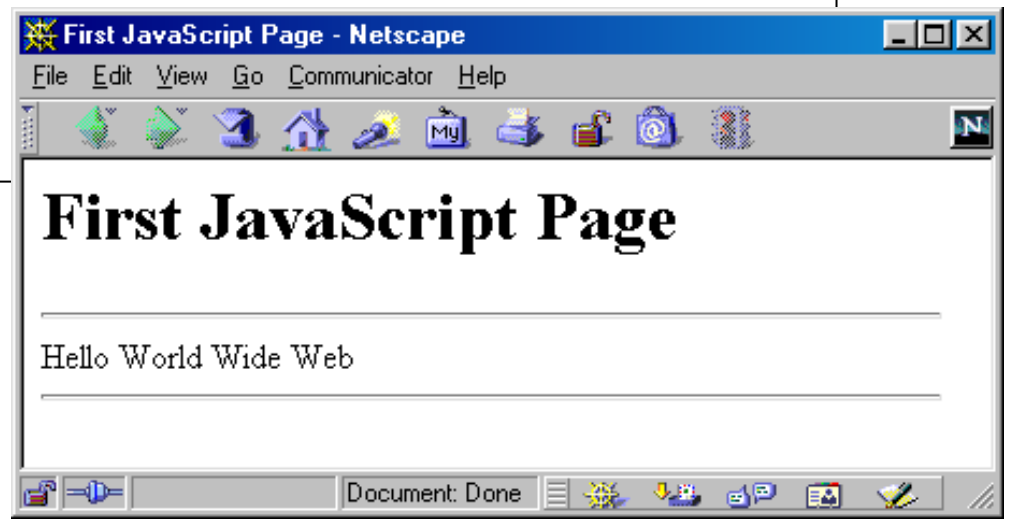
```
<html>
<head>
<script type="text/javascript">
    ..... Script code here
<script>
</head>
<body>
</body>
</html>
```

Script in body section

```
<html>
<body>
<script
type="text/javascript">
    ..... Script code here
<script>
</body>
</html>
```

A Simple Script

```
<html>
<head><title>First JavaScript Page</title></head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
  document.write("<hr>");
  document.write("Hello World Wide Web");
  document.write("<hr>");
</script>
</body>
</html>
```



Creating external script

- To **run same script on several pages** without having to write the script on each page.
- To simplify this, **write external script & save .js extension**. To use external script **specify .js file in src attribute of script tag**.

```
<html>
<head><title>First JavaScript Program</title></head>
<body>
<script type="text/javascript"
        src="your_source_file.js"></script>
</body>
</html>
```

Inside your_source_file.js

```
document.write("<hr>");
document.write("Hello World Wide Web");
document.write("<hr>");
```

- Use the **src** attribute to include JavaScript codes from an external file.
- The included code is inserted in place.

JavaScript code in the Dev Console

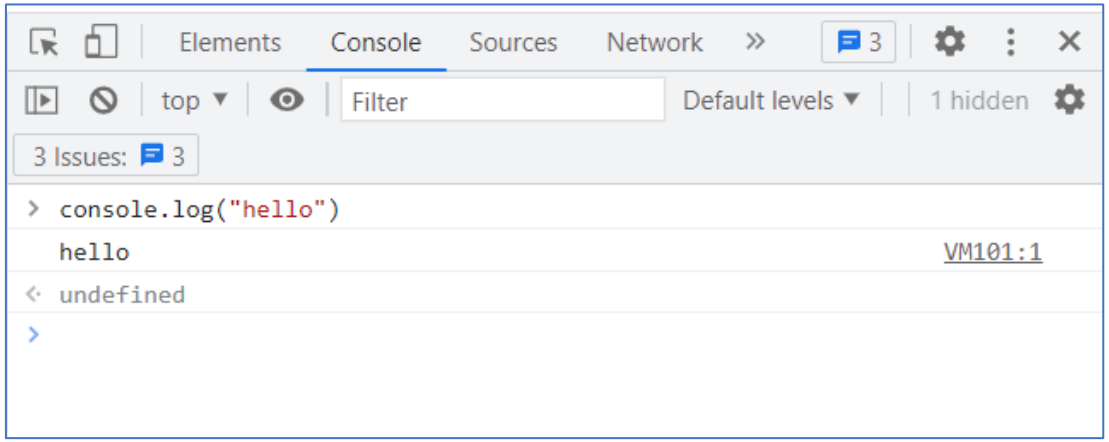
To run JavaScript code in the **Dev Console**, you can follow these steps:

1. Open the **Dev Tools** by pressing the **F12 key** (or Ctrl+Shift+I on Windows or Command+Option+I on Mac).
2. Click on the **"Console"** tab to view the console.
3. Type your JavaScript code directly into the console prompt and press Enter to run it.

For example:

- Type **console.log("Hello, world!");** into the console prompt and press Enter

- Output : **"Hello, world!"** to the console.



Declaring variable

Variable is a **memory location** where **data can be stored**.

- In java script variables with any **type of data** are declared by using the keyword **var**.
- All keywords are small letters only.
  - **var a; a=20;**
  - **var str; str= “Sunil”;**
  - **var c; c=“a“;**
  - **var d; d=30.7;**

But the keyword is not mandatory when declare of the variable.

- During the script, we can change value of variable as well as type of value of variable.
- **Ex:** a=20;
- a=30.7;

Declaring variable - JavaScript syntax rules:

- **JavaScript is case sensitive language.**

In this upper case lower case letters are differentiated (not same).

Ex: - a=20;
A=20;

The variable name "a" is different from the variable named "A"

Ex: - myf( ) // correct
myF( ) // incorrect

" ; " is optional in general JavaScript.

Ex: - a=20 // valid
b=30 // valid
A=10; b=40; // valid

- However, it is required when you put **multiple statements** in the same line.
- JavaScript **ignore white space**. In java script white space, tab space & empty lines are not preserved.
- To display special symbols we use \.

JavaScript Data Types

- **JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.**

    1. **Primitive data type**
    2. **Non-primitive (reference) data type**

- JavaScript is a **dynamic type language,** means you don't need to specify type of the variable because it is dynamically used by JavaScript engine.
- You need to use **var** here to specify the data type.
- It can hold any type of values such as numbers, strings etc. For example:

    - var a=40; //holding number
    - var b="Rahul"; //holding string

JavaScript primitive data types

- **There are five types of primitive data types in JavaScript. They are as follows:**

Data Type	Description	

String	represents sequence of characters
	e.g. "hello"
		`// Strings:`
		`let color = "Yellow";`
		`let lastName = "Johnson";`

Number	represents numeric values
	e.g. 100
		`// Numbers:`
		`let length = 16;`
		`let weight = 7.5;`

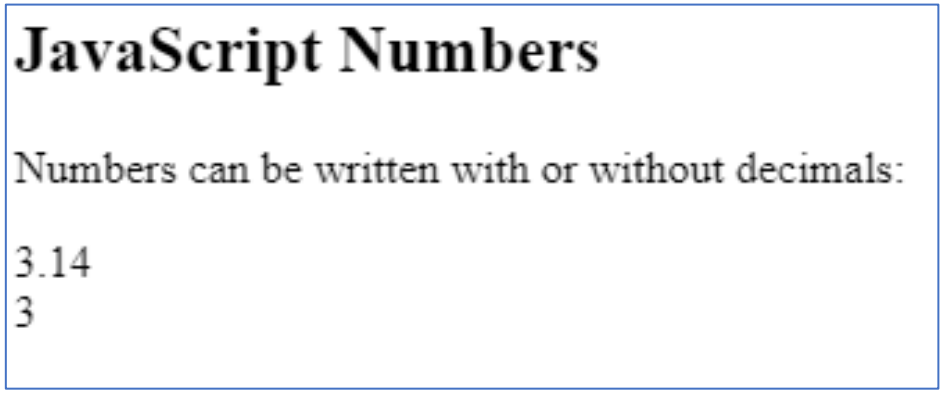
Boolean	represents boolean value either
	false or true
		`// Booleans`
		`let x = true;`
		`let y = false;`

Undefined	represents undefined value	
Null	represents null i.e. no value at all	

JavaScript primitive data types

- JavaScript has only **one type of number**.
- Numbers can be **written with or without decimals**.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Numbers</h2>
<p>Numbers can be written with or without decimals:</p>
<p id="demo"></p>
<script>
let x = 3.14;
let y = 3;
document.getElementById("demo").innerHTML = x + "<br>" + y;
</script>
</body>
</html>
```

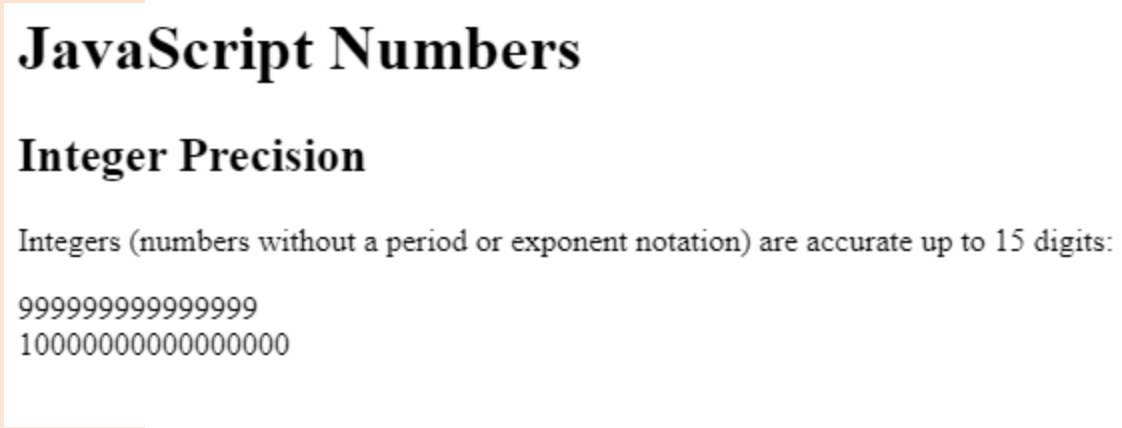


JavaScript primitive data types

Integer Precision

- Integers (**numbers without a period** or exponent notation) are accurate up to **15 digits**.

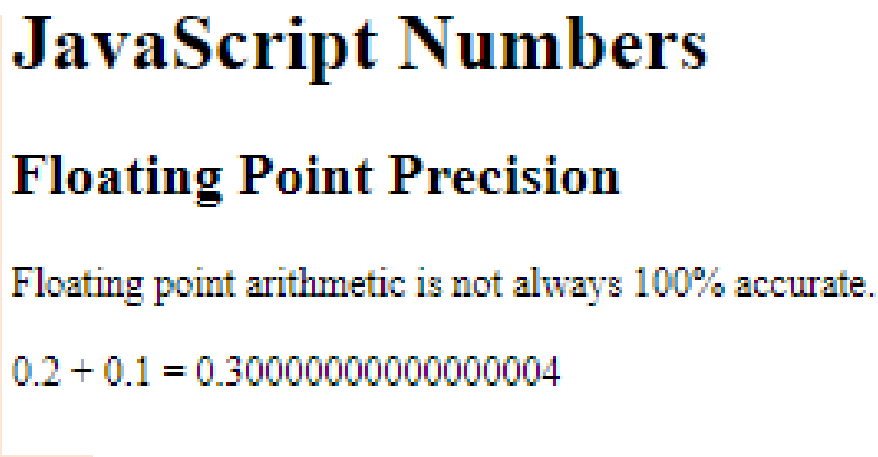
```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Numbers</h1>
<h2>Integer Precision</h2>
<p>Integers (numbers without a period or exponent
notation) are accurate up to 15 digits:</p>
<p id="demo"></p>
<script>
let x = 999999999999999;
let y = 9999999999999999;
document.getElementById("demo").innerHTML = x + "<br>"
+ y;
</script>
</body>
</html>
```



JavaScript primitive data types

Floating Precision

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Numbers</h1>
<h2>Floating Point Precision</h2>

<p>Floating point arithmetic is not always 100%
accurate.</p>

<p id="demo"></p>
<script>
let x = 0.2 + 0.1;
document.getElementById("demo").innerHTML = "0.2 + 0.1
= " + x;
</script>

</body>
</html>
```



JavaScript Data Types

- **A JavaScript string is zero or more characters written inside quotes.**

```
<!DOCTYPE html>
<html>
<body>
<h1>JavaScript Strings</h1>
<p id="demo"></p>
<script>
let text = "Chitkara University";  // String
written inside quotes
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

JavaScript Strings

Chitkara University

JavaScript non-primitive data types

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values

```
// Object:
const person = {firstName:"John", lastName:"Doe"};
```

```
// Array object:
const cars = ["Saab", "Volvo", "BMW"]
```

JavaScript Operators

- **JavaScript operators are symbols that are used to perform operations on operands**

- There are following types of operators in JavaScript.

    1. Arithmetic Operators
    2. Comparison (Relational) Operators
    3. Bitwise Operators
    4. Logical Operators
    5. Assignment Operators
    6. Special Operators

JavaScript Operators- Arithmetic operators

- **Arithmetic operators** are used to perform **arithmetic operations** on the operands.
- The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0

++	Increment	var a=10;
		a++;
		Now a = 11

--	Decrement	var a=10; a--; Now a = 9

```
<-- JS arithmetic operations -->
<html>
<head>
<script>
let a = 100;
let b = 50;
let x = a + b;
let y = a * b;
document.writeln("Addtion"+x);
document.writeln("<br>Multiplicaiton"+y);
</script>
</head>
<body>
</body>
</html>
```

JavaScript Operators

- Relational operators compares the two operands.

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

- The bitwise operators perform bitwise operations on operands.

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Operators

JavaScript Logical Operators

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

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

How to get input and displays output:

- **prompt()** → used to take input in a java script code

```
<script type="text/javascript">
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```


```
<html>
<head>
    <script>
        let a=prompt("Enter your name");
        document.writeln(a);
    </script>
</head>
<body>
</body>
</html>
```

prompt()

```
var num=parseInt(prompt("Enter a number"));
```

- If the user click the "OK" button, **prompt()** returns the value in the input as a string.
- If the user click the "Cancel" button, **prompt()** returns null.
- To covert the string input to number we use **parseInt()**

```
<html>
<head>
    <script>
        let a=parseInt(prompt("Ener number1"));
        let b=parseInt(prompt("Ener number2"));
        document.writeln(a+b);
    </script>
</head>
<body>
</body>
</html>
```

JavaScript Display Possibilities

- Writing into the **HTML output** using **document.write().**
- Writing into an **alert box**, using **window.alert().**
- Writing into an **confirm box** using **window.confirm()**
- Writing into an **HTML element**, using `innerHTML`

- **document.write()** provides user the **functionality to write multiple expressions (HTML or JavaScript)** directly to a document.
- This method **overwrites HTML code in a document**, if any and does not appends arguments to a new line.

```
document.write("Hello World!");
document.write("<h2>Hello World!</h2><p>Have a nice day!</p>");
```

JavaScript Display Possibilities- alert() and confirm()

A alert() and confirm() is used to take input in a java script code

```
alert(" This is an Alert method");
```

alert() box

- Display a message in a **dialog box**.
- The **dialog box will block the browser**.

```
var answer = confirm("Are you sure?");
```

confirm() box

- Display a message in a dialog box with two buttons: "OK" or "Cancel".
- `confirm()` returns `true` if the user click "OK". Otherwise, it returns `false`.

JavaScript Display Possibilities- alert() and confirm()

Writing into an HTML element, using `innerHTML`:

- To access an HTML element, JavaScript can use the **document.getElementById(id)** method.
- The id attribute defines the HTML element.

The innerHTML property defines the HTML content:

```
<!- innerHTML - ->
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
5 + 6;
</script>
</body>
</html>
```

Using console.log()

For debugging purposes, you can call the **console.log()** method in the browser to display data

```
<!- console.log() -->
<html>
<body>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

Conditional Statements

- To perform different actions for different decisions, we can **use conditional statements** in your code to do this.

- In JavaScript we have the following conditional statements:

    1. If Statement
    2. If else statement
    3. if else if statement
    4. Switch statement

1. If statement

- It evaluates the content only if expression is true.

```
if (condition) {
  //  block of code to be executed if the condition is true
}
```

```
<html>
<body>
<script>
var age=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
</body>
</html>
```

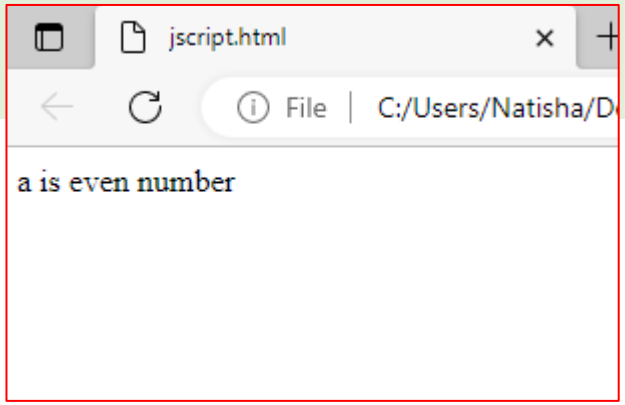
Conditional Statements- If...else Statement

2) If...else Statement

- It evaluates the content whether **condition is true of false**
- Use the else statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
  //code to be executed if the condition is true
} else {
  //code to be executed if the condition is false
}
```

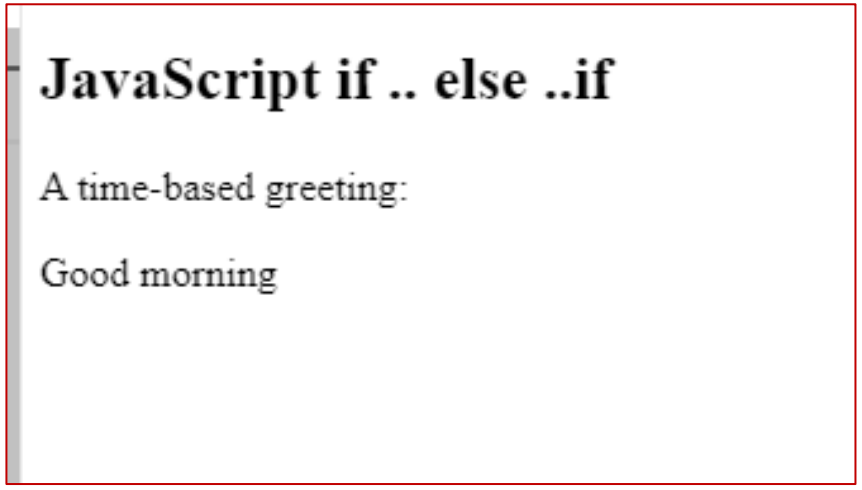
```
<!- JS program to find the ever or odd -->
<html>
<head>
</head>
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
<body>
</body>
</html>
```



Conditional Statements- else if statement

- Use the **else if statement** to specify a new condition if the first condition is false.

```
if (condition1) {
  // code to be executed if condition1 is true
} else if (condition2) {
  // code to be executed if the condition1 is
false and condition2 is true
} else {
  // code to be executed if the condition1 is
false and condition2 is false
}
```




```
<html>
<body>
<h2>JavaScript if .. else</h2>
<p>A time-based greeting:</p>
<p id="demo"></p>
<script>
const time = new Date().getHours();
let greeting;
if (time < 10)
{
  greeting = "Good morning";

} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTML = greeting;
</script>
</body>
</html>
```

Loops

- Loops can **execute a block of code a number of times.**

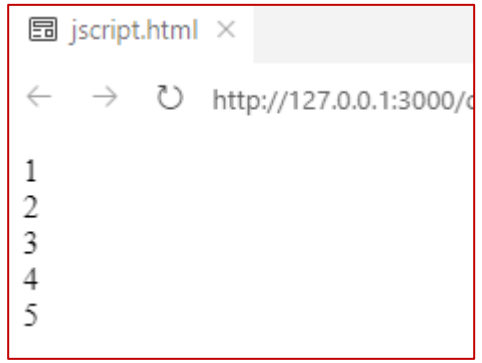
There are four types of loops in JavaScript.
1.for loop
2.while loop
3.do-while loop
4.for-in loop

1) JavaScript For loop:

The **JavaScript for loop** *iterates the elements for the fixed number of times*.

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

```
<html>
<body>
<script>
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
</body>
</html>
```



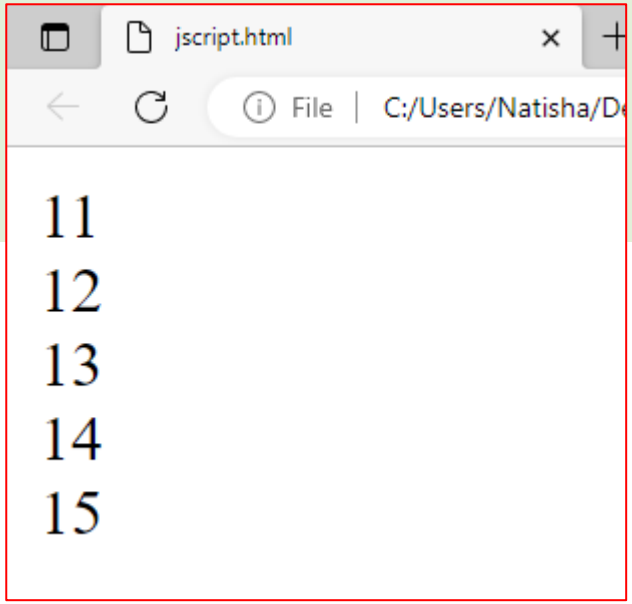
Loops - while

2) JavaScript while loop

- The JavaScript while **loop iterates the elements for the infinite number of times**.
- It should be used if number of iteration is not known.

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

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
</body>
</html>
```



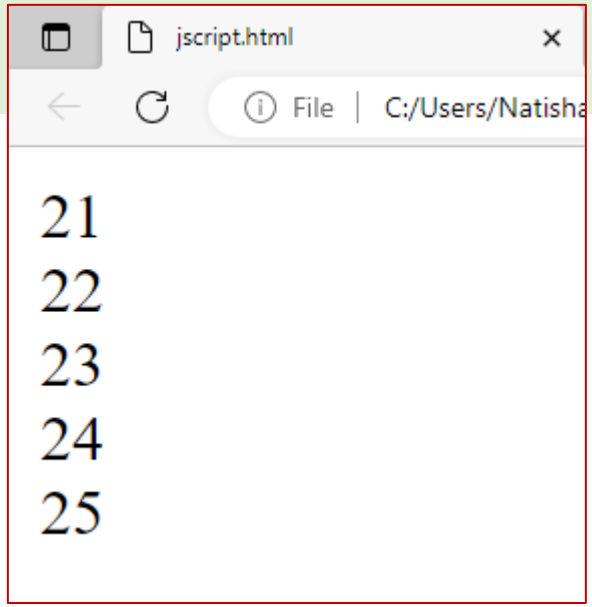
Loops – do while

3) JavaScript do while loop

- The JavaScript **do while loop** iterates the elements for the infinite number of times like while loop.
- But, code is executed at least once whether condition is true or false

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

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
</body>
</html>
```

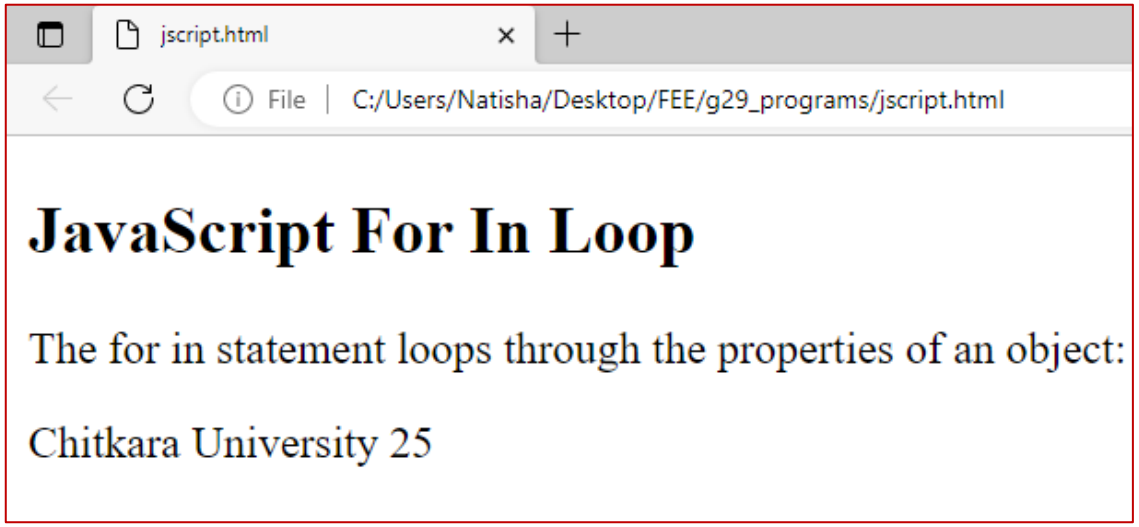


Loops – for in loop

The For In Loop

- The JavaScript for in statement loops through the properties of an Object:

```
for (key in object) {
  // code block to be executed
}
```




```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For In Loop</h2>
<p>The for in statement loops through the
properties of an object:</p>
<p id="demo"></p>
<script>
const person = {fname:"Chitkara",
lname:"University", age:25};

let txt = "";
for (let x in person) {
  txt += person[x] + " ";
}

document.getElementById("demo").innerHTML = txt;
</script>

</body>
</html>
```

JavaScript variable Scope

Scope determines the accessibility (visibility) of variables.

JavaScript has 3 types of scope:

1. Block scope
2. Function scope
3. Global scope

1. Block Scope

Variables declared inside a { } block cannot be accessed from outside the block:

```
{
  let x = 2;
}
// x can NOT be used here
```

```
{
  var x = 2;
}
// x CAN be used her
```

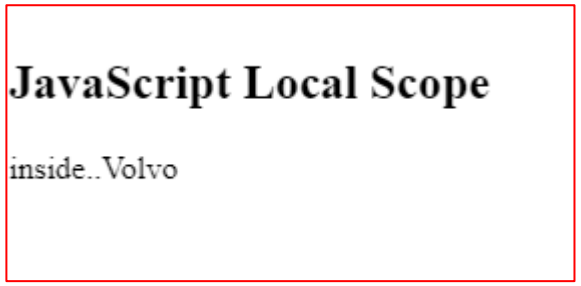
Variables declared with the `var` keyword can NOT have block scope.

JavaScript variable Scope

2. Function scope/Local Scope

- Variables declared within a JavaScript function, become **LOCAL** to the function.

```
// code here can NOT use carName
function myFunction() {
  let carName = "Volvo";
  // code here CAN use carName
}
// code here can NOT use carNam
```



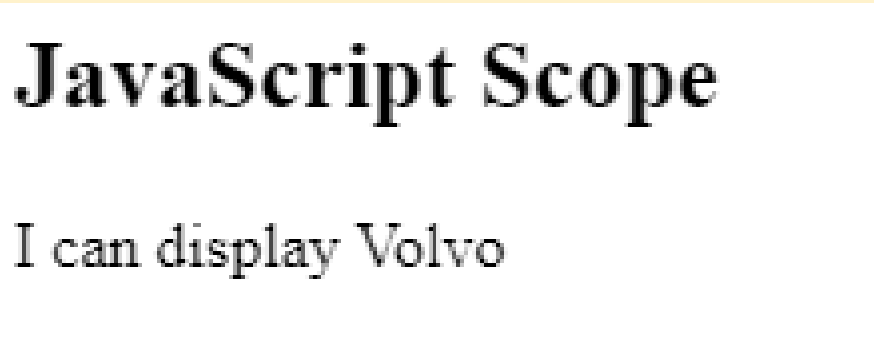

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Local Scope</h2>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
myFunction();
function myFunction() {
  let carName = "Volvo";
  document.getElementById("demo1").innerHTML
="inside.."+carName;
}
document.getElementById("demo2").innerHTML
="outside"+carName;
</script>

</body>
</html>
```

JavaScript variable Scope

3.Global JavaScript Variables

- A variable declared outside a function, becomes GLOBAL

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Scope</h2>
<p id="demo"></p>
<script>
let carName = "Volvo";
myFunction();
function myFunction() {
  document.getElementById("demo").innerHTML = "I can
display " + carName;
}
</script>
</body>
</html>
```



JavaScript Functions

- A JavaScript function is a **block of code** designed to perform a particular task.

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

Generally we can place **script containing function head section** of web page.

Ex: -

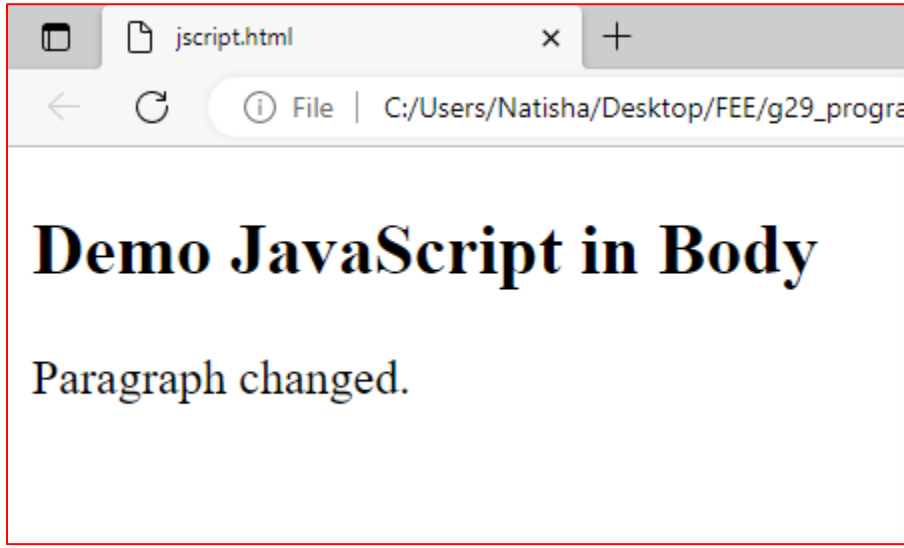
```
<HTML>
<HEAD>
<TITLE> Function direct call</TITLE>
<script language="JavaScript">
 function add(x,y)
        {
         z=x+y
        return z
        }
</script>
```

```
</HEAD>
<BODY>
<script>
        var r=add(30,60)
        document.write("addition is :"+r);
</script>
</BODY>
</HTML>
```

JavaScript Functions

- JavaScript in <body>
- A JavaScript function is placed in the **<body>** section of an HTML page.

```
<!DOCTYPE html>
<html>
<body>
<h2>Demo JavaScript in Body</h2>
<p id="demo">A Paragraph.</p>
<script>
function disp()
{
  document.getElementById("demo").innerHTML = "Paragraph
changed.";
}
disp()
</script
</body>
</html>
```



Invoking a JavaScript Function

- The code inside a function is not executed when the function is defined.
- **The code inside a function is executed when the function is invoked**.
    1. **Function call**
    2. **Events handlers**

1. Invoking a function by function call:

Ex>

```
function f1()
{
--
}

f1();
```

```
<HTML>
  <HEAD>
  <TITLE> Function direct call</TITLE>
  <script>
   function add(x,y) //function definition
   {
    z=x+y
    return z
   }
  </script>
</HEAD>
<BODY>
<script>
  var r=add(30,60)  //function call
  document.write("addition is :"+r);
</script>
</BODY>
</HTML>
```

2. Events handlers to call the function dynamically.

- To add dynamical effects, java script provide a **list of events** that call function dynamically.

```
<HTML>
<HEAD>
<TITLE> Function dynamically</TITLE>
<script language="JavaScript">
function add( )
{
      x=20;
      y=30;
      z=x+y;
      document.write("addition is :"+z);
}
</script>
```

```
</HEAD>
<BODY> to call function:
<input type="button“ value="clickhere" onclick="add( )">
</script>
</BODY>
</HTML>
```

O/P: - to call function: addition is :90