JavaScript

- JavaScript is the most popular script-based programming language that support the development of both client and server components of web based applications.
- It works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.
- JavaScript was originally developed by Netscape replacing LiveScript.

What is JavaScript?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Are Java and JavaScript the same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- JavaScript is a very free-form language compared to Java whereas Java is a class-based programming language designed for fast execution and type safety.

JavaScript	Java
1) JavaScript is interpreted (not complied) by client.	1) Java is compiled byte codes downloaded from server, executed on client.
2) JavaScript is object-oriented. There is no distinction between types of objects. Inheritance is through the prototype mechanism, and properties and methods can be added to any object dynamically.	2) Java is class-based. Objects are divided into classes and instances with all inheritance through the class hierarchy. Classes and instances cannot have properties or methods added dynamically.
3) In JavaScript, code is integrated with and embedded in HTML.	3) Java uses applets that are distinct from HTML (accessed from HTML pages).
4) Variable data types are not declared (dynamic typing).	4) Variable data types must be declared (static typing).

Database Connectivity:

- Netscape has a product called ‘Live wire’, which permits server side , JavaScript code, to connect to Relational Database management systems (RDBMS) like oracle, My SQL server, My SQL etc ‘live wire’ database drivers also support a no of non-relational database.

Client Side JavaScript:

- Client side java Script is embedded into a standard HTML program. JavaSript is embedded between the <script>…….</script> html tags.
- These tags are embedded within the <head>……</head> or <body>…..</body> tags of the html program.
- Java script is embedded into an html program because JavaScript uses the filename.html and HTTP protocol to transport itself from the web serve to the client's browser where the JavaScript executes and processes client information.
- Only a browser that is JavaScript enabled will be able to interpret JavaScript code.
- Netscape communicator does this best as javascript is the natural language of NetScape communicator.

Server Based JavaScript

- Server Based JavaScript programs are complied with HTML documents into a platform independent byte code format.
- The server based scripts remain with the server and are loaded to perform any server side processing.
- The HTML document loaded by the browser communicate with the server scripts to implement advance web applications that are distributed between the browser server and other server side programs such as database and electronic commerce applications.

Advantages of JavaScript:

1. **An interpreted language:**

   It do not requires compilation steps.

2. **Embedded within HTML:**

   JavaScript does not require any separate editor for program to be written, edited or compiled. It can be written in any text editor like notepad along with html tags and saved as fielname.html.

3. **Minimal Syntax- easy to learn:**

   By learning few commands, simple rules of syntax, complete application can be built using JavaScript.

4. Quick development:

As JavaScript does not require time consuming compilation, scripts can be developed in short period of time.

5. Designed for simple, small programs:

Small programs can be easily written executed at an acceptable speed using JavaScript.

6. Performance:

Java Script can be written such that the html files are compact and quite small. It minimize storage requirements on the web server and download time for client.

7. Procedural capabilities:

Every programming language needs to support facilities such as condition checking, looping and branching.

8. Designed for programming user events:

It support object/event based programming.

9. Easy debugging and Testing:

Being an interpreted language, JavaScript are tested line by line, and errors are listed when encountered. Thus it is easy to locate errors, make changes and test without any difficulty.

10. Platform independence/Architectural Neutral:

Java Script is a programming language that is completely independent of the hardware on which it works. It is a language understood by any JavaScript enabled browser.

Thus JavaScript applications work on any machine that has an appropriate JavaScript enabled browser installed.

Limitation Of JavaScript

The major drawbacks of JavaScript are as follows:

- JavaScript cannot be loaded in any and every browser. It requires a JavaScript enabled web browser like Netscape Navigator 2.0 or above and Internet Explorer 3.0 or above running on Windows95/98/NT. Older versions of browsers do not fully support JavaScript.
- JavaScript cannot be used to write stand-alone applications. It has to be embedded within HTML for the browser to do its processing.

- JavaScript stores data by virtue of HTTP cookies. Some of the information (such as, name, email-id etc.) can be stored in cookie, which can be used by the site when the user visits the site again. But it lacks other effective and persistent methods (such as, storing in a file, database etc.) of storing local data.
- JavaScript has no support for Table objects.
- It does not have any option of implementing threads within programs.
- JavaScript does not support all of OOPS concept. It is object based, not object oriented. Inheritance is a very strong concept in any object oriented programming language. But JavaScript does not support inheritance as well.

Data Types:

- JavaScript supports four primitive types of value and supports complex types such as arrays and objects.
- Primitive types are number, string, Boolean and null type.
- **Number:**
  - Consists of integer and floating point numbers and the special NaN (Not a Number) value.

- **Boolean:**
  - Consists of the logical value true and false.
  - If true, value=1
  - If false, value=0
- **String:**
  - Consists of string values that are enclosed in a single or double quotes.
- **Null:**
  - Consists of a single value, null, which identifies a null, empty or nonexistent reference.
  - It is automatically converted to default values of other types when used in an expression

Embedding JavaScript In HTML

<script language="text/JavaScript">

JavaScript statements

</script>

Output:

Output1:

Comments

- same as that of C++ and Java
- // - for a single line comment.
- /* ……

.. */ - for a multi-line comment

Output:

Variables

- JavaScript variables can be declared by preceding the variable name with a keyword **var.**
- For example,

```
var num;
var num1=100;
Radius=20;   -  is also valid
```

Output:

Scope of a variable

```
<script language="text/JavaScript">
var num = 100                    global variable, scope global
function student()
{
var num = 40          local variable, scope within the
                      function.
   // JavaScript statements
   document.write("Value of local variable is " + num)
}
   document.write("Value of global variable is " + num)
</script>
```

Output:

Data Type Conversion

- **parseInt(string), parseFloat(string)**
- E.g.

  var myint = 20

  var mystring = “55”

  result1= myint + mystring -------- results “2055”

  result2 = myint + parseInt(mystring) --- results 75

Output:

Operators

The operators can be organized in the following categories:

- **Assignment**
- **Arithmetic**
- Unary
- String
- **Logical**
- **Comparison**
- Bit Manipulation
- **Conditional**
- Special

Assignment

- a = b
- x += 5 // similar to x = x + 5
- x -= 2 // similar to x = x – 2
- x *= 5 // similar to x = x * 5
- x /= 3 // similar to x = x / 3
- x %= 10 // similar to x = x % 10
- x = y = z = 10

Arithmetic

Operator	Operation
+	*Addition*
–	*Subtraction*
*	*Multiplication*
/	*Division*
%	*Modulo*

Output:

Unary

Operator	Operation
++	*Increment*
--	*Decrement*
-	*Unary Negation*
~	*Bit wise Complement*
!	*Not*

String

e.g.

mystr1 = “Khwopa”

mystr2 = “Engg”

name = mystr1 + mystr2

- the + operator between the two strings concatenates them and **name will have a** value of **“KhwopaEngg”**

Logical

Operator	Description
&&	Logical And
||	Logical Or
!	Logical Not

expression1&& expression2

For example;

```
If (basic>1000 && age>35)
{
   JavaScript statements
}
```

Comparison

Operator	Description
==	Equal
===	Strictly equal
!=	Not equal
!==	Strictly not equal
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to

Bit Manipulation

Operator	Operator Name	Description
&	AND	Returns a one bit if both operands are one.
|	OR	Returns a one bit if either operand is one.
^	XOR	Returns a one bit if only one of the two operands is one.
~	NOT	Returns a one bit if operand is zero; zero bit if operand is one.

Operator	Operator Name
<<	Left Shift
>>	Sign-Propagating Right Shift
>>>	Zero-Fill Right Shift

The following is an example of bit wise logical operators.

```
x = 7                    // 00000111 ……..in binary
y = 9                    // 00001001 ……..in binary
resultand = x & y        // result ……..00000001 in binary
resultor = x | y         // result ……..00001111 in binary
resultxor = x ^ y        // result ……..00001110 in binary
resultnot = ~ x          // result ……..11111000 in binary

a = 11          // 00001011 ……..in binary
b = a << 3      //  result  ……..01011000.The bits have moved 3
places to the left.
```

Output:

Output1:

Conditional

- conditional operator - “**? :**”
- p < q ? 20 : 25

Output:

Special

- **Delete Operator:**
  - delete myarray[10]
- **New Operator**
- **Typeof Operator**

Output

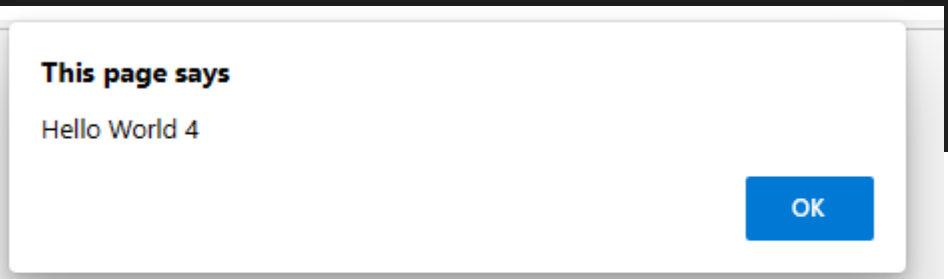
- **Void Operator**

JavaScript Output/Display

JavaScript can "display" data in different ways:

- Writing into an HTML element,
  using innerHTML.
- Writing into the HTML output
  using document.write().
- Writing into an alert box,
  using window.alert().
- Writing into the browser console,
  using console.log().

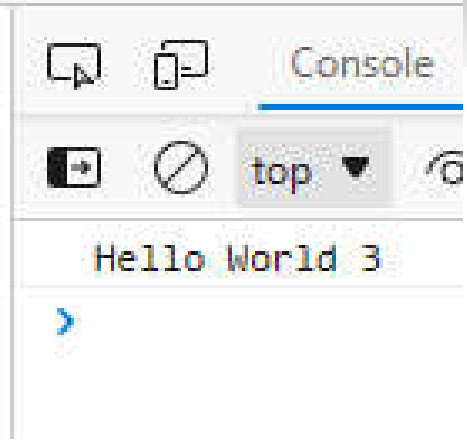
```html
<html>
    <head>
        <title>JavaScript Display</title>
    </head>
    <body>
        <h1 style="background-color:rgba(255, 99, 71, 0.2);">JavaScript Output</h1>
        <P id="pid">Welcome</P>
        <p id="aid"></p>
        <script>
            document.write("Hello World 1");
            document.getElementById("pid").innerHTML="Hello World 2";
            console.log("Hello World 3");
            window.alert("Hello World 4");
        </script>
    </body>
</html>
```



JavaScript Output

Hello World 2

Hello World 1



```html
<html>
    <head>
        <title>JavaScript Display</title>
    </head>
    <body>
        <h1>JavaScript Output</h1>
        <P id="pid">Welcome</P>
        <p id="aid"></p>
        <script language="JavaScript">
            document.getElementById("aid").innerHTML="Computer Department!";
        </script>
    </body>
</html>
```

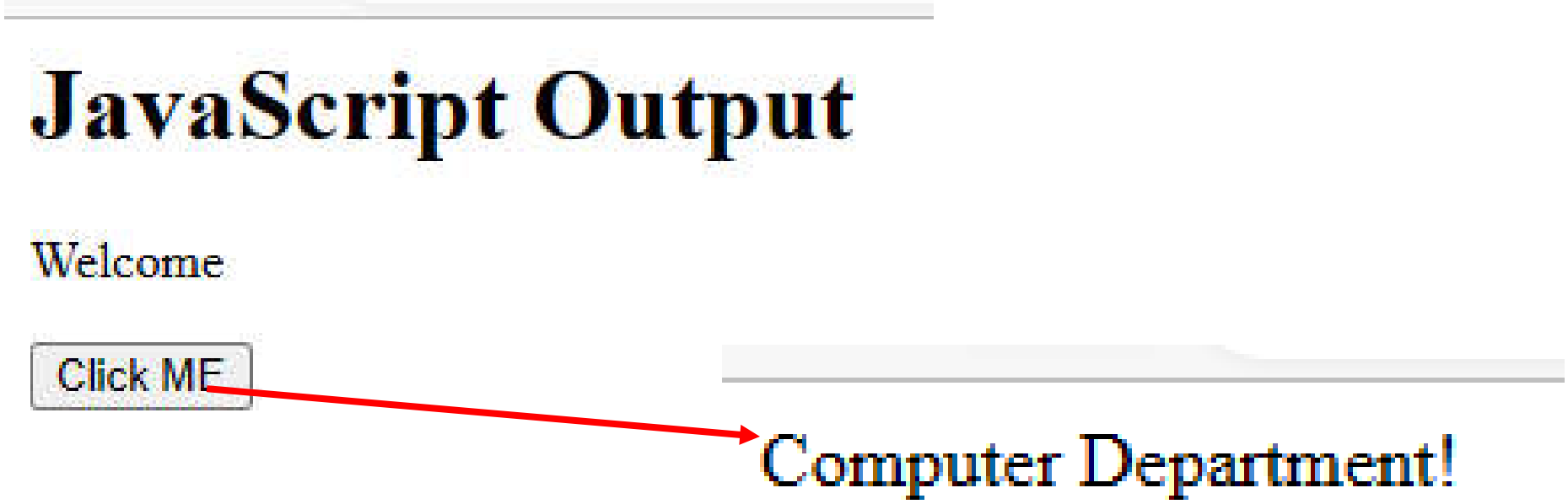
JavaScript Output

Welcome

Computer Department!

```html
<html>
    <head>
        <title>JavaScript Display</title>
    </head>
    <body>
        <h1>JavaScript Output</h1>
        <P id="pid">Welcome</P>
        <p id="aid"></p>
        <script language="JavaScript">
            document.write("Computer Department!");
        </script>
    </body>
</html>
```

```html
<html>
    <head>
        <title>JavaScript Display</title>
    </head>
    <body>
        <h1>JavaScript Output</h1>
        <P id="pid">Welcome</P>
        <p id="aid"></p>
        <input type="button" value="Click ME" onclick="document.write('Computer Department!');">
    </body>
</html>
```



JS Examples

1_HelloWorld.html

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#1</title>
</head>
<body>
    <script type="text/javascript">
        document.write("Hello World!");
    </script>
</body>
</html>
```

2_HelloWorld_h1.html

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#2</title>
</head>
<body>
    <script type="text/javascript">
        document.write("<h1>Hello World!</h1>");
    </script>
</body>
</html>
```

3_HTML_Write.html

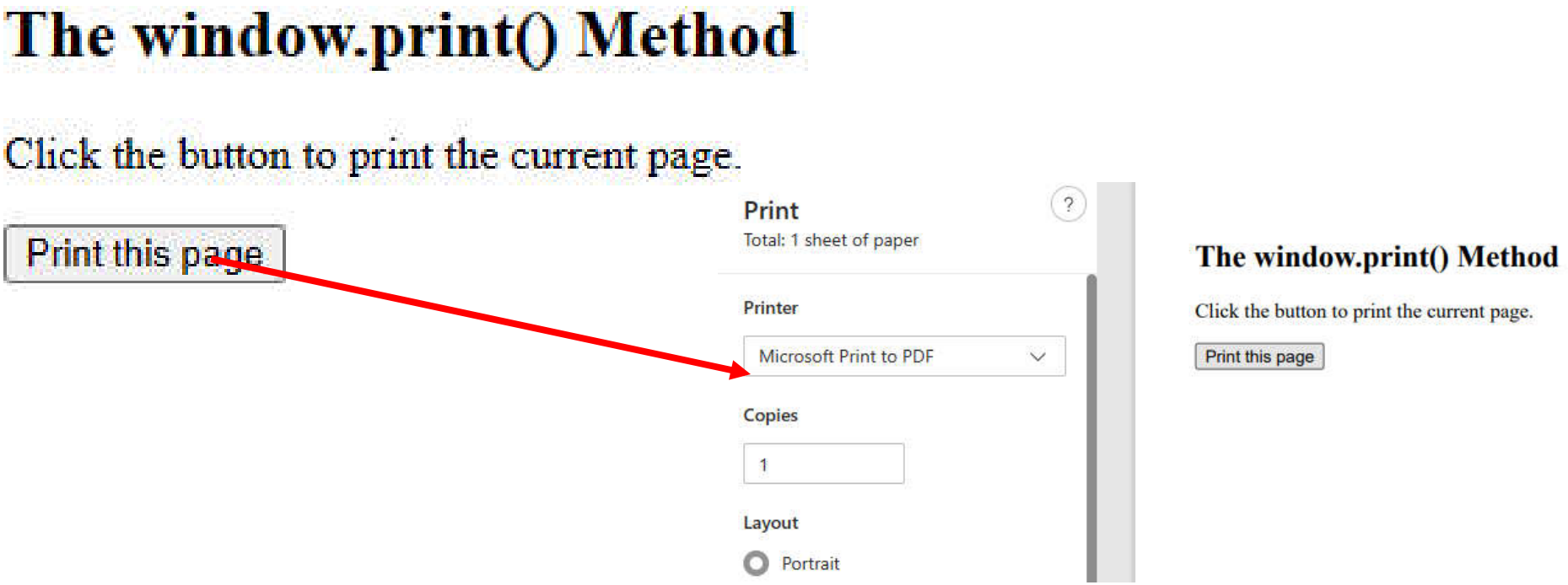
```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#3</title>
</head>
<body>
    <script type="text/javascript">
        document.write("<h1>This is a heading</h1>");
        document.write("<p>This is a paragraph.</p>");
        document.write("<p>This is another paragraph.</p>");
    </script>
</body>
</html>
```

4_JS_Comments.html

```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#4 - Comments</title>
</head>
<body>
    <script type="text/javascript">
        // Write a heading
        document.write("<h1>Heading#1</h1>"); // Comment in Line's E
        /* Write two paragraphs:
           Multi-line Comments
        */
        document.write("<p>This is a paragraph.</p>");
        document.write("<p>This is another paragraph.</p>");
    </script>
</body>
</html>
```

window.print()

```html
<html>
    <head>
        <title>JavaScript Display</title>
    </head>
    <body>
        <h2>The window.print() Method</h2>
        <p>Click the button to print the current page.</p>
        <input type="button" onclick="window.print()" value="Print this page">
    </body>
</html>
```



```html
<html>
    <head>
        <title>JavaScript Variables</title>
    </head>
    <body>
        <h1>JavaScript Variables</h1>
        <script type="text/JavaScript">
            var firstname = "Computer";
            document.write(firstname);
            document.write("<br/>");
            firstname = "Department";
            document.write(firstname);
        </script>
        <p>The script above declares a variable, assigns a value,
            displays it and again changes the value and displays it.
        </p>
    </body>
</html>
```

```html
<html>
    <head>
        <title>JavaScript Variable</title>
    </head>
    <body>
        <h2>JavaScript Variable</h2>
        <p>JavaScript let keyword</p>
        <p id="demo1"></p>
        <script>
            let a, b, c;
            a = 5;
            b = 6;
            c = a + b;
            document.getElementById("demo1").innerHTML = c;
        </script>
    </body>
</html>
```

JavaScript Variable

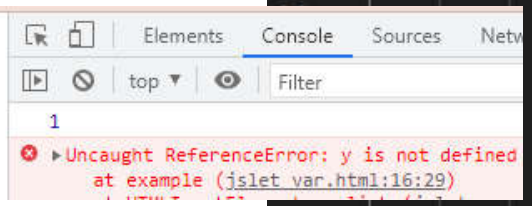
JavaScript let keyword

```html
<html>
    <head>
        <title>JavaScript Variable</title>
    </head>
    <body>
        <h2>JavaScript Variable</h2>
        <p>JavaScript let vs var keywords</p>
        <p id="demo1"></p>
        <script>
            function example() {
                if (true) {
                    var x = 1;
                    let y = 2;
                }
                console.log(x); // 1
                console.log(y); // ReferenceError: y is not defined
            }
        </script>
        <input type="button" value="Enter" onclick="example();">
    </body>
</html>
```

JavaScript Variable

JavaScript let vs var keywords

Enter



let vs. var variables

The main difference between **let** and **var** is that **let** is block-scoped, meaning that the variable is only accessible within the block (i.e., the code within the curly braces **{}**) in which it is declared. On the other hand, **var** is function-scoped, meaning that the variable is accessible within the entire function in which it is declared, or the global scope if it's declared outside any function.

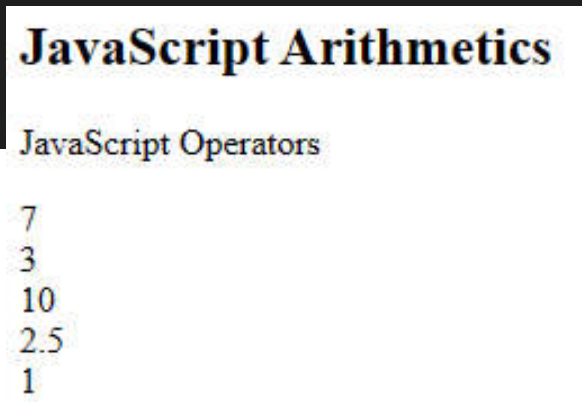
Another important difference is that **var** can be redeclared within the same scope without any error, whereas **let** cannot.

Therefore, it's generally recommended to use **let** instead of **var** in modern JavaScript development, especially in cases where you need to ensure proper scoping and avoid potential bugs caused by variable redeclaration.

let vs var variables

- The main difference between **let** and **var** is that **let** is block-scoped, meaning that the variable is only accessible within the block (i.e., the code within the curly braces **{}**) in which it is declared.
- On the other hand, **var** is function-scoped, meaning that the variable is accessible within the entire function in which it is declared, or the global scope if it's declared outside any function.
- Another important difference is that **var** can be redeclared within the same scope without any error, whereas **let** cannot.
- Therefore, it's generally recommended to use **let** instead of **var** in modern JavaScript development, especially in cases where you need to ensure proper scoping and avoid potential bugs caused by variable redeclaration.

```html
<html>
    <head>
        <title>JavaScript Arithmetics</title>
    </head>
    <body>
        <h2>JavaScript Arithmetics</h2>
        <p>JavaScript Operators</p>
        <p id="demo1"></p>
        <script>
            let a, b, c, d, e, f, g;
            a = 5;
            b = 2;
            c = a + b;
            d = a - b;
            e = a * b;
            f = a / b;
            g = a % b;
            document.getElementById("demo1").innerHTML = c + "<br/>" + d
            + "<br/>" + e + "<br/>" + f + "<br/>" + g;
        </script>
    </body>
</html>
```



```html
<html>
    <head>
        <title>JavaScript Arithmetics</title>
    </head>
    <body>
        <h1>JavaScript Assignments</h1>
        <h2>Bitwise AND & OR Assignment</h2>
        <h3>The &= |= Operators</h3>
        <p id="pid"></p>
        <p id="aid"></p>
        <script type="text/JavaScript">
        let x = 10;
        x &= 2;
        document.getElementById("pid").innerHTML = "Value of x is: " + x;
        let y = 10;
        y |= 5;
        document.getElementById("aid").innerHTML = "Value of y is: " + y;
        </script>
    </body>
</html>
```

JavaScript Assignments

Bitwise AND & OR Assignment

The &= |= Operators

Value of x is: 2

Value of y is: 15

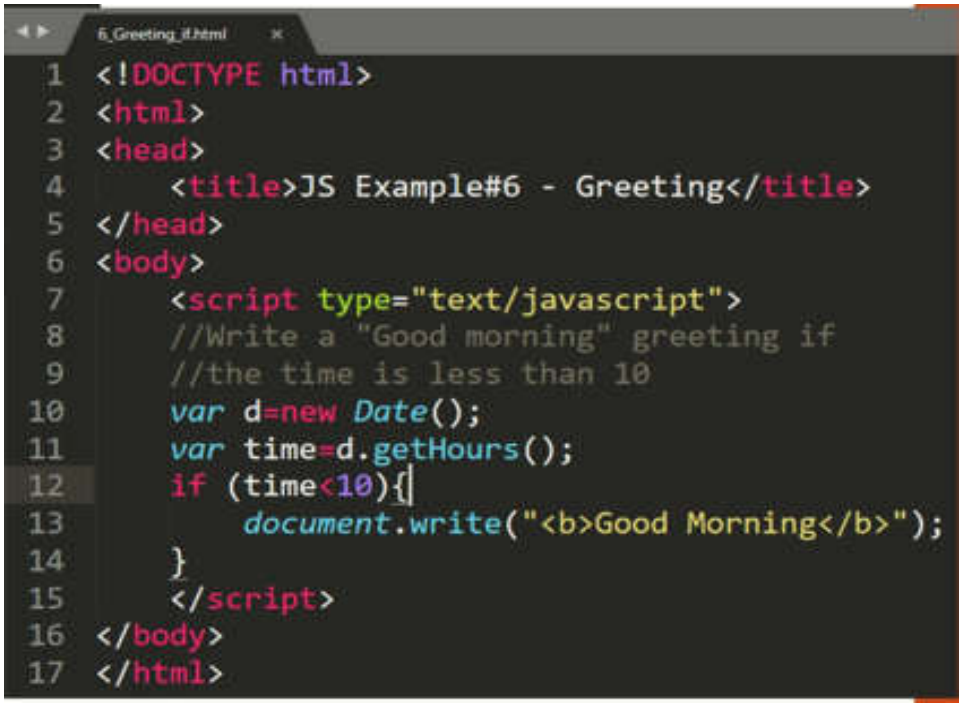
Programming Constructs

Programming constructs in JavaScript include:

- Conditional statements,
- Loop statements,
- Break statements, and
- Continue statements.

- **The If Statement**

```
if (condition)
{
   statements
}
```




```
if(i==2)
{
   basic=1000;
   salary=basic + 0.2*basic;
   document.write(i + "." +" salary is " + salary +
   "<br>");
}
```

- **If...else Statement**

Syntax

if (*condition*)

{

code to be executed if condition is true

}

else

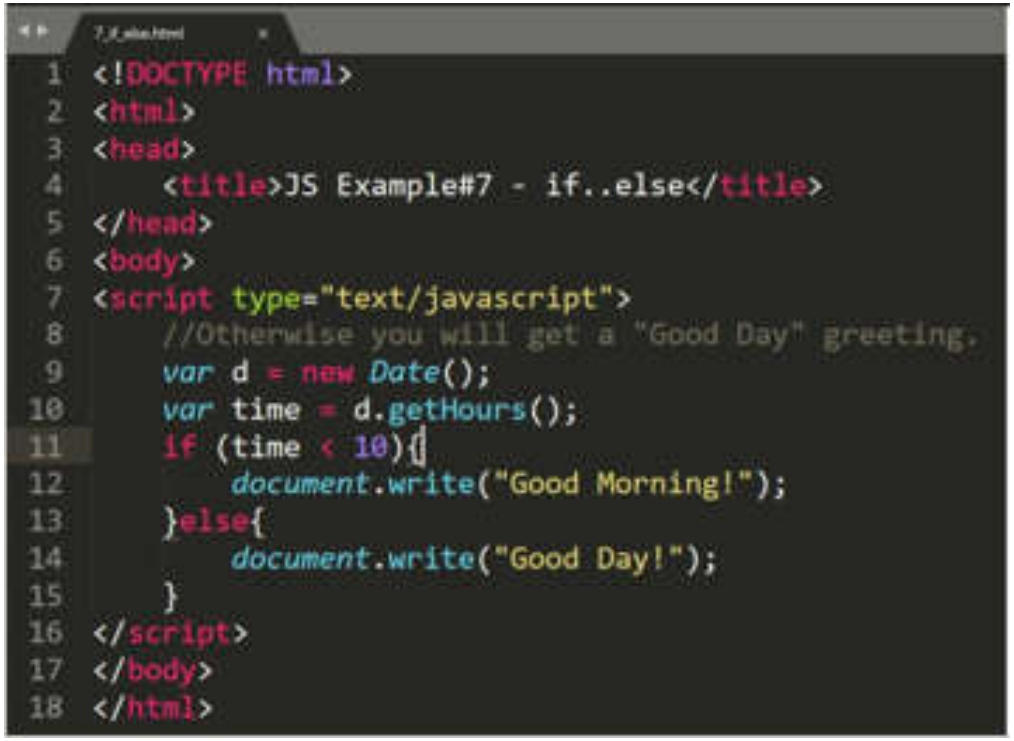
{

code to be executed if condition is not true

}

- **Example**

```
<script type="text/javascript">
    var d = new Date();
    var time = d.getHours();
    if (time < 10) {
        document.write("Good
    morning!");
        }
    else {
        document.write("Good
    day!");
        }
</script>
```



```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#7 - if..else</title>
</head>
<body>
<script type="text/javascript">
    //Otherwise you will get a "Good Day" greeting.
    var d = new Date();
    var time = d.getHours();
    if (time < 10){
        document.write("Good Morning!");
    }else{
        document.write("Good Day!");
    }
</script>
</body>
</html>
```

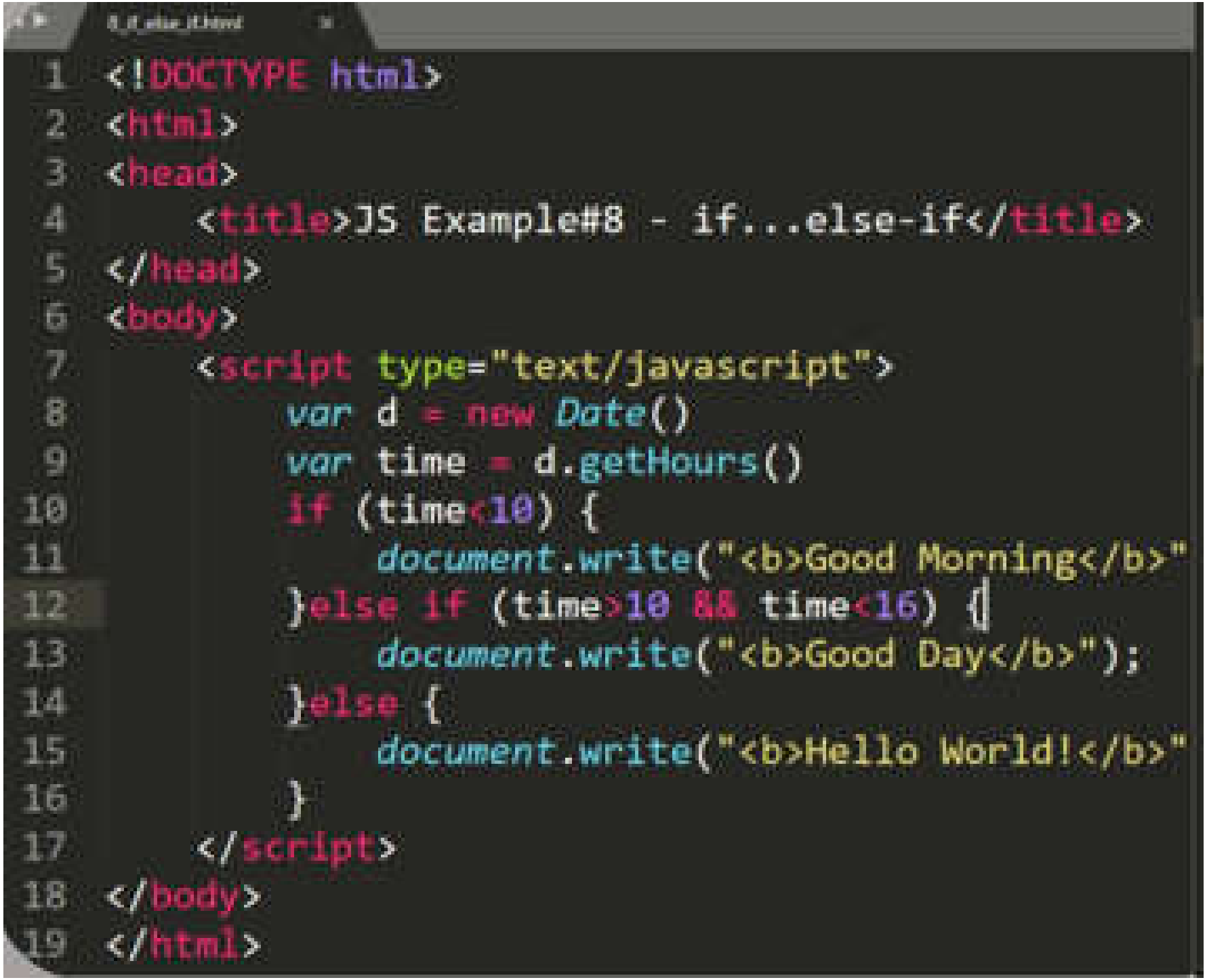
- **If...else if...else Statement**

Syntax

```
if (condition1)
{
code to be executed if condition1 is true
 }
else if (condition2)
{
code to be executed if condition2 is true
}
else
{
code to be executed if condition1 and condition2 are not
    true
}
```

- **Example**

```
<script type="text/javascript">
    var  d = new Date();
    var  time = d.getHours();
    if (time<10) {
        document.write("<b>Good morning</b>");
        }
    else if (time>10 && time<16) {
        document.write("<b>Good day</b>"); }
    else {
        document.write("<b>Hello World!</b>");
        }
</script>
```



```
<!DOCTYPE html>
<html>
<head>
    <title>JS Example#8 - if...else-if</title>
</head>
<body>
    <script type="text/javascript">
        var d = new Date()
        var time = d.getHours()
        if (time<10) {
            document.write("<b>Good Morning</b>"
        }else if (time>10 && time<16) {
            document.write("<b>Good Day</b>");
        }else {
            document.write("<b>Hello World!</b>"
        }
    </script>
</body>
</html>
```

- **The for Loop**
- **Syntax**

```
for(var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
 }
```

Example:

```
<html> <body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{ document.write("The number is " + i);
   document.write("<br />"); }
</script>
</body> </html>
```

- **The while loop**
- **syntax:**

```
while (var<=endvalue)
{
code to be executed
}
```

- **The do...while Loop**
- **syntax:**

```
do {
code to be executed
} while (var<=endvalue);
```

16_HeadingControl.html

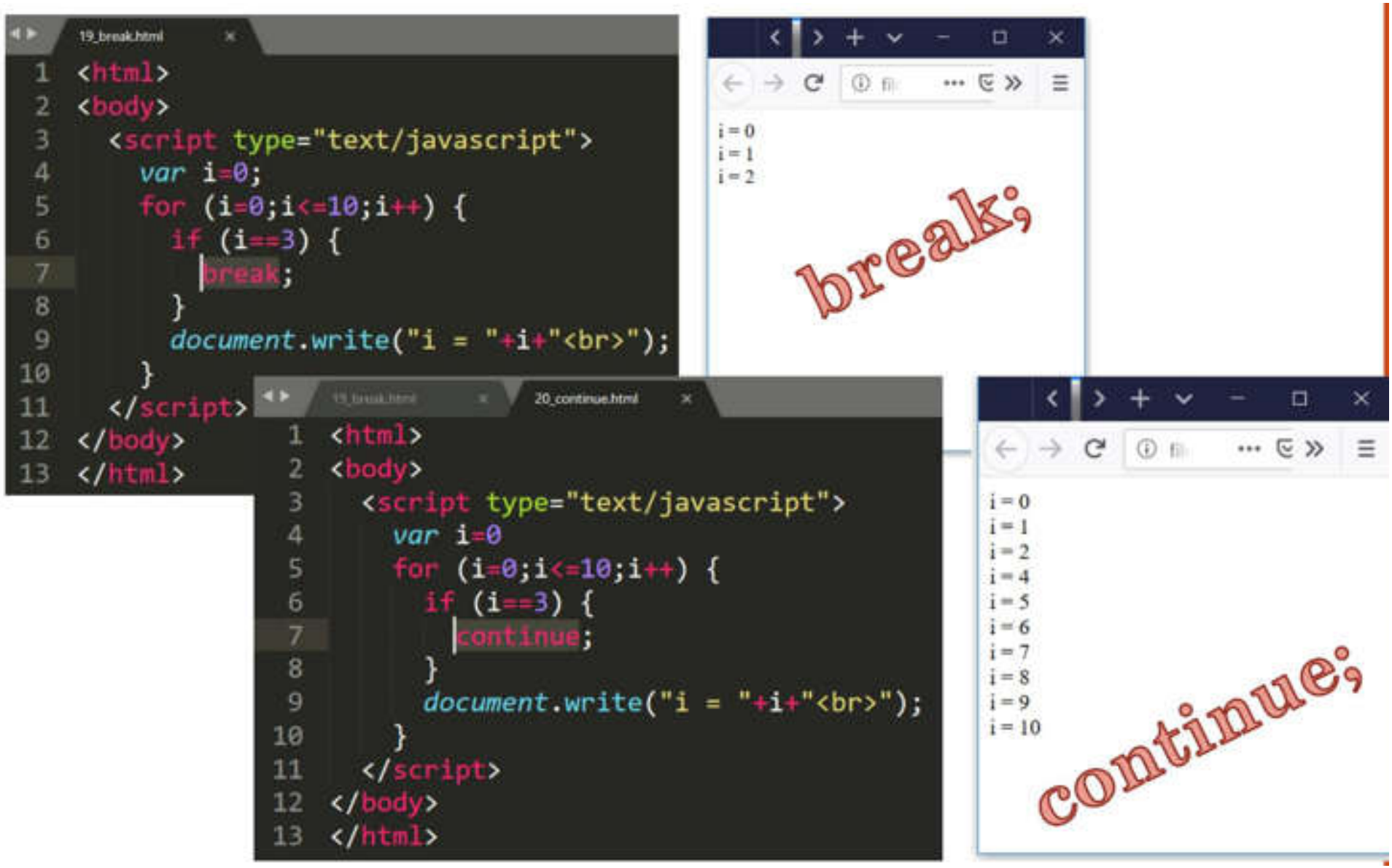
```
<html>
<body>
  <script type="text/javascript">
    for (i = 1; i <= 6; i++){
      document.write("<h" + i + ">This is heading " + i);
      document.write("</h" + i + ">");
    }
  </script>
</body>
</html>
```

17_while_Loop.html

```
<html>
<body>
  <script type="text/javascript">
    var i=0;
    while (i<=5) {
      document.write("n = "+i+"<br>");
      i++;
    }
  </script>
</body>
</html>
```

18_do_while_Loop.html

```
<html>
<body>
  <script type="text/javascript">
    var i=0;
    do{
      document.write("i = "+i+"<br>");
      i++;
    }
    while (i<=5);
  </script>
</body>
</html>
```


19_break.html
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++) {
if (i==3) {
break;
}
document.write("i = "+i+"
");
}
</script>
</body>
</html>
i = 0
i = 1
i = 2
break;
20_continue.html
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++) {
if (i==3) {
continue;
}
document.write("i = "+i+"
");
}
</script>
</body>
</html>
i = 0
i = 1
i = 2
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
i = 10
continue;

- **Switch Statement**
- **Syntax**

```
switch(n)
{
case 1:
   execute code block 1
   break;
case 2:
   execute code block 2
   break;
default:
   code to be executed if n is different from case 1 and
   2
}
```

```html
<html>
<body>
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.
var d=new Date();
theDay=d.getDay();
switch (theDay){
    case 5:
        document.write("Finally Friday");
        break;
    case 6:
        document.write("Super Saturday");
        break;
    case 0:
        document.write("Sleepy Sunday");
        break;
    default:
        document.write("I'm looking forward to this weekend!");
}
</script>
</body>
</html>
```

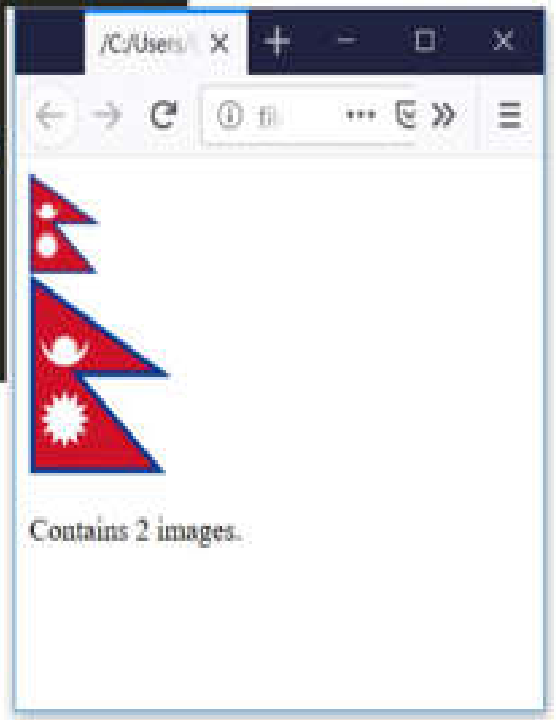
Switch Case – Example

```
<!DOCTYPE html>
<html>
<body>
<p>Write Banana, Orange or Apple in the input field and click the button.</p>
<p>The switch statement will execute a block of code based on your input.</p>
<input id="myInput" type="text">
<button onclick="myFunction()">Try it</button>

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

<script>
function myFunction() {
  var text;
  var fruits = document.getElementById("myInput").value;

  switch(fruits) {
    case "Banana":
        text = "Banana is good!";
        break;
    case "Orange":
        text = "I am not a fan of orange.";
        break;
    case "Apple":
        text = "How you like them apples?";
        break;
    default:
        text = "I have never heard of that fruit...";
  }
  document.getElementById("demo").innerHTML = text;
}
</script>
</body>
</html>
```

Image Count – Within Webpage

25_Count_img_of_webpage.html

```
1  <html>
2  <body>
3    <img border="0" src="np.png" height="50"><br />
4    <img border="0" src="np.png" height="100"><br /><br />
5    <script type="text/javascript">
6      var l = document.images.length;
7      document.write("Contains " + l + " images.");
8    </script>
9  </body>
10 </html>
```



The typeof Operator

- The JavaScript typeof operator is used to find the type of a JavaScript variable.
- The typeof operator returns the type of a variable or an expression.
- **Primitive Data**
- A primitive data value is a single simple data value with no additional properties and methods.
- The typeof operator can return one of these primitive types:
  - string
  - number
  - boolean
  - undefined

The typeof Operator

- **Complex Data**
- The typeof operator can return one of two complex types:
  - function
  - object
- The typeof operator returns "object" for objects, arrays, and null.
- The typeof operator does not return "object" for functions.

Examples

- **Primitive Examples**

```
typeof "John"          // Returns "string"
typeof 3.14            // Returns "number"
typeof true            // Returns "boolean"
typeof false           // Returns "boolean"
typeof x               // Returns "undefined" (if x has no value)
```

- **Complex Examples**

```
typeof {name:'John', age:34}    // Returns "object"
typeof [1,2,3,4]                // Returns "object" (not "array")
typeof null                     // Returns "object"
typeof function myFunc(){}      // Returns "function"
```

```html
<html>
    <head>
        <title>JavaScript typeof Operator</title>
    </head>
    <body>
        <h1>JavaScript Operators</h1>
        <h2>The typeof Operator</h2>
        <p>The typeof operator returns the type of a variable, object, function or expression:</p>
        <p id="demo"></p>
        <script type="text/JavaScript">
            document.getElementById("demo").innerHTML =
            "'John' is " + typeof "John" + "<br>" +
            "3.14 is " + typeof 3.14 + "<br>" +
            "NaN is " + typeof NaN + "<br>" +
            "false is " + typeof false + "<br>" +
            "[1, 2, 3, 4] is " + typeof [1, 2, 3, 4] + "<br>" +
            "{name:'John', age:34} is " + typeof {name:'John', age:34} + "<br>" +
            "new Date() is " + typeof new Date() + "<br>" +
            "function () {} is " + typeof function () {} + "<br>" +
            "myCar is " + typeof myCar + "<br>" +
            "null is " + typeof null;
        </script>
    </body>
</html>
```

JavaScript Operators

The typeof Operator

The typeof operator returns the type of a variable, object, function or expression:

'John' is string
3.14 is number
NaN is number
false is boolean
[1, 2, 3, 4] is object
{name:'John', age:34} is object
new Date() is object
function () {} is function
myCar is undefined
null is object

- **JavaScript Functions**
  - To keep the browser from executing a script when the page loads
  - A function contains code that will be executed by an event or by a call to the function.
  - You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file)
  - Functions can be defined both in the <head> and in the <body> section of a document.
  - However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

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).
- A JavaScript function is defined with the function keyword, followed by a **name**, followed by parentheses **()**.
- Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- The parentheses may include parameter names separated by commas:
  (*parameter1, parameter2, ...*)
- The code to be executed, by the function, is placed inside curly brackets: **{}**

- The syntax for creating a function is:

function *functionname*(*var1,var2,...,varX*)
{
    //some code
}

or

function *name*(*parameter1, parameter2, parameter3*)
{
    // code to be executed
}

or

function *functionname*()
{
    //some code
}

- Function **parameters** are listed inside the parentheses () in the function definition.
- Function **arguments** are the **values** received by the function when it is invoked.
- Inside the function, the arguments (the parameters) behave as local variables.
- *Note:*

  A Function is much the same as a Procedure or a Subroutine, in other programming languages.

Function Invocation

- The code inside the function will execute when "something" **invokes** (calls) the function:
  - When an event occurs (when a user clicks a button)
  - When it is invoked (called) from JavaScript code
  - Automatically (self invoked)

Function Return

- When JavaScript reaches a return statement, the function will stop executing.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.
- Functions often compute a **return value**.
- The return value is "returned" back to the "caller“.

- **The return Statement**
- **Example**

```
function prod(a,b)
   {
        x=a*b;
        return x;
   }
```

- **To call the function**
- **Example**

```
product=prod(2,3);
```

Output:

```html
<html>
    <head>
        <title>JavaScript typeof Operator</title>
    </head>
    <body>
        <h2>JavaScript Functions</h2>
        <p>This example calls a function which performs a calculation, and returns the result:</p>
        <p id="pid"></p>
        <script type="text/JavaScript">
            function myFunction(p1, p2) {
            return p1 * p2;
            }
            document.getElementById("pid").innerHTML = myFunction(4, 3);
        </script>
    </body>
</html>
```

JavaScript Functions

This example calls a function which performs a calculation, and returns the result:

12

The Lifetime of JavaScript Variables

- If you declare a variable within a function, the variable can only be accessed within that function.
- When you exit the function, the variable is destroyed. These variables are called local variables.
- You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- If you declare a variable outside a function, all the functions on your page can access it.
- The lifetime of these variables starts when they are declared, and ends when the page is closed.

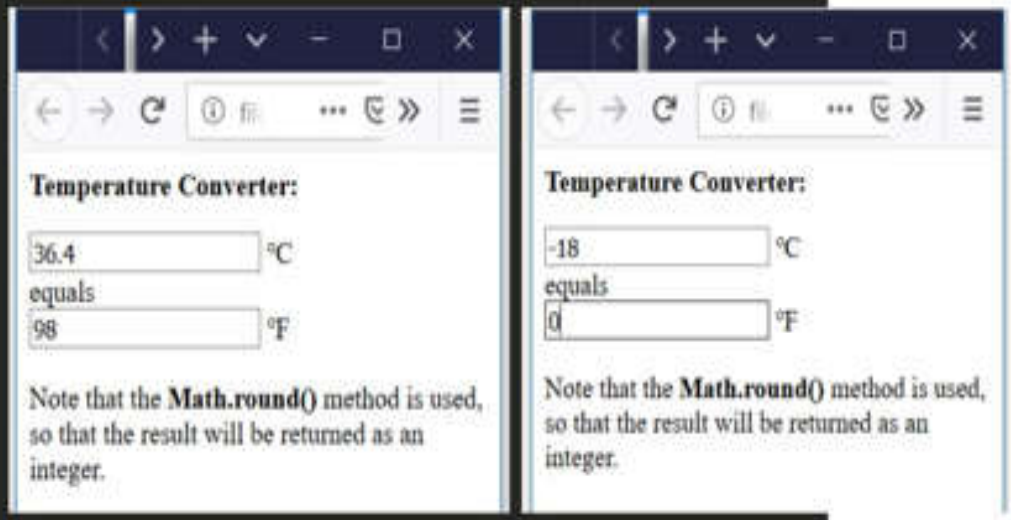
Local Variables

- Variables declared within a JavaScript function, become **LOCAL** to the function.
- Local variables can only be accessed from within the function.
- Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.
- Local variables are created when a function starts, and deleted when the function is completed.

Temperature Converter

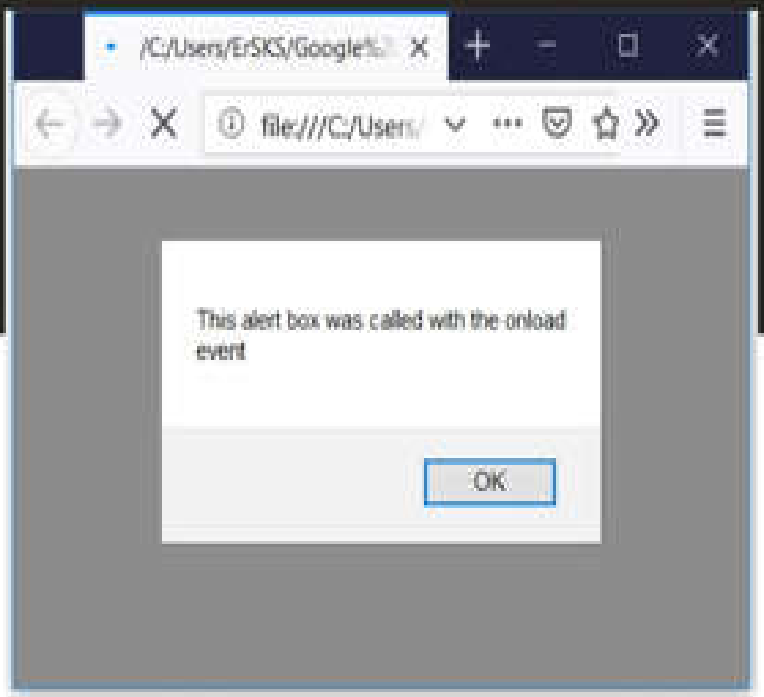
26_DegreeCelcius.html

```
<html>
<head>
  <script type="text/javascript">
    function convert(degree){
      var c_ID = document.getElementById("c");
      var f_ID = document.getElementById("f");
      if (degree == "Ce"){
        F = c_ID.value * 9 / 5 + 32;
        f_ID.value = Math.round(F);
      }else {
        C = (f_ID.value -32) * 5 / 9;
        c_ID.value = Math.round(C);
      }
    }
  </script>
</head>
<body>
  <p><b>Temperature Converter:</b></p>
  <form>
    <input id="c" name="c" onkeyup="convert('Ce')"> &#8451;<br>equals<br>
    <input id="f" name="f" onkeyup="convert('Fa')"> &#8457;
  </form>
  <p>Note that the <b>Math.round()</b> method is used, so that the result will be
  returned as an integer.</p>
</body>
</html>
```



Body onload Event

```html
<html>
<head>
    <script type="text/javascript">
        function message(){
            alert("This alert box was called with the onload event");
        }
    </script>
</head>
<body onload="message()">

</body>
</html>
```

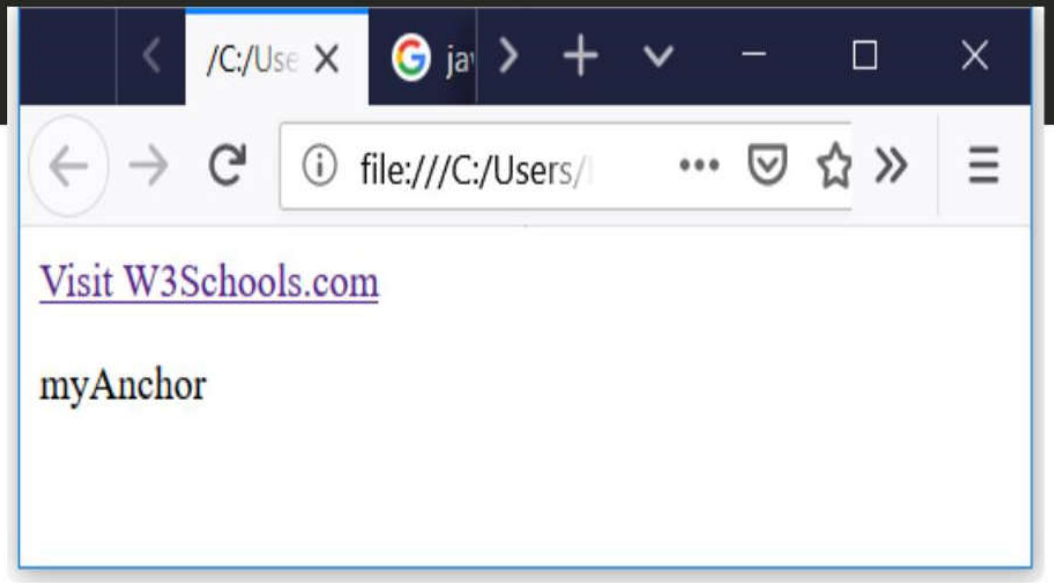


getElementById(), innerHTML & Date()

getElementById()

02_getelementbyid.html

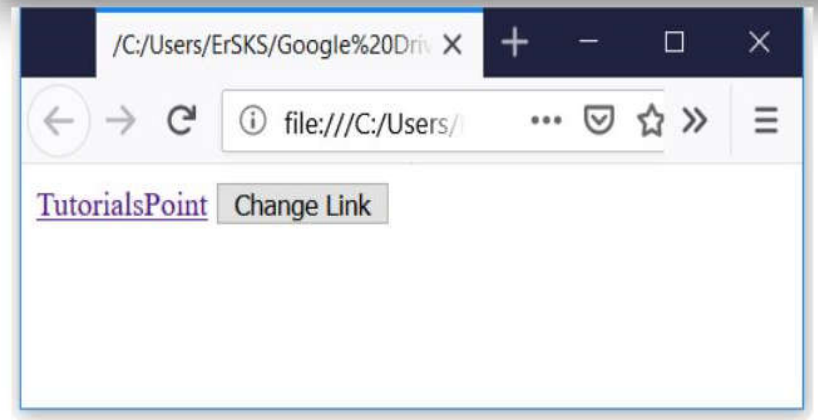
```
<html>
<body>
    <p>
        <a id="myAnchor" href="http://www.w3schools.com">Visit W3Schools.com</a>
    </p>
    <script type="text/javascript">
        var x=document.getElementById("myAnchor");
        document.write(x.id);
    </script>
</body>
</html>
```

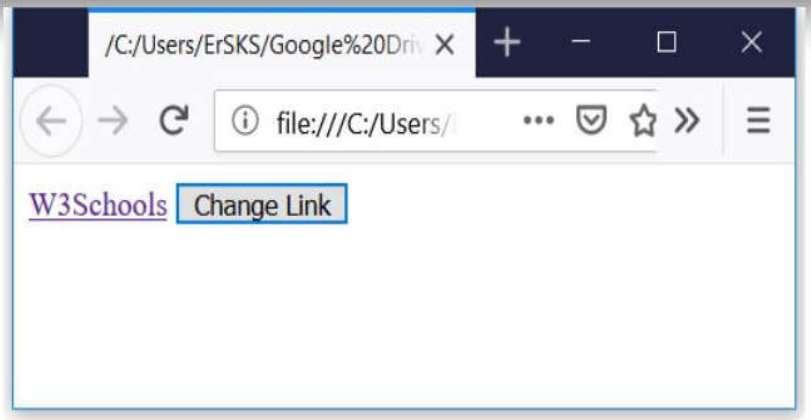


innerHTML

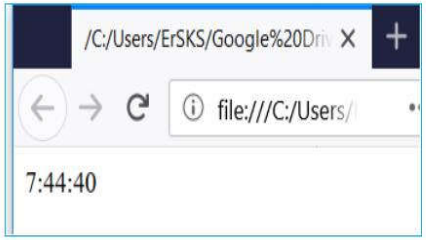
02_getelementbyid.html × 02_innerHTML.html ●

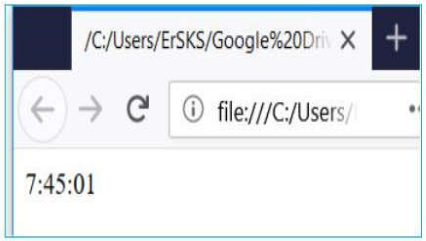
```
<html>
<head>
    <script type="text/javascript">
        function change_link(){
            document.getElementById('myAnchor').innerHTML="W3Schools";
            document.getElementById('myAnchor').href="https://www.w3schools.com/js/";
            document.getElementById('myAnchor').target="_blank";
        }
    </script>
</head>
<body>
    <a id="myAnchor" href="https://www.tutorialspoint.com/javascript/">TutorialsPoint</a>
    <input type="button" onclick="change_link()" value="Change Link">
</body>
</html>
```

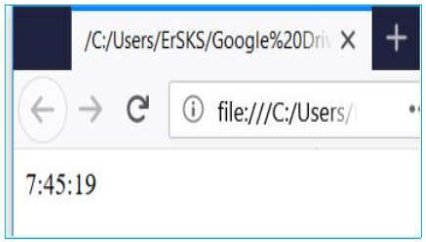




Digital Watch







02_getelementbyid.html | 02_innerHTML.html | 02_currentTime.html

```
<html>
<head>
    <script type="text/javascript">
        function startTime(){
            var today=new Date();
            var h=today.getHours();
            var m=today.getMinutes();
            var s=today.getSeconds();
            m=checkTime(m);
            s=checkTime(s);
            document.getElementById('time').innerHTML=h+":"+m+":"+s;
            t=setTimeout('startTime()',500);
        }
        function checkTime(i){
            if (i<10){
                i="0" + i;
            }
            return i;
        }
</script>
</head>
<body onload="startTime()">
    <div id="time"></div>
</body>
</html>
```

```html
<html>
<head>
  <script type="text/javascript">
    function disp_msg(){
      alert("Hello World!");
    }
  </script>
</head>
<body>
  <form>
    <input type="button" value="Click me!"
    onclick="disp_msg()" />
  </form>
</body>
</html>
```

14_Return_from_fxn.html

```html
<html>
<head>
  <script type="text/javascript">
    function my_func(){
      return ("Hello World!");
    }
  </script>
</head>
<body>
  <script type="text/javascript">
    document.write(my_func())
  </script>
</body>
</html>
```

12_ReturnStatement.html

```html
<html>
<head>
  <script type="text/javascript">
    function product(a,b){
      return a*b;
    }
  </script>
</head>
<body>
  <script type="text/javascript">
    document.write(product(4,3));
  </script>
</body>
</html>
```

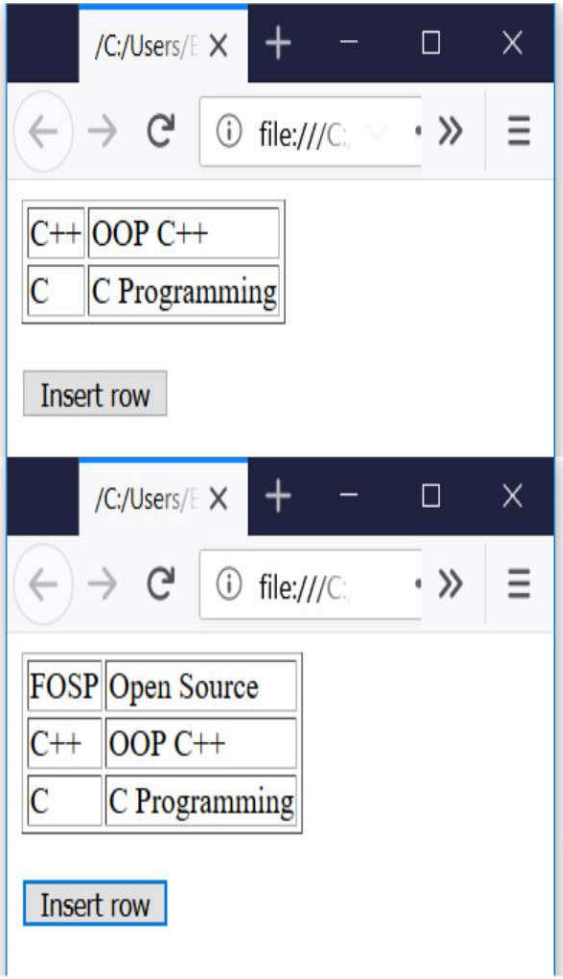
13_MsgPassing.html

```html
<html>
<head>
  <script type="text/javascript">
    function func(txt){
      alert(txt);
    }
  </script>
</head>
<body>
  <form>
    <input type="button" onclick="func('Hello')"
    value="Call Me">
  </form>
</body>
</html>
```

15_for_Loop.html

```html
<html>
<body>
  <script type="text/javascript">
    var i=0;
    for(i=0;i<=5;i++){
      document.write("The number is "+i+"<br>");
    }
  </script>
</body>
</html>
```

JS to Insert Row

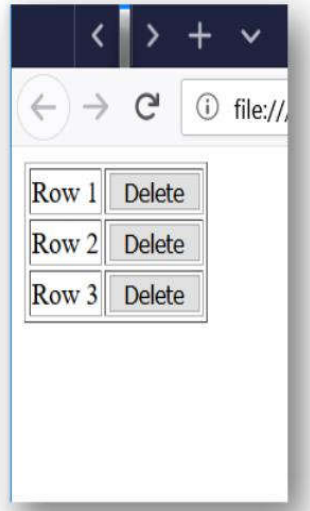


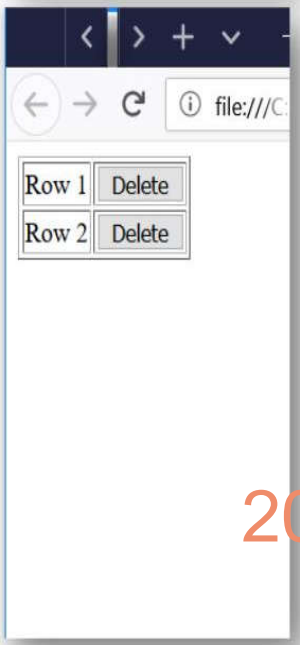
23_Insert_row_in_Table.html

```
<html>
<head>
  <script type="text/javascript">
    function insRow(){
      var x=document.getElementById('myTable').insertRow(0);
      var y=x.insertCell(0);
      var z=x.insertCell(1);
      y.innerHTML="FOSP";
      z.innerHTML="Open Source";
    }
  </script>
</head>
<body>
  <table id="myTable" border="1">
    <tr >
      <td>C++</td>
      <td>OOP C++</td>
    </tr>
    <tr >
      <td>C</td>
      <td>C Programming</td>
    </tr>
  </table>
  <input type="button" onclick="insRow()" value="Insert Row">
</body>
</html>
```

24_Del_Row.html

```
<html>
<head>
  <script type="text/javascript">
    function fxn(r){
      var i=r.parentNode.parentNode.rowIndex;
      document.getElementById('myTable').deleteRow(i);
    }
  </script>
</head>
<body>
  <table id="myTable" border="1">
    <tr >
      <td>Row 1</td>
      <td><input type="button" value="Delete" onclick="fxn(this)"></td>
    </tr>
    <tr >
      <td>Row 2</td>
      <td><input type="button" value="Delete" onclick="fxn(this)"></td>
    </tr>
    <tr >
      <td>Row 3</td>
      <td><input type="button" value="Delete" onclick="fxn(this)"></td>
    </tr>
  </table>
</body>
</html>
```

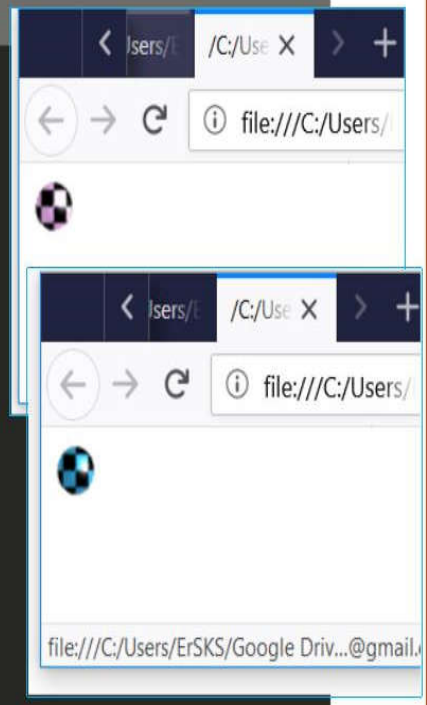




Mouse – Over & Out

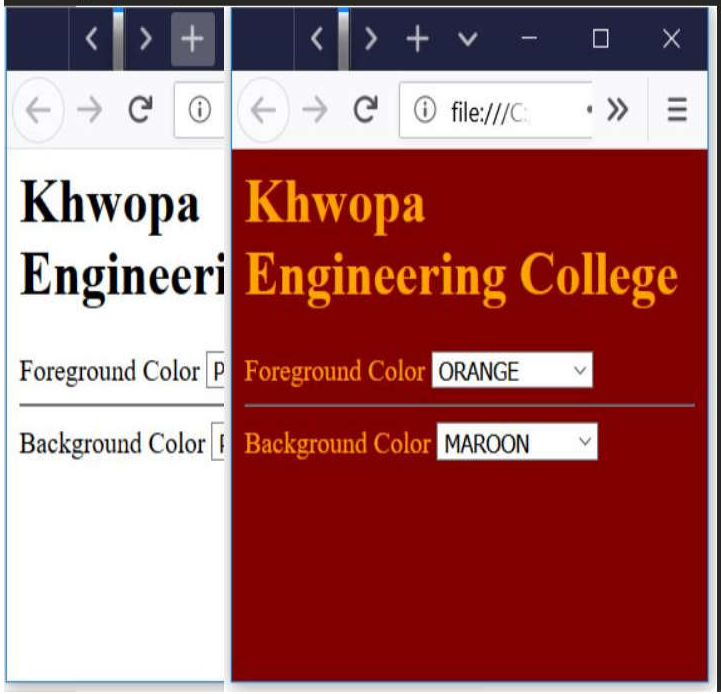
03_mouseover.html

```
<html>
<head>
    <script type="text/javascript">
        function mouseOver(){
            document.getElementById("b1").src ="b_blue.gif";
        }
        function mouseOut(){
            document.getElementById("b1").src ="b_pink.gif";
        }
    </script>
</head>
<body>
<a href="fb.com" target="_blank">
    <img border="0" alt="Ball" src="b_pink.gif" id="b1" width="26" height="26"
    onmouseover="mouseOver()" onmouseout="mouseOut()" />
</a>
</body>
</html>
```



28_Change_bg_fg_color.html

```
<html>
<head>
  <title>Foreground and Background Changing</title>
  <script type="text/javascript">
    function fg_color(abc){
      if(abc == "--")
        alert("Please Select Color !");
      else
        document.fgColor = abc;
    }
    function bg_color(abc){
      if(abc == "--")
        alert("Please Select Color !");
      else
        document.bgColor = abc;
    }
  </script>
</head>
```




```
<body>
  <h1>Khwopa Engineering College</h1>
  <form name="frm1" method="post" action="">
    Foreground Color
    <select name="fcolor" onchange="fg_color(this.value);">
      <option value="--">Please Select</option>
      <option value="RED">RED</option>
      <option value="GREEN">GREEN</option>
      <option value="BLUE">BLUE</option>
      <option value="PURPLE">PURPLE</option>
      <option value="MAROON">MAROON</option>
      <option value="ORANGE">ORANGE</option>
      <option value="WHITE">WHITE</option>
      <option value="PINK">PINK</option>
    </select><hr/>
    Background Color
    <select name="bcolor" onchange="bg_color(this.value);">
      <option value="--">Please Select</option>
      <option value="RED">RED</option>
      <option value="GREEN">GREEN</option>
      <option value="BLUE">BLUE</option>
      <option value="PURPLE">PURPLE</option>
      <option value="MAROON">MAROON</option>
      <option value="BLACK">BLACK</option>
      <option value="ORANGE">ORANGE</option>
      <option value="WHITE">WHITE</option>
      <option value="PINK">PINK</option>
    </select>
  </form>
</body>
</html>
```

Bg & Fg Color Changer 24

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

<h2 onclick="showCoords(event)">Click this heading to get the x (horizontal) and y (vertical) coordinates of the mouse pointer when it was clicked.</h2>

<p><strong>Tip:</strong> Try to click different places in the heading.</p>

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

<script>
function showCoords(event) {
  var x = event.clientX;
  var y = event.clientY;
  var coords = "X coords: " + x + ", Y coords: " + y;
  document.getElementById("demo").innerHTML = coords;
}
</script>

</body>
</html>
```

Result Size: 635 x 601

Click this heading to get the x (horizontal) and y (vertical) coordinates of the mouse pointer when it was clicked.

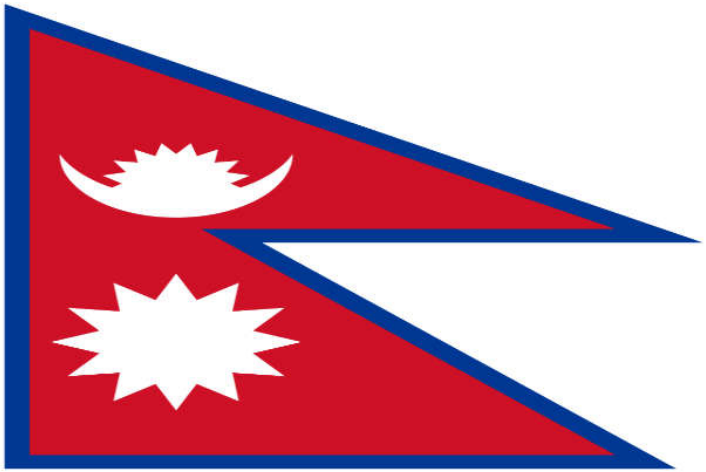
Tip: Try to click different places in the heading.

X coords: 509, Y coords: 46

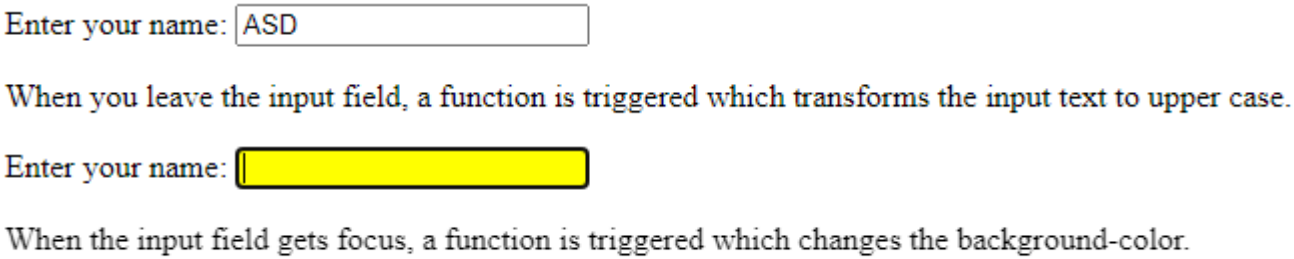
```html
<html>
    <head><title>onMouseMove Event</title></head>
<body>

<h2>Move Mouse/Cursor to get the x (horizontal) and y (vertical) coordinates of the mouse pointer.</h2>

<p id="pid"></p>
<img src="np.png" onmousemove="disp();">
<script type="text/javaScript">
function disp() {
  let x = event.screenX;
  let y = event.screenY;
  let coord = "X coords: " + x + ", Y coords: " + y;
  document.getElementById("pid").innerHTML = coord;
}
</script>

</body>
</html>
```

Move Mouse/Cursor to get the x (horizontal) and y (vertical) coordinates of the mouse pointer.

X coords: 374, Y coords: 437



```html
<html>
<head>
<script>
function myFunction1() {
  var x = document.getElementById("fname");
  x.value = x.value.toUpperCase();
}
function myFunction(x) {
  x.style.background = "yellow";
}
</script>
</head>
<body>
Enter your name: <input type="text" id="fname" onblur="myFunction1()">
<p>When you leave the input field, a function is triggered which transforms the input text to upper case.</p>
Enter your name: <input type="text" onfocus="myFunction(this)">
<p>When the input field gets focus, a function is triggered which changes the background-color.</p>
</body>
</html>
```



```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML =
"You selected some text";
}
</script>
</head>
<body>

Some text: <input type="text" value="Hello
world!" onselect="myFunction()">

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

</body>
</html>
```

Some text: Hello world!

You selected some text

Arrays

- JavaScript arrays are used to store multiple values in a single variable.
- An array is a special variable, which can hold more than one value at a time.
- An array can hold many values under a single name, and one can access the values by referring to an index number.

Defining Arrays

var *array_name* = [*item1*, *item2*, ...];

or

var *array_name* =new Array();

or

var *array_name* =new Array(n);

Assigning Values in Arrays with new keyword

```
var mycars =new Array();
```

or

```
var mycars =new Array(3);

mycars[0]="Saab";
mycars[1]="Volvo";
mycars[2]="BMW";
```

or

```
var mycars=new Array("Saab","Volvo","BMW");
```

Assigning Values in Arrays

```
var cars = ["Saab", "Volvo", "BMW"];
```

or

```
var cars = [
    "Saab",
    "Volvo",
    "BMW"
  ];
```

Accessing Arrays

- document.write(mycars[0]);

- var name = cars[0];

```
<script language=javaScript>
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById(“ID_Name").innerHTML = cars[0];
</script>
```

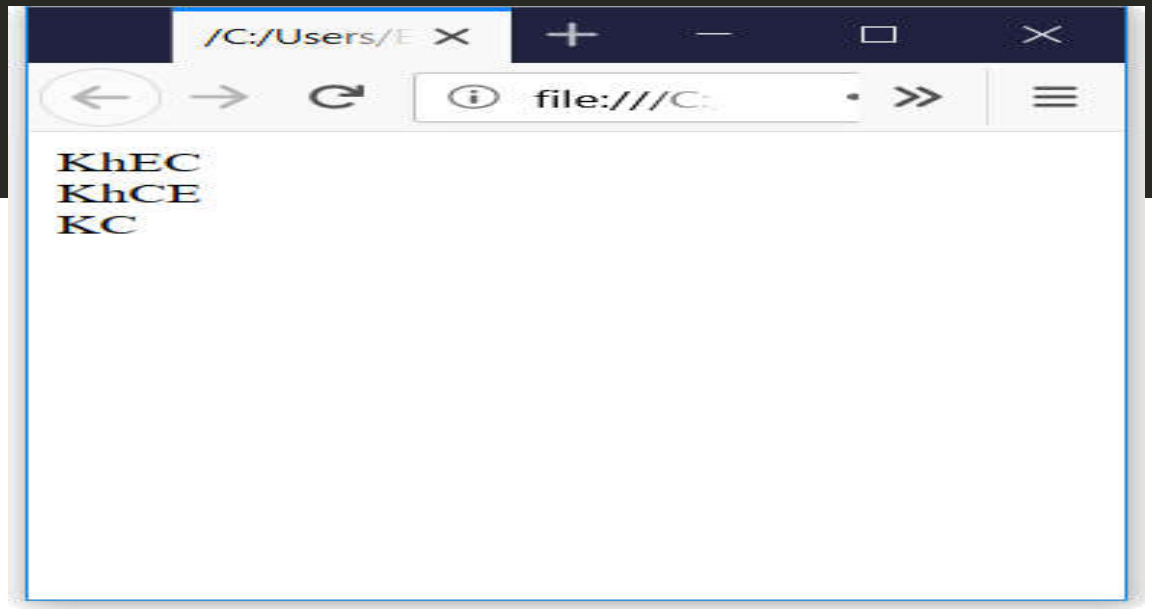
- **JavaScript For...In Statement**
  - is used to loop (iterate) through the elements of an array or through the properties of an object.
- **Syntax**

```
for (variable in object)
{
code to be executed
}
```

```
<html> <body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars) {
document.write(mycars[x] + "<br />");
}
</script>
</body> </html>
```

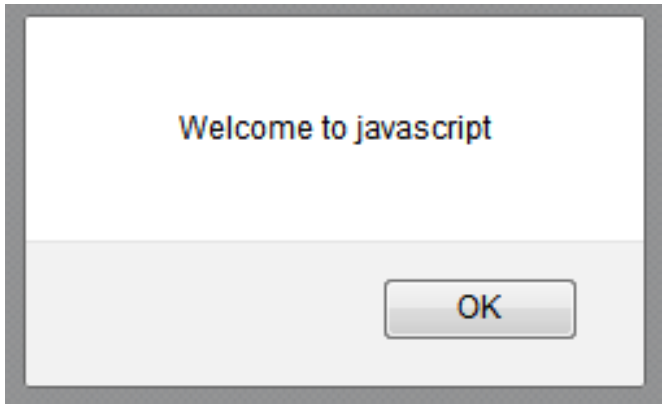
21_for_Loop_in.html

```
1  <html>
2  <body>
3    <script type="text/javascript">
4      var x;
5      var co = new Array("KhEC", "KhCE", "KC");
6      for (x in co) {
7        document.write(co[x] + "<br />");
8      }
9    </script>
10 </body>
11 </html>
```



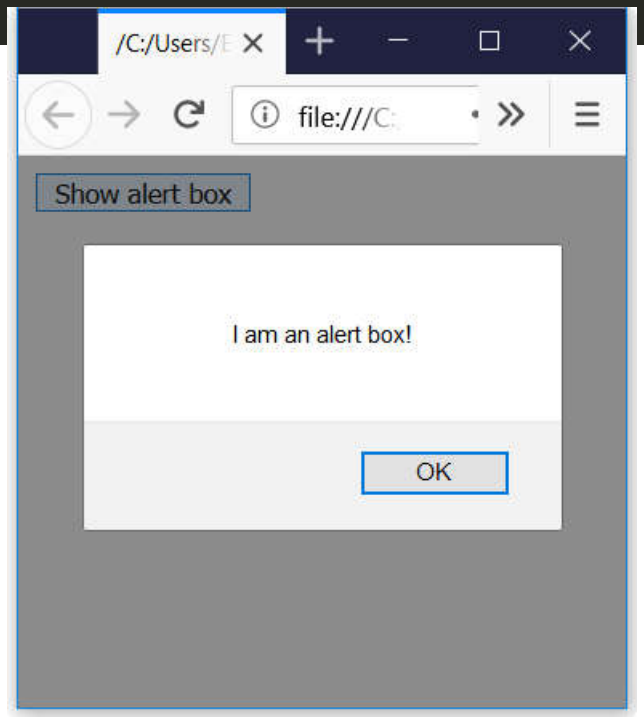
- **JavaScript Popup/Dialog Boxes**
- **Alert Box**
- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- **Syntax:**
- alert("sometext");
- alert('Welcome to javascript');

Output:



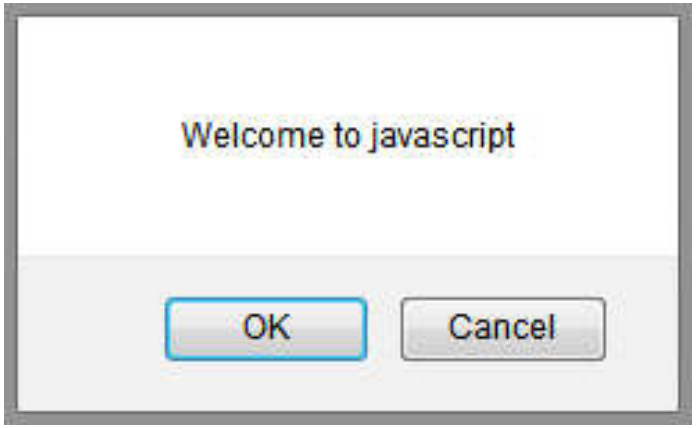
31_AlertBox.html

```
1  <html>
2  <head>
3    <script type="text/javascript">
4      function show_alert(){
5        alert("I am an alert box!");
6      }
7    </script>
8  </head>
9  <body>
10   <input type="button" onclick="show_alert()" value="Show alert box" />
11 </body>
12 </html>
```



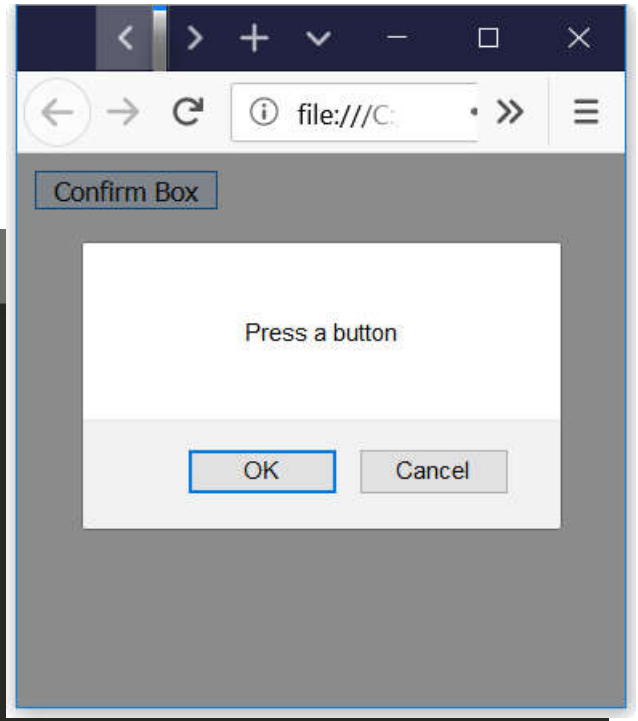
- **Confirm Box**
- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.
- **Syntax:**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

Output:



31_AlertBox.html | 32_ConfirmBox.html

```
1  <html>
2  <head>
3    <script type="text/javascript">
4      function show_confirm(){
5        var r=confirm("Press a button");
6        if (r==true) {
7          alert("You pressed OK!");
8        }
9        else {
10         alert("You pressed Cancel!");
11       }
12     }
13   </script>
14 </head>
15 <body>
16   <input type="button" onclick="show_confirm()" value="Confirm Box" />
17 </body>
18 </html>
```

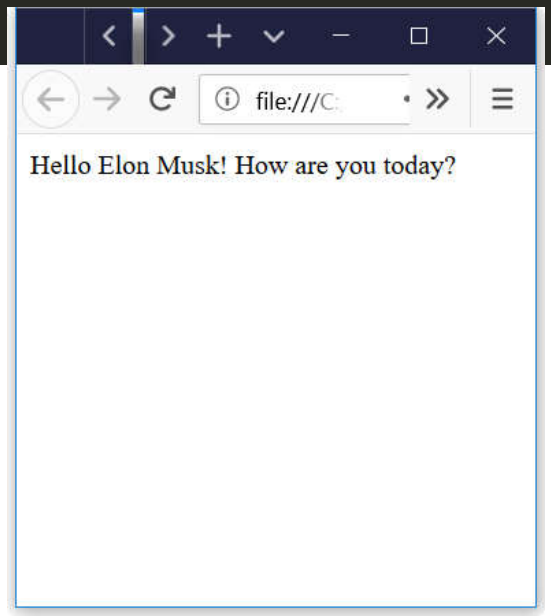


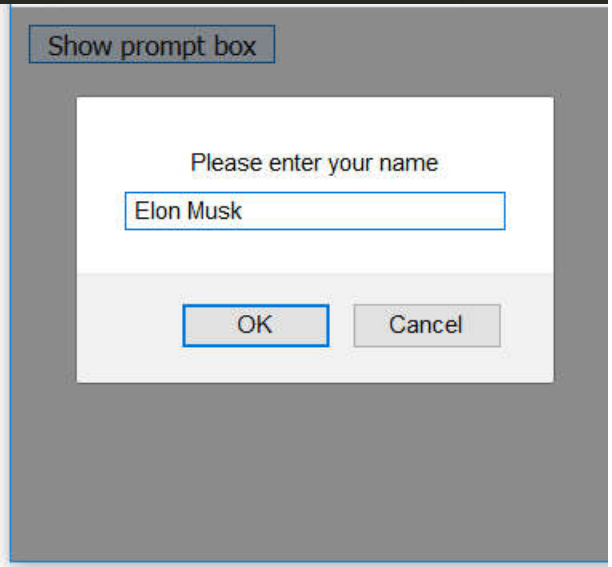
- **Prompt Box**
- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user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

Output:

31_AlertBox.html | 32_ConfirmBox.html | 33_PromptBox.html

```
<html>
<head>
  <script type="text/javascript">
    function show_prompt(){
      var name=prompt("Please enter your name","Harry Potter");
      if (name!=null && name!="") {
        document.write("Hello " + name + "! How are you today?");
      }
    }
  </script>
</head>
<body>
  <input type="button" onclick="show_prompt()" value="Show prompt box" />
</body>
</html>
```





Built-in Array Properties and Methods

- **The length Property**
- The length property of an array returns the length of an array (the number of array elements).
- var x = cars.length; // The length property returns the number of elements
- **Example**
- var fruits = ["Banana", "Orange", "Apple", "Mango"];
- document.write(fruits.length);
- // the length of fruits is 4

- **Accessing the First Array Element**
- **Example**
- fruits =
  ["Banana", "Orange", "Apple", "Mango"];
  var first = fruits[0];
- **Accessing the Last Array Element**
- **Example**
- fruits =
  ["Banana", "Orange", "Apple", "Mango"];
  var last = fruits[fruits.length - 1];

- **Sorting an Array**
- The sort() method sorts an array alphabetically:
- Example
- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort(); // Sorts the elements of fruits
- **Reversing an Array**
- The reverse() method reverses the elements in an array.
- You can use it to sort an array in descending order:
- Example
- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort(); // First sort the elements of fruits
  fruits.reverse(); // Then reverse the order of the elements

Popping and Pushing

- When you work with arrays, it is easy to remove elements and add new elements.
- This is what popping and pushing is:
  - Popping items **out** of an array, or pushing items **into** an array.
- These are done by **pop()** and **push()** methods.

Popping

- The pop() method removes the last element from an array:
- Example
- var fruits =
  ["Banana", "Orange", "Apple", "Mango"];
  var x = fruits.pop(); // Removes the last element ("Mango") from fruits
- Document.write(x); //Returns Mango

Pushing

- The push() method adds a new element to an array (at the end):
- Example
- var fruits =
  ["Banana", "Orange", "Apple", "Mango"];
  fruits.push("Kiwi"); // Adds a new element ("Kiwi") to fruits
- Document.write(fruits);
- // returns Banana,Orange,Apple,Mango,Kiwi
- //Document.write(fruits.push(“Kiwi"));
- // Returns 5

Shifting Elements

- Shifting is equivalent to popping, working on the first element instead of the last.
- The shift() method removes the first array element and "shifts" all other elements to a lower index.
- **Example**
- var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.shift(); // Removes the first element "Banana" from fruits
- Document.write(fruits);
- // returns Orange,Apple,Mango

Unshifting Elements

- The unshift() method adds a new element to an array (at the beginning), and "unshifts" older elements:
- **Example**
- var fruits = ["Banana", "Orange", "Apple", "Mango"]; fruits.unshift("Lemon"); // Adds a new element "Lemon" to fruits
- Document.write(fruits);
- // returns Lemon,Banana,Orange,Apple,Mango
- //Document.write(fruits.unshift("Lemon"));
- // Returns 5