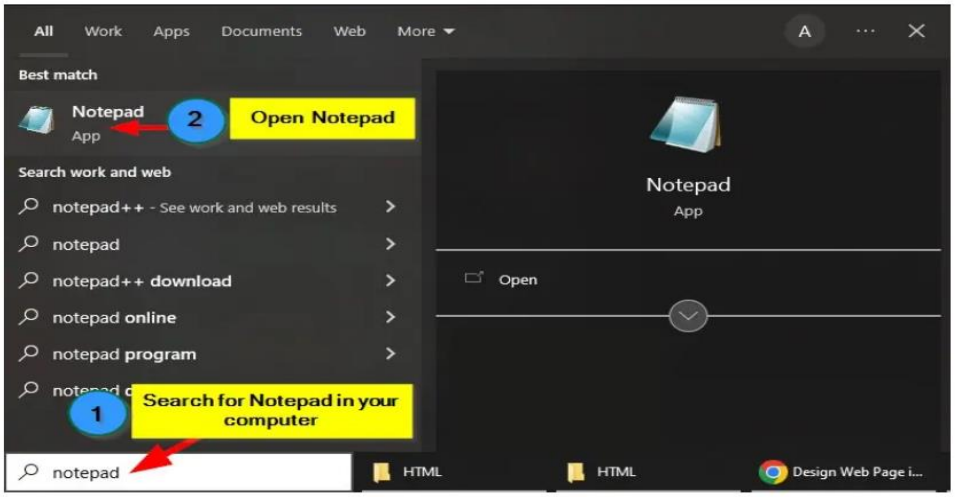
## UNIT-I: SCRIPTING

### ❖ Web page Design using HTML (Step-by-Step):

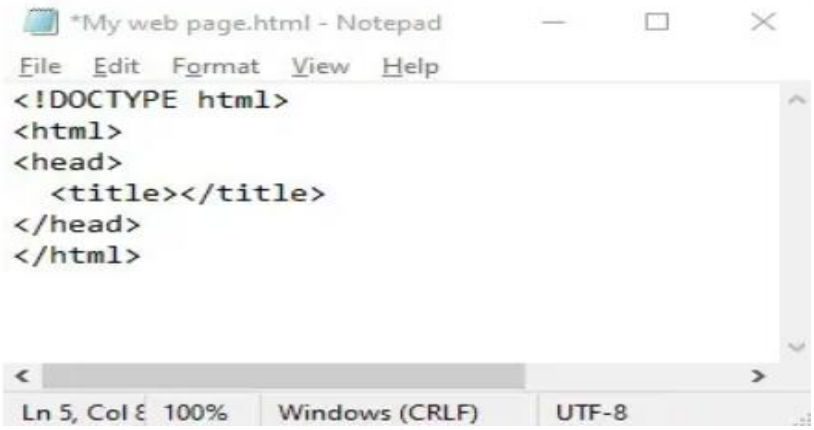
### Step 1: Set up Your Project

- Create a new folder on your computer to store the files for your web page.
- Open a text editor like Notepad, Sublime Text, or Visual Studio Code in order to write your HTML code.




### Step 2: Start with the HTML Structure

- Begin your HTML file by adding the **<!DOCTYPE html>** declaration at the top. This tells the browser that you're using HTML5.
- Create the opening and closing HTML tags: **<html></html>**.
- Inside the HTML tags, create the opening and closing head tags: **<head></head>**.
- Within the head tags, add the opening and closing titletags: **<title></title>**. This is where you'll write the title of your web page.



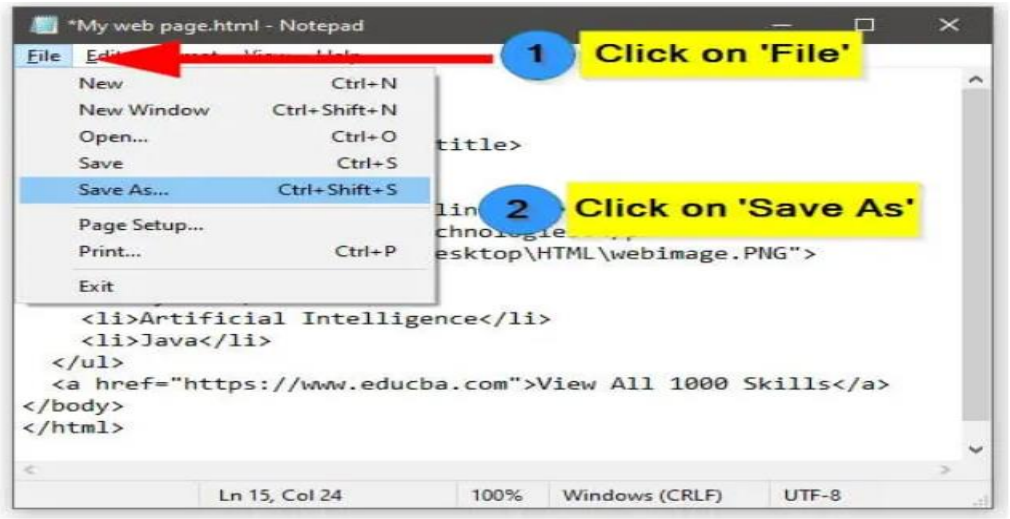

**Step 3: Add Content to the Body**

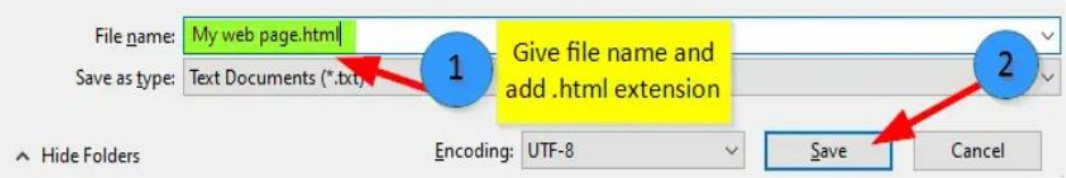
- After the closing head tag, create the opening and closing body tags: <body></body>. This is where you'll put all the visible content of your web page.
- Inside the body tags, you can start adding elements such as headings, paragraphs, images, and links.
- To add a heading, use the **<h1></h1>** tags for the main heading and **<h2></h2>**, **<h3></h3>**, and so on for subheadings.
- To add a paragraph, use the ** **

```
<!DOCTYPE html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>WELCOME TO SRI INDU</h1>
<P>WECOME TO AI&DS CLASS</P>
</body>
</html>
```

## Step 4: Save your HTML File

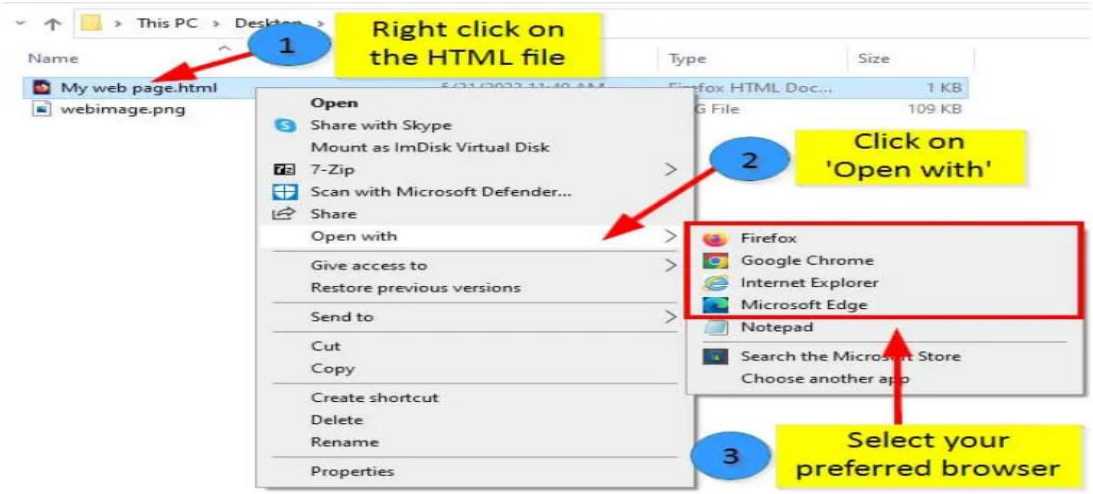
- Save your file with a **.html extension** in the folder you created earlier.Choose a descriptive name for your file, such as **my web page.html**




## Step 5: View your Web Page

- Open the HTML file in a web browser such as Chrome, Firefox, or Safari.



- You should see the below web page displayed in the browser, showing the content we added.

**OUTPUT:**


❖ **Client -Side and Server -side Scripting:**

1.**Client -Side:**

- Web browsers execute client-side scripting.
- It is used when browsers have all code.
- Source code is used to transfer from web server to user's computer over the internet and run directly on browsers.
- It is also used for validations and functionality for user events.

**2.Server-side:**

- Web servers are used to execute server-side scripting.
- They are basically used to create dynamic pages.
- It can also access the file system residing at the web server.
- A server-side environment that runs on a scripting language is a web server.
- The server sends pages to the request of the user/client.

❖ **Difference between client-side scripting and server-side scripting :**

| **Client-side scripting** | **Server-side scripting** |
|---|---|
| Source code is visible to the user. | source code is not visible to the user because its output of server-side is an HTML page. |
| It usually depends on the browser and its version. | In this any server-side technology can be used and it does not depend on the client. |
| It runs on the user's computer. | It runs on the webserver. |
| It does not provide security for data. | It provides more security for data. |
| HTML, CSS, and java script are used. | PHP, Python, Java, Ruby are used. |
| No need of interaction with the server. | It is all about interacting with the servers. |
| It reduces load on processing unit of the server. | It surge the processing load on the server. |

❖ **JavaScript Objects:**

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

**Creating Objects in JavaScript:**

There are 3 ways to create objects.

Advertisement

1. By object literal
2. By creating instance of Object directly (using new keyword)
3. By using an object constructor (using new keyword)

**1.JavaScript Object by object literal:**

The syntax of creating object using object literal is given below:

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

As you can see, property and value is separated by : (colon).

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>object</title>
<body>
<script>
emp={id:551,name:"Anu",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
</head>
```

```
</body>
</html>
```

**Output:**

551 Anu 40000

**2.By creating instance of Object:**

Here, new keyword is used to create object. The syntax of creating object directly is given below:

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

**Example:**

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

**Output:**

101 Ravi 50000

**3.By using an Object constructor**

Here, you 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.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>object</title>
<body>
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal",30000);
document. write(e.id+" "+e.name+" "+e.salary);
</script>
</head>
</body>
</html>
```

**Output:**

103 Vimal  30000

**JavaScript Literals:**

In JavaScript, literals are fixed values written directly into the code. Unlike variables or complex expressions, literals offer an explicit and concrete value, making coding more straightforward and manageable.

These are differenttypes  literals,

1. **Array literals** :A collection of values, each of which is an array element, contained within square brackets.

   **Example:** var colors = [“Red”, “Blue”, “Green”, “Yellow”]

2. **Boolean literals** :Boolean literal can have two values, either true or false.
3. **Floating point literals** : It store values with or with out decimals, fractions, and exponents
4. **Integer literals** : It represent positive or negative numbers with at least one digit from 0–9, excluding spaces or commas
5. **Object literals** :A collection of zero or more {name:value} pairs, enclosed in curly brackets
6. **String literals:** A sequence of characters enclosed in double or single quotes, like "web flow," "12345," and "an example of a string literal"

**JavaScript Identifiers / Names:**

- Identifiers are JavaScript names. Identifiers are used to name variables and keywords, and functions.
- The rules for legal names are the same in most programming languages.

A JavaScript name must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

Subsequent characters may be letters, digits, underscores, or dollar signs.

❖ **JavaScript Operators:**

JavaScript operators are Special 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. String Operators
4. Bitwise Operators
5. Logical Operators
6. Assignment Operators
7. Special Operators

1. **Arithmetic Operators:**

- Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.
- Assignment operators are like Addition, Subtraction, Multiplication, Division, Modulus, Increment, Decrement etc

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Arithmetic</title>
<script>
var x= 10,y=5;
document. write("Addition: x + y = ", x + y);
document. write("<br>Subtraction: x - y =", x - y);
document. write("<br>Multiplication: x * y =", x * y);
document. write("<br>Division: x / y =", x / y);
document. write("<br>Remainder: x % y =", x % y);
document. write("<br>Increment: ++x =", ++x);
document. write("<br>Decrement: --x =", --x);
</script>
</head>
</body>
</html>
```

**Output:**

```
Addition: x + y =  15
Subtraction: x - y = 5
Multiplication: x * y = 50
Division: x / y = 2
```

Remainder: x % y = 0

Increment: ++x = 11

Decrement: --x = 10

**2.Realtional Operators:**

- The JavaScript relational operator compares the two operands and return value either true or false.
- The comparison operators are as follows Greater than(>),Less than(<),Greater than or equal to(>=),Less than or equal to(<=),Is equal to(==),Not equal to(!=) etc.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Realtional</title>
<script>
var x=20,y=10;
document. write("<br>Equal to: x == y is", x == y);
document. write("<br>Not equal to: x != y is", x != y);
document. write("<br>Greater than: x > y is", x > y);
document. write("<br>Less than: x > y is", x > y);
document. write("<br>Greater than or equal to: x >= y is", x >= y);
document. write("<br>Less than or equal to: x <= y is", x <= y);
</script>
</head>
</body>
</html>
```

**Output:**

Equal to: x == y is false

Not equal to: x != y is true

Greater than: x > y is true

Less than: x > y is true

Greater than or equal to: x >= y is true

Less than or equal to: x <= y is false

**3.String operators:**

- String operator is used to perform concatenate(join) two or more than string.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>String</title>
<script>
var x="Anjali",y="laxmi";
document.write("string: x + y is :", x +y);
</script>
</body>
</head>
</html>
```

**Output:**

string: x + y is :Anjalilaxmi

**4.Bitwise operators:**

- Bitwise operators are used to perform operations on binary numbers(0's and 1's)
- The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows Bitwise AND ,Bitwise OR,Bitwise NOT, Bitwise XOR, Bitwise Left Shift ,Bitwise Right Shift Bitwise Right Shift with Zero etc.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Bitwise</title>
<script>
    // 1 = 00000001
    // 2 = 00000010
    // ------------
    //     00000000 = 0
document. write(1^2);
document. write(1&2);
document. write(1|2);
document. write(~1);
document. write(1<< 2);
document. write(1>> 2);
</script>
</body>
</head>
</html>
```

**Output:**

```
    3
    0
    3
    -2
    4
    0
```

**5.Logical Operators:**

- Logical operator is used to perform compare two relational expression and return Boolean values either true or false.
- Logical operators are like logical AND, logical OR, logical NOT etc.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Logical</title>
<script>
var x = 5,y=2;
document. write((x < y) && (x > y));
document. write ((x < y) && (x > y));
document.  write ((x > y) || (x > y));
document. write ((x > y) || (x < y));
document. write (!(x == y));
</script>
</body>
</head>
</html>
```

**Output:**

false

false

true

true

true

**6.Assignment Operator:**

- Assignment operator is used to perform assign value of variable.

- Assignment operators are like Addition assignment, Subtraction assignment, Multiplication assignment, Division assignment, Remainder assignment etc.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Assignment</title>
<script>
var a = 7;
document. write("Assignment: a = 7, a =", a);
a += 5;  // a = a + 5
document. write("Addition Assignment: a += 5, a =", a);
a -= 5;  // a = a - 5
document. write("Subtraction Assignment: a -= 5, a =", a);
a *= 2;  // a = a * 2
document. write("Multiplication Assignment: a *= 2, a =", a);
a /= 2;  // a = a / 2
document. write("Division Assignment: a /= 2, a =", a);
a %= 2;  // a = a % 2
document. write("Remainder Assignment: a %= 2, a =", a);
</script>
</body>
</head>
</html>
```

**Output:**

```
Assignment: a = 7, a = 7
Addition Assignment: a += 5, a = 12
Subtraction Assignment: a -= 5, a = 7
```

Multiplication Assignment: a *= 2, a = 14

Division Assignment: a /= 2, a = 7

Remainder Assignment: a %= 2, a = 1

**7.Special Operators:**

- Special operator is used to perform special operations on the operand. The following operators are known as JavaScript special operators.

**Expression:**

- An expression is a combination of operators, constants and variables.
- An expression value will be stored in this variables.

RESULT=X+Y/Z;

❖ **Control Statements:**

Control statements are used to decide the flow of execution based on different conditions.

These are different types of control statements,

1.If Statement

2.If-else Statement

3. If-else-if Statement

4.Nested if-else Statement

5.Switch Statements

**1.If Statement:**

The JavaScript if statement is used to only if expression is true.

**Syntax:**

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

**Example:**

```
<DOCTYPE html>
<html>
```

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

**Output:**

value of a is greater than 10

**2.If-else Statement:**

The JavaScript if-else statement is used to execute the code whether condition is true or false.

**Syntax:**

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

**Example:**

```
<DOCTYPE html>
<html>
```

```
<head>
<title>If-else statement</title>
<body>
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
</head>
</body>
</html>
```

**Output:**

```
a is even number
```

**3.If-else-if Statement**:

The java script if-else-if statement is used to only if expression is true from several expressions.

**Syntax**:

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
```

```
}
else{
//content to be evaluated if no expression is true
}
```

**Example:**

```
<DOCTYPE html>
<html>
<head>
<title>If-else-if statement</title>
<body>
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
</head>
</body>
</html>
```

**Output:** a is equal to 20

**4.Nested if-else Statement:**

The java script Nested if -else statement is used to more than two expressions based on whether the conditions are true or false.

**Syntax:**

```
if(expression1){
if(expression2){
//content to be evaluated if expression1&2 is true
}
else{
//content to be evaluated if else expression is true
}
}
else{
if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
}
```

**Example:**

```
<DOCTYPE html>
<html>
<head>
<title>Nested if -else statement</title>
<body>
<script>
var x = 10,y=20,z=30;
if (x<y) {
```

```
if (x<z) {
document.write ("X is smallest number");
}
else {
document.write("z is smallest number");
}
}
else {
if (y <z) {
document.write ("y is smallest number");
}
else{
document.write ("z is smallest number");
}
}
</script>
</script>
</head>
</body>
</html>
```

**Output:**

More than one-digit positive number

**5.Switch Statements:**

The JavaScript switch statement is used to execute one code from multiple expressions.

**Syntax:**

```
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;
}
```

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Switch</title>
<body>
<script>
var grade='B';
var result;
switch(grade){
case 'A':
result="A Grade";
break;
case 'B':
result="B Grade";
break;
case 'C':
result="C Grade";
break;
default:
```

```
result="No Grade";
}
document.write(result);
</script>
</body>
</head>
</html>
```

**Output:**

B Grade

**Features:**

1. Object-Centered Script Language
2. Client edge Technology
3. Validation of User's Input
4. Else and If Statement
5. Interpreter Centered
6. Ability to perform In Built Function
7. Case Sensitive format
8. Light Weight and delicate
9. Statements Looping
10. Handling Events

❖ **Events:**

- JavaScript Events are actions or occurrences that happen in the browser. They can be triggered by various user interactions or by the browser itself.
- Common events include mouse clicks, keyboard presses, page loads, and form submissions.
- Event handlers are JavaScript functions that respond to these events, allowing developers to create interactive web applications.

**Syntax:**

<HTML-element Event-Type = "Action to be performed">

Some of the HTML events and their event handlers are:

1. Mouse events
2. Keyword events
3. Form events
4. Window/document events

**1. Mouse events:**

Events that occurs when the mouse interacts with the html document.

Important mouse events are:

| Event Performed | Event Handler | Description |
|---|---|---|
| Click | onclick | When mouse click on an element |
| doubleclick | ondblclick | When mouse double 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. |

**Example: ONCLICK&ONDBLCLICK**

```
<!DOCTYPE html>
<html>
<head>
<title>Javascript Events</title>
<body>
<script >
function clickevent()
{
document.write("This is Javascript");
}
</script>
<form>
<input type="button" onclick="clickevent()" value="Who's this?"/>
<input type="button" ondblclick="clickevent()" value="Who's this?"/>
</form>
</head>
</body>
</html>
```

**Output:**

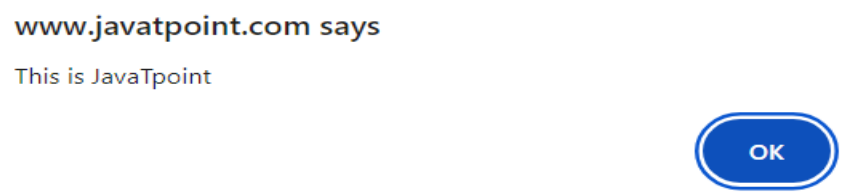

This is Javascript This is Javascript

**Example: MOUSE OVER EVENT**

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Javascript Events</title>
<body>
<script >
function mouseoverevent()
{
alert("This is JavaTpoint");
}
</script>
<p onmouseover="mouseoverevent()"> Keep cursor over me</p>
</head>
</body>
</html>
```

**Output:**




Keep cursor over me

2. **Keyboard events:**

The key events happen whenever a user interacts with keyboard. Event that occurs when user presses a key on the keyboard

These are mainly three key events:

| Event Performed | Event Handler | Description |
|---|---|---|
| Keydown | onkeydown | When the user press the key |
| Keyup | onkeyup | When the user release the key |

| | | |
|---|---|---|
| Keypress | onkeypress | When the user presses the key |

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<body>
<title>JavaScript  events</title>
<h2> Enter something here</h2>
<input type="text" id="input1" onkeydown="keydownevent()"/>
<input type="text" id="input1" onkeyup="keyupevent()"/>
<input type="text" id="input1" onkeypress="keypressevent()"/>
<script>
function keypressevent()
{
document.getElementById("input1");
alert("Pressed a key");
}
</script>
</head>
</body>
</html>
```

**Output:**

**3.Form events:**

Event that occurs through the interaction of a user with an html form are called java script form events

Important form events are:

| Event Performed | Event Handler | Description |
|---|---|---|
| Focus | onfocus | When the user focuses on an element |
| submit | onsubmit | When the user submits the form |
| Reset | onreset | When the user sresets 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 |

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Javascript Events</title>
<script>
function event()
{
alert("The form was submitted  ")
}
function events()
{
```

```
alert("The form was reset ")
}
</script>
<body>
<form>
<form onsubmit="event" onreset="events">
Enter name:<input type="text" name="fname"><br>
Enter password:<input type="text" name="fpswd"><br>
<input type="submit" value="submit">
<input type="reset" value="reset">
</form>
</head>
</body>
</html>
```

**Output:**

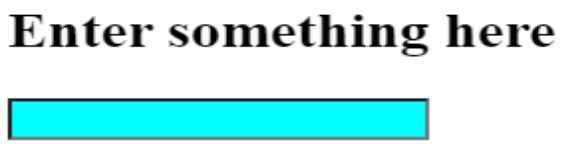
Enter name:
Enter password:
submit reset

**Example: FOCUS EVENT**

```
<!DOCTYPE html>
<html>
<head>
<title>event focus</title>
<body>
<h2> Enter something here</h2>
<input type="text" id="input1" onfocus="focusevent()"/>
<script>
function focusevent()
```

```
{
document.getElementById("input1").style.background=" aqua";
}
</script>
</head>
</body>
</html>
```

**Output:**




**4.window/document events:**

Events triggered for the window object(applies to the <body>tag). Important window events are:

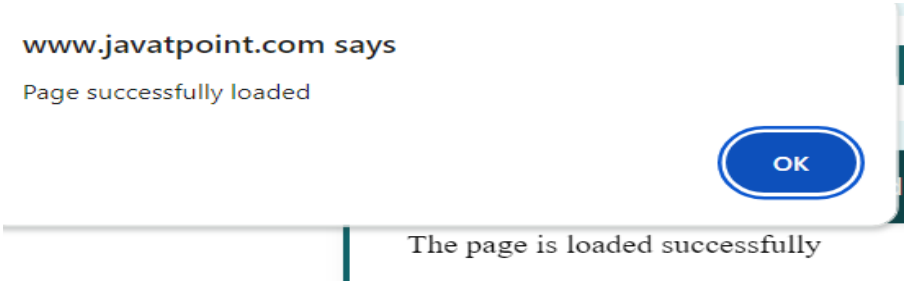
| Event Performed | Event Handler | Description |
|---|---|---|
| Load | onload | When the browser finishes the loading of the page |
| unload | onunload | When the visitor leaves the current webpage, the browser unloads it |
| Resize | onresize | When the visitor resizes the window of the browser |

**Example:**

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>document events</title>
<body>
</br>
<body onload="window.alert('Page successfully loaded');">
body onunload="window.alert('Page successfully unloaded');">
<script>
document.write("The page is loaded successfully");
</script>
</head>
</body>
</html>
```

**Output:**




The page is loaded successfully

- **Frames:**

- HTML frames are used to divide your browser window into multiple sections where each section can load a separate HTML document independently.
- A collection of frames in the browser window is known as a frameset.
- The window is divided into frames in a similar way the tables are organized: into rows and columns.

**Attributes of Frameset Tag:**

Attributes are different types of frameset tags,

1.Rows

2.Columns

3.Border

4.Border Colour

**1.Rows:** It is used to divide a single web pages into multiple rows horizontally.

**Example: Frame1.html**

```
<!DOCTYPE html>
<html>
<head>
<style>
div{
background-color: #7fffd4;
height: 500px;
}
</style>
</head>
<body>
<div>
<h2>This is first frame</h2>
</div>
</body>
</html>
```

**Frame2.html**

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
div{
background-color: #2f4f4f;
height: 500px;

}
</style>
</head>
<body>
<div>
<h2>This is Second frame</h2>
</div>
</body>
</html>
```

**Frame3.html**

```
<!DOCTYPE html>
<html>
<head>
<style>
div{
background-color: #c1ffc1;
height: 500px;
}
</style>
</head>
<body>
<div>
<h2>This is Third frame</h2>
</div>
```

```
</body>
</html>
```

**Fr.html**

```
<!DOCTYPE html>
<html>
<head>
<title>Frame tag</title>
</head>
<frameset cols="25%,50%,25%">
<frame src="frame1.html" >
<frame src="frame2.html">
<frame src="frame3.html">
</frameset>
</html>
```

**Output:**

| | | |
|---|---|---|
| This is first frame | This is Second frame | This is Third frame |

**2.Columns:** It is used to divide a single web pages into multiple columns Vertically.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Frame tag</title>
<frameset rows="30%, 40%, 30%">
<frame name="Top" src="frame1.html" >
<frame name="Main" src="frame2.html">
<frame name="Bottom" src="frame3.html">
</frameset>
</head>
</html>
```

**Output:**

**Top** **Main** **Bottom**

| | | |
|---|---|---|
| This is first frame | This is Second frame | This is Third frame |

**3. Frame Border:** It is used to specify each frame border is separately.

**4.Border Colour:** It is used to specify the border colour.

- **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

1. **primitive data types:**

The predefined data types provided by JavaScript language are known as primitive data types. There are five types of primitive data types in JavaScript.

1. **Number:** Numbers are always stored in double-precision 64-bit binary format IEEE 754.

   These are two types of numbes,

   **1.Integer:**It stores with out decimal point. **Ex:**100

   **2.Float:** It stores with decimal point. **Ex:**100.5
2. **String:** String is a sequence of characters e.g. "hello".
3. **Boolean:** It represents boolean value either false or true
4. **Null:** It represents null i.e. no value at all.
5. **Undefined:** A variable that has not been assigned a value is undefined.

**2. Non-Primitive Data Types:**

The data types that are derived from primitive data types of the JavaScript language are known as non-primitive data types. It is also known as derived data types or reference data types.

There are two types of Non primitive data types in JavaScript.

1. **Object:** It represents instance through which we can access members
2. **Array:** It represents group of similar types of elements.
3. **RegExp:** It represents regular expression.A regular expression is a character sequence defining a search pattern.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Dtatypes</title>
```

```
<body>
let integer_number =20;
document.write(integer_number);
let float_number = 3.15;
document.write(float_number);
let fruit = 'Apple';
document.write(fruit)
let name;
document.write(name);
let number = null;
document.write(number);
let dataChecked = true;
document.write(dataChecked);
let valueCounted = false;
document.write(valueCounted);
</body>
</head>
</html>
```

**Output:**

20

3.15

Apple

undefined

null

true

false

- **Functions:**

- A function is a reusable block of code that performs a specific task.

- You define it once, and then you can run (or “call”) it whenever you need that task done in your program.
- A JavaScript function runs when it is “called” by some part of your code

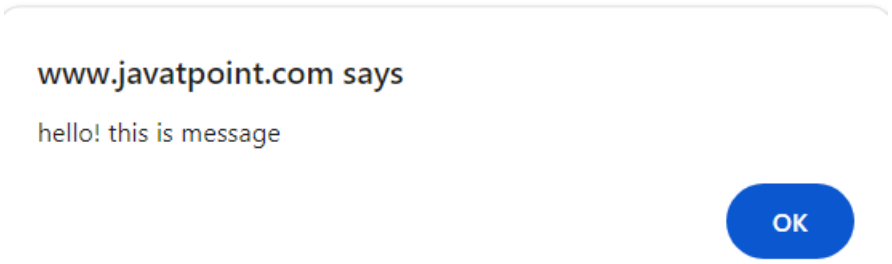
**Syntax:**

```
function functionName(Parameter1, Parameter2, ...)
{
// Function body
}
```

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Functions</title>
<body>
<script>
function msg(){
alert("hello! this is message");
}
</script>
<input type="button" onclick="msg()" value="call function"/>
</body>
</head>
</html>
```

**Output:**




we can categorize functions into two categories:

1. User defined Function
2. Built-in Function

**1.User-defined Function:**

- User-defined Function allows us to define our own functions as per our requirement.
- Let's understand how to create our own functions and call them in our script to use them.

**Function with Parameters**:

` We have a simple function for multiply defined which will take two numeric values as input and will multiply them and return the result.

**Example**:

```
<html>
<head>
<title>Parameter</title>
<body>
<script type="text/javaScript">
function multiplyThem(x, y)
{
return x * y;
}
document.write(multiplyThem(2, 7));
</script>
</head>
```

```
</body>
</html>
```

**Output:**14

**Function Arguments:**

We can call function by passing arguments.While calling a function, when we provide values for the function parameters, we call them arguments.

**Example:**

```
<html>
<head>
<title>Arguments</title>
<body>
<script>
function getcube(number){
alert(number*number*number);
}
</script>
<form>
<input type="button" value="click" onclick="getcube(4)"/>
</form>
</body>
</head>
</html>
```

**Output:**

**Function with Return Value:**

- We can call function that returns a value and use it in our program.
- A return statement is used to specify the value/result/output that is returned from a function.

**Example:**

```
<html>
<head>
<title>Return value</title>
<body>
<script>
function getInfo(){
return "hello javatpoint! How r u?";
}
</script>
<script>
document.write(getInfo());
</script>
</body>
</head>
</html>
```

**Output:**

hello javatpoint! How r u?

**2.Built-In Functions:**

- Functions that are provided by JavaScript itself as part of the scripting language, are known as built-in functions.
- JavaScript provides a rich set of the library that has a lot of built-in functions.
- Some examples of the built-in functions are : alert(), prompt(), parseInt(), eval() etc.

**Function Object:**

- In JavaScript, the purpose of Function constructor is to create a new Function object. It executes the code globally.
- However, if we call the constructor directly, a function is created dynamically but in an unsecured way.

**Syntax:**

1. new Function ([arg1[, arg2[, ....argn]],] functionBody)

**Example:**

```
<html>
<head>
<title>Objects</title>
<body>
<script>
var add=new Function("num1","num2","return num1+num2");
document.writeln(add(2,5));
</script>
</body>
</head>
</html>
```

**Output:**7

❖ **Browser Object Model(BOM):**

- BOM represents the hierarchy of entire browser along with the toolbar, menu bar, status bar, web page and many more.
- We can use the properties and methods of objects in the BOM to change the window or elements displayed in the browser.
- We do not have to create an object in BOM, they Are Created Automatically when the browser open the web page.

**Window Object:**

- The window object represents a window in browser. An object of window is created automatically by the browser.
- Window object is a global as it contain all other objects of the BOM within it.
- We use methods and attributes of window objectss to control the web browser window.

**Methods of window object:**

The important methods of window object are as follows:

**1. alert() in javascript:**

It displays alert dialog box. It has message and ok button.

**Example:**

```
<script type="text/javascript">
function msg(){
alert("Hello Alert Box");
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

**Output :**


**2.confirm() in javascript:**

It displays the confirm dialog box. It has message with ok and cancel buttons.

**Example:**

```
<script type="text/javascript">
function msg(){
var v= confirm("Are u sure?");
if(v==true){
alert("ok");
}
else{
alert("cancel");
}
}
</script>
<input type="button" value="delete record" onclick="msg()"/>
```

**Output :**


**3. prompt() in javascript:**

It displays prompt dialog box for input. It has message and textfield.

**Example:**

```
<script type="text/javascript">
function msg(){
var v= prompt("Who are you?");
alert("I am "+v);
}
</script>
```

<input type="button" value="click" onclick="msg()"/>

**Output :**


**4. open() in javascript:**

It displays the content in a new window.

**Example:**

```
<script type="text/javascript">
function msg(){
open("http://www.javatpoint.com");
}
</script>
<input type="button" value="javatpoint" onclick="msg()"/>
```

**Output :**


**5. setTimeout() in javascript:**

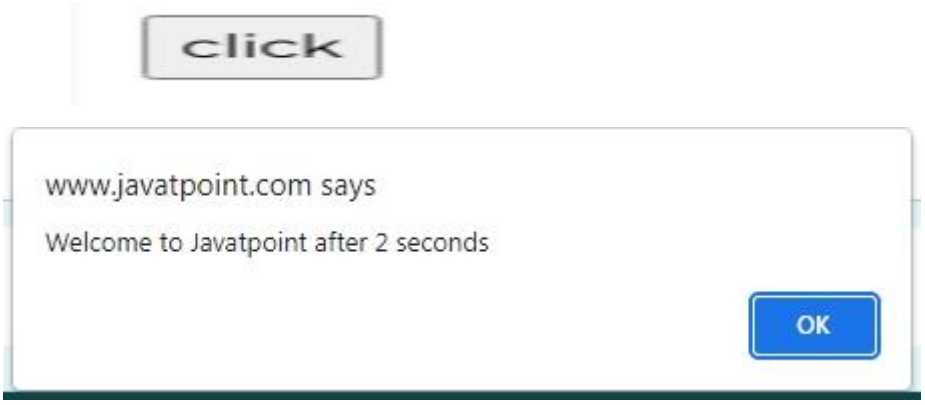
It performs its task after the given milliseconds.

**Example:**

```
<script type="text/javascript">
```

```
function msg(){
setTimeout(
function(){
alert("Welcome to Javatpoint after 2 seconds")
},2000);
}
</script>
<input type="button" value="click" onclick="msg()"/>
```

**Output :**




**Document Object:**

- The most significant part of window object is  a document object.
- It is used to represent the web page displayed in the browser window.
- All the elements of the web page like forms, images, links and others are contained within the document object.

**Form Object:**

- Document object contain the form object ,similarly the forn object contains element object.
- The element object is used to reference each element in the form.
- We can access the element object using its parental hierarchy from document object.
- Here the element object and its parent object is separated by dot operator.

**Example:** document.Form_name.textbox_name.attribute;

❖ **Verifying or validating Forms:**

- Form Validation is a way to ensure that the data users enter into a form is correct before it gets submitted.
- This helps ensure that things like emails, passwords, and other important details are entered properly, making the user experience smoother and the data more accurate.

**Types of Form Validation:**

1. **Client-side Validation:**

   This is done in the user's browser before the form is submitted.It provides quick feedback to the user, helping them fix errors without sending data to the server first.

2. **Server-side Validation:**

   Even though client-side validation is useful, it's important to check the data again on the server.This ensures that the data is correct, even if someone tries to bypass the validation in the browser.

**Steps for Form Validation:**

When we validate a form in JavaScript, we typically follow these steps:

**1.Data Retrieval:**

- The first step is to get the user's values entered into the form fields (like name, email, password, etc.).
- This is done using document.forms.RegForm, which refers to the form with the name "RegForm".

2. **Data Validation:**

- **Name Validation:** We check to make sure the name field isn't empty and doesn't contain any numbers.
- **Address Validation:** We check that the address field isn't empty.
- **Email Validation:** We make sure that the email field isn't empty and that it includes the "@" symbol.

- **Password Validation:** We ensure that the password field isn't empty and that the password is at least 6 characters long.
- **Course Selection Validation:** We check that a course has been selected from a dropdown list.

3. **Error Handling:**
    - If any of the checks fail, an alert message is shown to the user using window.alert, telling them what's wrong.
    - The form focuses on the field that needs attention, helping the user easily fix the error.
4. **Submission Control:**
    - If all the validation checks pass, the function returns true, meaning the form can be submitted.
5. **Focus Adjustment:**
    - The form automatically focuses on the first field that has an error, guiding the user to fix it.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
<title>Form Validation</title>
<link rel="stylesheet" href="style.css" />
</head>
<body>
<h1>REGISTRATION FORM</h1>
<form name="RegForm" onsubmit="return validateForm()">
```

```
<p>
<label for="name">Name:</label>
<input type="text" id="name" name="Name"
placeholder="Enter your full name" />
<span id="name-error" class="error-message"></span>
</p>
<p>
<label for="address">Address:</label>
<input type="text" id="address" name="Address"
placeholder="Enter your address" />
<span id="address-error" class="error-message"></span>
</p>
<p>
<label for="email">E-mail Address:</label>
<input type="text" id="email" name="EMail"
placeholder="Enter your email" />
<span id="email-error" class="error-message"></span>
</p>
<p>
<label for="password">Password:</label>
<input type="password" id="password" name="Password" />
<span id="password-error" class="error-message"></span>
</p>
<p>
<label for="subject">Select Your Course:</label>
<select id="subject" name="Subject">
<option value="">
Select Course
</option>
```

```
<option value="BTECH">BTECH</option>
<option value="BBA">BBA</option>
<option value="BCA">BCA</option>
<option value="B.COM">B.COM</option>
</select>
<span id="subject-error" class="error-message"></span>
</p>
<p>
<label for="comment">College Name:</label>
<textarea id="comment" name="Comment"></textarea>
</p>
<p>
<input type="checkbox" id="agree" name="Agree" />
<label for="agree">I agree to the aboveinformation</label>
<span id="agree-error" class="error-message"></span>
</p>
<p>
<input type="submit" value="Send" name="Submit" />
<input type="reset" value="Reset" name="Reset" />
</p>
</form>
</body>
</html>
```

**Output:**


❖ **HTML5:**

- It provides details of all 40+ HTML tags including audio, video, header, footer, data, data list, article etc.
- This HTML tutorial is designed for beginners and professionals.
- HTML5 is a next version of HTML. Here, you will get some brand new features which will make HTML much easier.
- These new introducing features make your website layout clearer to both website designers and users.
- There are some elements like <header>, <footer>, <nav> and <article> that define the layout of a website.

**Why use HTML5**

It is enriched with advance features which makes it easy and interactive for designer/developer and users.

- It allows you to play a video and audio file.
- It allows you to draw on a canvas.
- It facilitate you to design better forms and build web applications that work offline.

- It provides you advance features for that you would normally have to write JavaScript to do.
- The most important reason to use HTML 5 is, we believe it is not going anywhere. It will be here to serve for a long time according to W3C ( World Wide Web Consortium)recommendation.

**Example:**

```
<!DOCTYPE>
<html>
<head>
<title>Html5</title>
<body>
<h1>Write Your First Heading</h1>
<p>Write Your First Paragraph.</p>
</head>
</body>
</html>
```

**Output:**

**Write Your First Heading**

Write Your First Paragraph.

❖ **Cascading Style Sheets(css):**

Cascading Style Sheets (CSS) is used to style HTML elements on web pages. It defines how elements are displayed, including layout, colours, fonts, and other properties. CSS targets HTML elements and applies style rules to dictate their appearance.

There are three types of CSS which are given below:

1. Inline CSS
2. Internal or Embedded CSS

3. External CSS

**1. Inline CSS:**

Inline CSS is used to style the elements of HTML documents. It is used in HTML to style the attributes without using the selectors. It is challenging to manage the inline function in websites compared to other types. It is very helpful in Html in some situations.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Inline CSS</title>
<body>
<pstyle="color:#009900;
font-size:50px;
font-style:italic;
text-align:center;">
Web Programming  </p>
</body>
</head>
</html>
```

**Output:**

Web Programming

**2. Internal CSS:**

Internal CSS is used to design the style single page effectively. It is more time-consuming because we can only work on one page or we need to style each web page. In internal CSS, we style a single webpage uniquely.

**Syntax:**

```
<style>
--- required styles--
```

```
</style>
```

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Internal CSS</title>
<style>
.main {
text-align: center;
}
.WP {
color: #009900;
font-size: 50px;
font-weight: bold;
}
.JAVA {
font-style: bold;
font-size: 20px;
}
</style>
</head>
<body>
<div class="main">
<div class="WP">WEB PROGRAMMING</div>
<div class="JAVA">
Artificial Intelligence and Data Science
</div>
</div>
</body>
```

```
</html>
```

**Output:**


**3. External CSS:**

External CSS is used to link all webpage with an external file. CSS, which can be created in a text file. It is more efficient for styling an extensive webpage. It also increases the readability of the CSS files.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>External CSS</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
<div class="main">
<div class="pp">Python Programming</div>
<div id="python">
A computer science portal for geeks
</div>
</div>
</body>
</html>
```

**Output:**

Python Programming

A computer science portal for geeks

**Advantages of CSS:**

- Web designers needs to use few lines of programming for every page improving site speed.
- It is less complex therefore the effort are significantly reduced.
- CSS changes are device friendly. With people employing a batch of various range of smart devices to access websites over the web, there's a requirement for responsive web design.
- Easy for the user to customize the online page
- It reduces the file transfer size.

**Disadvantages of CSS:**

- The programming language world is complicated for non-developers and beginners. Different levels of CSS i.e. CSS, CSS 2, CSS 3 are often quite confusing.
- Browser compatibility (some styles sheet are supported and some are not).
- CSS works differently on different browsers. IE and Opera supports CSS as different logic.
- There might be cross-browser issues while using CSS.
- There are multiple levels which creates confusion for non-developers and beginners.

❖ **HTML5 canvas:**

- The HTML <canvas> element is used to draw graphics on a web page.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas is supported by all major browsers.
- The graphic to the left is created with <canvas>.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

- It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Canvas</title>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #000000;">
Your browser does not support the HTML canvas tag.
</canvas>
</body>
</html>
```

**output:**

**Add a JavaScript:**

After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

**Draw a Line:**

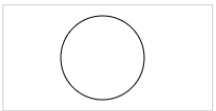
**Example:**

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>
<body>Line</title>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
</head>
</body>
</html>
```

**Output:**

**Draw a Circle:**

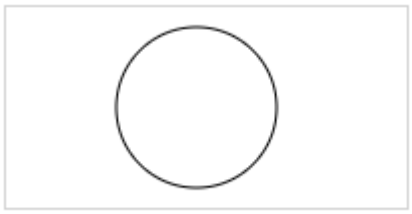


**Example:**

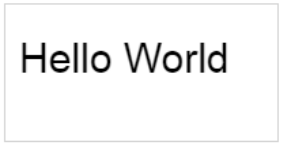
```
<!DOCTYPE html>
<html>
<head>
<title>Circle</title>
```

```
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>
</head>
</body>
</html>
```

**Output:**



**Draw a Text:**

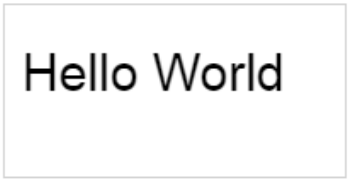



**Example:**

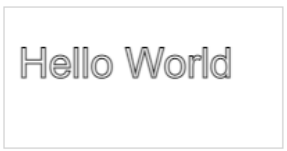
```
<!DOCTYPE html>
<html>
<head>
<title>Text</title>
```

```
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>
</head>
</body>
</html>
```

**Output:**




**Stroke Text:**




**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Stroke Text</title>
<body>
```

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",10,50);
</script>
</head>
</body>
</html>
```

**Output:**


**Draw Linear Gradient:**

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Linear Gradient</title>
<body>
```

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var grd = ctx.createLinearGradient(0,0,200,0);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>
</head>
</body>
</html>
```

**Output:**

**Draw Circular Gradient:**

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Circular Gradient</title>
<body>
```

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var grd = ctx.createRadialGradient(75,50,5,90,60,100);
grd.addColorStop(0,"red");
grd.addColorStop(1,"white");
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>
</head>
</body>
</html>
```

**Output:**

**Draw Image:**

**Example:**

```
<!DOCTYPE html>
<html>
<head>
<title>Image</title>
<body>
<p>Image to use:</p>
<img id="scream" src="img_the_scream.jpg" alt="The Scream" width="220" height="277">
```

```
<p>Canvas to fill:</p>
<canvas id="myCanvas" width="250" height="300"
style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>
<p><button onclick="myCanvas()">Try it</button></p>
<script>
function myCanvas() {
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img,10,10);
}
</script>
</head>
</body>
</html>
```

**Output:**

Image to use:

Canvas to fill:

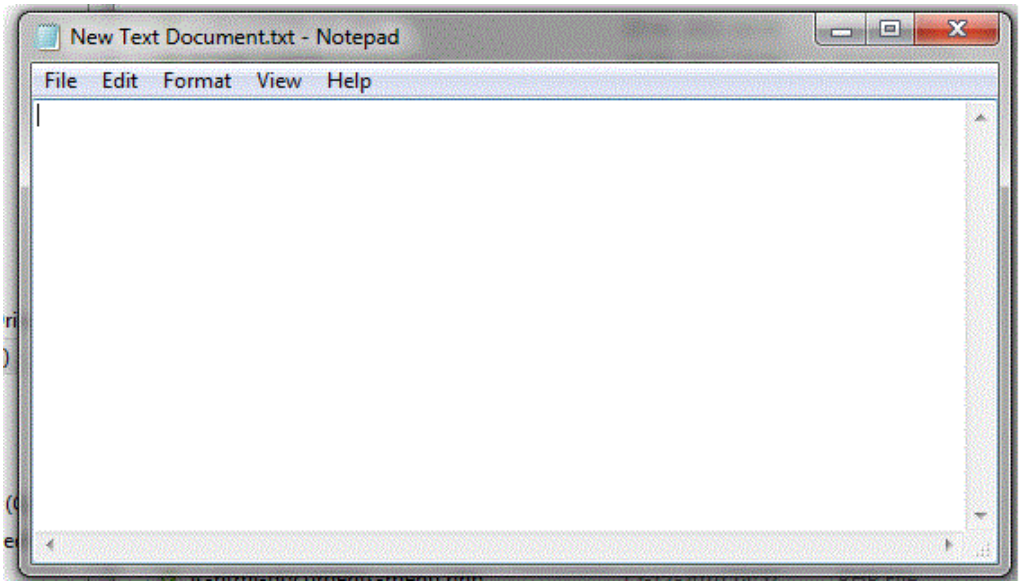
Try it

- **Web Site Creation Using Tools:**

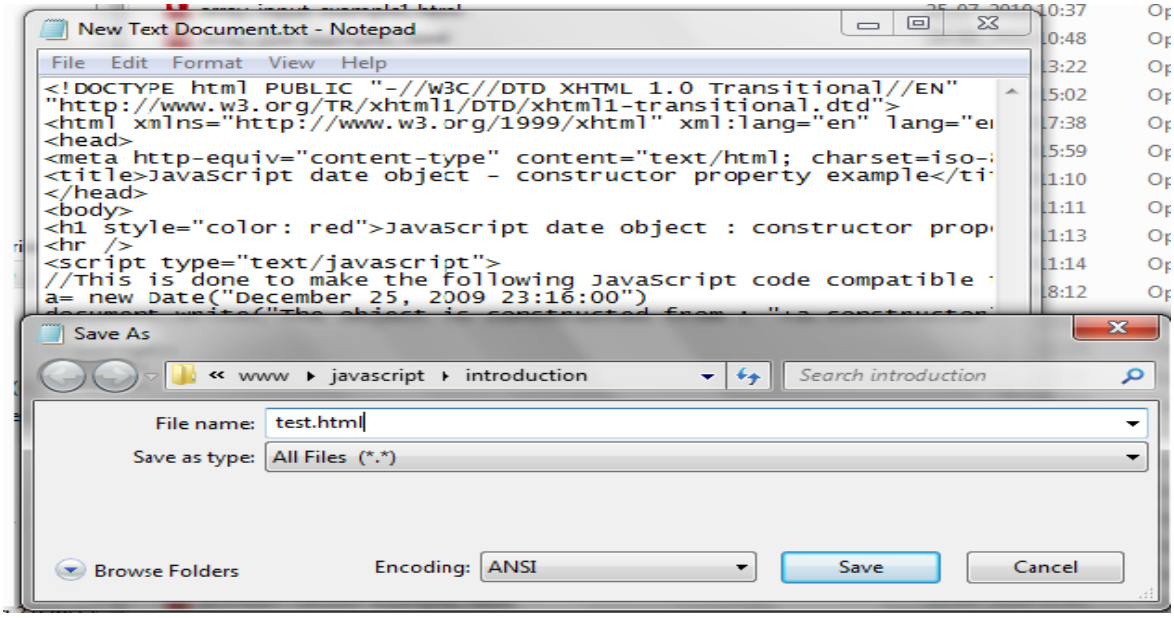
**The System Tools:**

- To create a JavaScript code for web applications you need a simple text editor. F
- or windows environment, you can use Windows Notepad and for Mac OS X, the Text Edit application is usable.
- There are lots of good tools in the market for developing a web-based application like Macromedia's Dreamweaver.
- Initially, it will be best to write the code by hand in a simple text editor rather than use advanced tools.
- An important factor to remember that Windows Notepad saves files with an extension of .txt, if you do not mention another extension.
- Therefore at the time of saving the HTML document files, we must mention .html extension.

**Open a file:**




**Save a file:**




**Select a Browser:**

To view your web pages you need a browser like firefox, opera, internet explorer etc.

**Enable JavaScript in various Browser**

**Internet Explorer (8.0)**

1. Select 'Tools' from the top menu.
2. Choose 'Internet Options'.
3. Click on the 'Security' tab.
4. Click on 'Custom Level'.
5. Scroll down until you see a section labelled 'Scripting'.
6. Under 'Active Scripting', select 'Enable' and click OK.

**Mozilla Firefox (3.6.13)**

1. Select 'Tools' from the top menu.
2. Choose 'Options'.
3. Choose 'Web Features' from the left navigation.
4. Select the checkbox next to 'Enable JavaScript' and click OK.

**Apple Safari (1.0)**

1. Select 'Safari' from the top menu.
2. Choose 'Preferences'.
3. Choose 'Security'.
4. Select the checkbox next to 'Enable JavaScript'.

**Opera version 9**

1. On the Tools menu, click Preferences.
2. On the Advanced tab, click Content.
3. Click to select the Enable JavaScript check box, and then click OK.
4. Click the Back button to return to the previous page, and then click the Reload button to run scripts.

**Writing first JavaScript:**

We have already learned the JavaScript syntax, the tools to create scripts and how to enable JavaScript in the browser. Lets create the first JavaScript. Here is an example and explanation.

**HTML Code:**

```
<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>JavaScript  First Example</title>

</head>

<body>

<script type="text/javascript">

document.write("First JavaScript");

</script>

</body>

</html>
```

**Explanation:**

The first nine and last three lines are HTML codes.