UNIT IV: JavaScript

4.1 Client side scripting with JavaScript
4.2 variables
4.3 functions
4.4 conditions, loops and repetition
4.5 Pop up boxes,
4.6 Advance JavaScript: Javascript and objects
4.7 JavaScript own objects
4.8 The DOM and web
4.9 browser environments
4.10 Manipulation using DOM
4.11 forms and validations
4.12 DHTML: Combining HTML, CSS and Javascript
4.13 Events and buttons

4.1 Client side scripting with JavaScript

Client-side scripting is when the server sends the code along with the HTML web page to the client. The script is referred to by the code.

Client-side scripting is mostly used for dynamic user interface components including pull-down menus, navigation tools, animation buttons, and data validation.

It is currently quickly expanding and evolving on a daily basis.

Client-side scripts may also include instructions for the web browser to follow in response to user activities like clicking a page button.

JavaScript is allows developers to wrap HTML and CSS code in it to make web apps interactive.

It enables the interaction of users with the web application. It is also used for making animations on websites

JavaScript is the most commonly used client-side scripting or programming language. It is written in the ECMAScript programming language.

JavaScript is a dynamically typed (also known as weakly typed) object-oriented scripting language. It uses an integrated interpreter to run directly in the browser.

Weakly typed indicates that the variables can be implicitly transformed from one data type to another.

The Benefits of Client-Side Scripting

Client-side scripting is a simple language to learn and utilize.
Client-side scripting has the advantage of being lightweight and reasonably simple to implement
Data validation on the client side can be accomplished using a client-side scripting language such as JavaScript.
Client-side scripting can also be used for mathematical assessment.
Script code that is only performed by the browser and not by the server.
Executing script code takes far too little time.
When a user taps a key, moves the mouse, or clicks, the browser responds promptly.

4.2 variables

A JavaScript variable is simply a name of storage location.

There are two types of variables in JavaScript:

1. Local variable

A JavaScript local variable is declared inside block or function. It is accessible within the function or block only.

2. Global variable.

A JavaScript global variable is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable.

There are some **rules** while declaring a JavaScript variable

1. **Name must start with a letter** (a to z or A to Z), underscore (_), or dollar ($) sign.

2. After **first letter we can use digits** (0 to 9), for example value1.

3. JavaScript variables are **case sensitive**, for example x and X are different variables.

```
var x = 10;
var _name="gopal";  //Correct

var  123=30;
var *aa=320;  //Incorrect
```

Example

```
<script>
var data=200;    //gloabal variable
function a(){
var data=100;
document.writeln(data);
}

a(); //calling JavaScript function

</script>
```

4.3 functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

Code reusability: We can call a function several times so it save coding.

Less coding: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

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

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

JavaScript Function Arguments

We can call function by passing arguments. Let's see the example of function that has one argument.

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

Function with Return Value

We can call function that returns a value and use it in our program. Let's see the example of function that returns value.

4.4 conditions, loops and repetition

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

1. If statement

It evaluates the content only if expression is true.

Syntax

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

Example

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

2. If...else Statement

It evaluates the content whether condition is true of false.

Syntax

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

Example

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

3. If...else if statement

It evaluates the content 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

```
<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>
```

4. Switch

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement

But it is convenient than if..else..if because it can be used with numbers, characters etc.

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

```
<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>
```

B] loops
The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.

There are four types of loops in JavaScript.

for loop
The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known.

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

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

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

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

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

do-while loop

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

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

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

4.5 Pop up boxes,

JavaScript has three kind of popup boxes:

1. Alert box
2. Confirm box
3. Prompt box.

1 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");
```

Example

```
alert("I am an alert box!");
```

2 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");
```

Example

```
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
```

3 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","defaultText");
```

Example

```
let person = prompt("Please enter your name", "Harry Potter");
let text;
if (person == null || person == "") {
  text = "User cancelled the prompt.";
} else {
  text = "Hello " + person + "! How are you today?";
}
```

C] repetition

The repeat() method returns a string with a number of copies of a string.

The repeat() method returns a new string.

The repeat() method does not change the original string.

Syntax

string.repeat(count)

Parameters

Count - Required. The number of copies.

Return Value
A string - The copies of the original string.

Example

```
<body>

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

<script>
let text = "Hello world!";
let result = text.repeat(2);

document.getElementById("demo").innerHTML = result;
</script>
</body>
```

4.6 Advance JavaScript: Javascript and objects

A javaScript object is an entity having state and behavior

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.

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.

By object literal

The syntax of creating object using object literal

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

```
<script>
emp={id:102,name:"Shyam Kumar",salary:40000}
document.write(emp.id+" "+emp.name+" "+emp.salary);
</script>
```

By creating instance of Object directly (using new keyword)

The syntax of creating object directly

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

new keyword is used to create object.

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

By using an object constructor (using new keyword)

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.

The example of creating object by object constructor

```
<script>
function emp(id,name,salary){
this.id=id;
this.name=name;
this.salary=salary;
}
e=new emp(103,"Vimal Jaiswal",30000);

document.write(e.id+" "+e.name+" "+e.salary);
</script>
```

4.7 JavaScript own objects

an object is a collection of properties, where each property is a key-value pair.

This example creates a new object called person

```
const person = {
   firstName: 'John',
   lastName: 'Doe'
};
```

The person object has two properties: firstName and lastName.

a property of an object can be either own or inherited.

A property that is defined directly on an object is own while a property that the object receives from its prototype is inherited.

The following creates an object called employee that inherits from the person object:

```
const employee = Object.create(person, {
   job: {
      value: 'JS Developer',
      enumerable: true
   }
});
```

The employee object has its own property job, and inherits firstName and lastName properties from its prototype person.

The hasOwnProperty() method returns true if a property is own.

```
console.log(employee.hasOwnProperty('job')); // => true
console.log(employee.hasOwnProperty('firstName')); // => false
console.log(employee.hasOwnProperty('lastName')); // => false
console.log(employee.hasOwnProperty('ssn')); // => false
```

A property that is directly defined on an object is an own property.

The obj.hasOwnProperty() method determines whether or not a property is own.

4.8 The DOM and web

The document object represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the root element that represents the html document.

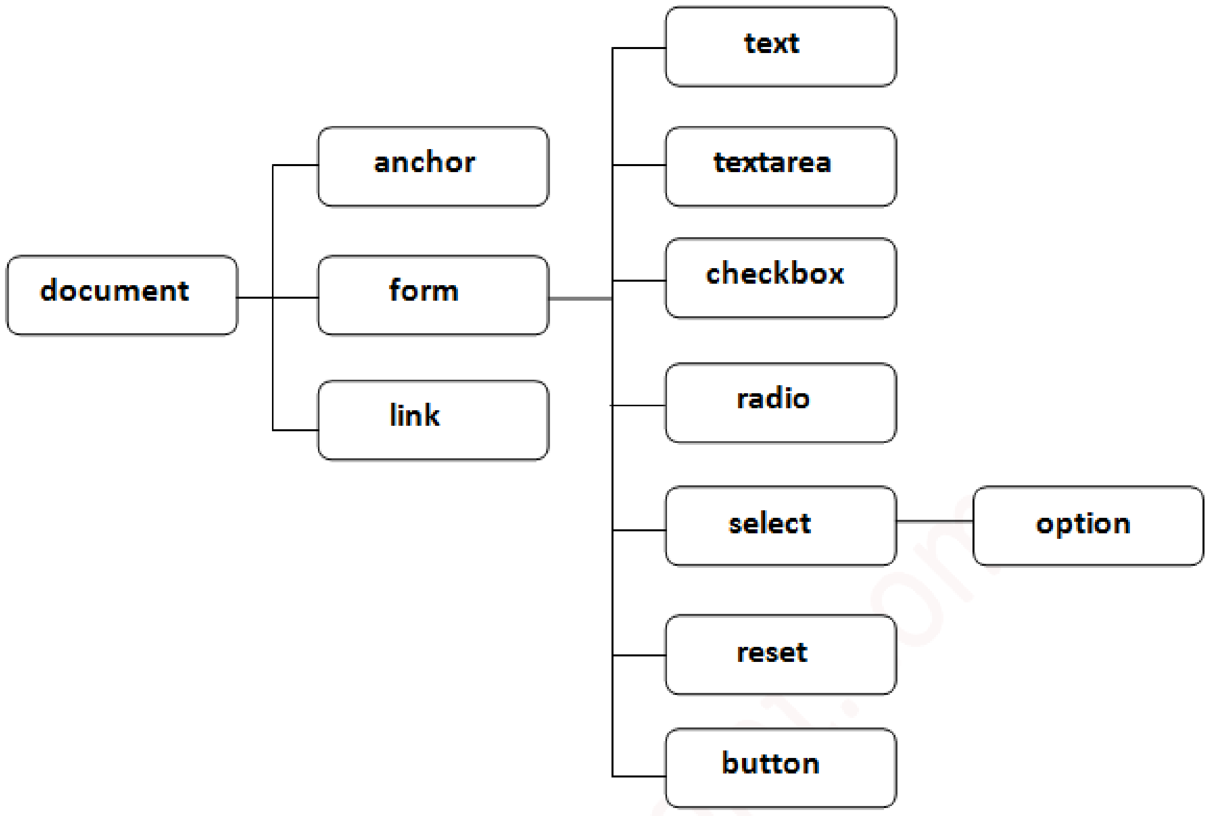
It has properties and methods. By the help of document object, we can add dynamic content to our web page.

it is the object of window

window.document Is same as **document**

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

Properties of document object



Methods of document object

We can access and change the contents of document by its methods.

write("string") - writes the given string on the doucment.

writeln("string")- writes the given string on the doucment with newline character at the end.

getElementById() - returns the element having the given id value.

getElementsByName() - returns all the elements having the given name value.

getElementsByClassName() - returns all the elements having the given class name.

Accessing field value by document object

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

4.9 browser environments

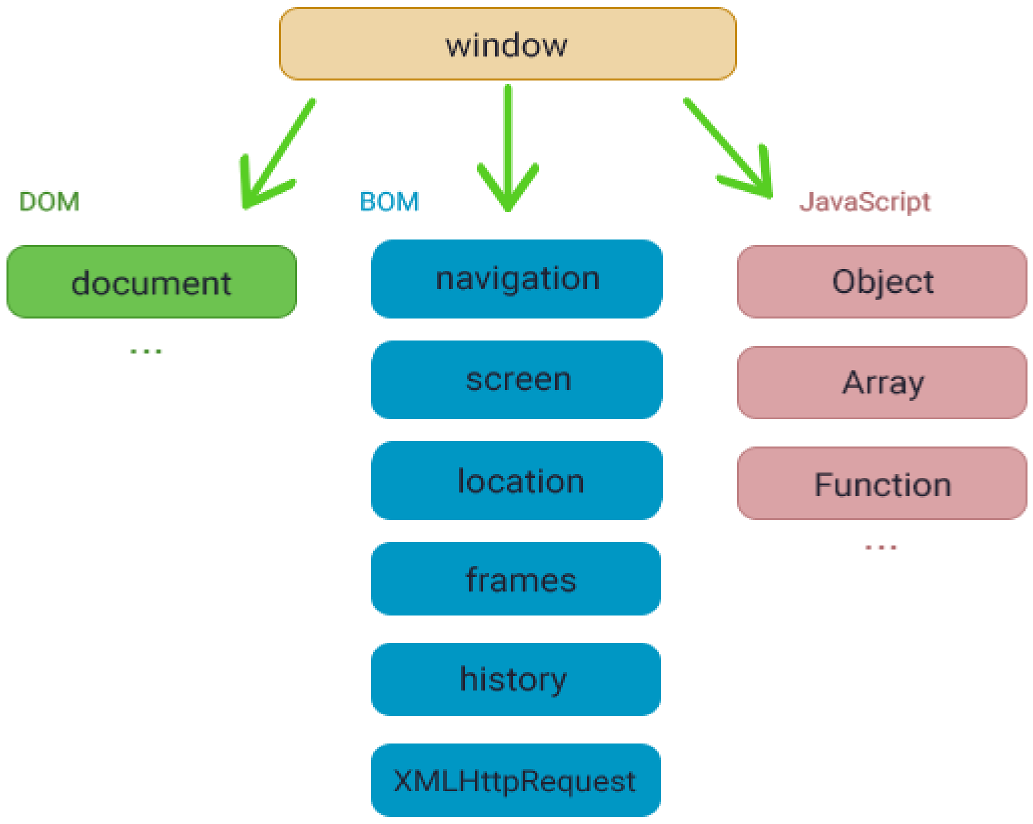
JavaScript was created for web browsers. But, since then, it has expanded and become a language with different platforms and uses.

A platform can be a browser, a web-server, or another host. Each of them includes a platform-specific functionality.

For JavaScript specification, it is a host environment.

A host environment has its own objects and functions in addition to the language core.

Web browsers allow controlling web pages.



The "root" object named window has two roles

It is considered a global object for JavaScript code.

It presents the "browser window", providing methods to control it.

```
function welcome() {
  console.log("Welcome to W3Docs");
}
```

```
// global functions are global object methods:
window.welcome();
```

Document Object Model (DOM)

Document Object Model (DOM) is targeted at representing the overall page content as objects that can be transformed.

The document object is considered the primary “entry point” to the page. It allows you to change or create anything on the page.

```
// change the background color to green
document.body.style.background = "green";
// change it back after 2 second
setTimeout(() => document.body.style.background = "", 2000);
```

The DOM specification explains a document structure, providing objects to manipulate it. Some non-browser instruments also use DOM.

Browser Object Model (BOM)

The Browser Object Model (BOM) represents extra objects provided by the host environment to work with anything excluding the document.

```
alert(location.href); // shows current URL
if (confirm("Go to Google?")) {
  location.href = "https://www.google.com/"; // redirect the browser to another
URL
}
```

The functions such as alert/confirm/prompt are also a part of BOM.

4.10 Manipulation using DOM

each Element object in the DOM has properties and methods that you can use to interact with that element.

The following are the most common and practical ways you might want to manipulate the Element object:

Change the Content of an Element

You can change the value or content of an element by setting the innerText property of that element.

```
<p class="myParagraph">This is a paragraph</p>
```

Next, you select the element and change its innerText value

```
const p = document.querySelector('.myParagraph');

p.innerText = 'A new day is dawning';
```

Manipulate the Class Attribute

You can add a new class attribute to an Element by using the add() method of the classList object:

```
Element.classList.add('myClass');
```

You can remove a class using the remove() method:

```
Element.classList.remove('myClass');
```

Setting CSS Styles Using JavaScript

you can control the style of an element by adding or removing classes that change the style rules applied to an element.

```
.color-primary {
  color: #007bff;
}

.color-secondary {
  color: #6c757d;
}

.bold {
  font-weight: 700;
}
```

If you have an element with the color-primary class applied, you can replace it with color-secondary class, or add the bold class.

```
<p class="myParagraph">A new day is dawning</p>

const p = document.querySelector('.myParagraph');

// add a class to the element
p.classList.add('color-primary');

// replace a class
p.classList.replace('color-primary', 'color-secondary');

// remove a class
p.classList.remove('color-secondary');
```

or

```
const p = document.querySelector('.myParagraph');

p.style.fontWeight = '700'; // set font weight
p.style.textTransform = 'uppercase'; // set to uppercase
p.style.color = '#007bff'; // set color
```

Create, Add, and Remove Elements

Besides creating a DOM tree out of your HTML file, you also have the ability to create DOM elements programmatically using JavaScript.

```
const p = document.createElement('p');
p.innerText = 'This paragraph is created using JavaScript';
const body = document.querySelector('body');

body.append(p);
```

Insert Element at a Specific Position

The append() method that we explored above will insert a new element as the last child of the parent element.

```
let p2 = document.createElement('p');

p2.innerText = 'The second paragraph';

let body = document.querySelector('body');
let p1 = document.querySelector('#first');

body.insertBefore(p2, p1);
```

Manipulating Element Attributes

The classList object only provide methods to change the class of an element. If you want to change other attributes like id, href, or src, you can use the setAttribute() method.

The setAttribute() method accepts two arguments:

The name of the attribute to set
The value of the attribute to set

```
<img id="profile-pic" src="feature-image.png" />

const img = document.querySelector('#profile-pic');

img.setAttribute('src', 'new-image.jpg');
```

Manipulating Data Attributes

The data attribute is used to store extra information on HTML elements. How you use the data is up to you.

```
let myDiv = document.querySelector('#intro');

// Access the dataset property
console.log(myDiv.dataset.session) // 2022

// Use camelCase when your data attribute is more than one word
console.log(myDiv.dataset.attributeTheme) // light
```

4.11 forms and validations

The code gets the values of various form fields (name, email, what, password, address) using Form.

Data Validation:
Name Validation: Checks if the name field is empty or contains any digits.
Address Validation: Checks if the address field is empty.
Email Validation: Checks if the email field is empty or lacks the '@' symbol.
Password Validation: Checks if the password field is empty or less than 6 characters long.
Selection Validation: Checks if a is selected.
Error Handling:
Displays alerts if any of the fields fail validation criteria.
Sets focus back to the respective field.

eturns true if all validation checks pass, indicating that the form can be submitted. Otherwise, it returns false, preventing form submission.

Sets focus to the first field that failed validation, ensuring the user's attention is drawn to the problematic field.

Ex.

```
  if (address === "") {
     window.alert
        ("Please enter your address.");
     address.focus();
     return false;
  }

  if (email === "" || !email.includes('@')) {
     window.alert
        ("Please enter a valid e-mail address.");
     email.focus();
     return false;
  }

  if (password.length < 6) {
     alert ("Password should be atleast 6 character long");
     password.focus();
     return false;

  }
```

4.12 DHTML: Combining HTML, CSS and Javascript

DHTML, or Dynamic HTML, is a technology that differs from traditional HTML.

DHTML combines HTML, CSS, JavaScript, and the Document Object Model (DOM) to create dynamic content.

It uses the Dynamic Object Model to modify settings, properties, and methods.

It is supported by some versions of Netscape Navigator and Internet Explorer 4.0 and higher.

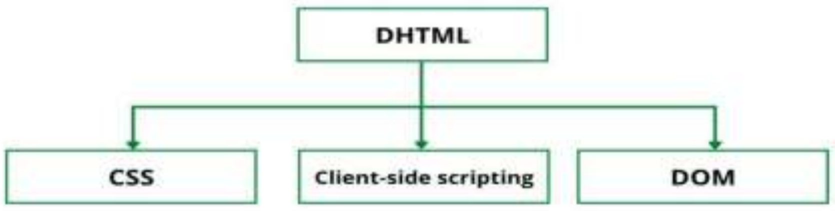
HTML: HTML stands for Hypertext Markup Language and it is a client-side markup language. It is used to build the block of web pages.

Javascript: It is a Client-side Scripting language. Javascript is supported by most of browsers, and also has cookie collection to determine the user’s needs.

CSS: The abbreviation of CSS is Cascading Style Sheet. It helps in the styling of the web pages and helps in designing of the pages. The CSS rules for DHTML will be modified at different levels using JS with event handlers which adds a significant amount of dynamism with very little code.

DOM: It is known as a Document Object Model which acts as the weakest link in it. The only defect in it is that most of the browsers does not support DOM. It is a way to manipulate the static content.

DHTML is not a technology; rather, it is the combination of three different technologies, client-side scripting (JavaScript or VBScript), cascading style sheets and document object model.



Advantages:
Size of the files are compact in compared to other interactional media like Flash or Shockwave, and it downloads faster.

It is supported by big browser manufacturers like Microsoft and Netscape.

Highly flexible and easy to make changes.

Disadvantages:
It is not supported by all the browsers. It is supported only by recent browsers such as Netscape 6, IE 5.5, and Opera 5 like browsers.

Implementation of different browsers are different. So if it worked in one browser, it might not necessarily work the same way in another browser.

4.13 Events and buttons

An event is defined as changing the occurrence of an object.

It is compulsory to add the events in the DHTML page. Without events, there will be no dynamic content on the HTML page.

1 onchange - It occurs when the user changes or updates the value of an object.

2 onclick - It occurs or triggers when any user clicks on an HTML element.

3 ondblclick - when user clicks on an HTML element two times together.

4 onload- It occurs when an object is completely loaded.

5 onkeydown - It triggers when a user is pressing a key on a keyboard device.

6 onkeypress - It triggers when the users press a key on a keyboard.

7 onmouseover - It occurs when a user moves the cursor over an HTML object.

8 onsubmit - It is triggered when user clicks a button after submission of a form.

Ex.
```
<script type="text/javascript">
function ChangeText(ctext)
{
```

```
ctext.innerHTML=" Hi Gopal! ";
}
</script>

<body>
<font color="red"> Click on the Given text for changing it: <br>
</font>
<font color="blue">
<h1 onclick="ChangeText(this)"> Hello Gopal! </h1>
</font>
</body>
```

buttons

In dhtml we can create button using <button> Tag.

```
<body>
  <p id="demo"> This text changes color when click on the following different
buttons. </p>
  <button onclick="change_Color('green');"> Green </button>
  <button onclick="change_Color('blue');"> Blue </button>
<script type="text/javascript">

function change_Color(newColor) {
  var element = document.getElementById('demo').style.color = newColor;
}
</script>
```

The End