JavaScript

- JavaScript is an object-based scripting language. It is light weighted. Using HTML we can only design a web page
- It is first implemented by Netscape (with help from Sun Microsystems)
- JavaScript was created by **Brendan Eich** at Netscape in **1995** for the purpose of allowing code in web-pages (performing logical operation on client side).

JavaScript features are as follows

1. Client-Side Technology
2. Platform Independent
3. Validating User’s Input
4. Control Statement
5. Handling Dates and Time
6. Generating HTML on the fly.
7. Detecting the User’s Browser and OS
8. Performing simple computations on the client side.

Client-Side Technology:

- JavaScript is very useful while using forms. It has the capability to validate user input for errors and also saves time. If the user leaves a required field empty or the information is incorrect, JavaScript checks for them before sending the data over to the server.

Platform Independent

- Since browsers interpret JavaScript, it solves the problem of compilation and compatibility.
- Thus it can run on Windows, Macintosh, and other Netscape-supported systems.
- Also, it is possible to embed them in any other script like HTML that keeps JavaScript into use.

Validating User's Input

- JavaScript is very useful while using forms. It has the capability to validate user input for errors and also saves time. If the user leaves a required field empty or the information is incorrect, JavaScript checks for them before sending the data over to the server.

Control Statement

- JavaScript provides greater control to the browser rather than being completely dependent on the web servers.
- JavaScript provides various browsers with additional functionalities that help reduce server load and network traffic.

Handling Dates and Time

- JavaScript has built-in functions to determine the date and time. Thus it is very easy to code only by using methods
- **Like:** .getDate()

Generating HTML on the fly

- JavaScript has very handy features to dynamically generate HTML content for the web.
- It allows us to add text, links, images, tables, etc after an event occurrence

Example: - Mouse Click

Detecting the User's Browser and OS

- JavaScript is very capable in the detection of the user's browser and OS information.
- Though JavaScript runs on every platform, there may occur a situation where we need the user's browser before processing.
- This can be helpful for writing code that result in different outputs in different browsers

Performing simple computations on the client side:

- it can perform basic calculations on the browser. The browser does not need to ask server time for every task.
- This is especially helpful when a user needs to perform these calculations repeatedly.
- In these cases, connecting to the server would take a lot more time than performing the actual calculations.

Uses of JavaScript are:

- Client-side validation
- Dynamic drop-down menus
- Displaying date and time
- Validate user input in an HTML form before sending the data to a server.
- Build forms that respond to user input without accessing a server.
- Change the appearance of HTML documents and dynamically write HTML into separate Windows.
- Open and close new windows or frames.
- Manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser window.
- Build small but complete client side programs.
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

JavaScript have some limitations which are given below:

- Client-side JavaScript does not allow the reading or writing of files.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.

There are three places to put the JavaScript code.

1. Inline JavaScript
2. Internal JavaScript
3. External JavaScript

Inline JavaScript (Between the <body> </body> tag of html)

- When java script was written within the html element using attributes related to events of the element then it is called as inline java script.

Example of Inline JavaScript:

```
<html>
<form>
<input type="button" value="Click" onclick="alert('Button Clicked')"/>
</form>
</html>
```

Output:

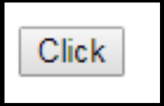
Internal JavaScript

- When java script was written within the section using element then it is called as internal java script.

Example of Internal JavaScript

```
<html>
<head>
<script>
function msg()
{
alert("Welcome in JavaScript");
}
</script>
</head>
<form>
<input type="button" value="Click" onclick="msg()"/>
</form>
</html>
```

Output:



External JavaScript

- Writing java script in a separate file with extension .js is called as external java script. For adding the reference of an external java script file to your html page, use tag with src attribute as follows

Syntax of External JavaScript:

```
<script type="text/javascript" src="filename.js"/>
```

Save file name: message.js

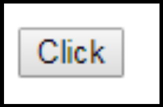
```
<script>
        function msg()
        {
                alert("Welcome in MU");
        }
<script>
```

Save another file name: **sawan.html**

```
<html>
<head>
<script type="text/javascript" src="message.js"></script>
</head>
<body>

<form>
<input type="button" value="click" onclick="msg()"/>
</form>
</body>
</html>
```

Output:



Variable IN JavaScript

- Java script did not provide any data types for declaring variables and a variable in java script can store any type of value. Hence java script is loosely typed language. We can use a variable directly without declaring it.
- Only var keywords are use before variable name to declare any variable.

Syntax of Variable:

```
var x;
```

Example of variable:

```
<html>
<Script>
var a=10;
var b=20;
var c=a+b;
document.write(c);
</script>
</html>
```

Output: 30

Rules to declaring a variable in JS:

- Name must start with a letter (a to z or A to Z), underscore (_), or dollar ($) sign.
- After first letter we can use digits (0 to 9), for example value1.
- JavaScript variables are case sensitive, for example x and X are different variables.

Example of Variable:

- var x = 10; // Valid
- var 123=20; // Invalid
- var #a=30; // Invalid
- var *b=40; // Invalid
- var _value="Sawan"; // Valid

Types of variable in JavaScript are:

1. Local Variable
2. Global Variable

Global Variable:

- A 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.

For example:

```
<script>
        var value=10;//global variable
        function a()
        {
        alert(value);
        }
        function b()
        {
        alert(value);
        }
</script>
```

Window Object

- The **window object** represents a window in browser. An object of window is created automatically by the browser.
- Window is the object of browser, it is not the object of JavaScript. The javascript objects are string, array, date etc.
- It is used to display the popup dialog box such as alert dialog box, confirm dialog box, input dialog box etc.

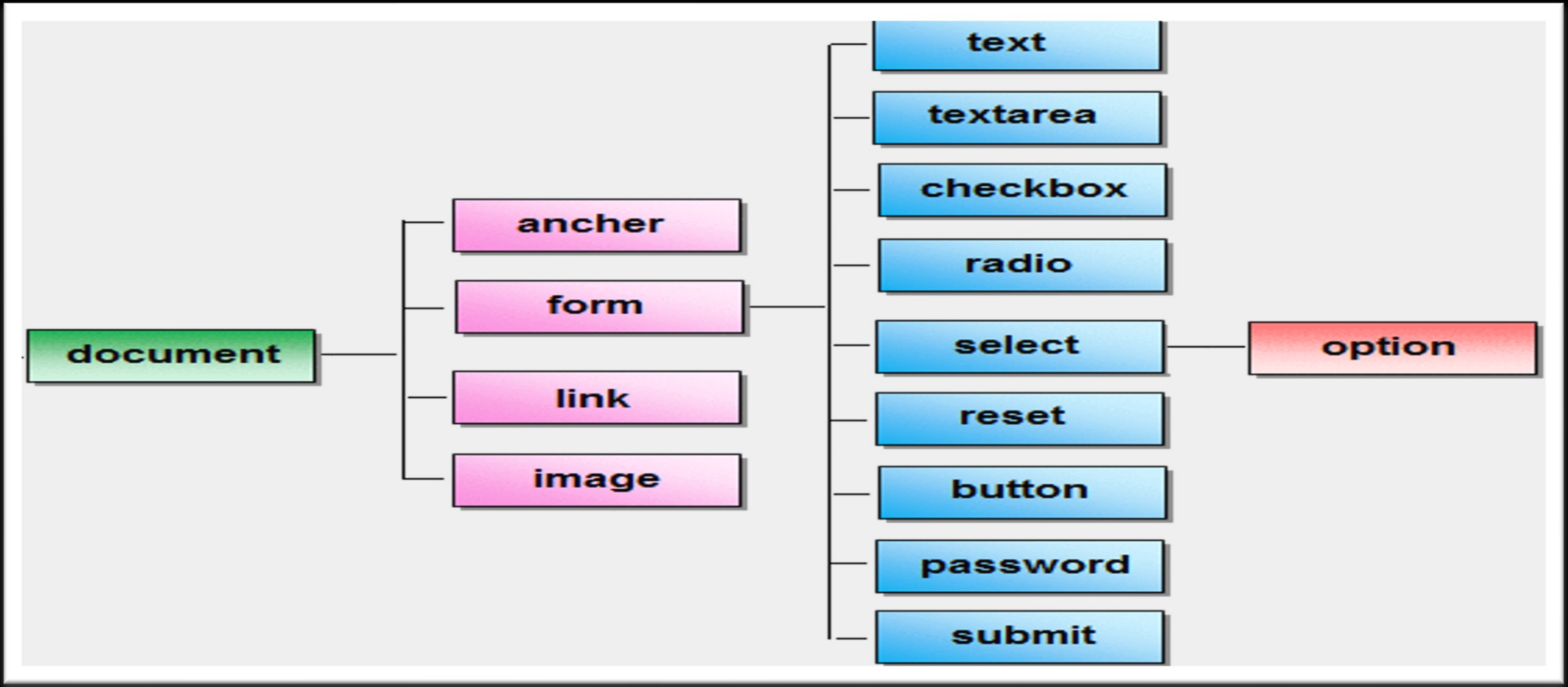
- The window methods are mainly for opening and closing new windows. The following are the main window methods. They are:

Methods of window object

- alert(): displays the alert box containing message with ok button.
- confirm(): Displays the confirm dialog box containing message with ok and cancel button.
- prompt(): Displays a dialog box to get input from the user.
- open(): Opens the new window.
- close(): Closes the current window
- setTimeout():Performs action after specified time like calling function, evaluating expressions etc.

Document Object

- 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.



Methods of document object

- write("string"): writes the given string on the document.
- writeln("string"): Same as write(), but adds a newline character after each statement.
- getElementById(): returns the element having the given id value.
- getElementsByName(): returns all the elements having the given name value.
- getElementsByTagName(): returns all the elements having the given tag name.
- getElementsByClassName(): returns all the elements having the given class name.

Event in JavaScript

- All objects have properties and methods. In addition, some objects also have "events". Events are things that happen, usually user actions, that are associated with an object.
- The "event handler" is a command that is used to specify actions in response to an event. Below are some of the most common events:

Event in JavaScript

- onload: Occurs when a page loads in a browser
- onUnload: occurs just before the user exits a page
- onMouseOver: occurs when you point to an object
- onMouseOut: occurs when you point away from an object
- onSubmit: occurs when you submit a form
- onClick: occurs when an object is clicked

Events and Objects

- Events are things that happen, actions, that are associated with an object.

  Below are some common events and the object they are associaated with:

Events	Object
onload	Body
onUnload	Body
onMouseOver	Link, Button
onMouseOut	Link, Button
onSubmit	Form
onClick	Button, Checkbox, Submit, Reset, Link

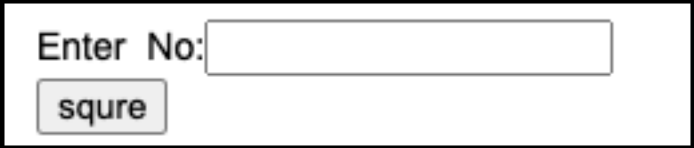
getElementById

- The **document.getElementById()** method returns the element of specified id.
- In below example we receive input from input field by using document.getElementById() method, here document.getElementById() receive input value on the basis on input field id.
- In below example "number" is the id of input field.

Example

```
<html>
<body>
<head>
        <script type="text/javascript">
                 function squre()
                {
                var num=document.getElementById("number").value;
                alert(num*num);
                }
        </script>
 </head>
                 <form> Enter No:<input type="text" id="number"name="number"/><br/>
                 <input type="button" value="squre" onclick="squre()"/>
 </form>
</body>
 </html>
```

Result



Code Explanation

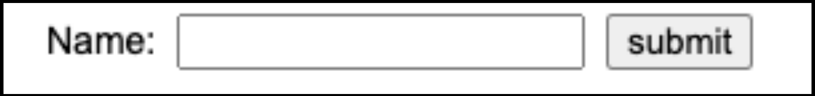
- **var num=document.getElementById("number").value;**
- In this code getElementById received value from input field which have id number.

JavaScript Forms Validation

- Forms validation is important things other wise user input wrong data, so we need to validate given data by user before submit it on server.
- The JavaScript provides the facility to validate the form on the client side so processing will be fast than server-side validation. So, most of the web developers prefer client side form validation using JavaScript.
- Here i will show you how to validate any input field like user name can not be blank. In below example input are can not be blank.

Example

```
<html>
<head>
<script> function form_validation()
{
var name=document.myform.name.value; //var x =
document.forms["myform"]["name"].value; if (name==null || name=="")
{
alert("Name can't be blank"); return false; }
}
</script>
</head>
 <body>
 <form name="myform" method="post" action="register.php" onsubmit="return
form_validation()" >
Name: <input type="text" name="name"> <input type="submit" value="submit">
</form>
 </body>
 </html>
```



Email validation

- We can validate the email with the help of JavaScript. Here we check the condition related to any email id, like email if must have "@" and "." sign and also email id must be at least 10 character.

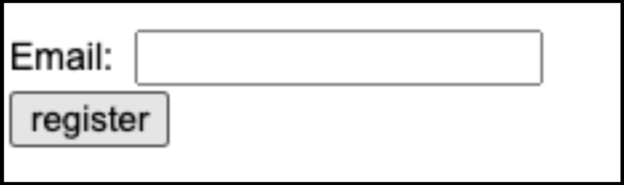
There are many criteria that need to be follow to validate the email id such as:

- email id must contain the @ and . character
- There must be at least one character before and after the @.
- There must be at least two characters after . (dot).

Example

```
<html>
<head>
<script>
        function email_validation()
        {
                var x=document.myform.email.value;
                var atposition=x.indexOf("@");
                var dotposition=x.lastIndexOf(".");
                if (atposition<1 || dotposition<atposition+2 || dotposition+2>=x.length)
                {
                alert("Please enter a valid e-mail address \n atpostion:"+atposition+"\n
                dotposition:"+dotposition);
                return false;
        }
         }
 </script>
```

```
</script>
</head>
<body>
<form name="myform" method="post" action="#" onsubmit="return
email_validation();">
Email: <input type="text" name="email"><br/>
<input type="submit" value="register">
</form>
</body>
 </html>
```

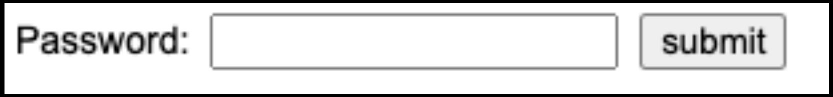


JavaScript Password Validation

- Here i will show you how to validate any password field like password field can not be blank and length of password is minimum 8 character.

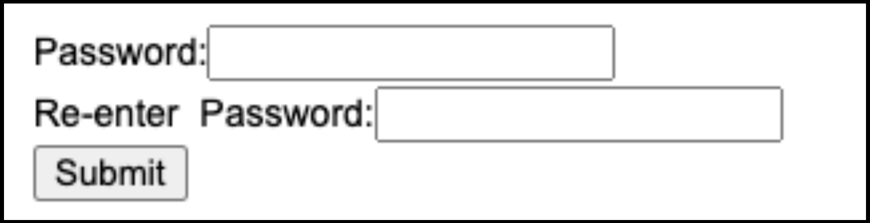
Example

```
<html>
 <head>
 <script>
        function pass_validation()
         {
        var password=document.myform.password.value; if (password==null ||
        password=="")
        {
                alert("password can't be blank"); return false;
        }
else if(password.length<8)
        {
        alert("Password must be at least 8 characters long.");
        return false;
        }
         }
</script>
</head>
 <body>
<form name="myform" method="post" action="register.php" onsubmit="return
pass_validation()" > Password: <input type="password" name="password">
 <input type="submit" value="submit">
</form>
</body>
</html>
```



Re-type Password Validation

```
<html>
 <head>
<script>
 function pass_validation()
{
 var firstpassword=document.f1.password1.value;
 var secondpassword=document.f1.password2.value;
if(firstpassword==secondpassword)
{
 return true;
}
Else
{
        alert("password one and two must be same!");
       return false;
}
 }
</script>
 </head>
<body>
<form name="f1" action="/JavaScript/Index" onsubmit="return
pass_validation()"> Password:<input type="password" name="password1" />
<br/>
Re-enter Password:<input type="password" name="password2"/><br/>
<input type="submit">
</form>
 </body>
 </html>
```



Thank You