Javascript is a scripting language that will allow you to add real programming to your web pages.

You can create small application type processes with javascript, like a calculator or a primitive game of some sort.

However, there are more serious uses for javascript:

- **Browser Detection**
  Detecting the browser used by a visitor at your page. Depending on the browser, another page specifically designed for that browser can then be loaded.
- **JavaScript Can Change HTML Content**
- **Cookies**
  Storing information on the visitor's computer, then retrieving this information automatically next time the user visits your page. This technique is called "cookies".
- **JavaScript Can Change HTML Attributes**
- **JavaScript Can Change HTML Styles (CSS)**
  document.getElementById("demo").style.fontSize = "25px";
- **Control Browsers**
  Opening pages in customized windows, where you specify if the browser's buttons, menu line, status line or whatever should be present.
- **Validate Forms**
  Validating inputs to fields before submitting a form.
  An example would be validating the entered email address to see if it has an @ in it, since if not, it's not a valid address.

WHERE TO PLACE IT

Since javascript isn't HTML, you will need to let the browser know in advance when you enter javascript to an HTML page. This is done using the <script> tag.

The browser will use the <script type="text/javascript"> and </script> to tell where javascript starts and ends.

NOTE: The type attribute is not required. JavaScript is the default scripting language in HTML.

Consider this example:

```
<html>
<head>
<title>My Javascript Page</title>
</head>

<body>
<script type="text/javascript">
alert("Welcome to my world!!!");
</script>
</body>
</html>
```

The word **alert** is a standard javascript command that will cause an alert box to pop up on the screen. The visitor will need to click the "OK" button in the alert box to proceed.

If we had not entered the <script> tags, the browser would simply recognize it as pure text, and just write it on the screen.

You can enter javascript in both the <head> and <body> sections of the document.
In general however, it is advisable to keep as much as possible in the <head>section.

THE FIRST SCRIPT

Knowing that javascript needs to be entered between <script> tags, is a start. But there are a few other things you need to know before writing your first javascript:

- Javascript lines end with a semicolon.
- Always put the text within " ".
  When entering text to be handled by javascript, you should always put the text within " ".
  If you forget to enclose your text in " ", javascript will interpret your text as being variables rather than text.
- JavaScript is a case-sensitive language.

Now consider this example:

Instead of having javascript write something in a popup box we could have it write directly into the document.

```
<html>
<head>
<title>My Javascript Page</title>
</head>

<body>
<script>
document.write("Welcome to my world!!!<br>");
var myvalue=2;
MyValue=5;
result=myvalue+MyValue;
document.write(result+"<br><br><br>");
</script>
Enjoy your stay...
<br>
</body>
</html>
```

The **document.write** is a javascript command telling the browser that what follows within the parentheses is to be written into the document.

The script in the example would produce this output on your page:

Welcome to my world!!! 7

As you can see, javascript simply writes the text to where the script is placed within the HTML codes. An interesting aspect is that you can write all kinds of HTML tags to webpages with the **document.write** method.

Note: When entering text in javascript you need to include it in " ".

POP UP BOXES

It is possible to make three different kinds of popup windows.

ALERT BOX

The syntax for an alert box is: **alert("yourtext");**

The user will need to click "OK" to proceed.

Typical use is when you want to make sure information comes through to the user.

Examples could be warnings of any kind.

CONFIRM BOX:

The syntax for a confirm box is: **confirm("yourtext");**

The user needs to click either "OK" or "Cancel" to proceed.

Typical use is when you want the user to verify or accept something.

Examples could be age verification like "Confirm that you are at least 57 years old" or technical matters like "Do you have a plug-in for Flash?"

- If the user clicks "OK", the box returns the value **true**.
- If the user clicks "Cancel", the box returns the value **false**.

```
if (confirm("Do you agree"))
{alert("You agree")}
else{alert ("You do not agree")};
```

PROMPT BOX:

The prompt box syntax is: **prompt("yourtext","defaultvalue");**

The user must click either "OK" or "Cancel" to proceed after entering the text.

Typical use is when the user should input a value before entering the page.

Examples could be entering user's name to be stored in a cookie or entering a password or code of some kind.

- If the user clicks "OK" the prompt box returns the entry.
- If the user clicks "Cancel" the prompt box returns **null**.

Since you usually want to use the input from the prompt box for some purpose it is normal to store the input in a variable, as shown in this example:

```
username=prompt("Please enter your name","Enter your name here");
```

ASSIGNING VALUES TO VARIABLES

Example	Resulting value

a=2;
a=2; a++;
a=2; a--;
a=2; b=3; c=a+b;
a=2; d=a+6;
First="MYname";
Last="ABC";
Full=First+" "+Last;
a=2*7;
b=20/5;
c=(20/5)*2;
d=20/(5*2);
	a=2
	a=3 (2+1)
	a=1 (2-1)
	c=5 (2+3)
	d=8 (2+6)
	First=Myname
	Last=ABC
	Full=Myname ABC
	a=14 (2*7)
	b=4 (20/5)
	c=8 (4*2)
	d=2 (20/10)

COMPARING VARIABLES

```
if (a==b) {alert("a equals b")};
if (lastname=="XYZ") {alert("Nice name!!!")};
```

FUNCTIONS

Javascript written into functions will not be performed until you specifically ask for it. This way you gain complete control of the timing.
Look at this example of script lines written as a function:

```
<html>
<head>
<script>
function myfunction()
{
alert("Welcome to my world!!");
}
</script>
</head>
<body>
<form name="myform">
<input type="button" value="Hit me" onclick="myfunction()">
</form>
</body>
</html>
```

Array

```
value=new Array;
for (number=1; number<=100; number=number+1)
    { value[number]=number*10};
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

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

Using window.alert()

You can use an alert box to display data:

Using document.write()

For testing purposes, it is convenient to use **document.write()**:

Using document.write() after an HTML document is fully loaded, will **delete all existing HTML**:

```
<html>
<body>
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<button onclick="document.write(5 + 6)">Try it</button>
</body>
</html>
```

Using innerHTML

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

Example

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

Using console.log()

In your browser, you can use the **console.log()** method to display data. Activate the browser console with F12, and select "Console" in the menu.

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

```
<h1>My First Web Page</h1>
<p>My first paragraph.</p>
<script> console.log(5 + 6);
</script>
</body>
</html>
```

EVENTS

Events are actions that can be detected by javascript.
An example would be the onmouseover event, which is detected when the user moves the mouse over an object.
Another event is the onload event, which is detected as soon as the page is finished loading.
Usually, events are used in combination with functions, so that the function does not start until the event happens.

The following are the most important events recognized by javascript:

Event	**Detected when**	**HTML tags**
onfocus=""	Form field gets focus	select, text, textarea
onblur=""	Form field looses focus	select, text, textarea
onchange=""	Content of a field changes	select, text, textarea
onselect=""	Text is selected	text, textarea
onmouseover=""	Mouse moves over a link	A
onmouseout=""	Mouse moves out of a link	A
onclick=""	Mouse clicks an object	A, button, checkbox, radio, reset, submit
onload=""	Page is finished loading	body, frameset
onunload=""	Browser opens new document	body, frameset
onSubmit=""	Submit button is clicked	form

Data Validation Using JavaScript

Data validation is the process of ensuring that computer input is clean, correct, and useful.
Typical validation tasks are:

- has the user filled in all required fields?
- has the user entered a valid date?
- has the user entered text in a numeric field?

Most often, the purpose of data validation is to ensure correct input to a computer application.
Validation can be defined by many different methods, and deployed in many different ways.

Server side validation is performed by a web server, after input has been sent to the server.
Client side validation is performed by a web browser, before input is sent to a web server.

```
function validateForm() {
  var x = document.forms["myForm"]["fname"].value;
  if (x == null || x == "") {
    alert("Name must be filled out");
    return false;
  }
}
```

The function can be called when the form is submitted:

HTML Form Example

```
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()" method="post">
Name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
```

The Document Object

When an HTML document is loaded into a web browser, it becomes a **document object**.
The document object is the root node of the HTML document and the "owner" of all other nodes:
(element nodes, text nodes, attribute nodes, and comment nodes).
The document object provides properties and methods to access all node objects, from within JavaScript.

Document Object Properties and Methods

The following properties and methods can be used on HTML documents:

Property/Method	Description
Document.activeElement	Returns the currently focused element in the document
document.cookie	Returns all name/value pairs of cookies in the document
Document.domain	Returns the domain name of the server that loaded the document
Document.getElementById()	Returns the element that has the ID attribute with the specified value
Document.getElementByName()	Returns a NodeList containing all elements with a specified name
Document.lastModified	Returns the date and time the document was last modified
Document.URL	Returns the full URL of the HTML document
Document.title	Sets or returns the title of the document