

# Web Engineering



Assiut University

# JavaScript

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

<h2>My First JavaScript</h2>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

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

</body>
</html>
```

## My First JavaScript

Click me to display Date and Time.

Sun Oct 22 2023 16:47:47 GMT+0300 (EEST)

# Why Use JavaScript?

JavaScript enables web designer to do programming in the HTML.

1. Read elements from documents and write new elements and text into documents
2. Manipulate or move text
3. Create pop-up windows
4. Perform mathematical calculations on data
5. React to events, such as a user rolling over an image or clicking a button

# Why Use JavaScript?

- 1 • Retrieve the current date and time from a user's computer.
- 2 • Determine the user's screen size, browser version, or screen resolution
- 3 • **Perform** actions based upon conditions such as alerting users if they enter the wrong information into a form or if they press a certain button.

# Adding a Script to Your Pages

- like CSS rules, JavaScript can either be **embedded** in a page or placed in an **external** script file.
- But in order to work in the browser, the browser must support JavaScript *and* must have it enabled
- You add scripts to your page inside the <script> element. The type attribute on the opening <script> tag indicates what scripting language will be found inside the element.
- Other languages exist like VB script and perl not only javascript.

# Where to Add a Script to Your Pages?

There are three places where you can put your JavaScript

**1-In the <head> of a page:** These scripts will be **called** when an event triggers them.

**2-In the <body> section:** These scripts will **run** as the **page loads**.

**3- In an external file:** You can also write JavaScript in external documents that have the file extension .js. you need to use the src attribute on the <script> element.

```
<script type="JavaScript" src="scripts/validation.js" />
```

# *Comments in JavaScript*

```
<script type="text/javascript">  
document.write("My first JavaScript") //to eof line comment goes here  
</script>
```

```
/* This whole section is commented  
out so it is not treated as a part of  
the script. */
```

# *The <noscript> Element*

- What I should do (as a browser) if I can't run js
- ✗ • Offers alternative content for users whose browsers do not support JavaScript.



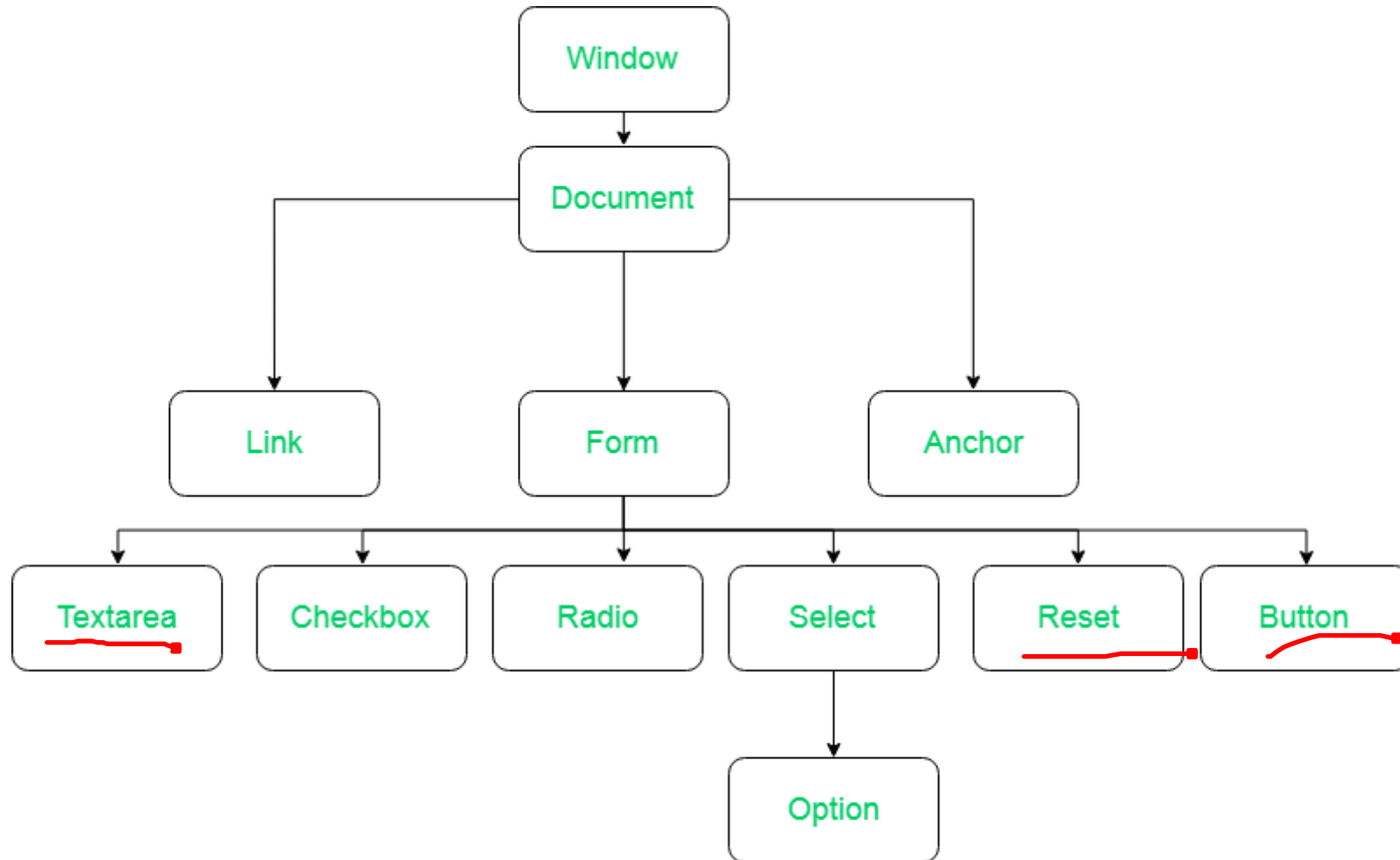
# Document Object Model (DOM)

The document object model explains what properties of a document a script can retrieve and which ones it can alter;

It also defines some methods that can be called upon the document.

The document object model specifies how you can retrieve values users have entered into a form. Once you have retrieved these values, you can use JavaScript to ensure the user has entered an appropriate value for that form control.

# DOM



# DOM Figure

The Figure shows you how the elements of a page are made available in scripts as scriptable objects. The Document object represents the whole document and then each of the child objects represents a collection of similar tags within that document:

- 1 • The *forms* collection contains all of the <form> tags in the document.
- 2 • The *image* collection represents all of the images in a document.
- 3 • The *link* collection represents all of the hyperlinks within a page.
- 4 • The *anchor* collection represents all of the anchors in a document (<a> elements with a name or id attribute rather than an href attribute).
- 5 • The *area* collection represents all of the image maps that use an <area> element in the document.
- The *applet* collection represents all of the applets within a document

## Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
document.body.style.backgroundColor = "red";
}
function myFunction2() {
document.body.style.backgroundColor = "green";
}
</script>
</head>
<body>
<h1>HTML Style Object</h1>
<h2>The backgroundColor Property</h2>
<button type="button" onclick="myFunction()">Set background color</button>
<button type="button" onclick="myFunction2()">Set background color</button>
</body>
</html>
```

# Example

## HTML Style Object

### The backgroundColor Property

Set background color

Set background color

## HTML Style Object

### The backgroundColor Property

Set background color

Set background color

# Example (get Element by ID and apply Style by JS)

```
document.getElementById("myCSSid").innerHTML = "Paragraph contents changed!";
```

```
document.getElementById("myCSSid").src = "OlympiaLogo.jpg";
```

```
document.getElementById("myCSSid").style.color = 'red';
```

# Example

The `classList` property is read-only, but you can use the methods listed below, to add, toggle or remove CSS classes from the list

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.body.classList.toggle("mode");
}
</script>
<style>
.mode{
background-color:red;
color:white;
}
</style>
</head>
<body>
<h1>HTML Style Object</h1>
<h2>The classList Property</h2>
<button type="button"
  onclick="myFunction()">Toggle</button>

</body>
</html>
```

## HTML Style Object

### The `classList` Property

## HTML Style Object

### The `classList` Property



# Back Button

```
<BUTTON onclick="history.back()"> GO BACK </BUTTON>
```

# An Example for further slides

```
<h1>User Registration</h1>
```

```
<form name="frmLogin" action="login.asp" method="post">
```

```
    Username <input type="text" name="txtUsername" size="12" /> <br />
```

```
    Password <input type="password" name="pwdPassword" size="12" /> <br />
```

```
    <input type="submit" value="Log In" />
```

```
</form>
```

```
<p>
```

```
    If you are a new user <a href="register.asp">Register here</a> |
```

```
    If you have lost your password you can <a href="lostPassword.asp">retrieve
```

```
your password here</a>
```

```
</p>
```

# DOM Usage

The DOM would therefore make the content of the form available for use in the script as part of the forms collection and the links as part of the links collection.

`document.links[0].href`

There are four parts of this statement

- 1-The word document indicates I am accessing the document object.
- 2-The word links corresponds to the links collection.
- 3-The [0] indicates that I want the first link in the document.
- 4- You want to retrieve the href property for this link.

# Alternative DOM Usage

An alternative approach is to use the names of elements to navigate a document. For an example, the following line requests the value of the password box:

```
document.frmLogin.pwdPassword.value
```

Again there are four parts to this statement:

- 1- The document comes first again as it is the top-level object.
- 2- Next you can see the name of the form frmLogin.
- 3- This is followed by the name of the form control pwdPassword.
- 4- Finally the property I am interested in is the value of the password box, and this property is called value

# Objects, Methods, and Properties

An object model (such as the document object model) is made up of several objects, each of which can have *properties* and *methods*:

A property tells you something about an object.

A method performs an action.

# Properties of the Document Object

Property	Description
document.anchors	Returns all <a> elements that have a name attribute.
document.applets	Deprecated
document.baseURI	Returns the absolute base URI of the document
document.body	Returns the <body> element
document.cookie	Returns the document's cookie
document.doctype	Returns the document's doctype
document.documentElement	Returns the <html> element
document.documentMode	Returns the mode used by the browser
document.documentURI	Returns the URI of the document
document.domain	Returns the domain name of the document server
document.domConfig	Obsolete.
document.embeds	Returns all <embed> elements

# Properties of the Document Object

document.forms	Returns all <form> elements
document.head	Returns the <head> element
document.images	Returns all <img> elements
document.implementation	Returns the DOM implementation
document.inputEncoding	Returns the document's encoding (character set)
document.lastModified	Returns the date and time the document was updated
document.links	Returns all <area> and <a> elements that have a href attribute
document.readyState	Returns the (loading) status of the document
document.referrer	Returns the URI of the referrer (the linking document)
document.scripts	Returns all <script> elements
document.strictErrorChecking	Returns if error checking is enforced
document.title	Returns the <title> element
document.URL	Returns the complete URL of the document

# Methods of the Document Object

**write(string)** Allows you to add text or elements into a document

**writeln(string)** The same as *write()* but adds a new line at the end of the output



# The Forms Collection

- \* The forms collection holds references corresponding to each of the `<form>` elements in the page.

If the login form is the first form in the document and you want to access the action property

**`document.forms[0].action`**

Alternatively, you can directly access that form object using its name:

**`document.frmLogin.action`**

# Properties of the Form Objects

action, length, method, name, target

5

## Methods of the Form Objects

**reset()** Resets all form elements to  
default values

**submit()** Submits the form

2

# Form Elements

When you access a form you usually want to access one or more of its elements.

- ✗ Each form element has an elements[] collection object as a property

# Example

```
<html>
<body>
<form id="myForm" action="/action_page.php">
First name: <input type="text" name="fname" value="Donald"><br>
Last name: <input type="text" name="lname" value="Duck"><br>
<input type="submit" value="Submit">
</form>
<p>Click the "Try it" button to display the number of elements in the form.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
var x = document.getElementById("myForm").elements.length;
document.getElementById("demo").innerHTML = "Found " + x + " elements in the form.";
}
</script>
</body>
</html>
```

# Example

```
<html>
<head>
<script>
function myFunction() {
var x = document.getElementById("myForm").elements.length;
document.getElementById("demo").innerHTML = "Found " + x + " elements in the form.";
}
</script>
<\head>
<body>
<form id="myForm" action="/action_page.php">
First name: <input type="text" name="fname" value="Donald"><br>
Last name: <input type="text" name="lname" value="Duck"><br>
<input type="submit" value="Submit">
</form>
<p>Click the "Try it" button to display the number of elements in the form.</p>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
</body>
```

First name:

Last name:

Click the "Try it" button to display the number of elements in the form.

Found 3 elements in the form.

# Thank U

# What to do with the Elements?

Here are some of the things you might want to do with the elements in a form:

- 1- **Text fields:** Read data a user has entered or write new text to these elements.
- 2- **Checkboxes and radio buttons:** Test if they are checked and check or uncheck them.
- 3- **Buttons:** Disable them until a user has selected an option.
- 4- **Select boxes:** As with checkboxes and radio buttons some of the properties and methods of individual form element objects.



# Properties of Form Elements



- Checked      Checkboxes and radio buttons  
Returns true when checked or false  
when not  
Read/write
- disabled      All except hidden Returns true  
when disabled and user cannot  
interact with it (supported in IE 4  
and Netscape 6 and later versions  
only)

# Properties of Form Elements

- Form All elements Returns a reference to the form it is part of  
Read only
- length Select boxes Number of options in the `<select>` element
- name All elements Accesses the name attribute of the element (Read only)

# Properties of Form Elements

- `selectedIndex` Select boxes Returns the index number of the currently selected item (Read/Write)
- `type` All Returns type of form control Read only
- `value` All Accesses the value attribute of

# Different Types of Objects in JavaScript

1-W3C DOM objects:

like document object, browser object,...

2-Built-in Objects:

such as date object which deal with date and time , math object

3-Custom objects:

for advanced JavaScript ,you can write your own objects an example

→ for form validation

# Start Programming in JavaScript

Programming is the capability to make calculations in the HTML documents and it has many of the features and concepts in programming languages:

- 1- A variable is used to store data of different data types.
- 2-Operators: Arithmetic, Relational, Logic, Assignment, String.
- 3-Conditional statements
- 4-Loops
- 5-Functions

# Variables

There are a few rules you must remember about variables in JavaScript:

- 1 • Variable names are case-sensitive.
- 2 • They must begin with a letter or the underscore character.
- 3 • Avoid giving two variables the same name within the same document as one might override the value of the other, creating an error.
- 4 • Try to use descriptive names for your variables.

# Variable Declaration

Variables are declared by var keyword e.g

```
var userName = "Ahmed Aly"
```

```
userName = "Mohamed Aly"
```

```
var x, y, z
```

```
x=" javaScript programming "
```

```
y=1240
```

```
z=1234.567
```

# Lifetime of a Variable

Variables declared in a function are local to this function scope.

The same variable name can be used in different functions.

Variables declared outside functions are global variables and can be used by any function.



# Arithmetic Operators

## Symbol Description Example (x = 10) Result

- + Addition  $x+5$  15
- - Subtraction  $x-2$  8
- \* Multiplication  $x*3$  30
- / Division  $x/2$  15
- % Modulus (division remainder)  $x\%3$
- ++ Increment (increments the variable by 1—this technique is often used in counters) if  $x = 10$  then  $x++ \rightarrow 11$
- -- Decrement (decreases the variable by 1)  $x--$  9

# Assignment Operators ( = )

Variable = variable or expression

total = total – profit

Shorthand


Symbol	Using Sh	Equi Without SH
--------	----------	-----------------

- |      |      |       |
|------|------|-------|
| • += | x+=y | x=x+y |
| • -= | x-=y | x=x-y |
| • *= | x*=y | x=x*y |
| • /= | x/=y | x=x/y |
| • %= | x%=y | x=x%y |

# Comparison Operators

Operator	Description	Example
• ==	is equal to	1==2 returns false 3==3 returns true
• !=	not equal to	1!=2 returns true 3!=3 returns false
• >	greater than	1>2 false
• >=	grater than or eq to	3>=2 true
• <	less than	2< 3 true
• <=	less than or equal to	3<=3 true

# Logical Operator

Operator	Name	Desc	Example ( x=1 and y=2)
• &&	And	Allows you to check if both of two conditions are met (x < 2 && y > 1) Returns true	
•	Or	Allows you to check if one of two conditions are met (x < 2    y < 2) Returns true	
• ! 	Not	Allows you to check if something is not the case ! (x > y) returns true	

✓  
.  
!  
!

# String Operator

```
firstName = "Bob"
```

```
lastName = "Stewart"
```

```
name = firstName + lastName
```

# Functions

- A function is just like a method, except a method belongs to an object, and a function lives on its own at the top of a document—but both perform actions.

# How to Define a Function

There are three parts to creating or defining a function:

1-Define a name for it.

2-Indicate any values that might be required as arguments.

3-Add statements.

```
function calculateArea(width, height)  
{  
    area = width * height  
    return area  
}
```

# How to Call a Function

```
<form name="frmArea" action="">
```

Enter the width and height of your rectangle to calculate the size:<br />

Width: <input type="text" name="txtWidth" size="5" /><br />

Height: <input type="text" name="txtHeight" size="5" /><br />

<input type="button" value="Calculate area"

onclick="alert(calculateArea(document.frmArea.txtWidth.value, document.frmArea.txtHeight.value))" />

```
</form>
```



# Conditional Statements

- if statements, which are used when you want the script to execute if a condition is true
- if...else statements, which are used when you want to execute one set of code if a condition is true and another if it is false
- switch statements, which are used when you want to select one block of code from many

# Conditional Statements Example

```
<script type="text/JavaScript">  
    date = new Date();  
    time = date.getHours();  
    if (time < 12)  
        document.write('Good Morning');  
    else  
        document.write('Good Afternoon');  
</script>
```

# Conditional Statements Example

```
switch (expression)
{
    case option1:
        code to be executed if expression is what is written in option1
        break
    case option2:
        code to be executed if expression is what is written in option2
        break
    .....
    default:
        code to be executed if expression is different from option1, option2, and option3
}
```

# Conditional (or Ternary) Operator

A conditional operator (also known as the ternary operator) assigns a value to a variable based upon a condition:

```
variablename=(condition)?value1:value2
```



# JS cont.

Digger deeper into JS

# Looping

- A ***while*** loop runs the same block of code while or until a condition is true.
- A ***do while*** loop runs once before the condition is checked. If the condition is true, it will continue to run until the condition is false.
- A ***for*** loop runs the same block of code a specified number of times.

# The For Loop

The **for** statement creates a loop with 3 optional expressions:

```
for (expression 1; expression 2; expression 3) {  
    // code block to be executed  
}
```

**Expression 1** is executed (one time) before the execution of the code block.

**Expression 2** defines the condition for executing the code block.

**Expression 3** is executed (every time) after the code block has been executed.

## Example

Instead of writing:

```
text += cars[0] + "<br>";  
text += cars[1] + "<br>";  
text += cars[2] + "<br>";  
text += cars[3] + "<br>";  
text += cars[4] + "<br>";  
text += cars[5] + "<br>";
```

You can write:

```
for (let i = 0; i < cars.length; i++) {  
  text += cars[i] + "<br>";  
}
```



# Example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For Loop</h2>
<p id="demo"></p>
<script>
  cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
  text = "";
  for (let i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
  }
  document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

## JavaScript For Loop

BMW2  
Volvo  
Saab  
Ford  
Fiat  
Audi



## Loop scope

```
var i = 5;  
  
for (var i = 0; i < 10; i++) {  
    // some code  
}  
  
// Here i is 10
```

# Using Let

- Using `let`, the variable declared in the loop does not redeclare the variable outside the loop.
- When `let` is used to declare the `i` variable in a loop, the `i` variable will only be visible within the loop.

```
let i = 5;

for (let i = 0; i < 10; i++) {
    // some code
}

// Here i is 5
```

# The While Loop

The **while** loop loops through a block of code as long as a specified condition is true.

## Syntax

```
while (condition) {  
    // code block to be executed  
}
```

# Example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript While Loop</h2>
<p id="demo"></p>
<script>
let text = "";
let i = 0;
while (i < 10) {
text += "<br>The number is " + i;
i++;
}
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

## JavaScript While Loop

The number is 0  
The number is 1  
The number is 2  
The number is 3  
The number is 4  
The number is 5  
The number is 6  
The number is 7  
The number is 8  
The number is 9

# The Do While Loop

- The do while loop is a variant of the while loop.
- This loop will execute the code block **once**, before checking if the condition is true, then it will repeat the loop as long as the **condition** is true.

## Syntax


- do {
- // code block to be executed
- }
- while (condition);

# Example

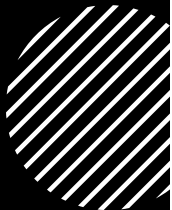

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Do While Loop</h2>
<p id="demo"></p>
<script>
let text = ""
let i = 0;
do {
text += "<br>The number is " + i;
i++;
}
while (i > 10);
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

## JavaScript Do While Loop

The number is 0



# Events: Types of events



Window Event Attributes



Form Events



Keyboard Events



Mouse Events



Clipboard Events



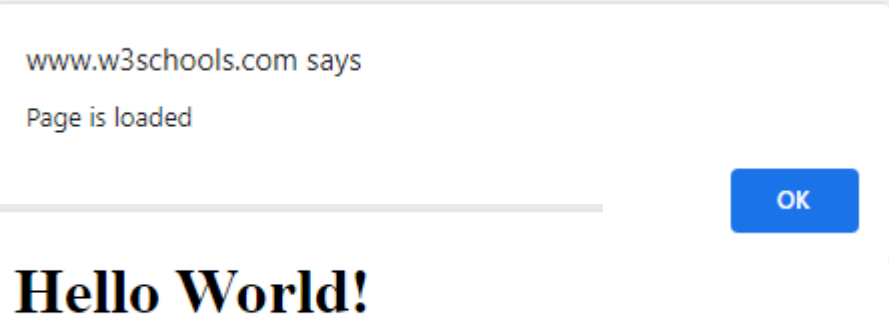
# Window Event Attributes

<b>onload</b>	<b>Fires after the page is finished loading</b>
---------------	---

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  alert("Page is loaded");
}
</script>
</head>

<body onload="myFunction()">
<h1>Hello World!</h1>
</body>

</html>
```



www.w3schools.com says  
Page is loaded

OK

**Hello World!**

# Form Events

onsubmit

Fires when a form is submitted

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>When you submit the form, a function is triggered which  
alerts some text.</p>
```

```
<form action="/action_page.php" onsubmit="myFunction()">
```

```
  Enter name: <input type="text" name="fname">
```

```
  <input type="submit" value="Submit">
```

```
</form>
```

```
<script>
```

```
function myFunction() {
```

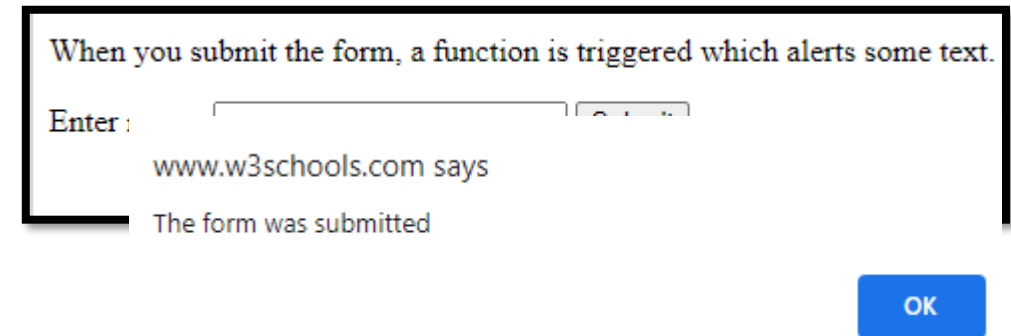
```
  alert("The form was submitted");
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```



The screenshot shows a web form with the text "When you submit the form, a function is triggered which alerts some text." Below this is a label "Enter:" followed by a text input field containing "www.w3schools.com" and a "Submit" button. Below the form, an alert box is displayed with the message "The form was submitted" and an "OK" button.

# Keyboard Events

[onkeydown](#)

Fires when a user is pressing a key

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>A function is triggered when the user is pressing a key in the input field.</p>
```

```
<input type="text" onkeydown="myFunction()">
```

```
<script>
```

```
function myFunction() {  
  alert("You pressed a key inside the input field");  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

A function is triggered when the user is pressing a key in the input field.

www.w3schools.com says

You pressed a key inside the input field

OK

# Mouse Events

[ondblclick](#)

Fires on a mouse double-click on the element

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<button ondblclick="myFunction()">Double-click me</button>
```

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

```
<p>A function is triggered when the button is double-clicked.  
The function outputs some text in a p element with  
id="demo".</p>
```

```
<script>
```

```
function myFunction() {  
  document.getElementById("demo").innerHTML = "Hello  
World";  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

Double-click me

A function is triggered when the button is double-clicked. The function outputs some text in a p element with id="demo".

Double-click me

Hello World

A function is triggered when the button is double-clicked. The function outputs some text in a p element with id="demo".

# Clipboard Events

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

```
<input type="text" oncopy="myFunction()" value="Try to copy
this text">
```

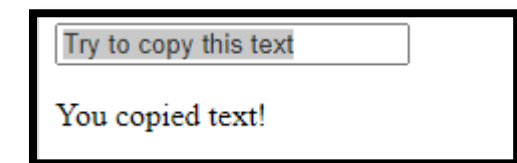
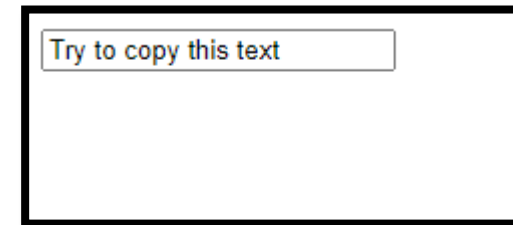
```
<p id="demo"></p>
```

```
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "You copied
text!"
}
```

```
</script>
```

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

Attribute	Description
<a href="#">oncopy</a>	Fires when the user copies the content of an element
<a href="#">oncut</a>	Fires when the user cuts the content of an element
<a href="#">onpaste</a>	Fires when the user pastes some content in an element



# Drag Events

---

Attribute	Value	Description
<a href="#"><u>ondrag</u></a>	<i>script</i>	Script to be run when an element is dragged
<a href="#"><u>ondragend</u></a>	<i>script</i>	Script to be run at the end of a drag operation
<a href="#"><u>ondragenter</u></a>	<i>script</i>	Script to be run when an element has been dragged to a valid drop target
<a href="#"><u>ondragleave</u></a>	<i>script</i>	Script to be run when an element leaves a valid drop target
<a href="#"><u>ondragover</u></a>	<i>script</i>	Script to be run when an element is being dragged over a valid drop target
<a href="#"><u>ondragstart</u></a>	<i>script</i>	Script to be run at the start of a drag operation
<a href="#"><u>ondrop</u></a>	<i>script</i>	Script to be run when dragged element is being dropped
<a href="#"><u>onscroll</u></a>	<i>script</i>	Script to be run when an element's scrollbar is being scrolled



# Extra Various Events



---

onload, onunload, onclick,  
ondblclick,

---

onmousedown, onmouseup,  
onmouseover,

---

onmousemove, onmouseout,  
onkeypress,

---

onkeydown, onkeyup, onfocus,  
onblur,

---

onsubmit, onreset, onselect,  
onchange



# Objects



# Built-in Objects

## *String*

- The string object allows you to deal with strings of text.
- Before you can use a built-in object you need to create an instance of that object.
- You create an instance of the string object by assigning it to a variable like so:

```
myString = new String('Here is some big text')
```

# String Properties and Methods

<a href="#"><u>charAt()</u></a>	Returns the character at a specified index (position)
<a href="#"><u>length</u></a>	Returns the length of a string
<a href="#"><u>replace()</u></a>	Searches a string for a pattern, and returns a string where the first match is replaced
<a href="#"><u>split()</u></a>	Splits a string into an array of substrings
<a href="#"><u>trim()</u></a>	Returns a string with removed whitespaces

# String HTML Wrapper Methods

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h1>JavaScript String Methods</h1>
```

```
<h2>The link() Method</h2>
```

```
<p id="demo1"></p>
```

```
<p>The link() method is deprecated in JavaScript.</p>
```

```
<p>Use an a tag instead:</p>
```

```
<p id="demo2"></p>
```

```
<script>
```

```
let text = "Free Web Building Tutorials!";
```

```
let result = text.link("https://www.w3schools.com");
```

```
document.getElementById("demo1").innerHTML = result;
```

```
result = "<a href='https://www.w3schools.com'>" + text +  
"</a>";
```

```
document.getElementById("demo2").innerHTML = result;
```

```
</script>
```

```
</body>
```

```
</html>
```

[link\(\)](#)

Displays a string as a hyperlink

## JavaScript String Methods

### The link() Method

[Free Web Building Tutorials!](#)

The link() method is deprecated in JavaScript.

Use an a tag instead:

[Free Web Building Tutorials!](#)

# *Date*

The ***date*** object helps you work with dates and times. You create a new date object using the date constructor like so:

```
new Date()
```

You can create a date object set to a specific date or time, in which case you need to pass it one of four parameters:

- **milliseconds:** This value should be the number of milliseconds since 01/01/1970.
- **dateString:** Can be any date in a format recognized by the `parse()` method.
- **yr\_num, mo\_num, day\_num:** Represents year, month, and day.
- **yr\_num, mo\_num, day\_num, hr\_num, min\_num, seconds\_num:** Represents the hours, days, minutes, seconds, and milliseconds.

# Date Examples

- **The first uses milliseconds and will read Thu Nov 27 05:33:20 UTC 1975:**

```
var birthDate = new Date(8298400000)
document.write(birthDate)
```

- **The second uses a dateString, and will read Wed Apr 16 00:00:00 UTC+0100 1975:**

```
var birthDate = new Date("April 16, 1975")
document.write(birthDate)
```

- **The third uses yr\_num, mo\_num, and day\_num, and will read Mon May 12 00:00:00 UTC+0100 1975:**

```
var birthDate = new Date(1975, 4, 28)
document.write(birthDate)
```

# Date Methods

<a href="#"><u>getDate()</u></a>	Returns the day of the month (from 1-31)
<a href="#"><u>getDay()</u></a>	Returns the day of the week (from 0-6)
<a href="#"><u>getFullYear()</u></a>	Returns the year
<a href="#"><u>getHours()</u></a>	Returns the hour (from 0-23)
<a href="#"><u>getMilliseconds()</u></a>	Returns the milliseconds (from 0-999)
<a href="#"><u>getMinutes()</u></a>	Returns the minutes (from 0-59)
<a href="#"><u>getMonth()</u></a>	Returns the month (from 0-11)
<a href="#"><u>getSeconds()</u></a>	Returns the seconds (from 0-59)

# Math

The math object helps in working with numbers

Property	Purpose
• E	Returns the base of a natural logarithm
• LN2	Returns the natural logarithm of 2
• LN10	Returns the natural logarithm of 10
• LOG2E	Returns the base-2 logarithm of E
• LOG10E	Returns the base-10 logarithm of E
• PI	Returns pi
• SQRT1_2	Returns 1 divided by the square root of 2
• SQRT2	Returns the square root of 2

# Example

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

<h2>JavaScript Math Constants</h2>

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

<script>
document.getElementById("demo").innerHTML =
"<p><b>Math.E:</b> " + Math.E + "</p>" +
"<p><b>Math.PI:</b> " + Math.PI + "</p>" +
"<p><b>Math.SQRT2:</b> " + Math.SQRT2 + "</p>" +
"<p><b>Math.SQRT1_2:</b> " + Math.SQRT1_2 + "</p>" +
"<p><b>Math.LN2:</b> " + Math.LN2 + "</p>" +
"<p><b>Math.LN10:</b> " + Math.LN10 + "</p>" +
"<p><b>Math.LOG2E:</b> " + Math.LOG2E + "</p>" +
"<p><b>Math.Log10E:</b> " + Math.LOG10E + "</p>";
</script>

</body>
</html>
```

## JavaScript Math Constants

**Math.E:** 2.718281828459045

**Math.PI:** 3.141592653589793

**Math.SQRT2:** 1.4142135623730951

**Math.SQRT1\_2:** 0.7071067811865476

**Math.LN2:** 0.6931471805599453

**Math.LN10:** 2.302585092994046

**Math.LOG2E:** 1.4426950408889634

**Math.Log10E:** 0.4342944819032518



# Math Methods

Method	Purpose
• <code>abs(x)</code>	Returns the absolute value of $x$
• <code>acos(x)</code>	Returns the arccosine of $x$
• <code>asin(x)</code>	Returns the arcsine of $x$
• <code>atan(x)</code>	Returns the arctangent of $x$
• <code>atan2(y,x)</code>	Returns the angle from the x-axis to a point
• <code>ceil(x)</code>	Returns the nearest integer greater than or equal to $x$
• <code>cos(x)</code>	Returns the cosine of $x$
• <code>exp(x)</code>	Returns the value of $E$ raised to the power of $x$

# Math Methods

Method	Purpose
• floor( $x$ )	Returns the nearest integer less than or equal to $x$
• log( $x$ )	Returns the natural log of $x$
• max( $x, y$ )	Returns the number with the highest value of $x$ and $y$
• min( $x, y$ )	Returns the number with the lowest value of $x$ and $y$
• pow( $x, y$ )	Returns the value of the number $x$ raised to the power of $y$

# Math Methods

Method	Purpose
• random()	Returns a random number between 0 and 1
• round(x)	Rounds $x$ to the nearest integer
• sin(x)	Returns the sine of $x$
• sqrt(x)	Returns the square root of $x$
• tan(x)	Returns the tangent of $x$

# *Array*

An *array* is like special variable

- `instruments = new Array("guitar", "drums", "piano")`
- `instruments = new Array(3)`

# Window

Every browser window and frame has a corresponding Window object that is created with every instance of a `<body>` or `<frameset>` element.

# Window Properties

## Property

## Purpose

- `closed` A Boolean determining if a window has been closed. If it has, the value returned is true.
- `defaultStatus` Defines the default message displayed in a browser window's status bar (usually at the bottom of the page on the left)
- `document` The document object contained in that window.
- `Frames` An array containing references to all named child frames in the current window.

# Methods

Method	Purpose
• <code>alert()</code>	Displays an alert box containing a message and an OK button.
• <code>back()</code>	Same effect as the browser's Back button.
• <code>close()</code>	Closes the current window or another window if a reference to another window is supplied.

# Tips for writing JavaScript

## ***1-Has Someone Already Written This Script?***

- <http://www.HotScripts.com>
- <http://www.JavaScriptKit.com>
- <http://www.JavaScriptSource.com>
- ***Reusable Functions***



# Form Validation

Forms are usually validated using the **onsubmit** event handler, which triggers a validation function stored in the head of the document, so the values are checked when the user presses the Submit button.

- You have probably seen forms on Web sites that first ask you to enter a password, and then to re-enter it to make sure you did not mistype something. It might look something like Figure 15-1.
- In such a form you might want to check a few things:
  - ☐ That the username is of a minimum length
  - ☐ That the password is of a minimum length
  - ☐ That the two passwords match

# Form Validation Example

```
function validate(form) {  
    var returnValue = true;  
    var username = frmRegister.txtUserName.value;  
    var password1 = frmRegister.txtPassword.value;  
    var password2 = frmRegister.txtPassword2.value;  
    if(username.length < 6) {  
        returnValue = false;  
        alert("Your username must be at least\n6 characters long.\nPlease try again.");  
        frmRegister.txtUserName.focus();  
    }  
    if (password1.length < 6) {  
        returnValue = false;  
        alert("Your password must be at least\n6 characters long.\nPlease try again.");  
    }  
}
```

# Form Validation Example

```
        frmRegister.txtPassword.value = "";
        frmRegister.txtPassword2.value = "";
        frmRegister.txtPassword.focus();
    }
    if (password1.value != password2.value) {
        returnValue = false;
        alter("Your password entries did not match.\nPlease try again.");
        frmRegister.txtPassword.value = "";
        frmRegister.txtPassword2.value = "";
        frm Register.txtPassword.focus();
    }
    return returnValue;
}
```

Text Control

Username:

Password:

Please confirm your password:

**Internet Explorer Script Alert**

! Your password must be at least 6 characters long. Please try again.

# Required Text Fields

Often you will want to ensure that a user has entered some value into a text field.

Use a loop to go through all of the required elements using a for loop, and if any of them are empty, return an error.

When you use this technique you need to have a class attribute that has a value of required on each form element that is required so the loop can tell if the text input must have a value, and you must have a name attribute whose value matches the label for the element (because this will be used in any error message).

# Example

Here is an example of what a text input should look like with the name and class attributes:

```
<input type="text" name="Username" size="5" class="required" />
```

# Example

```
function validate(form) {  
    var returnValue = true;  
    var formElements = form.elements;  
    for (var i=0; i<formElements.length; i++) {  
        currentElement = formElements[i];  
        if (currentElement.value=="" && currentElement.className=="required") {  
            alert("The required field \""+currentElement.name+"\" is empty. Please  
                provide a value for it.");  
            currentElement.focus();  
            returnValue = false;  
            break;  
        }  
    }  
    return returnValue;}}
```

```
<form name="frmEnquiry" method="post" action="register.asp" onsubmit="return  
validate(this);">  
    <table>  
    <tr>  
        <td>Name:</td>  
        <td><input type="text" class="required" name="Name" size="12" /></td>  
    </tr>  
    <tr>  
        <td>E-mail: </td>  
        <td><input type="text" class="required" name="Email" size="12" /></td>  
    </tr>
```



```

<tr>
  <td valign="top">Please enter your query here:</td>
  <td><textarea rows="8" class="required" cols="30" name="Query">
    </textarea></td> Creating a JavaScript Library
</tr>
<tr>
  <td>&nbsp;</td>
  <td><input type="submit" value="Submit your query" /></td>
</tr>
</table>
</form>

```

Form validation

Name:

E-mail:

Please enter your query here:

Internet Explorer Script Alert

! The required field " E-mail" is empty. Please provide a value for it.

OK

# Select Box & Radio

## Select Box Options

- If you want to check whether a user has selected one of the items from a select box you need to use the selectedIndex property of the select object that represents the select box.

## Radio Buttons

- If you want to ensure that a radio button has been selected you can either preselect one of the radio button values or you can loop through the RadioButton object's checked properties to see if one has been selected.

# An Example

```
<form name="frmCards" action="cards.asp" method="post" onsubmit="return  
validateForm(this)" >
```

```
    <p>Please select a suit of cards.</p>
```

```
    <p><input type="radio" name="radSuit" value="hearts" /> Hearts </p>
```

```
    <p><input type="radio" name="radSuit" value="diamonds" /> Diamonds
```

```
</p>
```

```
    <p><input type="radio" name="radSuit" value="spades" /> Spades </p>
```

```
    <p><input type="radio" name="radSuit" value="clubs" /> Clubs </p>
```

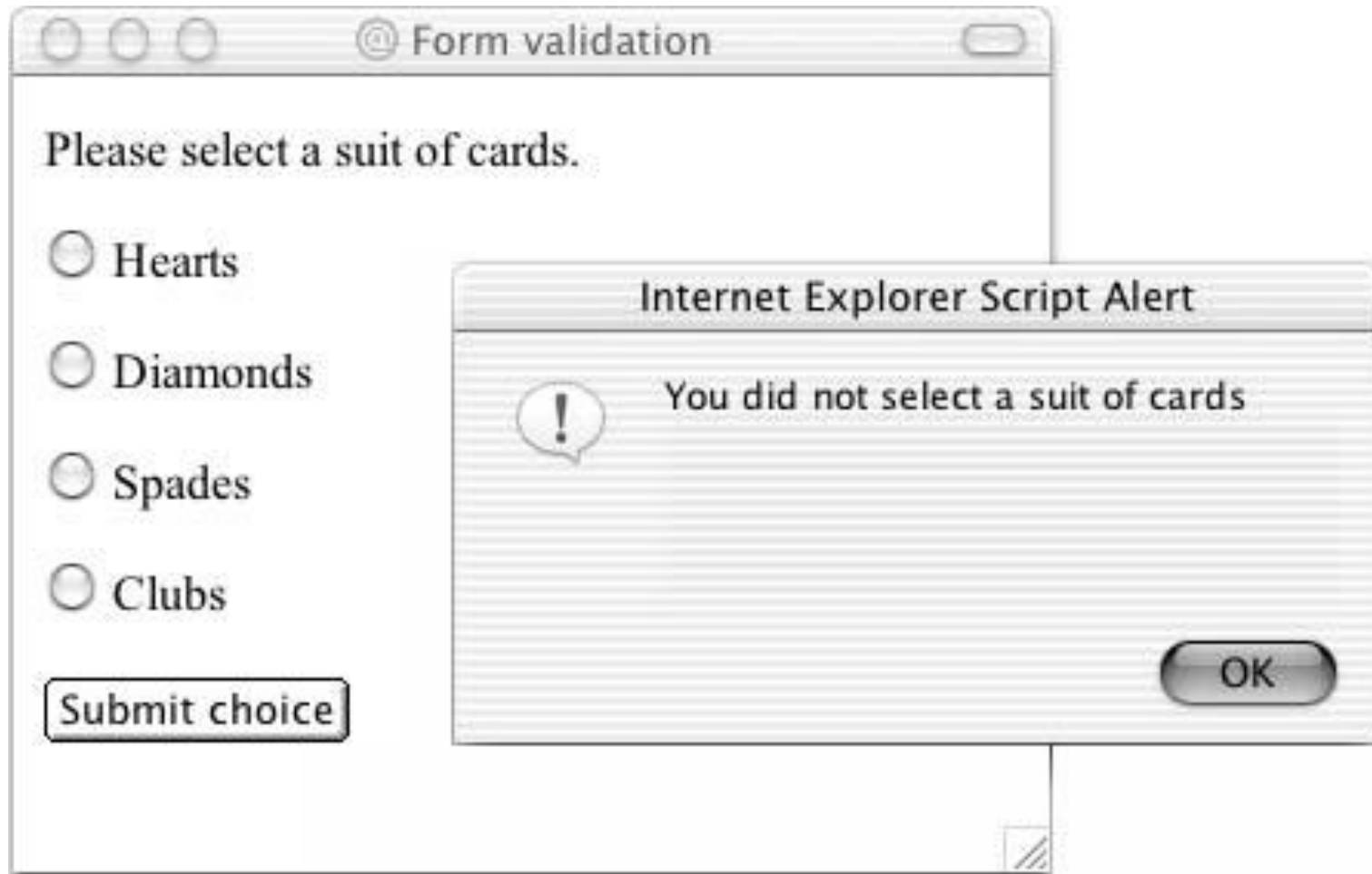
```
    <p><input type="submit" value="Submit choice" /></p>
```

```
</form>
```

```
function validate(form) {
```

- var radioButtons = form.radSuit;
- var radioChosen = false;
- for (var i=0; i<radioButtons.length; i++) {
  - if (radioButtons[i].checked)
    - {
      - radioChosen=true;
      - returnValue=true;
    - }
  - }
- if (radioChosen == false) {
  - returnValue = false;
  - alert("You did not select a suit of cards");
- }
- return returnValue;

```
}
```



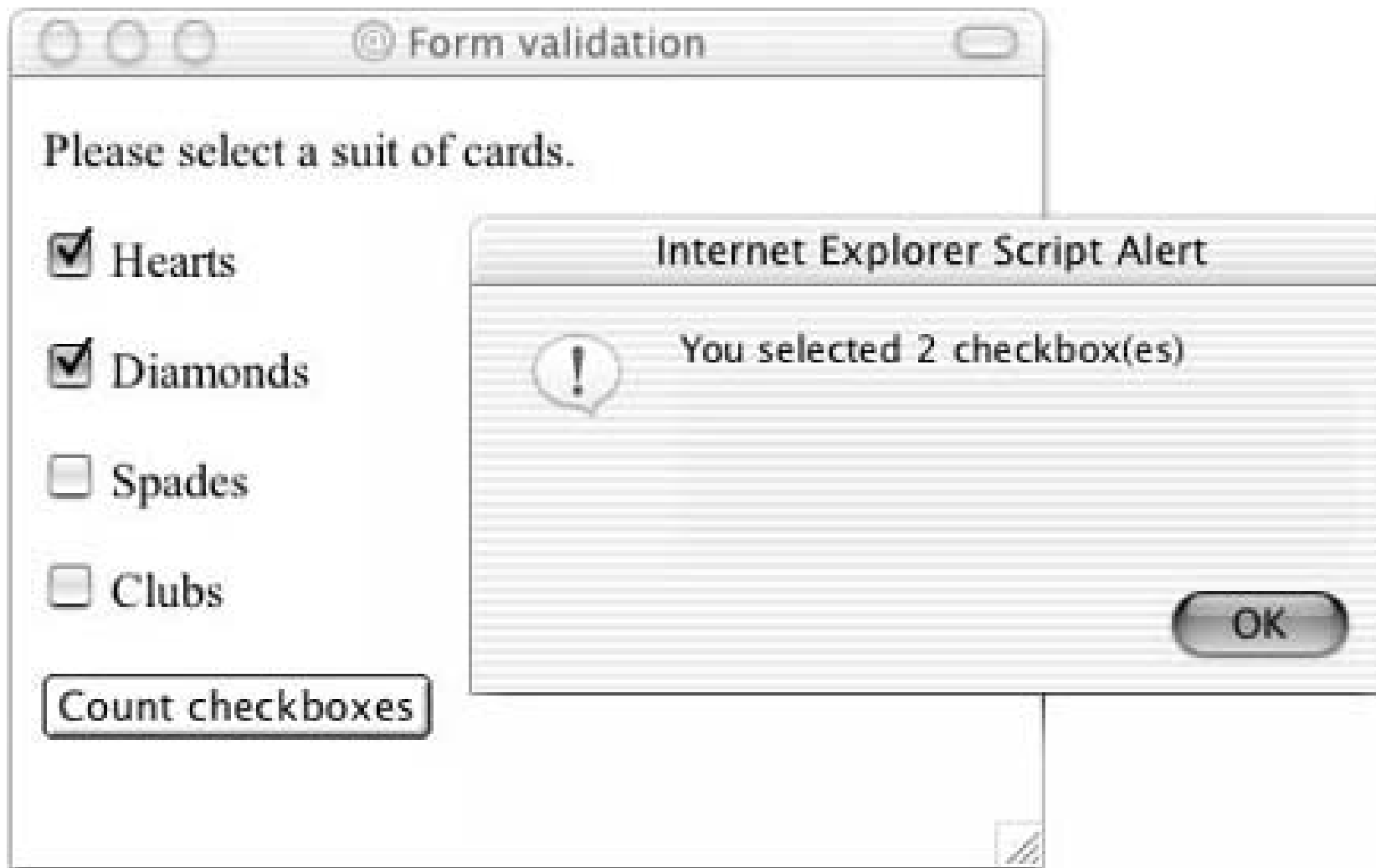
# Checkboxes

```
<form name="frmCards" action="cards.asp" method="post">
  <p>Please select one or more suits of cards.</p>
  <p><input type="checkbox" name="chkSuit" value="hearts" /> Hearts </p>
  <p><input type="checkbox" name="chkSuit" value="diamonds" />
  Diamonds </p>
  <p><input type="checkbox" name="chkSuit" value="spades" /> Spades
  </p>
  <p><input type="checkbox" name="chkSuit" value="clubs" /> Clubs </p>
  <p><input type="button" value="Count checkboxes"
  onclick="countCheckboxes(frmCards.chkSuit)" /></p>
</form>
```

# JS

```
function countCheckboxes(field) {  
    var intCount = 0  
    for (var i = 0; i < field.length; i++) {  
        if (field[i].checked)  
            intCount++;  
    }  
    alert("You selected " + intCount + " checkbox(es)");  
}
```





# Example

## Disabling a Submit Button Until a Checkbox Has Been Selected

If you want to ensure that a checkbox has been selected—for example, if you want a user to agree to certain terms and conditions

```
function checkCheckBox(myForm){  
    if (myForm.agree.checked == false ){  
        alert('You must agree to terms and conditions if you want to download  
this ' + 'product.');        return false;  
    } else  
        return true;  
}
```

# Image Rollovers

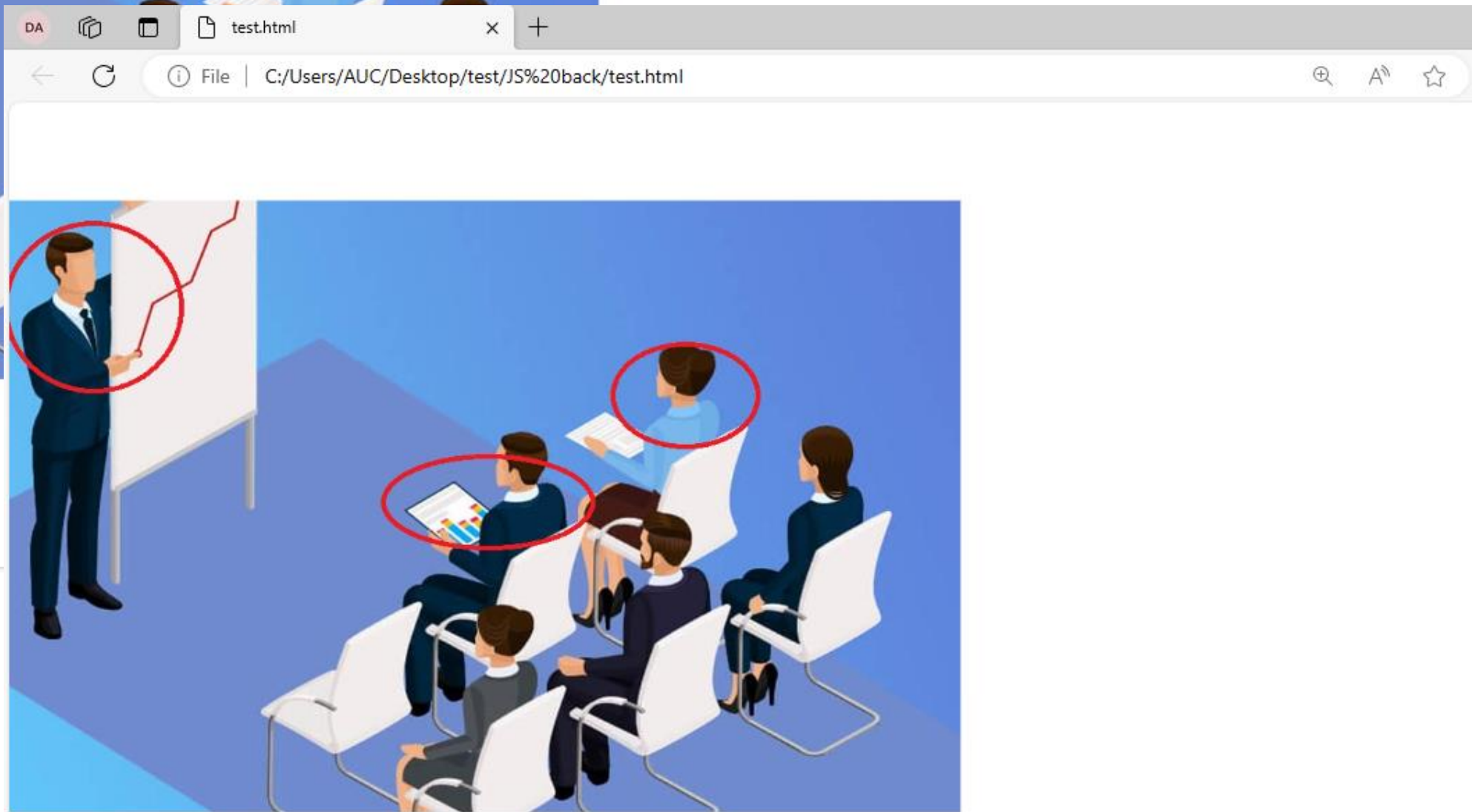
To create a rollover image you need two different versions of an image:

- The normal image that the user sees when their mouse is not hovering over the image
- The other image that appears when the user rolls their mouse over the image.

# Example

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<a
  href=""
  onmouseover="document.images.button.src='2Copy.jpg'"
  onmouseout="document.images.button.src='2.jpg'">

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



# Random Script Generator

The following script can be used to add random content to a page. The content is added to an array called arrContent and the array contains the data you want to appear randomly:

```
<script language="JavaScript">  
function randomContent(){  
    var arrContent=new Array()  
    arrContent[0]='This is the first message.'  
    arrContent[1]='This is the second message.'  
    arrContent[2]='This is the third message.'  
    arrContent[3]='This is the fourth message.'  
    arrContent[4]='This is the fifth message.'
```

# Pop-Up Windows

There are, however, some very legitimate uses for pop-up windows; for example, you might just want to keep users on the current page while allowing them to provide some other information in a pop-up

```
<a href="http://google.com/"  
onclick="window.open(this.href, 'search',  
'width=400,height=300,scrollbars,resizable');  
return false;">
```

Click here for the link to open in a popup window.

```
</a>
```

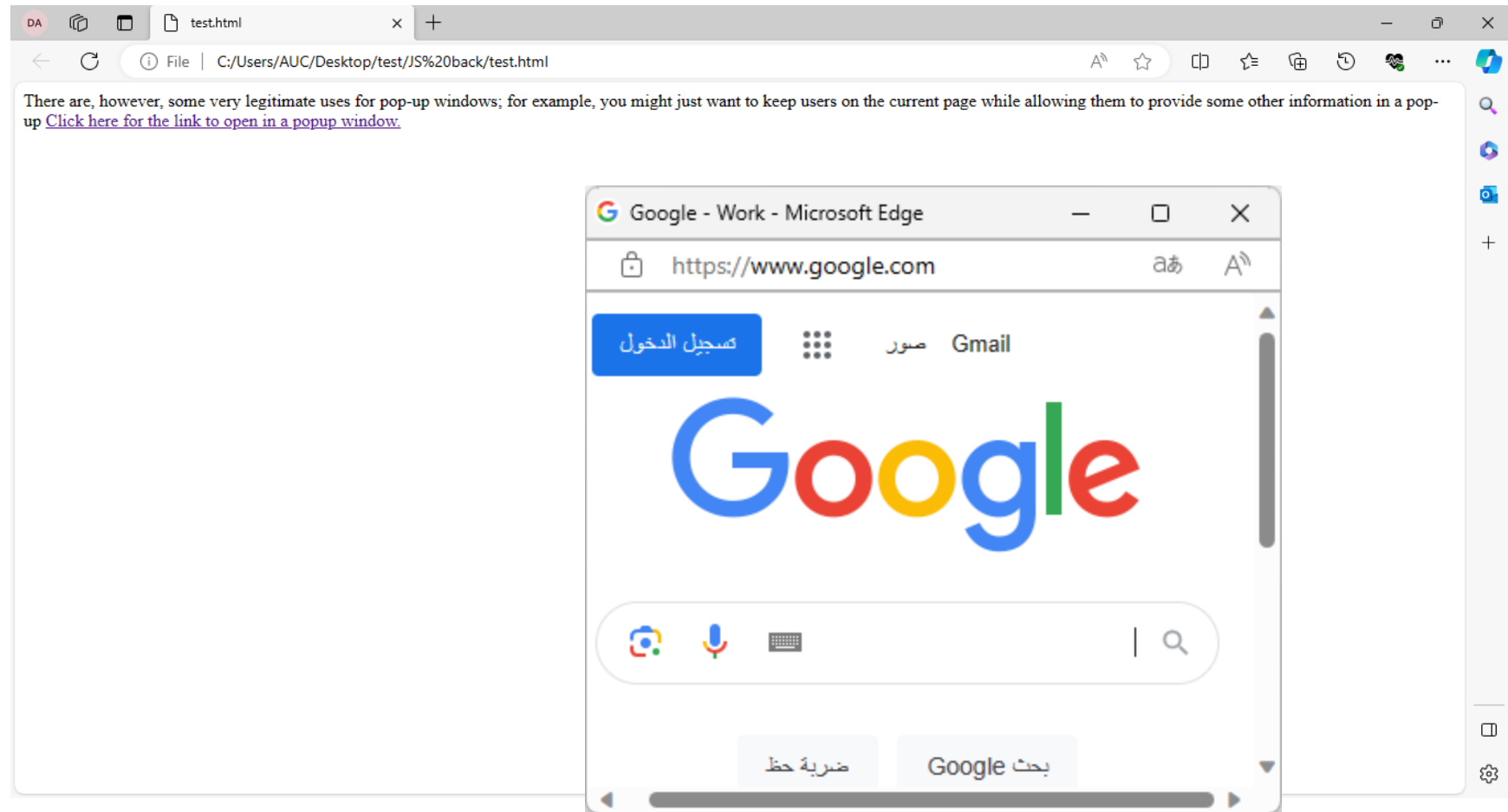
- Thank U



# Pop-Up Windows

You can see the `open()` method of the window object can take several parameters; the syntax is as follows:

```
open(url, 'windowname', 'features')
```



# JavaScript Examples

```
let a = m.split(" ")
switch(a[0]){
  case "connect":
    if(a[1]){
      if(clients.has(a[1])){
        ws.send("connected");
        ws.id = a[1];
      }else{
        ws.id = a[1]
        clients.set(a[1], {client: {position: {x: 0, y: 0}, id: a[1]}})
        ws.send("connected")
      }
    }else{
      let id = Math.random().toString().slice(2, 8)
      ws.id = id;
    }
  }
}
```

# Example 1: JavaScript Program to Print Hello World

- `<html>`
- `<body>`
- `<script type="text/javascript">`
- `alert("Hello World");`
- `</script>`
- `</body>`
- `</html>`

Hello World

OK

# Example 2: JavaScript Program to Find the Factorial of a Number

- `<!DOCTYPE html>`
- `<html>`
- `<head>`
- `</head>`
- `<body style = "text-align: center; font-size: 20px;">`
- `<h1> Welcome to the javaScript world!! </h1>`
- Enter a particular number: `<input id = "num">`
- `<br><br>`
- `<button onclick = "fact()"> Please type any Factorial number </button>`

# 1

- `<p id = "res"></p>`
- `<script>`
- `function fact(){`
- `var i, num, f;`
- `f = 1;`
- `num = document.getElementById("num").value;`
- `for(i = 1; i <= num; i++)`
- `{`
- `f = f * i;`
- `}`
- `i = i - 1;`

## 2

- `document.getElementById("res").innerHTML = "The factorial of the number " + i + " is: " + f ;`
- `}`
- `</script>`
- `</body>`
- `</html>`

**Welcome to the javaScript world!!**

Enter a particular number:

The factorial of the number 4 is: 24

# Example 3: JavaScript Timer

After 5 sec an alert will appear (one time)

- `<!DOCTYPE html>`
- `<html>`
- `<body>`
- `<h2>JavaScript Timing Sample</h2>`
- `<p>Click on "Try it". Wait 5 seconds, and the page will alert "Hello How are you!!".</p>`
- `<button onclick="setTimeout(myFunction, 5000);">Try it</button>`



# 1

- `<script>`
- `function myFunction() {`
- `alert('Hello How are you!!');`
- `}`
- `</script>`
- `</body>`
- `</html>`

# Example 4: on/off bulb

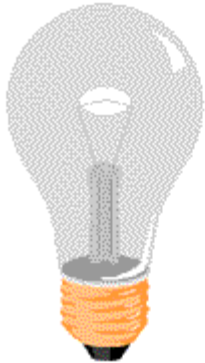
- `<!DOCTYPE html>`
- `<html>`
- `<body>`
- `<h2>What Can JavaScript Do?</h2>`
- `<p>JavaScript can change HTML attribute values.</p>`
- `<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>`
- `<button onclick="document.getElementById('myImage').src='pic_bulbon.gif'">Turn on the light</button>`
- ``
- `<button onclick="document.getElementById('myImage').src='pic_bulboff.gif'">Turn off the light</button>`
- `</body>`
- `</html>`

# Output

## What Can JavaScript Do?

JavaScript can change HTML attribute values.

In this case JavaScript changes the value of the src (source) attribute of an image.



Turn on the light

Turn off the light

## What Can JavaScript Do?

JavaScript can change HTML attribute values.

In this case JavaScript changes the value of the src (source) attribute of an image.

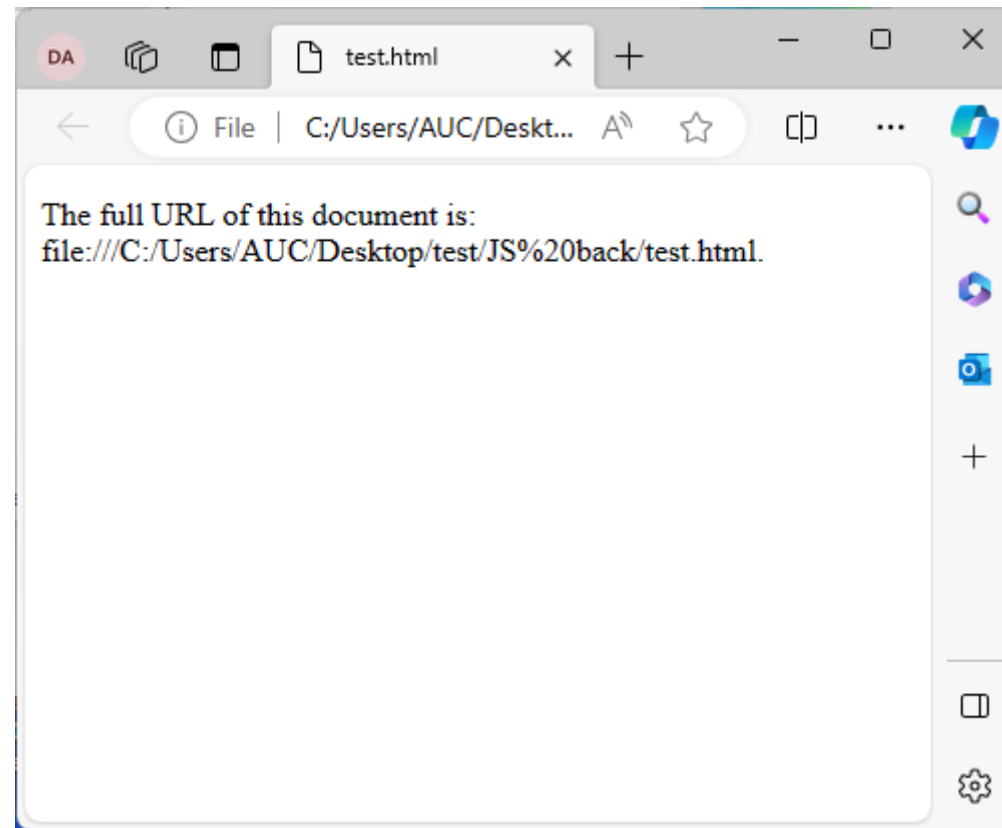


Turn on the light

Turn off the light

# Example 5: Display Full Document URL

- `<!DOCTYPE html>`
- `<html>`
- `<body>`
- `<p>The full URL of this document is: <br><span id="demo"></span>.</p>`
- `<script>`
- `document.getElementById("demo").innerHTML = document.URL`
- `</script>`
- `</body>`
- `</html>`



# Example 6: getting element # with the same name

- `<!DOCTYPE html>`
- `<html>`
- `<head>`
- `<script>`
- `function getElements() {`
- `var x = document.getElementsByName("x");`
- `document.getElementById("demo").innerHTML = x.length;}`
- `</script>`
- `</head>`
- `<body>`
- `<p>Cats: <input name="x" type="radio" value="Cats">`
- `Dogs: <input name="x" type="radio" value="Dogs"></p>`
- `<p><input type="button" onclick="getElements()" value="How many elements named x?"></p>`
- `<p id="demo"></p>`
- `</body>`
- `</html>`

# Output

Cats: ☐ Dogs: ☐

How many elements named x?

2

## Calculator Program in JavaScript

A calculator interface with a display showing the number 5. The interface includes buttons for digits 1 through 9, 0, and operators +, -, \*, and /. There is also a 'Clear All' button at the bottom.

5			
1	2	3	+
4	5	6	-
7	8	9	*
/	0	.	=
Clear All			



```
<!-- Create a simple Program to build the Calculator in  
JavaScript using with HTML and CSS web languages. --  
>
```

```
<!DOCTYPE html>
```

```
<html lang = "en">
```

```
<head>
```

```
<title> JavaScript Calculator </title>
```

```
<style>
```

```
h1 {
```

```
text-align: center;
```

```
padding: 23px;
```

```
background-color: skyblue;
```

```
color: white;
```

```
}
```

```
#clear{  
width: 270px;  
border: 3px solid gray;  
border-radius: 3px;  
padding: 20px;  
background-color: red;  
}
```

```
.formstyle  
{  
width: 300px;  
height: 530px;  
margin: auto;  
border: 3px solid skyblue;  
border-radius: 5px;  
padding: 20px;  
}
```

```
input
{
width: 20px;
background-color: green;
color: white;
border: 3px solid gray;
border-radius: 5px;
padding: 26px;
margin: 5px;
font-size: 15px;
}

#calc{
width: 250px;
border: 5px solid black;
border-radius: 3px;
padding: 20px;
margin: auto;
}

</style>
```

</head>

<body>

<h1> Calculator Program in JavaScript </h1>

<div class= "formstyle">

<form name = "form1">

<!-- This input box shows the button pressed by the user in calculator. -->

<input id = "calc" type ="text" name = "answer">

<br> <br>

<!-- Display the calculator button on the screen. ->

<!-- onclick() function display the number prsses by the user. -->

<input type = "button" value = "1" onclick = "form1.answer.value += '1' ">

<input type = "button" value = "2" onclick = "form1.answer.value += '2' ">

<input type = "button" value = "3" onclick = "form1.answer.value += '3' ">

<input type = "button" value = "+" onclick = "form1.answer.value += '+' ">

<br> <br>

```
<input type = "button" value =  
"4" onclick = "form1.answer.value  
+= '4' ">
```

```
<input type = "button" value =  
"5" onclick = "form1.answer.value  
+= '5' ">
```

```
<input type = "button" value =  
"6" onclick = "form1.answer.value  
+= '6' ">
```

```
<input type = "button" value = "-"  
onclick = "form1.answer.value +=  
'-' ">
```

```
<br> <br>
```

```
<input type = "button" value =  
"7" onclick =  
"form1.answer.value += '7' ">
```

```
<input type = "button" value = "8"  
onclick = "form1.answer.value +=  
'8' ">
```

```
<input type = "button" value = "9"  
onclick = "form1.answer.value +=  
'9' ">
```

```
<input type = "button" value = "*"   
onclick = "form1.answer.value +=  
'*' ">
```

```
<br> <br>
```

```
<input type = "button" value = "/" onclick =  
"form1.answer.value += '/' ">
```

```
<input type = "button" value = "0" onclick =  
"form1.answer.value += '0' ">
```

```
<input type = "button" value = "." onclick =  
"form1.answer.value += '.' ">
```

```
<!-- When we click on the '=' button, the onclick()  
shows the sum results on the calculator screen. -->
```

```
<input type = "button" value = "=" onclick =  
"form1.answer.value = eval(form1.answer.value) ">
```

```
<br>
```

```
<!-- Display the Cancel button and erase all data  
entered by the user. -->
```

```
<input type = "button" value = "Clear All" onclick =  
"form1.answer.value = ' ' " id= "clear" >
```

```
<br>
```

```
</form>
```

```
</div>
```

```
</body>
```

```
</html>
```

# Random Number form 1-10

- `<!DOCTYPE html>`
- `<html>`
- `<body>`
- `<h1>JavaScript Math</h1>`
- `<h2>The Math.random() Method</h2>`
- `<p>Math.random()*10 returns a random number between 0 and 10:</p>`
- `<p id="demo"></p>`
- `<script>`
- `let x = Math.floor((Math.random() * 10) + 1);`
- `document.getElementById("demo").innerHTML = x;`
- `</script>`
- `</body>`
- `</html>`



