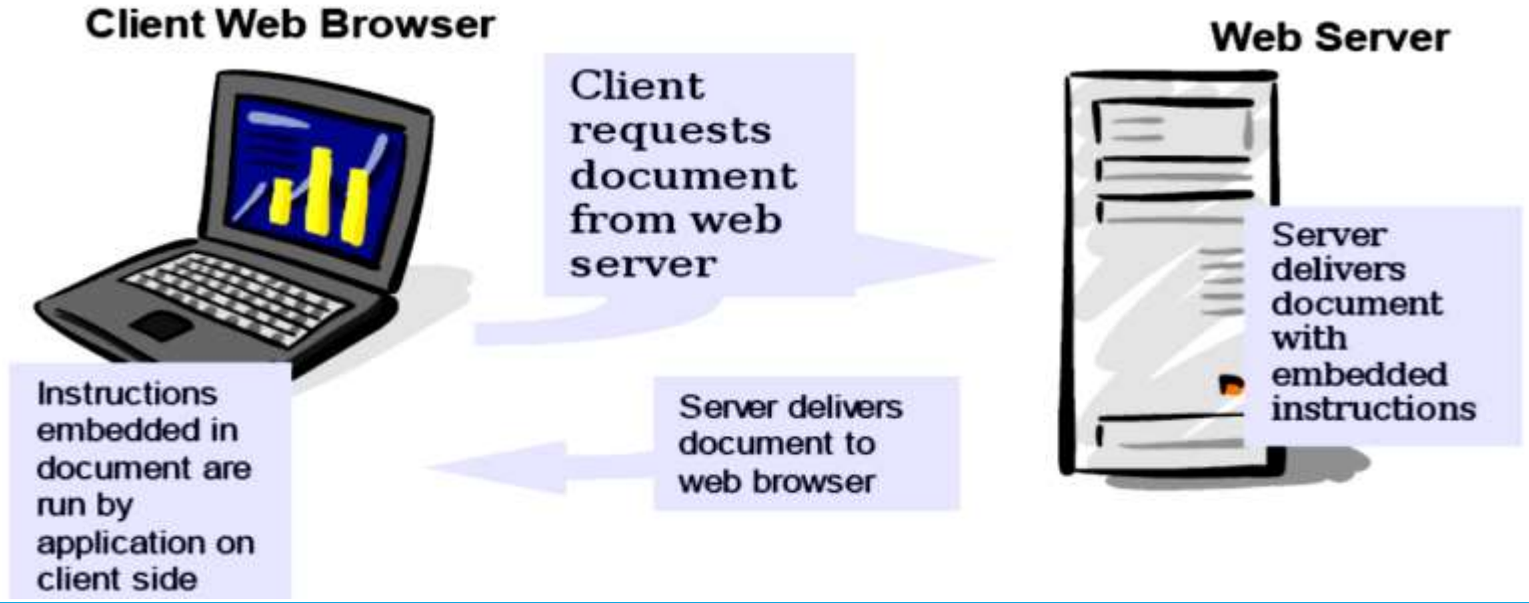
Introduction to JavaScript

Web Programming

- Programming for the World Wide Web involves
  - Server-side programming
  - Client-side (browser-side) programming


Client Web Browser
Web Server
Client requests document from web server
Server delivers document with embedded instructions
Server delivers document to web browser
Instructions embedded in document are run by application on client side

What is JavaScript?

- JavaScript is used to program the behaviour of web pages (performing dynamic tasks).
- Javascript are scripts (code) that is executed on the client’s browser instead of the web-server (Client-side scripts).

Why we need client side programming

- The user’s actions will result in an immediate response because they don't require a trip to the server.
  - Allows the creation of faster and more responsive web applications.
  - Make web pages more interactive
- Fewer resources are used and needed on the web-server.

Characteristics of JavaScript

Object-based

- Doesn't support all the features of OOPs like Polymorphism and Inheritance

Event-Driven

- Much of the JavaScript code you write will be in response to events generated by the user or the system.

Browser-Dependent

- JavaScript depends on the Web browser to support it. If the browser does not support it, your code will be ignored. Even worse, the JavaScript code itself may be displayed as text on your page.

Characteristics of JavaScript (cont.)

Interpreted language

- JavaScript is interpreted at runtime by the browser before it is executed. JavaScript is not compiled into an actual program like an .EXE file but remains part of the HTML document to which it is attached.

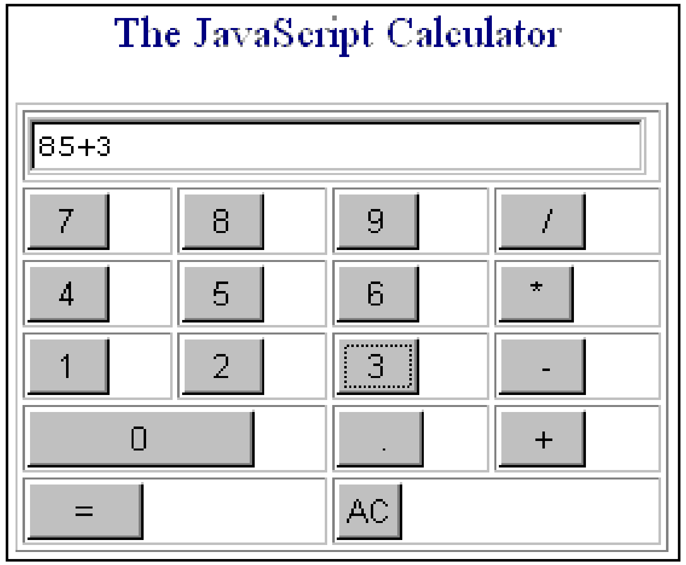
Dynamic

- You can declare variables of a specific type, but you do not need to

Case sensitive

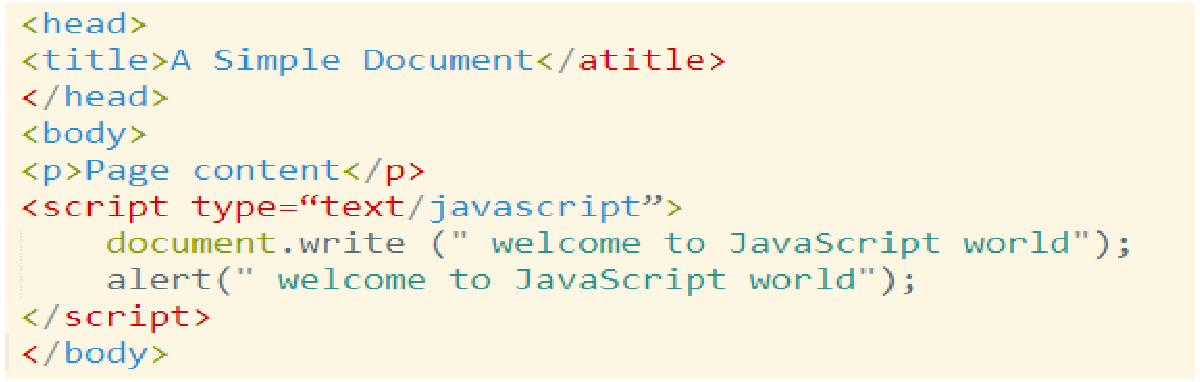
What Javascript can do?

- Javascript can change HTML Content
- Javascript can change HTML Attributes
- Javascript can change HTML Styles (CSS)
- Javascript can Validate Data
- Javascript can Make Calculations



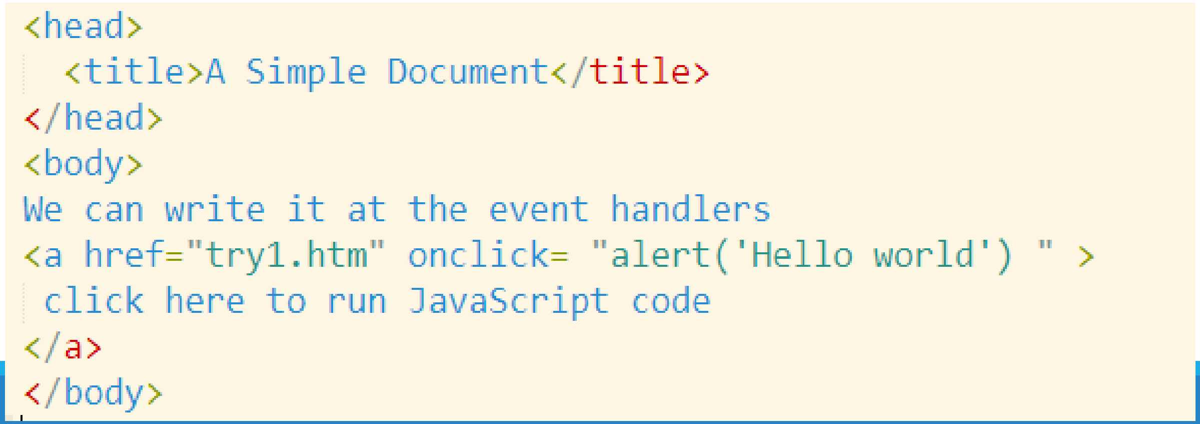
Embedding JavaScript in HTML

1. Anywhere in the html file between <script></script> tags.



```
<head>
<title>A Simple Document</atitle>
</head>
<body>
<p>Page content</p>
<script type="text/javascript">
    document.write (" welcome to JavaScript world");
    alert(" welcome to JavaScript world");
</script>
</body>
```

2. As the value of the event handler attributes.



```
<head>
  <title>A Simple Document</title>
</head>
<body>
We can write it at the event handlers
<a href="try1.htm" onclick= "alert('Hello world') " >
 click here to run JavaScript code
</a>
</body>
```

Embedding JavaScript in HTML

3. In an external file and refer to it using the SRC attribute.

```
<head>
   <title>A Simple Document</title>
   <script src= "MyJavascripFile.js"></script>
</head>
<body>
We can refer to JavaScript statements in another file.
</body>
```

Note:

Keeping all code in one place, is always a good habit.

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an alert box, using **window.alert()**, **window.prompt()**, **window.confirm()**.

- Writing into an HTML element, using **innerHTML**.

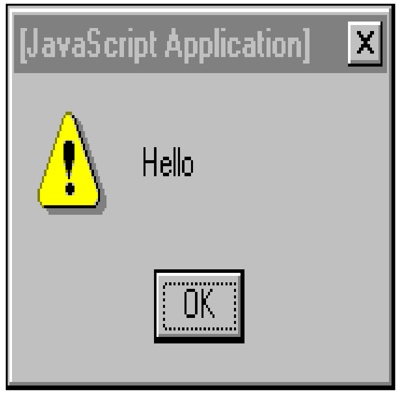
Alerts

```
<html>
<head><title>My Page</title></head>
<body>
<p>
<a href="myfile.html">My Page</a>
<br />
<a href="myfile.html"

                    onMouseover="window.alert('Hello');">
My Page</A>
</p>
</body>
</html>
```

An Event

JavaScript written inside HTML


[JavaScript Application]

Hello

Prompts and Confirm

- Prompts :The return is the data the user entered

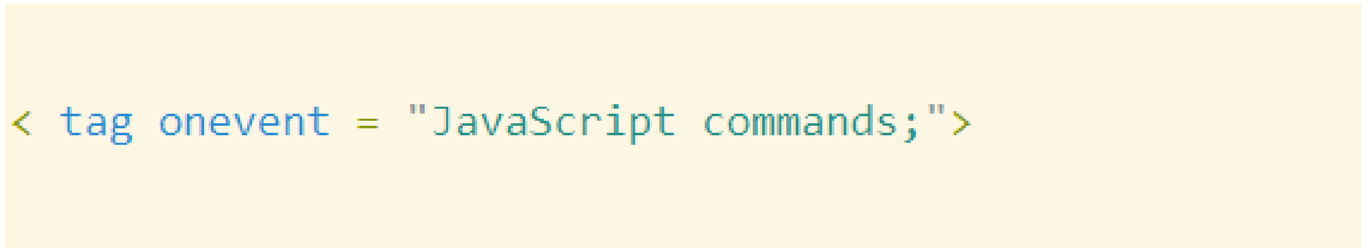
```
<script type="text/javascript">
window.prompt(‘Message', 'Initial Value’);
</script>
```

- The confirm returns true and false

```
<script type="text/javascript">
window.confirm(‘Message');
</script>
```

Events

- Event handlers are created as attributes added to the HTML tags in which the event is triggered.
- An Event handler adopts the event name and appends the word "on" in front of it.



```
< tag onevent = "JavaScript commands;">
```

- Thus the "click" event becomes the onclick event handler.

Mouse Events

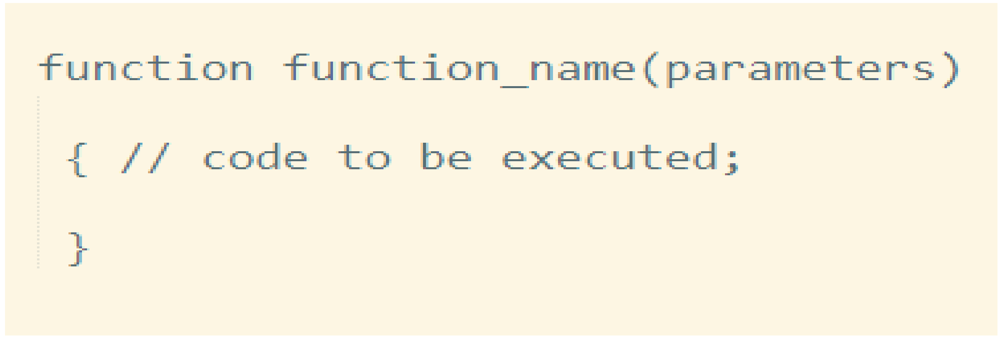
Event handler	Description
onmousedown	when pressing any of the mouse buttons.
onmousemove	when the user moves the mouse pointer within an element.
onmouseout	when moving the mouse pointer out of an element.
onmouseup	when the user releases any mouse button pressed
onmouseover	when the user moves the mouse pointer over an element.
onclick	when clicking the left mouse button on an element.
ondblclick	when Double-clicking the left mouse button on an element.
ondragstart	When the user has begun to select an element

Keyboard Events

Event handler	Description
onkeydown	When User holds down a key
onkeypress	When User presses a key
onkeyup	When User releases the pressed a key

JavaScript Functions

- A JavaScript function is a block of code designed to perform a particular task.



```
function function_name(parameters)
 { // code to be executed;
 }
```

- A function is executed when "something" invokes it (calls it).
  - When an event occurs (when a user clicks a button)
  - When it is invoked (called) from JavaScript code
  - Automatically (self invoked)

Function Invocation types

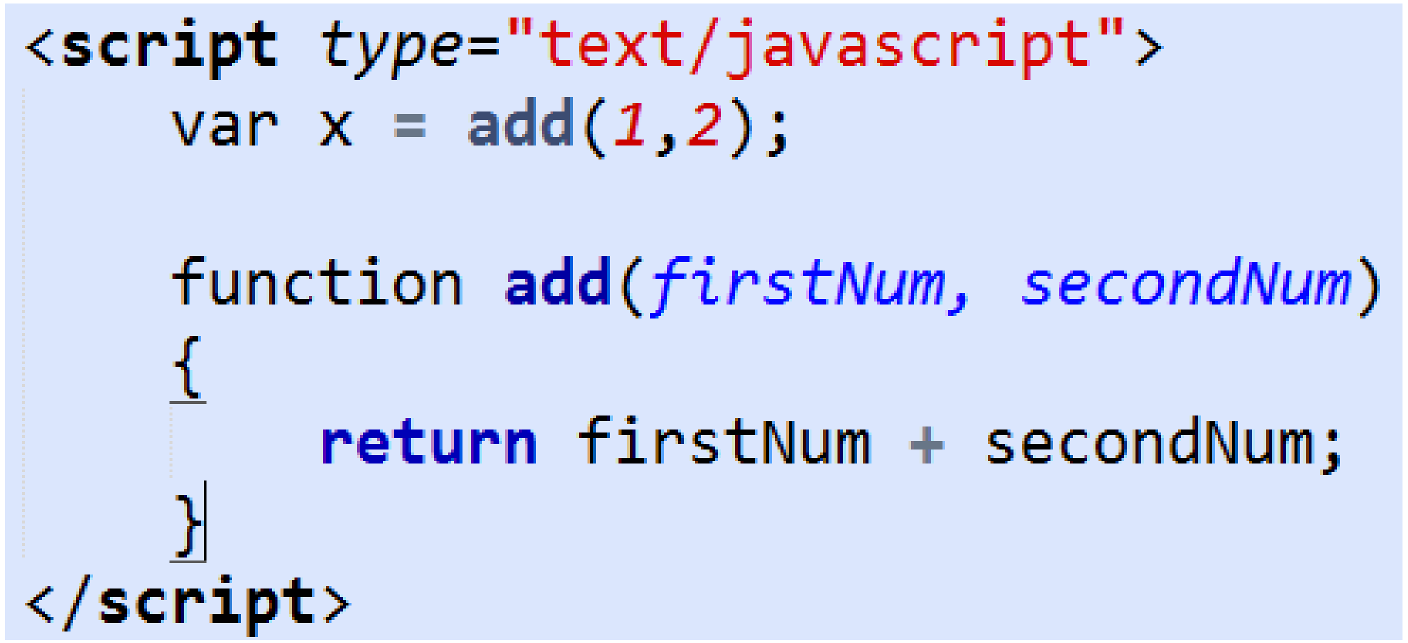
- When an event occurs (e.g. when a user clicks a button)

```html
<button type="button" onclick="call_me()">
    Click to call the function
</button>
```

```html
<script type="text/javascript">
    function call_me()
    {
        alert("The function is called");
    }
</script>
```

Function Invocation types(cont.)

- When it is invoked (called) from JavaScript code

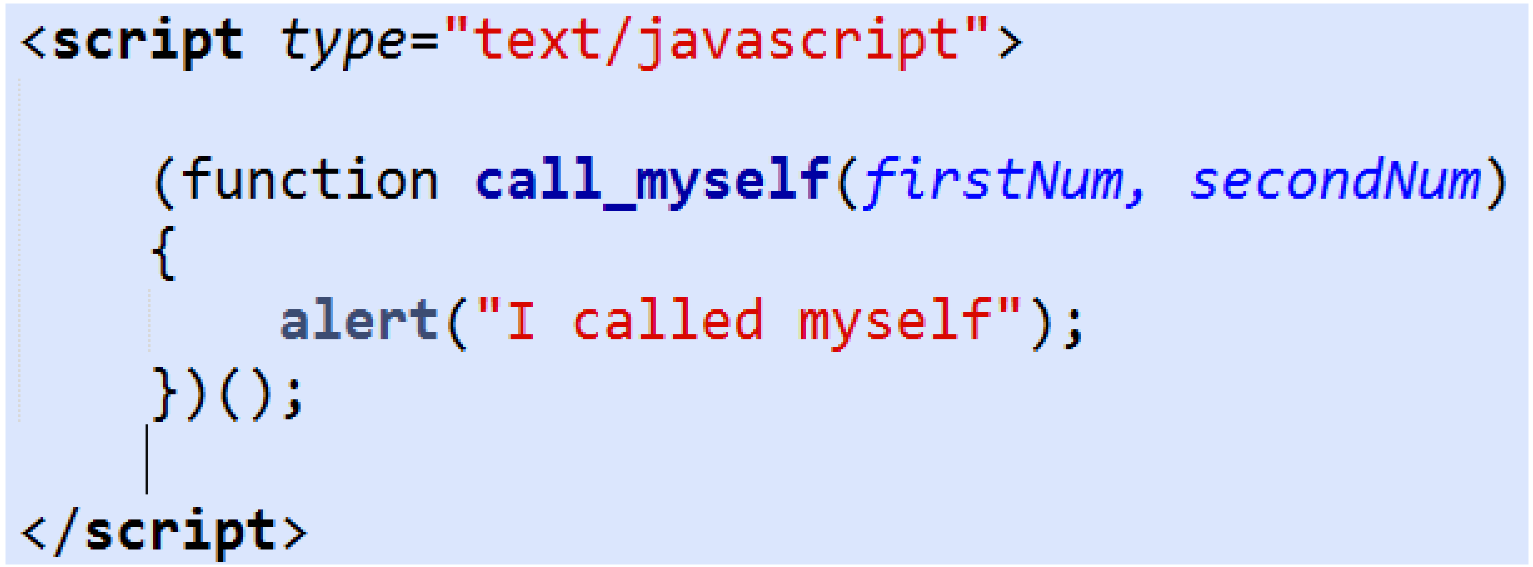


```
<script type="text/javascript">
    var x = add(1,2);

    function add(firstNum, secondNum)
    {
        return firstNum + secondNum;
    }
</script>
```

Function Invocation types (cont.)

- Automatically (self invoked)



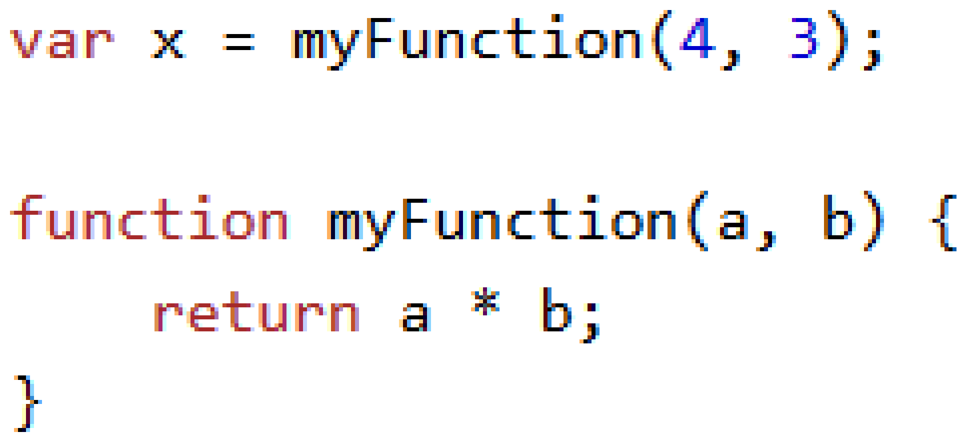
```
<script type="text/javascript">

    (function call_myself(firstNum, secondNum)
    {
        alert("I called myself");
    })();

</script>
```

Function Return

- When JavaScript reaches a **return statement**, the function will stop executing and returns some value to the invoker.
- If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.



```
var x = myFunction(4, 3);

function myFunction(a, b) {
    return a * b;
}
```

JavaScript Built-in functions

Name	Example

parseInt()	parseInt("3") //returns 3
	parseInt(“3a”) //returns 3
	parseInt(“a3”) //returns NaN
	parseInt("110", 2)// returns 6
	parseInt("0xD9", 16)// returns 217

parseFloat()	parseFloat("3.55") //returns 3.55
	parseFloat(“3.55a”) //returns 3.55
	parseFloat(“a3.55”) //returns NaN

Number()	myVar = new Boolean("true") document.write(Number(myVar)) // returns 1
String()	myVar = new Boolean(0) document.write(String(myVar)) // returns false

JavaScript Built-in functions

Name	Description	Example

isFinite(num)
(used to test number)
	returns true if the number is finite, else false	document.write(isFinite(2.2345))
		//returns true
		document.write(isFinite(“Hello”))
		//returns false

isNaN(val)
(used to test value)
	validate the argument for a number and returns true if the given value is not a number else returns false.	document.write(isNaN(0/0))
		//returns true
		document.write(isNaN("348"))
		//returns false

eval(expression)	evaluates an expression and returns the result.	f=999; w=777;
		document.write(eval(f + w));
		// returns 1776

JavaScript Built-in functions

Name	Description	Example
escape(string)	method converts the special characters like space, colon etc. of the given string in to escape sequences.	escape("test val"); //test%20val
unescape(string)	function replaces the escape sequences with original values. e.g. %20 is the escape sequence for space " ".	unescape("test%20val"); //test val

JavaScript Primitive Value types

Value	Example
Number	Any numeric value (e.g., 3, 5.3, or 45e8)
String	Any string of alphanumeric characters (e.g., "Hello, World!", "555-1212" or "KA12V2B334")
Boolean	True or False values only

JavaScript Special Values

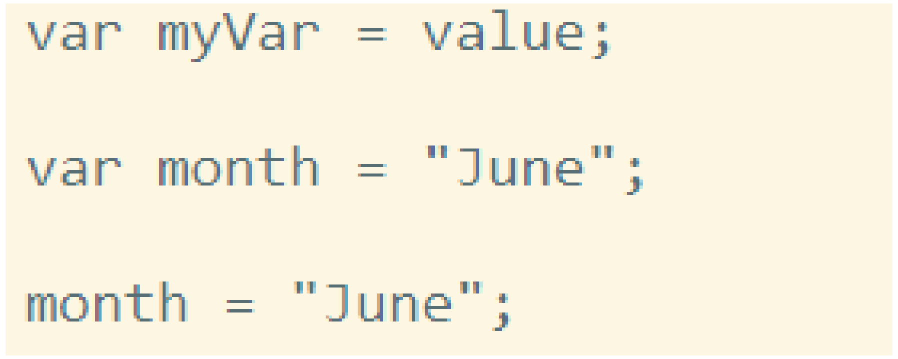
Value	Example
Null Eg: var x = null;	A special keyword for the null value (no value or empty variable) ** x has the type ‘object’
Undefined Eg: var x;	A special keyword means that a value hasn't even been assigned yet. ** x has the type ‘undefined’

JavaScript Variables

- Variables are containers that hold values.
- Variables are untyped
- The initial value for any variable is **undefined**.

```
var num;    //num = undefined
```

- While it is not technically necessary, variable declarations should begin with the keyword var.



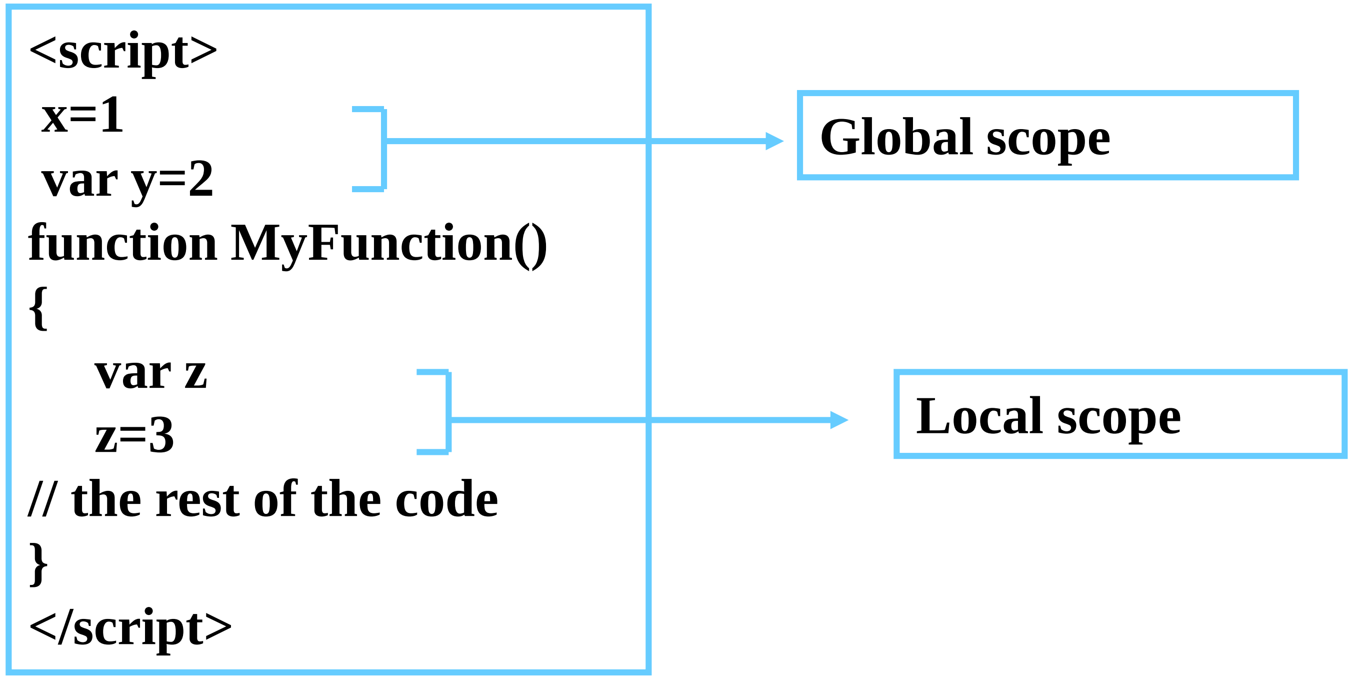
```
var myVar = value;

var month = "June";

month = "June";
```

JavaScript Variables Scope

- Global Scope
- Local Scope



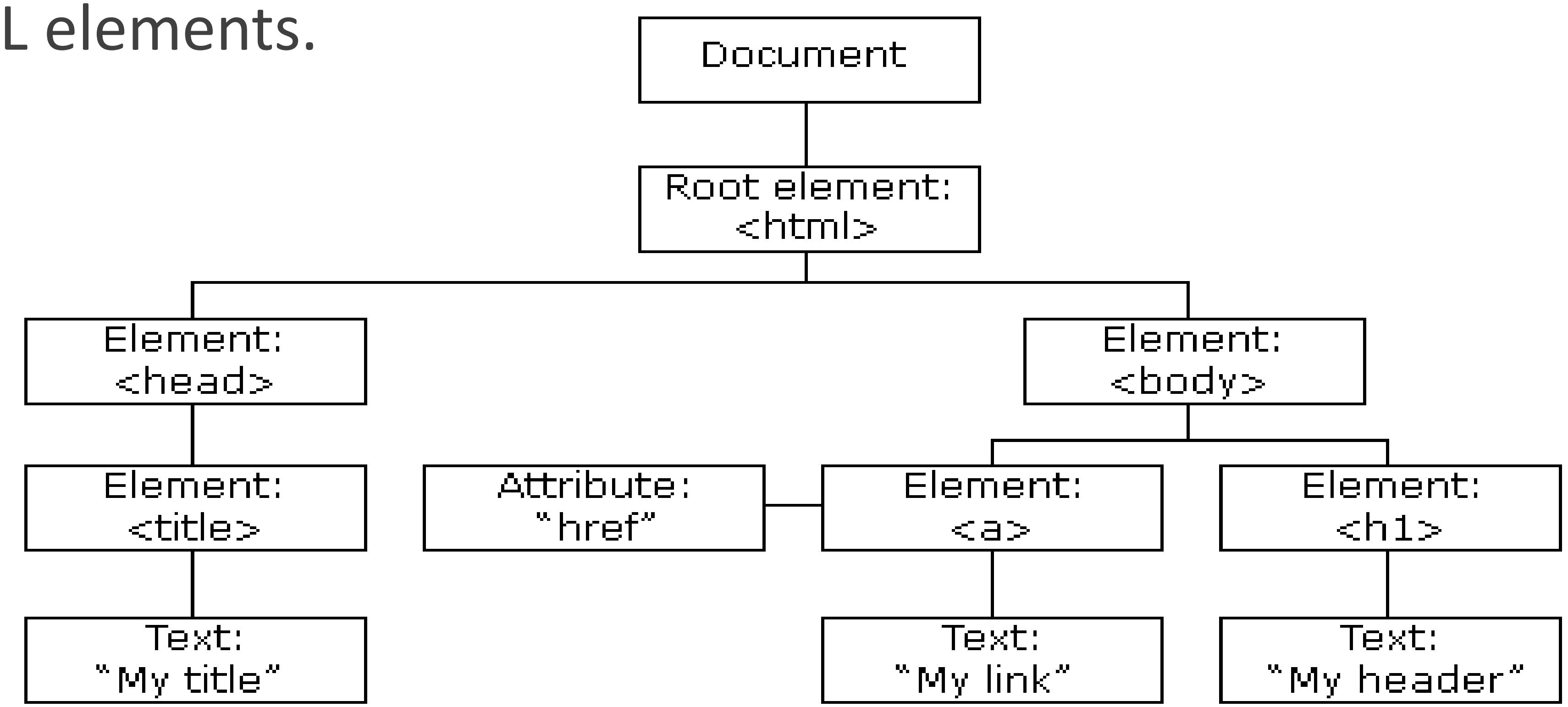
```
<script>
 x=1
 var y=2
function MyFunction()
{
    var z
    z=3
// the rest of the code
}
</script>
```

Controlling Program Flow

- Program flow is normally linear
- **Control Statements** can change the program flow
    1. Conditional Statements
        a. **if ….else**
        b. **switch/case**
    2. Loop Statements
        a. **for**
        b. **for..in** loops through the properties of an object
        c. **while**
        d. **do…while**

HTML DOM (Document Object Model)

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.



Finding HTML Elements

Method	Description
document.getElementById()	Find an element by element id
document.getElementsByTagName()	Find elements by tag name
document.getElementsByClassName()	Find elements by class name

Changing HTML elements

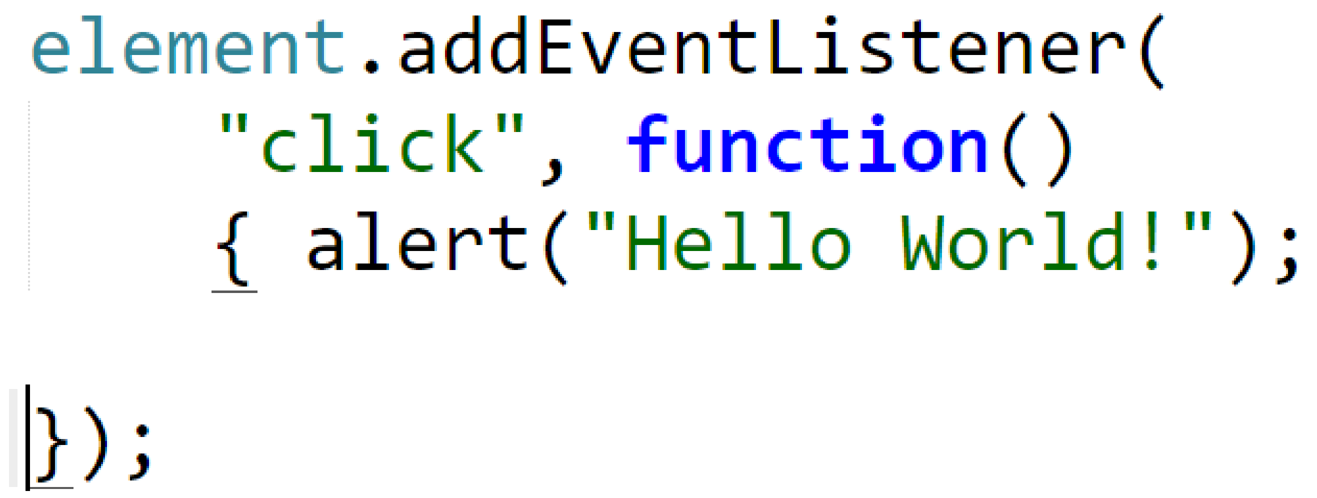
Method	**Description**
element.innerHTML=	Change the inner HTML of an element
element.attribute=	Change the attribute of an HTML element
element.setAttribute*(attribute,value)*	Change the attribute of an HTML element
element.style.*property=*	Change the style of an HTML element

Adding and deleting elements

Method	**Description**
document.createElement(...)	Create an HTML element
document.removeChild(...)	Remove an HTML element
document.appendChild(...)	Add an HTML element
document.replaceChild(...)	Replace an HTML element
document.write(*text*)	Write into the HTML output stream

The addEventListener() method

- Adds the function to an event attribute (onclick in this example) of the selected element



```
element.addEventListener(
    "click", function()
    { alert("Hello World!");

});
```

The addEventListener() method

- Another way is to add only the name of the function and define it separately in your scripts

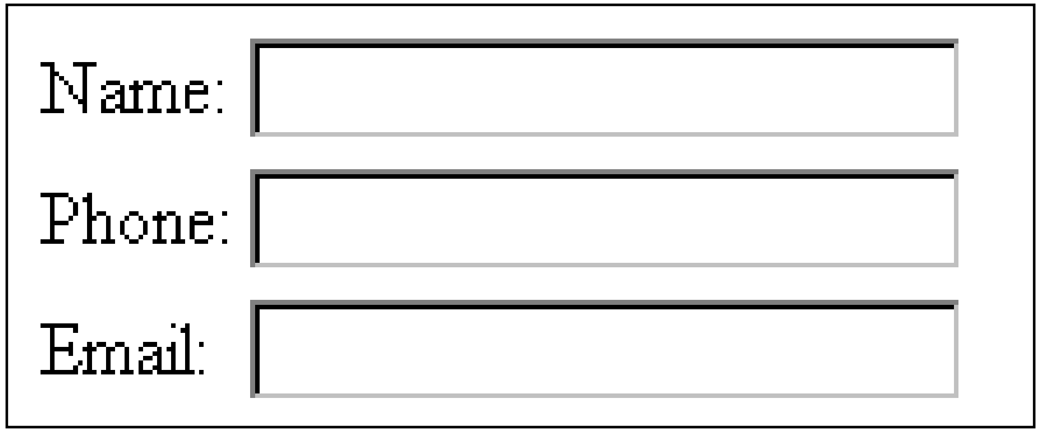
```
element.addEventListener("click", myFunction);


function myFunction() {
        alert ("Hello World!");
}
```

HTML Forms and JavaScript

- HTML `<form>` elements receive input
- JavaScript is very good at processing user input in the web browser
- Forms and form elements have unique names
  - Each unique element can be identified
  - Uses JavaScript Document Object Model (DOM)

Naming Form Elements in HTML



```
<form name="addressform">
Name:  <input name="yourname"><br />
Phone: <input name="phone"><br />
Email: <input name="email"><br />
</form>
```

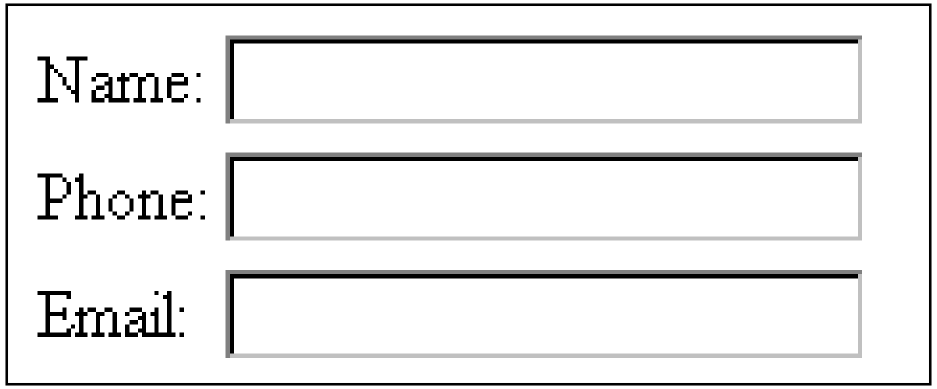
Using Form data

- To access the values of the form element use the following syntax:

*document.**formname.elementname.**value*

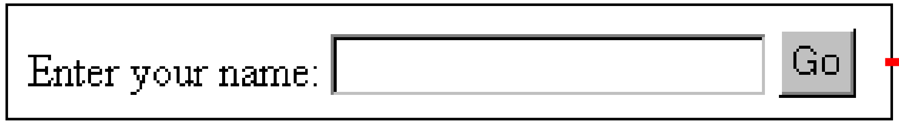
- Example

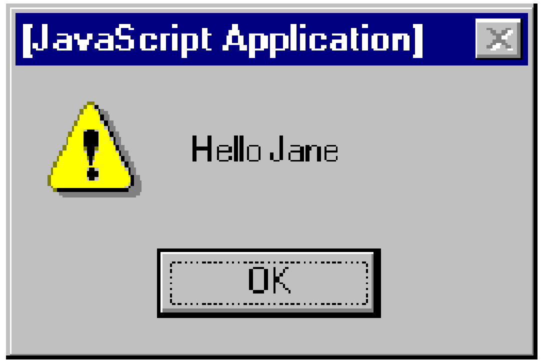
```
document.addressform.yourname.value
document.addressform.phone.value
document.addressform.email.value
```



Example

- Personalising an alert box


Enter your name:



Hello Jane

```
<form name="alertform">
Enter your name:
<input type="text" name="yourname">
<input type="button" value= "Go" onClick="window.alert('Hello ' + ▯ document.alertform.yourname.value);">
</form>
```

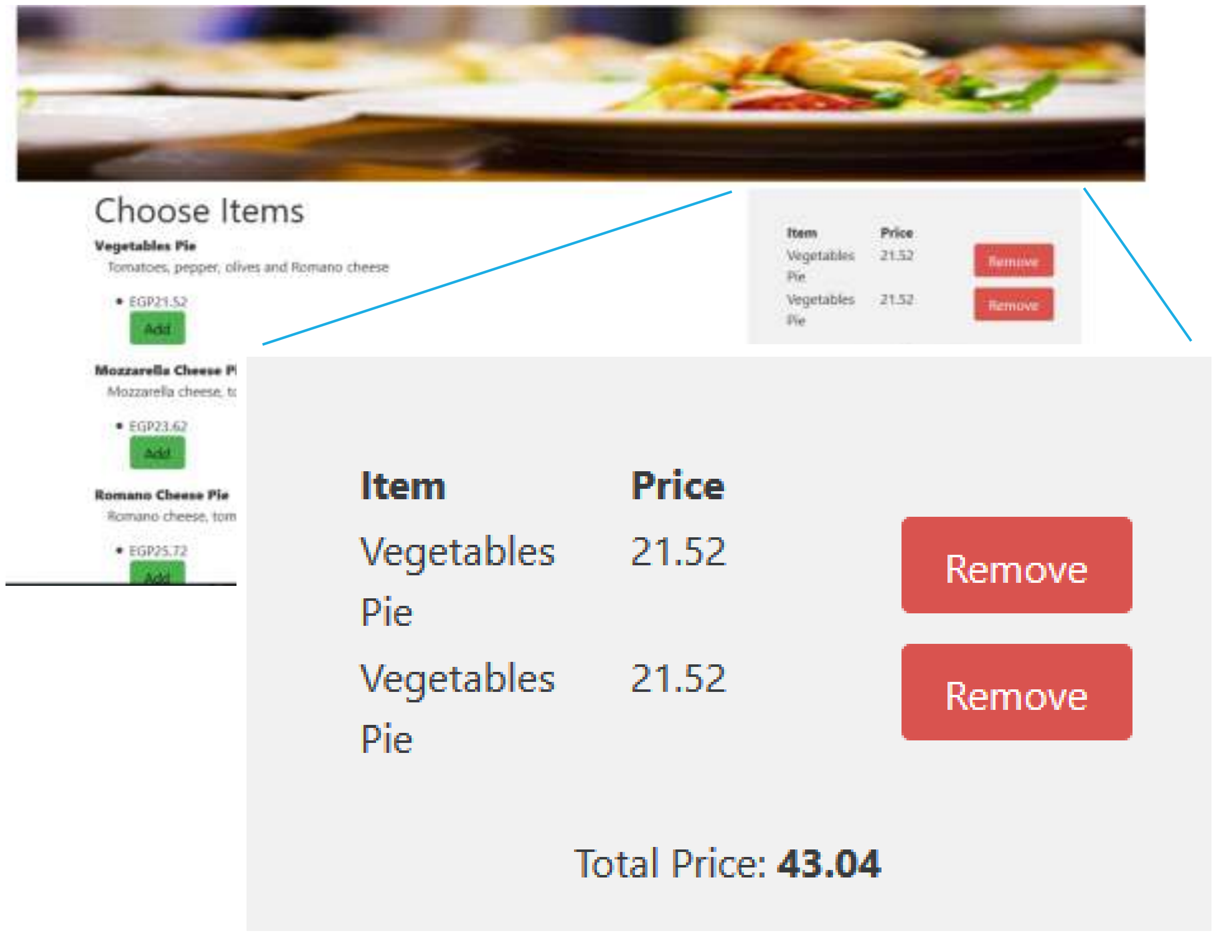
Assignment

Complete this course

- https://www.edx.org/course/javascript-intro
- Due date: 14-10-2017

Implement the following

- Implement the following cart:
- When the user clicks Add, the item name, price and a Remove button are added to the cart.
- When the user clicks Remove, the corresponding item is removed from the cart.
- There should be a Total Price that is affected when an item is added or removed.
- Due date: 14-10-2017



Choose Items

Vegetables Pie
Tomatoes, pepper, olives and Romano cheese
- EGP21.52

Add

Mozzarella Cheese Pi
Mozzarella cheese, to
- EGP23.62

Add

Romano Cheese Pie
Romano cheese, tom
- EGP25.72

Add

Item	Price	
Vegetables Pie	21.52	Remove
Vegetables Pie	21.52	Remove

Total Price: **43.04**