App Development & Modeling

BSc in Applied Computing

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit

Introduction to Javascript

Agenda

- Nature of Javascript
- Motivating Example
- The DOM
- Script Fragments in Detail:

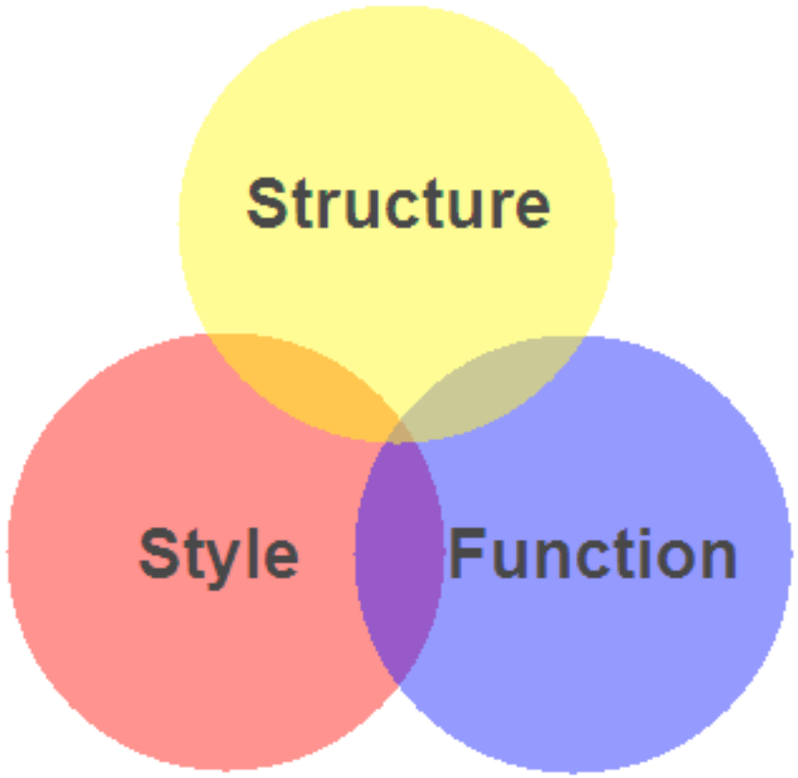
1. Functions
2. Variables & Operators
3. Date & Time
4. Events
5. Manipulating Lists

Agenda

- Nature of Javascript
- Motivating Example
- The DOM
- Script Fragments in Detail:

1. Functions
2. Variables & Operators
3. Date & Time
4. Events
5. Manipulating Lists

Structure of Client-Side Web



- Markup (HTML)
  - Structure
  - Content
- Style (CSS)
  - Style
  - Presentation
  - Appearance
- Function (Javascript)
  - Actions
  - Manipulations

Javascript

- JavaScript provides access to virtually all aspects of a page.
    - CSS properties
    - Markup content
    - Forms, communication to Server
- JavaScript is a scripting language most often used for client-side web development.
- It is a dynamic, weakly typed, prototype-based language with first-class functions.
- JavaScript was influenced by many languages and was designed to have a similar look to Java, but be easier for non-programmers to work with.

Javascript & Java: Similarities

- C-like syntax, the C language being their most immediate common ancestor language.
- Object-oriented, but with different approaches to classes
- Sandboxed - i.e. they run inside an all encompassing environment. For Java - the Java Virtual Machine, for Javascript, the Browser
- Available within browsers (Java requiring a browser plugin)
- JavaScript was designed with Java's syntax and standard library in mind - all Java keywords were reserved in original JavaScript.
- JavaScript's standard library follows Java's naming conventions, and JavaScript's Math and Date objects are based on classes from Java

Javascript & Java: Differences

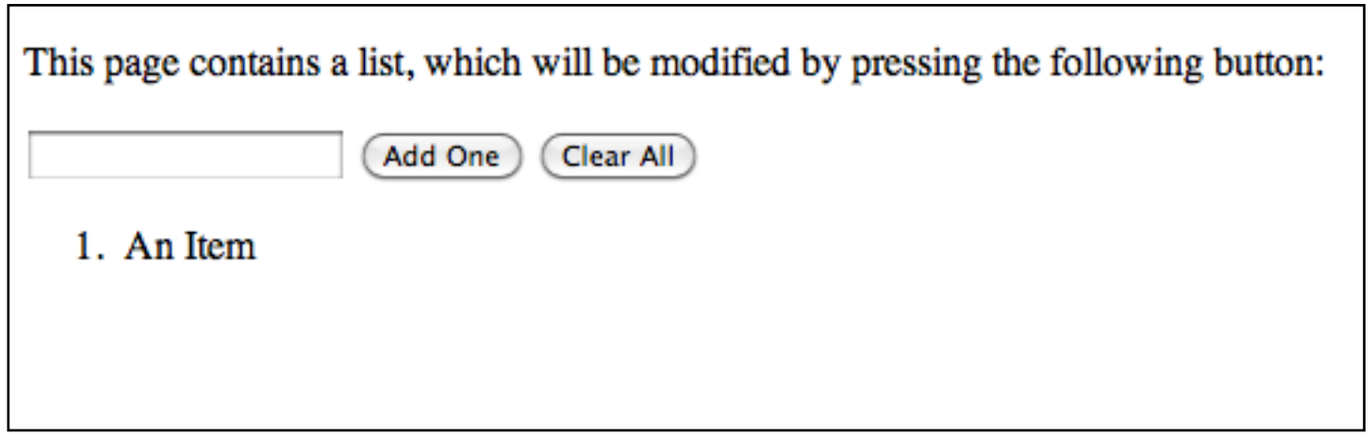
- Static vs Dynamic Typing:
  - Java has static typing - variable must be associated with specific type as soon as they are introduced
  - JavaScript's typing is dynamic - a variable can hold an object of any type and cannot be restricted
- Functional:
  - JavaScript also has many functional features based on the Scheme language, ultimately derived from the programming language LISP.
- Object Model
  - Java's objects are class-based;
  - JavaScript's objects are prototype-based.
- Java is loaded from compiled bytecode; JavaScript is loaded as human-readable source code.

Agenda

- Nature of Javascript
- Motivating Example
- The DOM
- Script Fragments in Detail:

1. Functions
2. Variables & Operators
3. Date & Time
4. Events
5. Manipulating Lists

Motivating Example



- This pages contains
  - A paragraph
  - 3 <input> elements
    - A Text Field
    - 2 Buttons
  - A list

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Changing the DOM</title>
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <p>
      This page contains a list, which will be modified by pressing the following button:
    </p>
    <input type="text" id="itemtext" />
    <input type="button" value="Add One" onclick="addElementById('itemtext')" />
    <input type="button" value="Clear All" onclick="clearList()" />
    <ol id="list">
      <li> An Item </li>
    </ol>
  </body>
</html>
```

Java Script Functions

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Changing the DOM</title>
    <script type="text/javascript" src="script.js"></script>
  </head>
```

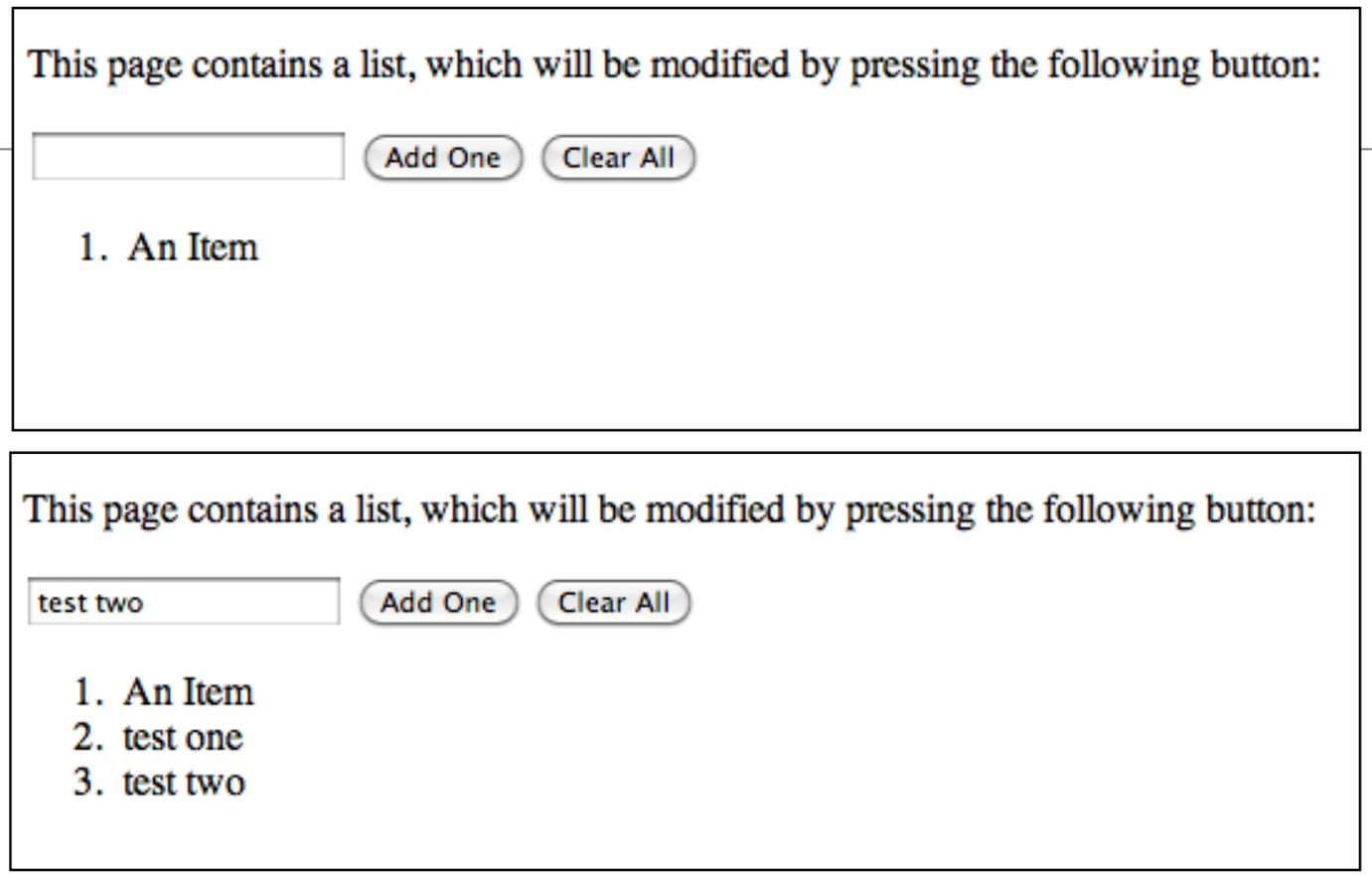
```
    <input type="button" value="Add One" onclick="addElementById('itemtext')" />
    <input type="button" value="Clear All" onclick="clearList()" />
```

- The script element identifies a file containing javascript functions
- The button elements identify the functions + parameters, to be called when the buttons are clicked
- The functions directly manipulate the DOM, changing the content of the current page

```
function addElementById(itemId)
{
  var list = document.getElementById('list');
  var itemText = document.getElementById(itemId);
  var newItem = document.createElement('li');
  newItem.innerHTML = itemText.value;
  list.appendChild(newItem);
}

function clearList()
{
  var list = document.getElementById('list');
  list.innerHTML = "";
}
```

script.js

Example



- For a static page, clicking on a link/button takes the browser to a new page (new url)

- With a dynamic page (javascript enabled), clicking on a button may change the *current* pages structure, content or style

Agenda

- Nature of Javascript
- Motivating Example
- The DOM
- Script Fragments in Detail:

1. Functions
2. Variables & Operators
3. Date & Time
4. Events
5. Manipulating Lists

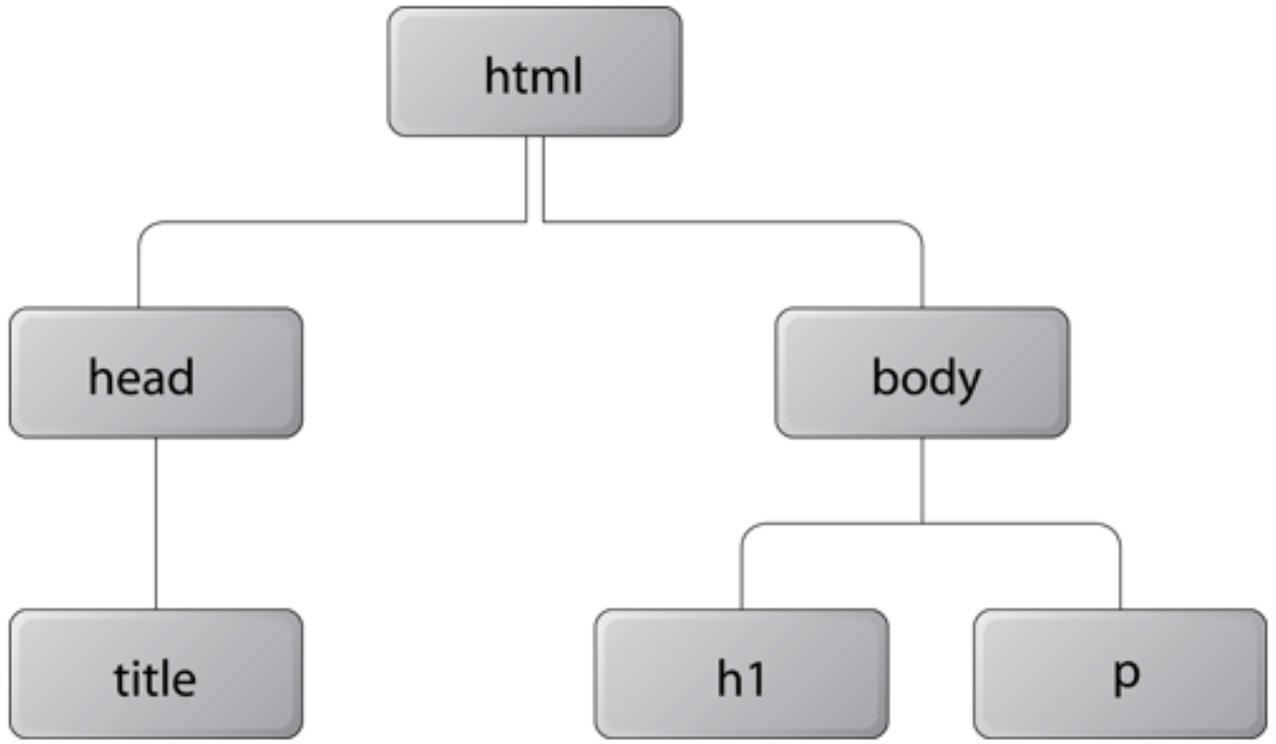
Parsing & Rendering

- A web browser typically reads and renders HTML documents in two phases:
  - the parsing phase
  - the rendering phase.
- During the parsing phase, the browser reads the markup in the document, breaks it down into components, and builds a Document Object Model (DOM) tree.
- The DOM is a in-memory data structure, typically traversed by Java Script code

DOM Tree

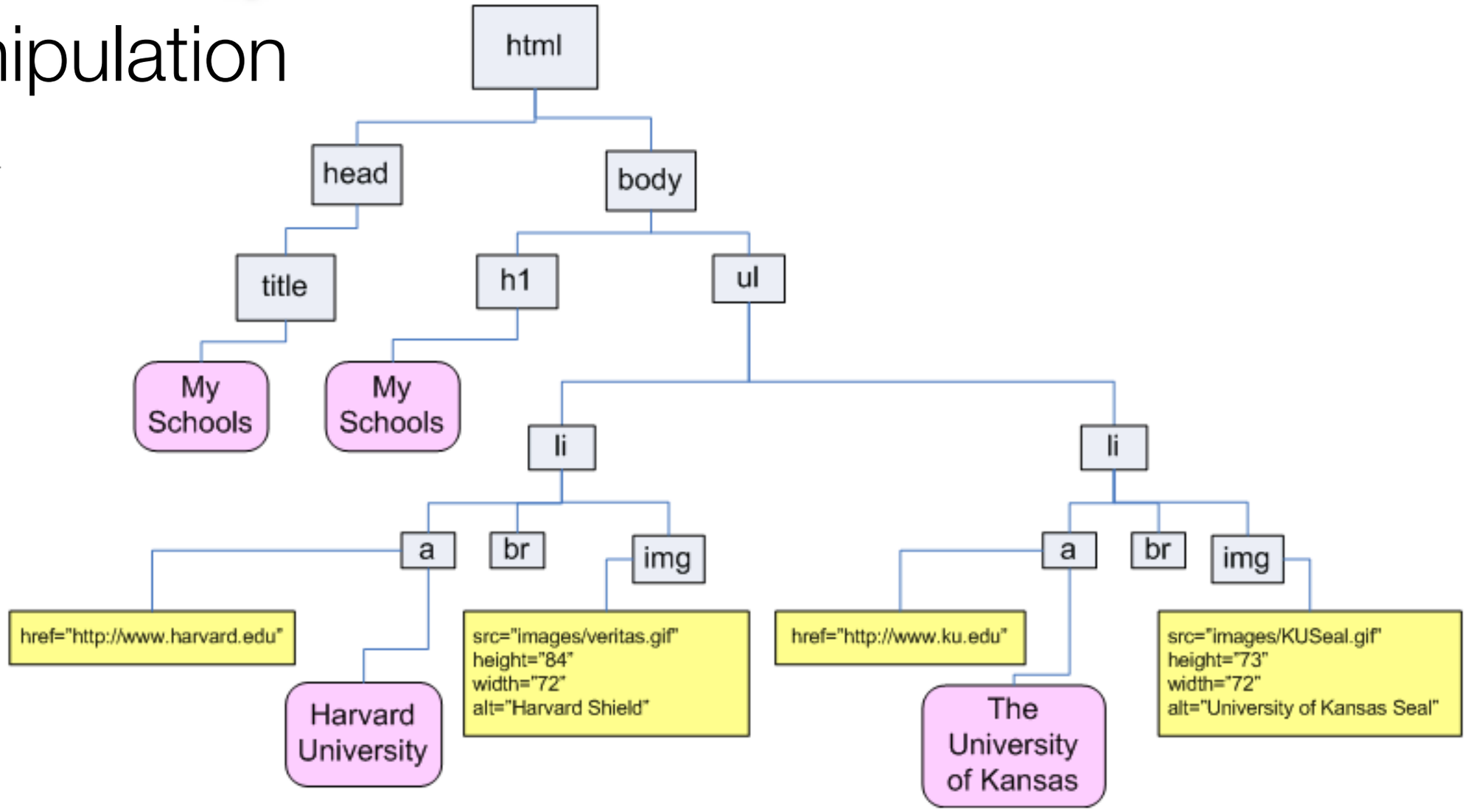
- Each object in the DOM tree is called a node.
- There are several types of nodes, including element nodes and text nodes.
- At the top of the tree is a document node, which contains an element node called the root node; this is always the html element in HTML documents.
- It branches into two child element nodes—head and body—which then branch into other children.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <title>Widgets</title>
  </head>
  <body>
    <h1>Widgets</h1>
    <p>Welcome to Widgets, the number one company
    in the world for selling widgets!</p>
  </body>
</html>
```




```
function clearList()
{
  var list = document.getElementById('list');
  list.innerHTML = "";
}
```

DOM Manipulation



- All aspects of the DOM - essentially a structured version of the html + the CSS loaded in the current page, can be accessed in javascript via the “document” object.

- If a script changes the DOM, the effects are immediately visible in the rendered document in the browser

Agenda

- Nature of Javascript
- Motivating Example
- The DOM
- Script Fragments in Detail:

1. Functions
2. Variables & Operators
3. Date & Time
4. Events
5. Manipulating Lists

Script Location

- Where a script can be provided, 3 options available

- External Script

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

- Script within HTML document

```
<script type="text/javascript">
 /*
   JavaScript code as content of script element
 */
</script>
```

- "Inline" scripts as values of event attributes

```
<a href="#"
  onclick="window.resizeTo(800,600))">
  Size Window to 800 x 600
</a>
```

Functions 1

- Functions are equivalent to “methods” in Java

- Introduced by the “function” keyword, and then scoped by { and }

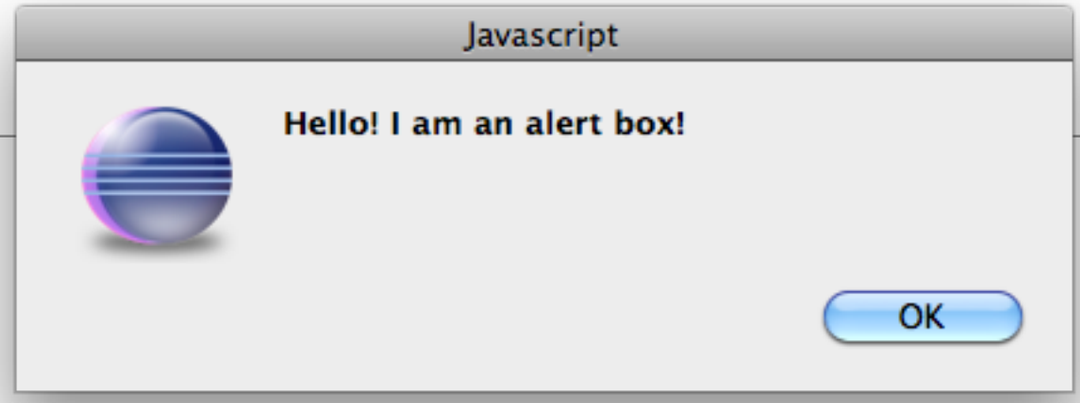
```
function message1()
{
  document.write ("Hello World");
}

function message2()
{
  document.write ("Hello World Again");
}
```

- These two functions write strings to the current document - the DOM

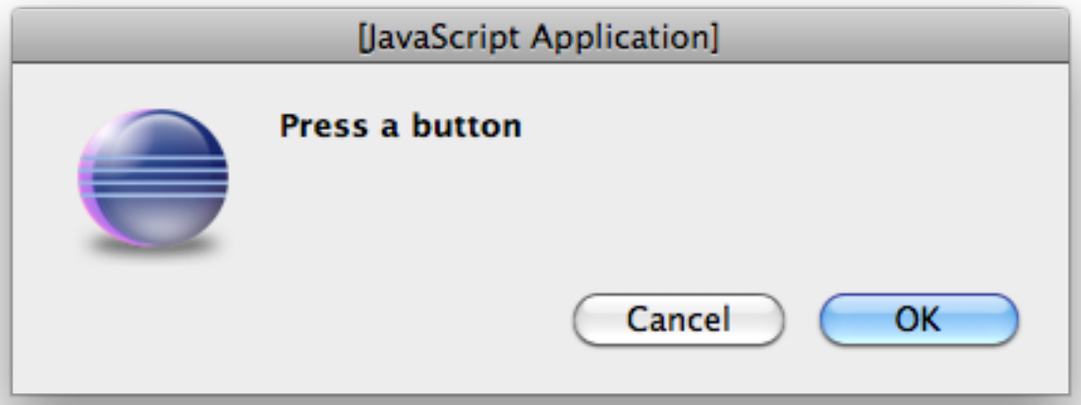
- There effect on the DOM depends on whether is page is currently loading, or has already loaded

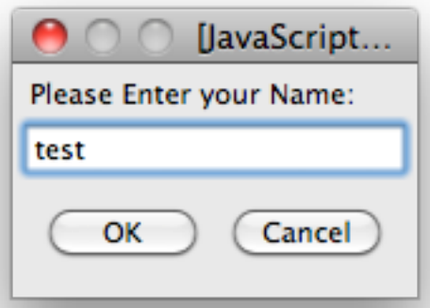
Functions 2

- These three functions display dialog boxes on top of the browser



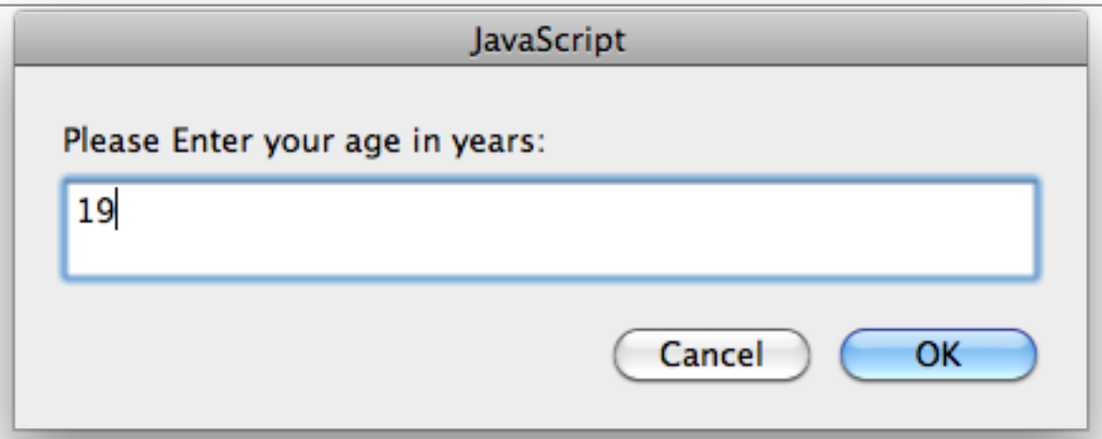

```
function showAlert()
{
  alert("Hello! I am an alert box!");
}

function showConfirm()
{
  confirm("Press a button");
}

function showPrompt()
{
  prompt("Please Enter your Name:", "none");
}
```



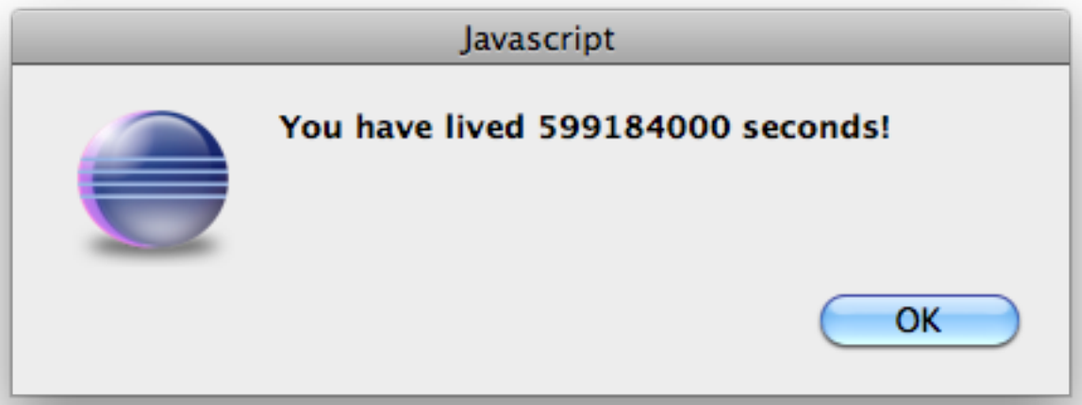


Functions 3

- Functions can perform computations, implement algorithms or arbitrary complexity




```
function calcAgeInSecs (age)
{
  var age = prompt("Please Enter your age in years:", "0");
  var seconds = age * 365 * 24 * 60 * 60;
  alert("You have lived " + seconds + " seconds!");
}
```



Variables

- JavaScript variables have some similarities to variables you have already encountered in the Java programming language
  - They have names, like “x”, or a more descriptive name, like “ageInSecs”.
  - The variable names are case sensitive (y and Y are two different variables)
  - They must begin with a letter or the underscore character
- The differ from Java variables in a number of important ways
  - there is no "type" associated with the variable
  - The variables can hold numbers, strings, boolean values (true,false) or null
  - The same variable can, at different times, hold different values, and different type (a number or a string for instance).
  - Variables are always introduced with the "var" keyword

```
var x=5;
var dvdname ="The Crazies";
var valid = false;
var go = true;
var cost = 23.99;
```

Arithmetic Operators

- Assume y contains 5

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

Assignment Operators

- Assume x is 10 and y is 5

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Comparison & Logical Operators

- Assume x contains 5

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

Date & Time

- The Date object is useful when you want to display a date or use a timestamp in some sort of calculation.

- In Javascript, you can either make a Date object by supplying the date of your choice, or you can let JavaScript create a Date object based on the system clock

```
function getDate()
{
  var currentTime = new Date();
  var month = currentTime.getMonth();
  var day = currentTime.getDate();
  var year = currentTime.getFullYear();
  var date = month + "/" + day + "/" + year;
  return date;
}
```

```
function getTime()
{
  var currentTime = new Date();
  var hours = currentTime.getHours();
  var minutes = currentTime.getMinutes();
  var seconds = currentTime.getSeconds();
  var time = hours + ":" + minutes + ":" + seconds;
  return time;
}
```

The current Date is

10/22/2010

Did you see the Date?

```
<body>
  <p> The current Date is </p>
  <script type="text/javascript">
    document.write(getDate());
  </script>
  <p> Did you see the Date? </p>
</body>
```

Events

- An event is some activity, usually initiated by the user, which can be detected and intercepted by the browser.

- Very often associated with the input device (keyboard or mouse)

- But may also be associated with other activities - e.g. a page load

- **click**
- **change**
- **submit**
- **load**
- **mouseout**
- **mouseover**
- **blur**
- **focus**
- **dblclick**
- **keydown**
- **keyup**
- **keypress**
- **unload**
- **mousedown**
- **mousemove**
- **mouseup**
- **reset**
- **select**

Input Element

- One of the most diverse elements in html.
- Many “Types” possible
  - button
  - text
  - checkbox
  - radio
  - password
  - etc

```
<input type="button" value="Click Me" />
```


```
<input type="text" value="Click Me" />
```

- Will often define and participate in “events”

onclick for Input Button

About to launch a script:

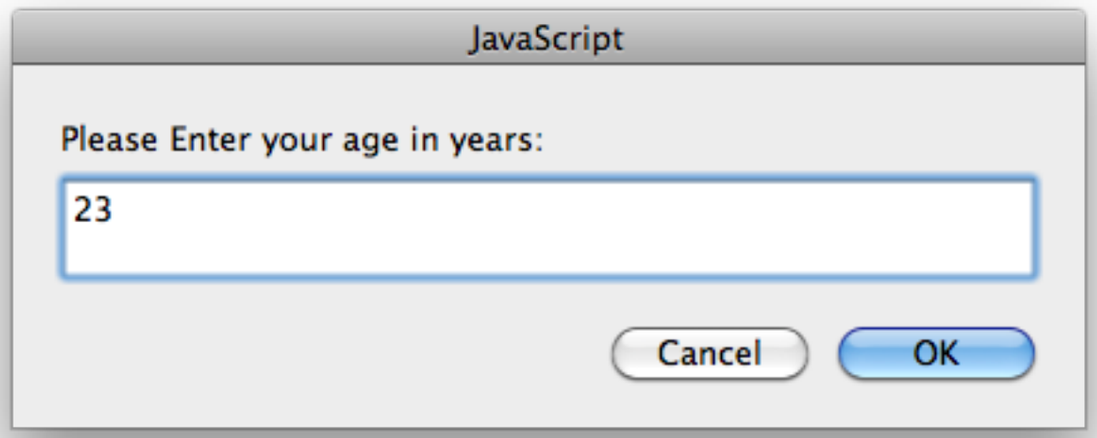
Hello World

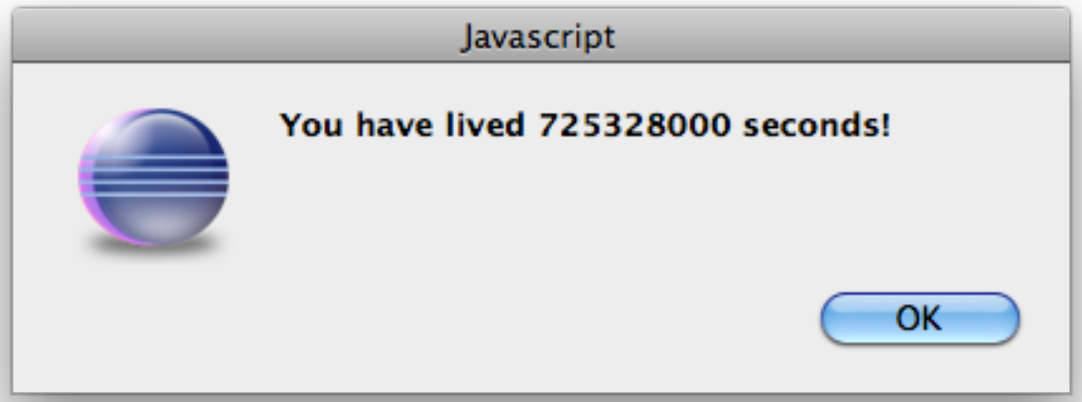
Script has finished - did you see the text?

Click Me

```html
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <script type="text/javascript" src="script4.js"> </script>
    <title>Insert title here</title>
  </head>
  <body>
    <p> About to launch a script: </p>
    <script type="text/javascript">
      document.write("Hello World")
    </script>
    <p> Script has finished - did you see the text? </p>
    <input type="button" value="Click Me" onclick="calcAgeInSecs()" />
  </body>
</html>
```

```javascript
function calcAgeInSecs (age)
{
  var age = prompt("Please Enter your age in years:", "0");
  var seconds = age * 365 * 24 * 60 * 60;
  alert("You have lived " + seconds + " seconds!");
}
```





onclick for Input Button

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

```
<body>
   //...
   <input type="button" value="Click Me" onclick="calcAgeInSecs()" />
 </body>
```

```
function calcAgeInSecs (age)
{
  var age = prompt("Please Enter your age in years:", "0");
  var seconds = age * 365 * 24 * 60 * 60;
  alert("You have lived " + seconds + " seconds!");
}
```

script.js

Changing Styles

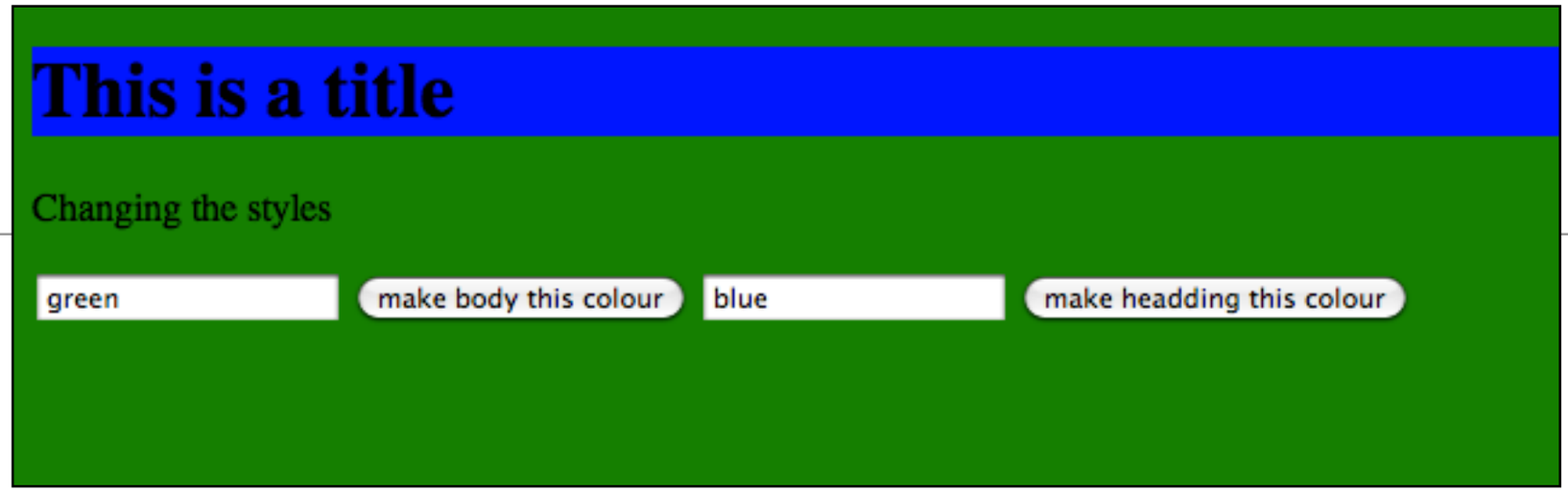
```
<body id="homepage">
  <h1 id="title"> This is a title </h1>
   <p>
     Changing the styles
   </p>
   <input type="button" value="make body red"
     onclick="changeBackground('homepage', 'red')" />
  <input type="button" value="make headding green"
     onclick="changeBackground('title', 'green')" />
</body>
```

- changeBackground() is method we have written, taking two parameters
  - Parameter 1: the Id of an element whose colour you wish to change
  - Parameter 2: the new colour for the background of the element

function changeBackground()

```
function changeBackground(id, colour)
{
  var element = document.getElementById(id);
  element.style.backgroundColor = colour;
}
```

- Two statements in the function:
  - Statement 1: ask the document (the DOM) for a reference to the element with a specific id. Store this in the variable called “element”
  - Statement 2: reach into this element, get its current style, and change the background colour to the one specified by the parameter




```
<body id="homepage">
  <h1 id="title"> This is a title </h1>
  <p>
    Changing the styles
  </p>
    <input type="text" id="bodycolour" />
    <input type="button" value="make body this colour" onclick="changeBackgroundById('homepage', 'bodycolour')" />
    <input type="text" id="headercolour" />
    <input type="button" value="make headding this colour" onclick="changeBackgroundById('title', 'headercolour')" />
</body>
```

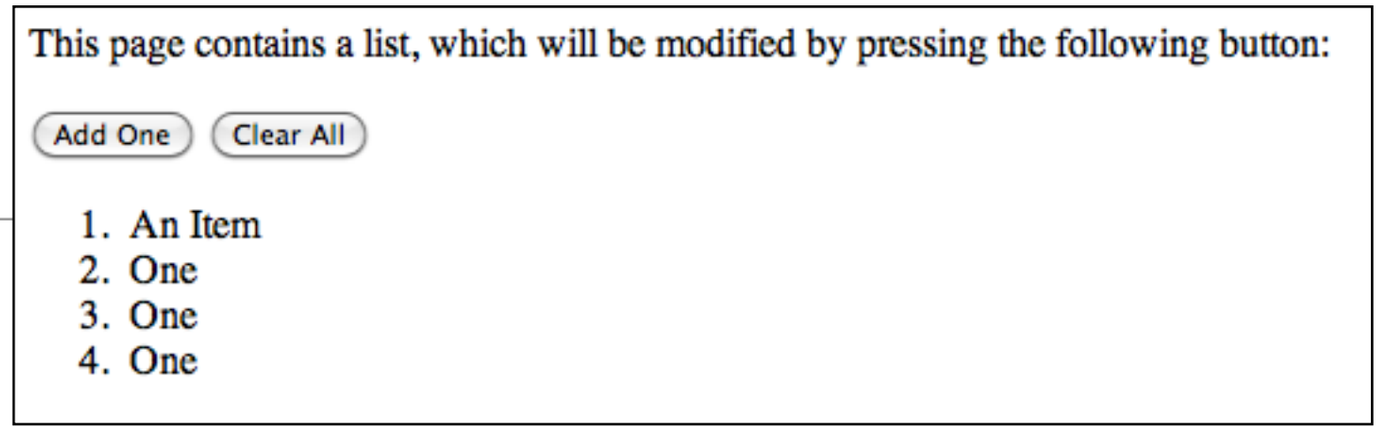
- Specify any valid colour in text fields
- changeBackgroundById() reads the colour from the text fields, and sets the style accordingly

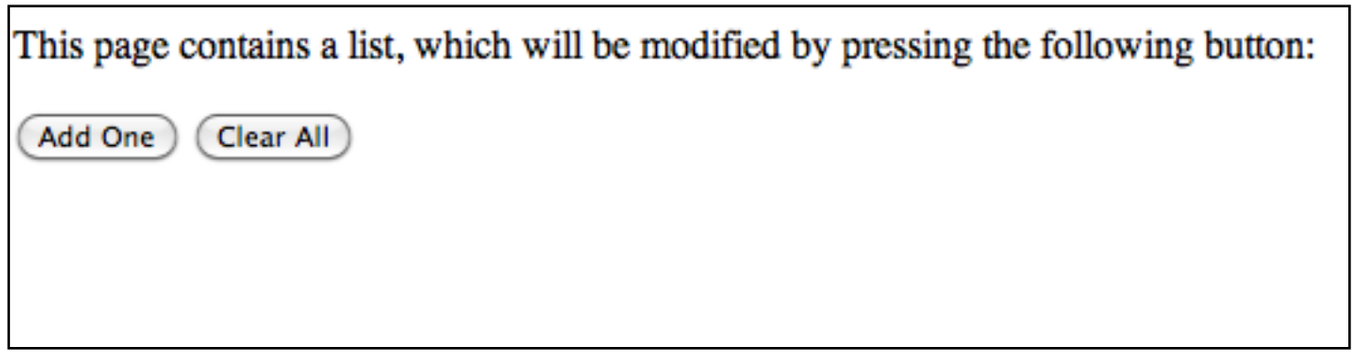
changeBackgroundById()

```
function changeBackgroundById(elementId, colourTextId)
{
  var element = document.getElementById(elementId);
  var colour = document.getElementById(colourTextId);
  element.style.backgroundColor = colour.value;
}
```

- Get reference to the element to the changed
- Get a reference to the text field containing the colour
- Set the element’s colour to the colour in that text field

Manipulating Lists





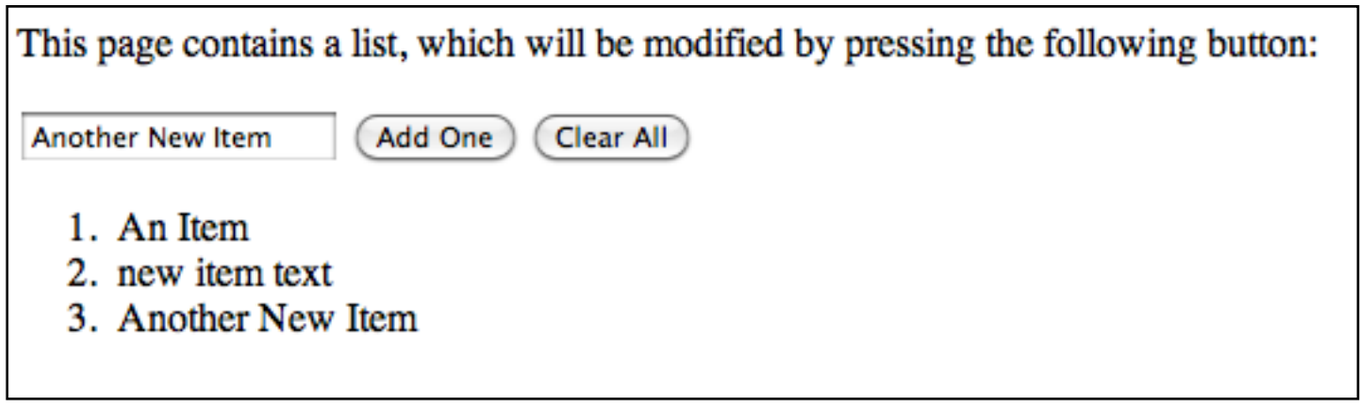

```
<body>
  <p>
    This page contains a list, which will be modified by
    pressing the following buttons:
  </p>
  <input type="button" value="Add One" onclick="addElement('One')" />
  <input type="button" value="Clear All" onclick="clearList()" />
  <ol id="list">
    <li> An Item </li>
  </ol>
</body>
```

functions addElement(), clearList()

```
function addElement(item)
{
  var list = document.getElementById('list');
  var newItem = document.createElement('li');
  newItem.innerHTML = item;
  list.appendChild(newItem);
}

function clearList()
{
  var list = document.getElementById('list');
  list.innerHTML = "";
}
```

- “document.createElement” will create a new element
- “appendChild” will attach an new element to an existing element
- “innerHTML” is a simple mechanism for changing the contents of an element

addElementById



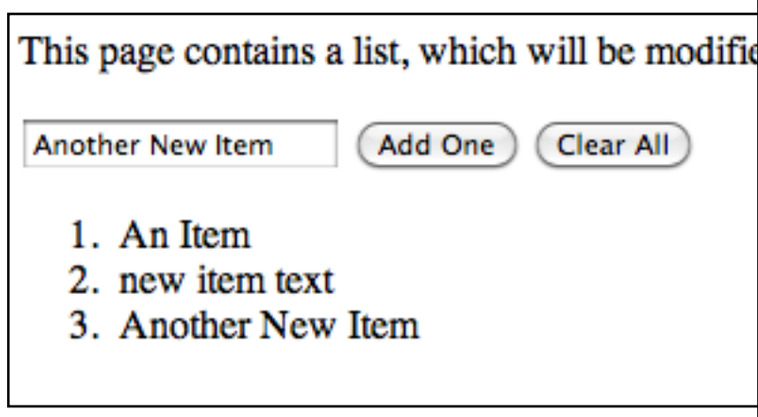

```
<input type="text" id="itemtext" />
<input type="button" value="Add One" onclick="addElementById('itemtext')" />
<input type="button" value="Clear All" onclick="clearList()" />
<ol id="list">
  <li> An Item </li>
</ol>
```

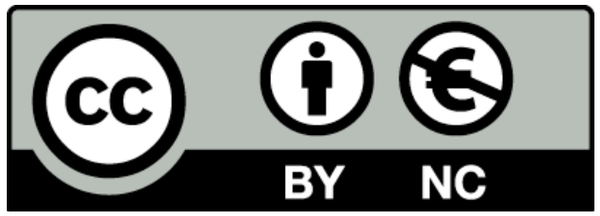
- Retrieve text from text field, and add to list

Function addElementById()

```
function addElementById(itemId)
{
  var list = document.getElementById('list');
  var itemText = document.getElementById(itemId);
  var newItem = document.createElement('li');
  newItem.innerHTML = itemText.value;
  list.appendChild(newItem);
}
```

1. Retrieve reference to the list
2. Retrieve reference to the text field
3. Create a new list item element
4. Insert the text in the text field into this new list item element
5. Append this new element into the list





Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

eLearning support unit