Programming on the Client Side: JavaScript/Dynamic HTML

Recap —- Perl/CGI:

- Server Side
- Essential to do hard work processing on the Internet (PHP similar)
  - Search Engine
  - Log Usage
  - Gather Data/Stats
  - Interface to Databases
  - Many more suitable applications ....

Return to Client Side (The Browser)

So far we have been developing

- **Static Web pages**

Static Pages:

- Very nice.
- Often Adequate for the task in hand
- Often useful and
- Often entertaining or informative.

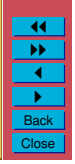
Making Web Pages Dynamic

Can replace some (**but not all**) CGI functionality?

E.g. HTML form:

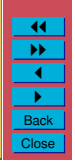
- Users enter data and submit it to the server
- a CGI script is used to verify and validate that data.
- passes data across the network — **slow**.

How much more interactive could a site be if data were checked by the browser and any error messages originated locally?



Other Typical Uses of Dynamic Web Page

- Detecting which browser accesses page
- Customising HTML for browsers
- User Customised HTML
- Opening new browser windows for new links
- Messages and confirmations, Alerts, Status Bar
- Writing to Frames
- Rollover Buttons, Moving Images
- Multiple pages in a single Download
- ..... Many More



Some More Definitions: Dynamic HTML and Document Object Model

Before delving into the programming there are two more topics to introduce.

The Document Object Model (DOM) :

- Scripts can only manipulate objects on the page because of DOM
- This was developed by the World Wide Web Consortium (W3C)
- Limited Cross-browser Support — neither of the big players yet adheres fully to it.

Dynamic HTML (DHTML) :

- A combination of content formatted using HTML, cascading style sheets, a scripting language and the DOM.

The Document Object Model (DOM)

DOM:

- The key element of DHTML is probably the document object model.
- DOM makes everything on the page into an object
- Elements can be manipulated by programmed scripts.

DOM of an HTML Document

The DOM model of an HTML document is a hierarchical structure which might reasonably be represented as a tree.

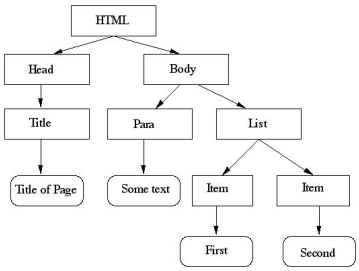
- **Does NOT** imply that DOM software must manipulate the document in a hierarchical manner,
- It is simply a representation.

Example HTML DOM

The relationship between some HTML code and the DOM

```
<html>
<head>
   <title>TITLE OF PAGE</title>
</head>
<body>
  <p>Some text...</p>
  <ul>
    <li>First
    <li>Second
  </ul>
</body>
</html>
```

Example HTML DOM (Cont.)



The DOM models, but does not implement:

- Objects that comprise a document
- Semantics, behaviour and attributes of those objects
- Relationships between those objects

JavaScript

JavaScript is NOT JAVA:

- The only similarity between the two languages is in their names!!!
- Basic idea : a client-side browser language not as complicated as Java.

JavaScript and JScript

- Meant to be implementations of the same thing.
- Not exactly the SAME!!!!

What is JavaScript and What is it Used For?

JavaScript is:

- A fairly simple language
- Only suitable for fairly simple tasks
- Best suited to tasks which run for a short time
- Most commonly used to manipulate the pieces of the document object model.

Pros and Cons of JavaScript

Advantages:

- Wide browser support **i.e.** Most Browsers speak a dialect.
- Easy access to document objects and their manipulation
- No long download times w.r.t java or graphic animations
- No Plug-in support required
- Relatively Secure

Disadvantages:

- Not standard support for Javascript across browsers esp. DOM
- Web page useless if script does not work!!
- Javascript may be disabled by browser reading HTML file no control over this
- JavaScripts can run slowly

The version of JavaScript that was used as the basis of the ECMA Script standard was 1.1. Therefore everything that is written using JavaScript 1.1 should comply with the standard.

Learning JavaScript

Borrowing Code

As with HTML (and Perl):

- A lot of code on Web.
- All the JavaScript that your browser encounters is freely available to you.
  - Part of HTML document (as we will see very soon)
  - So View Source, or
  - Save as ... HTML will give you access to lots of Examples.

One of the best resources on the web is **The JavaScript Source** at **http://javascript.internet.com**

First Steps in JavaScript

Running Javascript

- JavaScript can be run on some file and Web servers
- MOST COMMONLY: front-ends for Web pages.

Therefore:

- You **do not compile** it like other programs, Java, C, Pascal
- It is an interpreted language: It is a **script**
- **To Use**: Simply **embed** JavaScript in you HTML web page

A Simple Example 1 — Hello World

The script that follows could hardly be easier.

- We'll list the code first then explain what's going on.

```
<html>
 <head>
   <title> Hello World JavaScript</title>
 </head>

<body >
<script language="javascript">

   var str = "Hello World";

   window.alert(str);

   document.write("<h1>" + str + "</h1>");

</script>


</body>
</html>
```


The key points of JavaScript Programs

- The JavaScript program is contained between the `<script>` .... `</script>`
- The `language` attribute must be set to `"javascript"` **i.e**

```
<script language="javascript">
```

- JavaScript programs contain variables, objects and functions.
- Each line of code is terminated by a semi-colon;
- Blocks of code must be surrounded by a pair of curly brackets. A block of code is a set of instructions that are to be executed together as a unit. This might be because they are optional and dependent upon a boolean condition or because they are to be executed repeatedly;
- Functions have parameters which are passed inside parenthesis;
- Variables are declared using the keyword `var`;
- Scripts require neither a main function nor an exit condition.
- Functions may be used.

Our First Hello World Example

Specifically for the Hello World Program:

- We create a variable `str` and store the string `"Hello World"` in it.
- We create an alert window with the message `"Hello World"`. The `window` object is used and we call the `alert` method (function) of this object.

Figure 29: Hello World JavaScript Example

- We `write()` to the `document` model some HTML.

A Simple Example — Hello World with a Function

Let us now see how we use use funtions in JavaScript
Here wee also note that

- You may freely write JavaScript in any place in your Web page.
- However if you wish to trigger actions from certain events you may wish to put your JavaScript in functions in the `Head` section of the document.

Hello World with a Function

The Hello World program called from a function is as follows:

```
<html>
 <head>
   <script language="javascript">

   function init(){

   var str = "Hello World";

   window.alert(str);

   document.write("<h1>" + str + "</h1>");

   }

   </script>
 </head>

<body onLoad="init()">

</body>
</html>
```


Hello World with a Function — Explanation

The operation of the program is essentially the same as the first version except that

- We write a function

  `init()`

  to do the job.
- We call the function when the Web page loads with the

  `onLoad="init()"`

  attribute in the `body` tag.

More Functions

We can add more functions that get called on different events:

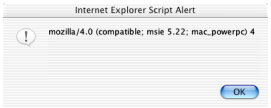
- We call a function `popup()` when the Web page loads
- We call a function `farewell()` when the page is unloaded:
  - The `onUnLoad="farewell()"` attribute is set.

Popup Browser Information Example

The program also introduces some more objects and some useful applications of JavaScript:

- The `navigator` contains data about the browser being used
- The program finds the version of the application `appVersion` and the `userAgent()`.
- This information is displayed in a pop up alert
- When you leave the Web page or Refresh the page an different `alert` is brought up

Safari Version



Explorer Version

Popup Browser Information Example

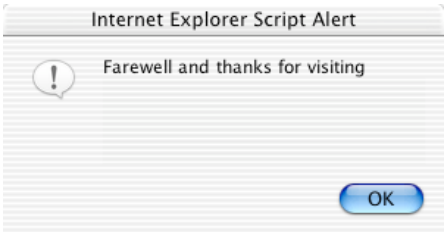


Figure 30: Browser Version Page Exit JavaScript Example (Internet Explorer Display Shown)

Popup Browser Information Example Code

The full code for this example is:

```
<html>
 <head>
   <script language="javascript">

   function popup(){
   var major = parseInt(navigator.appVersion);
   var minor = parseFloat(navigator.appVersion);
   var agent = navigator.userAgent.toLowerCase();

   document.write("<h1>Details in Popup</h1>");
   window.alert(agent + " " + major);
   }

   function farewell(){
   window.alert("Farewell and thanks for visiting");
   }
   </script>
 </head>

<body onLoad="popup()" onUnLoad="farewell()">
</body>
</html>
```

More Browser Information

So far our examples have done everything **inside JavaScript environments**:

- You can also **mix** HTML and JavaScript
- You can also **access JavaScript variables** in the HTML

A Second Browser Information Example

The program achieves a similar task as the previous example but in a completely different way:

- The program an HTML TABLE (see later) to a Browser with the Browser's details:

Figure 31: Browser Version 2 JavaScript Example (Explorer)

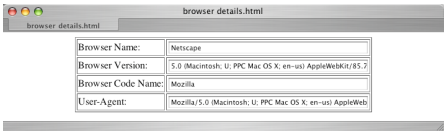


Figure 32: Browser Version 2 JavaScript Example (Safari)

- The program use the Javascript `document` object to determine which browser's `Name, Version, Code` and `Agent` are in operation.
- `Name, Version, Code` and `Agent` are **referenced in the body** of the HTML code

Browser Information Example 2: Javascript and HTML Mix

Browser Information Example 2: HTML HEAD

```
<HEAD>
<SCRIPT>

<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->

<!-- Begin
function whatBrowser() {
document.Browser.Name.value=navigator.appName;
document.Browser.Version.value=navigator.appVersion;
document.Browser.Code.value=navigator.appCodeName;
document.Browser.Agent.value=navigator.userAgent;
}
// End -->
</SCRIPT>
```

Browser Information Example 2: HTML BODY

```
<BODY onLoad="whatBrowser()">
<CENTER> <TABLE BORDER>
<FORM NAME="Browser">
<TR>
<TD> Browser Name: </TD>
<TD> <INPUT TYPE="txt" NAME="Name" Size="45"></TD>
</TR>
<TR>
<TD> Browser Version: </TD>
<TD> <INPUT TYPE="txt" NAME="Version" Size="45"></TD>
</TR>
<TR>
<TD> Browser Code Name: </TD>
<TD> <INPUT TYPE="txt" NAME="Code" Size="45"></TD>
</TR>
<TR>
<TD> User-Agent: </TD>
<TD> <INPUT TYPE="txt" NAME="Agent" Size="45"></TD>
</TR>
</FORM>
</TABLE> </CENTER>
</BODY>
</HTML>
```


JavaScript: Formal Syntax

Now that we have seen a few examples we should be able to pick up the main aspects of JavaScript.

But let us formally introduce these concepts:

- Using JavaScript with HTML Pages
- Formal JavaScript Syntax
  - Similar to many other languages at the command level

Writing small scripts

If use small scripts or only scripts in a web few pages then

- Include the script code in the HTML Page:

```
<html>
<head>
  <title>A Sample JavaScript</title>
  <script language=javascript>
    <!--
     the JavaScript code goes here...
    // -->
  </script>
</head>
<body>
...
</body>
</html>
```


Sharing A JavaScript Amongst Several Pages

What if you have a good script and you **wish/need to use it in several distinct pages?**:

- **Bad Idea**: include it every page literally
  - Copy and Paste into appropriate place or Retype Code?
  - What if you need to **change to source code**?
- **Good Idea**:
  - Store code in a **separate file**
  - Include that **separate file code** in the head of the page as shown below.
  - **Easier maintenance**: Only one file edited for code changes.

Including a JavaScript file

- By convention JavaScript programs are stored in files with the `.js` extension.
- To include the JavaScript `.js` file:
  - Use the `src` attribute in the `script` tag:

```
<html>
<head>
   <title>A Sample JavaScript File Include</title>
   <script language=javascript src="sample.js">
   </script>
</head>
<body>
    ...
</body>
</html>
```


Variables

- Like any programming language JavaScript has variables.
- Stores data items used in the script.
- Strict rules governing how you name your variables (Much like other languages):
  - Variable names **must** begin with a letter, digit or underscore;
  - You can't use spaces in names
  - Names are **case sensitive** so the variables `fred`, `FRED` and `frEd` all refer to **different** variables,
    - **HOWEVER** It is not a good idea to name variables with similar names
  - You **can't use a reserved word** as a variable name, **e.g.** `var`.

Creating Variables

- Use the keyword `var` before the variable name.
- No necessity to give the variable a value
- A value can be assigned with `=`.

Look at the following examples, **E.g**:

```
var num = 23;
var str = "Some words";
var another_str = str;
var first_boolean = true;
```

- As in Perl/Javascript, variables are context sensitive

Manipulating Variables

Having defined it you can manipulate variables in a variety of ways.

These are fairly standard operations such as.

- Simple arithmetic `+, -, */` etc. **E.g.**

```
num = num + 3;
```

- Autoincrement `++, --`

```
++num;
```

- String concatenation `+`:

```
str = str + another_str;
```

JavaScript Data Types

JavaScript has only four types of data,

- **Note:** you do not declare the type in `var`:

`Numeric` :

- Integers such as 2,22 and 2,222,000 or
- Floating point values like 23.42, -56.01 and 2E45.
- No need to differentiate between
- In fact variables can change type within program (Similar to Perl).

JavaScript Data Types (Cont.)

`Strings` :

- `String` **Definition:**
  A Collection of characters (that are not numbers).
- All of the following are strings:

```
"David",
"David Marshall"
"12345.432".
```

- Put quotes around the value to a assign a variable:

```
name = "David Marshall";
name = 'David Marshall';
str  = "34.45";
```


JavaScript Data Types (Cont.)

`Boolean` :

- Variables can hold the values true and false.
- Used a lot in conditional tests (later).

`Null` :

- Used when you don't yet know something.
- A `null` value means one that has not yet been decided.
- It does not mean **nil** or **zero** and **should NOT** be used in that way.

JavaScript Variable Scoping Rules

In JavaScript variables can be either **local** or **global**:

`Global` :

- Variable is available to all parts of the program

`Local` :

- Variables are declared **inside** a function.
- Can only be used **within** that function.

Simple Example: Difference between Global and Local Variables

The following (**meaningless**) code illustrates the difference between global and local variables:

```
// declare the global variables
var global_one = 32;
var global_two;

function test {
  // this function has a local variable
  // a global value is copied into it
  // the local is then passed into a
  // function as a parameter

  var local_var; // local variable
  local_var = global_one; // copy global
                          // into local
  func_two(local_var);
  }

function func_two(number) {
  // create a local variable and copy
  // a parameter into it

  var local_var = number;
  }

// copying a local variable into a global
// like this is not allowed
global_two = local_var;
```


Comments

The last **global/local scoping example** contained some **JavaScript comments**:

- Each line of a **comment** is defined by **two slashes** `//`, and
  - Comments continue from that point (`//`)to the **end of the line**.
  - All code form `//` to **end** of line are **ignored** by Javascript
- For example:

```
// this is a JavaScript comment

// var i = 1; this variable assignment is ignored
// as it is commented out
```

- No way of commenting large blocks of text as a block in JavaScript.
  - If you want a block comment then you have to comment **each and every line** – as in above example.

Objects in JavaScript

JavaScript **attempts** to be an object-oriented (OO) language:

- **NOT** a true OO language
- **BUT** Primitive JavaScript objects are quite useful
- Also many useful built-in functions

Objects and Classes

Objects are described in software and design constructs called **classes**.

A **class** usually contains:

- Some data items and
- Some methods.

JavaScript Objects

We do not really need to understand the **ins and outs** of objects for this course.

We essentially just use built-in JavaScript objects such as:

- `Document` and
- `Window`

You can create your own JavaScript objects but we'll leave this for another course (or see a good book or Web Site)

- The web site:

  `http://www.physiol.ox.ac.uk/Computing/Online_Documentat`

- contains a good reference for Javascript, and should be considered when

- studying the next few slides, or coding (or use a similar object properties listing)

The `Document` Object

A `document` is a Web page (object) that is being either

- Displayed or
- Created.

A `document` is controlled by a number of **properties**.

Document Properties

The `document`

- Has a **number of properties**
- Is **accessed by JavaScript programs**
- Is used to **manipulate the content** of the Web page.

Some of these properties can be used to

- Create HTML pages from **within** JavaScript
- Change the **operation** of the current page.

Some Common `document` Methods/Properties

`write, writeln`:

HTML pages can be created **live (dynamically)** using JavaScript. This is done by using the `write` or `writeln` methods of the document object.

E.g.:

```
document.write("<body>");
document.writeln("<h1>A test</h1>");
```

`writeln` causes a newline to be output with the output.

`bgcolor, fgcolor`:

- The same properties can be set in the HTML `<BODY>` tag.
- The values can be set from JavaScript.
- The methods accept either hexadecimal values or common names for colours:

```
document.bgcolor = "#e302334";
document.fgcolor = "coral";
```


`anchors`:

- This property is an array of names of all the HTML anchors
- List in the order in the HTML document.
- Anchors can be accessed like this:

```
document.anchors[0];
```

e.g, we could say

```
alert("The first anchor is " +
 document.anchors[0].name)
```

`links`:

- Another array – all HTML links
- Order as they appear on the Web page.
- Anchors can be accessed like this:

```
document.links[0];
```

- e.g. we could say


```
alert("The first hyperlink is " +
 document.links[0].href)
```

Note how this relates to the DOM

`forms` :

- An array in the order of the document.
- The array contains all of the HTML forms.
- Combining the array with individual form objects to access specific forms/form elements.

`layers` :

- A document can be made from a number of layers of content.
- An array contains the layer objects.
- Layers have many methods and properties of their own.

`close()` :

- A JavaScript `document` isn't **completely written** until the `close()` has been called.
- If you don't use this method then the browser will keep waiting for more data **even** if there is none — and may not complete the rendering of the Web page.

The `Window` Object

- The browser `window` is a mutable object
- Addressed by JavaScript.
- **See Examples soon**

Some Common `window` Methods and Properties

`open` :

- Opens a new window which contains the document specified by URL, **Syntax**:

```
win = window.open(URL ', 'name',
'prop1=X,prop2=X,propN=X' );
```

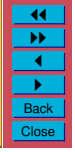
- The `window` is given an identifying `name` so that it can be manipulated individually.
- You can also manipulate the returned object, `win` in your JavaScript code..

```
win = window.open("","New Window","width=100,
height=100,resizable=0");
win.document.writeln("<h1>NEW WINDOW
MANIPULATION</h1>");
```

Note that there is no URL, also note the use of arguments such as width,resizable, etc

`close()` :

- Shuts the current window.



Some Common `window` Methods/Properties (Cont.)

Properties that may be toggled :

- Many of the attributes of a browser are undesirable in a pop-up window.
- Individually switched on and off.

```
toolbar=[1|0]
location=[1|0]
directories=[1|0]
status=[1|0]
menubar=[1|0]
scrollbars=[1|0]
resizable=[1|0]
```

Resizing a Window :

- You specify the size of a `window` in `pixel` units:

```
width= pixels
height= pixels
```


Some Common `window` Methods/Properties (Cont.)

`scroll(co-ord, co-ord)` :

- The `window` can be automatically scrolled to given position
- The `co-ord`inates are given in pixels.
- Start indexing from (0, 0) which is the **top left corner**.

Simple Example: Setting a Windows Properties

Frequently when a new window is being opened you may wish to set up how it is displayed in a Browser, with JavaScript:

- You may choose to open it at and set its properties very easily in JavaScript, for example:

```
newWin = open(address, "newWin",
       "width=100, height=100");
```

Creates a new window called `newWin` and of a size 100x100 pixels.

- **Many other properties (above) set in a similar way**

The `Form` Object

Two aspects of the form can be manipulated though JavaScript:

- The data that is entered onto your form can be checked at submission.
- You can actually build forms through JavaScript.

Form Elements:

- The elements of the form are held in an array
- Any of the properties of those form elements is now **accessible though JavaScript**.

Simple JavaScript `Form` Example

This example shows a form and a JavaScript function which reads the properties of the form elements

- If the name entered in the form's first element (the `name` text input field) is **not David** the form is reset!!!

Simple JavaScript `Form` Example HTML/JavaScript Code

```
<html>
<head>
  <script language="javascript">
    function validate() {
       var method = document.forms[0].method;
       var action = document.forms[0].action;
       var value = document.forms[0].elements[0].value;

       if(value != "David"){
         document.forms[0].reset();
       }
     else {
        alert("Hi David!!");
     }
   }
  </script>
</head>
<body>
  <form method="post">
    <input type="text" name="user" size=32>
    <input type="submit" value="Press Me!"
      onClick="validate()">
   </form>
</body>
</html>
```


Simple JavaScript `Form` Example Explained

The main points of this program are:

- We use the `onClick` event to trigger validation routines.
- This event is set in the `form`'s `input` tag for the `submit` type.
- The `onClick` event can be applied to all form elements — The event is triggered when the user clicks on that element.
- There other events that can triggered in a form:

  `onSubmit` : This event can only be triggered by the form itself and occurs when a form is submitted.

  `onReset` : Triggered when a form is reset by the user.

Validate/Submit to php/perl

processForm.php/.pl will only run if all fields are filled in.

```
<html>
<head>
 <script language="javascript">
 <!--
  function verifyForm(theForm) {
   if(theForm.username.value == "" ) {
     alert("Please enter a name");
     return false;}
   if(theForm.address.value == "" ) {
     alert("Please enter an address");
     return false;}
  }
 //-->
 </script>
</head>
<body>
 <form name="myForm" method="POST"
```

```
  action="processForm.php"
  onSubmit="return verifyForm(this)">
  Name: <input type="text" name="username"><br>
  Address:<input type="text" name="address"><br>
  <input type="submit" value="Send">
 </form>
</body>
</html>
```

- The function verifyForm can also be substituted with a different function
- E.g. validate() could be used, but it would require some modification

```
if(value != "David"){
        document.forms[0].reset();
        //This new line is required
        return false;
      }else {
       alert("Hi David!!");
```



```
    }
```

Now you can only visit processForm.php if 'David' is entered.

More Events

Some events have already been seen i.e. onLoad, onClick, onReset

- Events are often generated in response to mouse or keyboard actions
- Events can be used to run JavaScript functions in response to user actions, e.g. to validate user input from a form or to cause graphic effects such as rollovers
- Not all HTML elements support all events
- The set of HTML elements able to support events is different for different browsers.

Selected Events

- onClick: when the mouse button is clicked on an element (used with the button and link elements)
- onMouseOver/onMouseOut: when the mouse moves into/ out of an element (used with the link, image and layer elements).
- onMouseDown/onMouseUp: when the mouse button is pressed/released.
- onLoad/onUnload: when browser loads/finishes with a document (used with the body element).
- onFocus/onBlur : when an element is selected/deselected (i.e. another element is selected) with the mouse (used with the input, select, textarea and button elements).
- onSubmit: when the submit button pressed in a form (used with the form element).

Another onClick example

```
<head>
 <title>Example 3</title>
 <script language="javascript">
 <!--
  function change(col) {
    if(col=="red") {
      document.body.style.backgroundColor = "#ff0000";
    }
    if(col=="blue") {
      document.body.style.backgroundColor = "#0000ff";
    }
  }
 //-->
 </script>
</head>
<body>
 <form>
 <input type="button" value="Red" onClick=change("red")>
```



```
 <input type="button" value="Blue" onClick=change("blue")>
 </form>
</body>
```


Rollovers

- A rollover is an image that changes its appearance when the mouse moves over it, and returns to its original state when the mouse moves away
- The mouse movement is detected using the onMouseOver and onMouseOut event handlers

```
<img name="homeImage" src="home1.png">
```

The source of this image is stored in the object

```
document.homeImage.src
```

Mouse events trigger JavaScript commands that change the content of this object

Rollover Example

```
<head>
 <title>Rollovers</title>
</head>
<body>
 <a href="home.html"
   onMouseOver="document.homeImage.src='home2.png'"
   onMouseOut="document. homeImage.src='home1.png'">
 <img src="home1.png" name="homeImage"
   alt="home"  border="0" width="100">
 </a>
</body>
```

JavaScript Examples

Rollovers

Manipulating Windows

Common tasks based around the `window` object:

- Many practical uses
- Commonly seen on many web pages

Window Example 1: Opening a New Window

- Windows are independent of each other any windows
  - They can be **spawned** from our code
  - They can be made to **look** and **act** totally differently.

Recap: The `open()` method

Recall from last lecture `window` object definition :

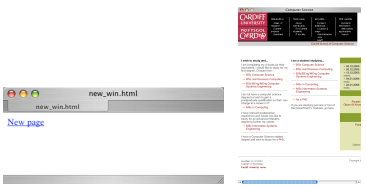
```
open("URL ", "name ")
close()

toolbar=[1|0]
location=[1|0]
directories=[1|0]
status=[1|0]
menubar=[1|0]
scrollbars=[1|0]
resizable=[1|0]
width=pixels
height=pixels
```


Window Example 1: Loading a URL into an Opening Customised Window

So our JavaScript Example can be developed to:

- Open a new window from a current one
- Loads specified a URL and
- Attributes of new window can be customised.



Window Example 1: JavaScript Code

The JavaScript code is as follows:

```
<html>
<head>
 <script language=javascript>
 <!--
  function Load(url){

    newwin = open(url, "newwin",
                  'status=0,
                   toolbar=0,
                   resizable=0,
                   width=500,
                   height=700');
    }
 //-->
 </script>
</head>

<body>
  <a href=""
   onClick="Load('http://www.cs.cf.ac.uk')">
   New page</a>
</body>
</html>
```

Window Example 1 Explained

The basic operation of the code is:

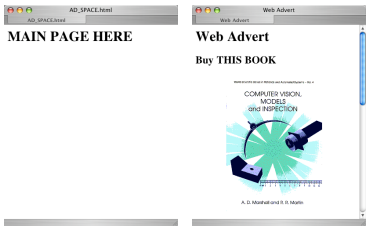
- The Web page Link is loaded into new window.
- The URL and window name are not optional
  - The URL can be replaced with empty quotes to open a blank window, E.g.:

```
open("","blankwin);
```

- Window has fixed given size:
  - It cannot be `resize`d
- No `toolbar` or `status` is present either.
- Set `resize`,`width height` **etc.** parameters:
  - The parameter list **must** be inside a single set of single quotes,
  - There cannot be line breaks or spaces in the parameter string (licence is taken in the notes for space considerations)
  - Don't have any spaces between the parameters,
  - Don't forget the commas between parameters.

Window Example 2: Web Advertising

- Javascript commonly used to create/control web adverts
- Here We access main page
- This will action a function to bring up a second window with adverts in
- Simply we call a `open()` on an `onLoad()` event.
- Load a web page `ad.html` in this window.



Window Example 2: JavaScript Code

```
<html>
<head>
 <script language=javascript>
 <!--
  function Load(url){

    newwin = open(url, "newwin",
             'status=0,toolbar=0,
              resizable=0,
              width=500,
              height=700');
     }
 //-->
 </script>
</head>

<body onLoad="Load('ad.html')">


  <h1> MAIN  PAGE HERE</h1>
</body>
</html>
```


Messages, Alerts, Prompts and Confirmations Example

- We have already met the `alert` message is previous examples.
- Other Message types exist.

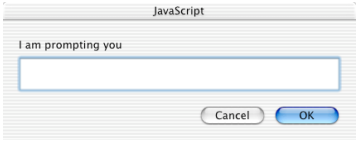
JavaScript provides three built-in window types:

- Prompts
- Confirms
- Alerts

`prompt("string, string ")` :

displays a simple window that contains a prompt and a textfield in which the user can enter data.

E.g. `prompt("You are being prompted", "");`



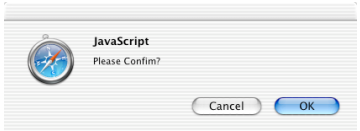
The method a has two parameters:

- a text string defining the prompt message and
- a string to use as the default value.
  If you don't want to display a default then simply use an empty string.

`confirm("string ")` :

Shows a window containing a message and two buttons: **OK** and **Cancel**.

E.g. `confirm("Please Confirm?");`



- Cancel will abort any pending action,
- OK will let the action proceed.

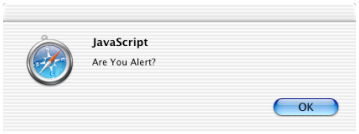
Sample applications:

- When submitting form data, or
- Possibly as the user tries to follow a link that leaves your site for another.

`alert("string ")` :

Displays the text string and an OK button.

E.g. `alert("Are You Alert?");`



The Status Bar

- At the bottom of the browser window.
- Non JavaScript Usage:
  - Displays URL of any link mouse is over
  - Connection status of web connection
- JavaScript can write to it:
  - Some Web developers like to use as part of the site.
  - Text strings can be displayed in the status bar but should be used with care.
  - Browser can't display your messages and useful messages about browser!!
  - anything that can be done in the status bar can be done more interestingly using elsewhere on Browser (Message box, body ...)

To write to the status do:

```
<html>
<head>
<script language="javascript">
<!--
function Init(){
self.status = "Some Message";
}
//-->
</script>
</head>
<body onLoad="Init()">
<h1>And the Status Bar Says...</h1>
</body>
</html>
```

Note: we will see another example of the `status()` function in the next browser check example

Checking for a Browser and Alerting the User

One final example for `alert`s and `status`es:

- We revisit the `navigator` object examples
- We detect what browser is being used and if the user is using **Internet Explorer** we inform the user via an `alert()` and `status()`

Checking for a Browser and Alerting the User Code Listing

The Code should be readily understood as we have covered all the elements before:

```
<html> <head>
<title>
Browser Check - Java Script E.G.
</title>
<SCRIPT LANGUAGE="JavaScript">

function Init() {
document.Browser.Name.value=navigator.appName;
document.Browser.Version.value=navigator.appVersion;
document.Browser.Code.value=navigator.appCodeName;
document.Browser.Agent.value=navigator.userAgent;

if (navigator.appName == "Microsoft Internet Explorer") {
 alert("You are using Microsoft Products. GO AWAY!!!");
 status = "You are using Microsoft Products. GO AWAY!!!";
}
else {
status = "You are using Safari. HOORAY";
}
}
</script></head>
```


Checking for a Browser and Alerting the User Code Listing (Cont.)

```
<body onLoad = "Init()">
<h1> Browser Information</h1>
<CENTER> <TABLE BORDER>
<FORM NAME="Browser">
<TR>
<TD> Browser Name: </TD>
<TD> <INPUT TYPE="txt" NAME="Name" Size="45"></TD>
</TR>
<TR>
<TD> Browser Version: </TD>
<TD> <INPUT TYPE="txt" NAME="Version" Size="45"></TD>
</TR>
<TR>
<TD> Browser Code Name: </TD>
<TD> <INPUT TYPE="txt" NAME="Code" Size="45"></TD>
</TR>
<TR>
<TD> User-Agent: </TD>
<TD> <INPUT TYPE="txt" NAME="Agent" Size="45"></TD>
</TR>
</FORM>
</TABLE></CENTER>
</body></html>
```

Writing to a Different Frame

We are yet to meet HTML frames so we'll defer this **until later**.

More Examples

Lots more at:

http://javascript.internet.com

Bookmark this site!!.