B211 Internet Computing

Client-side Scripting with JavaScript

Learning Objectives

1. Understand the purpose of JavaScript.
2. Learn to write JavaScript code and embed them in a HTML page.

Lecture Outline

- The purpose of JavaScript
- Embedding JavaScript in HTML
- JavaScript Language Constructs

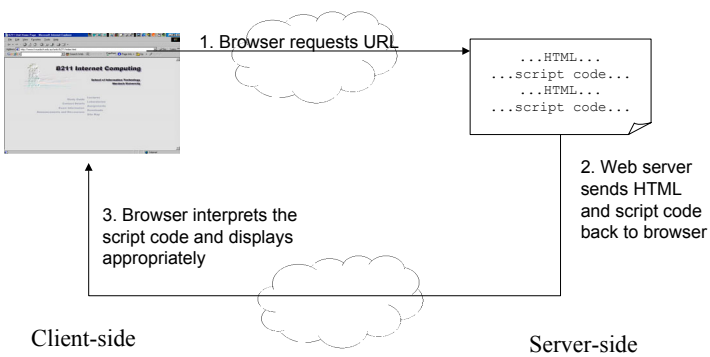
The Purpose of JavaScript

- The original reason Netscape developed JavaScript is to allow dynamic HTML (DHTML)
  - HTML only allows content to be marked up and "displayed", but does not include facilities for the person creating the web page to put in behaviours that change once it is on downloaded to the client-machine.
- We will go through the history and role of JavaScript and DHTML in *Topic 4: Internet Application Development - Programming, Scripting and Mark-up Languages.*

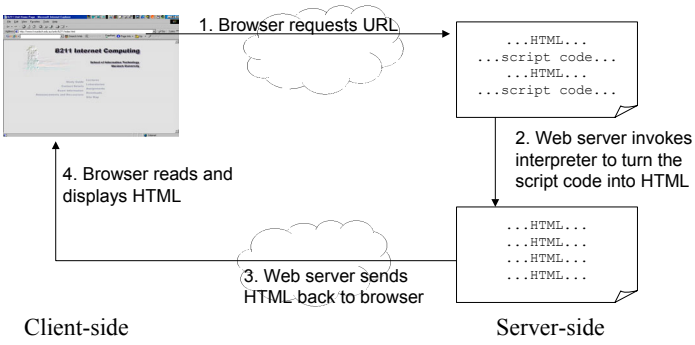
JavaScript as Client-side Technology

- JavaScript is commonly used as a client-side technology
  - The JavaScript code is downloaded to the web client and executed by the client.
  - The web client has full access to the original JavaScript source code - this means that users have the ability to read the original code.
  - JavaScript cannot do direct manipulation of resources on the server-side (eg, access to data in a central database).
- Netscape also has a language called server-side JavaScript, meant for processing on the server side
  - More on this in Topic 4..

How Client-side Scripting Works



How Server-side Scripting Works



Introductions to Programming

- The facilities available in the JavaScript language is similar to a lot of the most popular languages around today: *object-oriented* programming with *structured* programming constructs.
- Before we get into what JavaScript, let us look at these facilities.
  - The next few pages are mainly for students who haven't done ANY programming at all.
  - Those who have done previous programming, such as in B102, can skip to page 12.

Structured Programming

- In structured programming, we have facilities such as the following:
  - Statements: lines of code that are executed, one after another.
  - Data types: what kind of data is available (eg. characters, intergers, real numbers, etc)
  - Variables: places where data values can be stored.
  - Conditionals: blocks of statements that are executed when a certain conditions are true/false.
  - Loops: blocks of statements that are repeated done - how many times depends on a condition.
  - Functions/procedures: blocks of statements that are executed depending on certain parameters

Object-Oriented Programming

- In object-oriented (OO) programming, we have higher level facilities, principally
  - Objects: predefined things that you can use to do complicated things, without you having to explicitly program them.

The Art of Programming

- The task of programming then involves using the facilities mentioned in the last 2 pages, and come up with a program to do what we want.
- This generally involves learning to use methods others have derived for common purposes, and then using your own ingenuity to derive new ways of using the facilities available to accomplish novel operations.

The JavaScript Language

- Although JavaScript is limited in what it can do, it still contains the basic programming language constructs for us to define interesting behaviours.
- All of the the basic facilities mentioned before is available in the JavaScript language.

Storing Data in Variables

- In all programs, behaviours are defined by storing data, and manipulating data, and exporting data. The data are usually stored in variables.
- A variable is basically a name that has a value. You can change the value stored in that name.
- You declare (create) a variable by using the **`var`** keyword.

Example Code with Variables

```
var x ;

var y = 0 ;

var z = x + y ;

x = z - 1 ;
y = 2 * 2 ;

var my_name = "John" ;

var my_other_name = "Jane" ;
```


Data Types

- All variables have a data type:
  - numeric: a number
  - strings: a sequece of characters
  - Boolean: true or false
  - null: undefined
- In JavaScript, you do not need to explicitly say what data type a variable is - it is inferred from the values you give it.
  - Eg. "123" would be inferred as a numeric type, while "one" would be a string type.

Comments

- In the JavaScript code, when the string // is encountered, all characters after it in the same line will be ignored by the browser.
  - That can be used as comments.
- Eg.

```
var email ; // The email address typed in by the user
```

As far as the program goes, this section might as well not exist. It is only useful for humans reading it.

Control Statements

- The JavaScript code exists in statements.
- All simple one line statements separated by semi-colon are executed from top to bottom.
- We can do more interesting behaviour by using more interesting *control statements*.
  - if…else
  - for
  - while

if…else

- The if-statement is a *conditional*
  - It allows us to define a decision - under this condition, do this; under that condition, do that; etc.
- Format:

```
if (condition) {
      code
} else {
      code
}
```

for loops

- Loops are for executing a section of code repeatedly.
- The for loop controls how many times the code is repeated by having a control variable.
- General format:

```
for (counter=starting value ;
      counter <= ending value ;
            count++)
{
      code
}
```


while loops

- Another more general form of loops is the while loop.
- Format

```
while (condition)
{
      code
}
```


Operators

- JavaScript has a set of operators which can be used for
  - testing, or
  - manipulate variables
- Eg.
  - >, <, <=, >=, ==, != : to test the relationship between two numbers
  - &&, || : logical AND and logical OR
  - ++, -- : increase or decrease by 1
  - +=, -=, *=, /= : add/subtract/multiply/divide, followed by assign

Functions

- Sections of code can be put into functions.
- Functions are defined using the **function** keyword, of and has the form:

```
function name(parameters)
{
      code for the function
}
```


Function Return Values

- All functions can have a return value.
- Functions are defined using the **function** keyword, of and has the form:

```
function name(parameters)
{
      code for the function
      return value
}
```


Events and Event-Handlers

- JavaScript is an event-driven system. All code is executed based on triggering certain events (eg. the user clicks a “Submit” button).
- Programmers program the behaviours of different events by associating code to *event-handlers.*
  - Whatever you code you associate to an event-handler, that code will be invoke automatically when the event occurs.

Events and Event-Handlers

- Some event-handlers in JavaScript:
  - onBlur
  - onClick
  - onFocus
  - onLoad
  - onMouseover
  - onSelect
  - onSubmit
  - onUnload

Objects

- Objects is a programming language construct which puts together a few of the basic constructs above to allow us to encapsulate and manipulate concepts a lot easier.
- An object consist of:
  - A set of data
  - A set of methods/functions/operations which can manipulate the data.

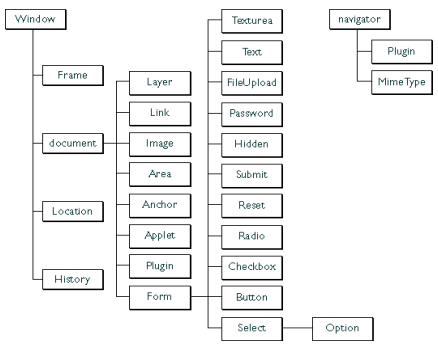
Objects in JavaScript

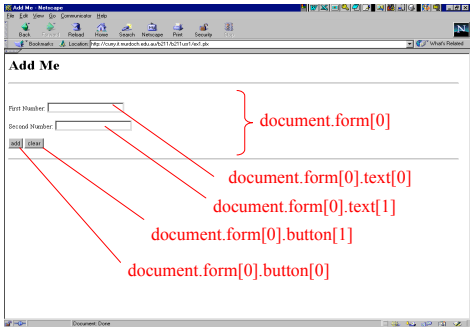
- In JavaScript, you have the option of creating your own objects.
  - You will not be required to do so in this unit.
- In this unit, you will follow the common practice of making full use of the objects already defined JavaScript and supported by the browsers.
  - These objects corresponding to the "elements" which exists in a browser windows and the downloaded HTML document.
- The objects are based on the *Document Object Model* (DOM)

The Document Object Model (DOM)

- The DOM details the characteristic properties of each element of a Web page
  - client-side dynamic technologies (such as JavaScript) makes use of DOM to refer to elements on the page.
  - DOM allows these technologies to define how to manipulate these elements and therefore manipulate the page.
- Originally released by Netscape, then Microsoft developed their own version (similar but not exactly the same). W3C currently develops a standardized version.

Example Objects in Netscape's DOM





You can also refer to the same object using the names given in the NAME attributes of the HTML tags

Embedding JavaScript in HTML

- There are four ways to add JavaScript to HTML

1. As statements and functions within a <SCRIPT> tag.
   Eg.
   ```
   <SCRIPT LANGUAGE="JavaScript1.1">
   …
   </SCRIPT>
   ```
   or
   ```
   <SCRIPT type="text/javascript">
   …
   </SCRIPT>
   ```


Embedding JavaScript in HTML (cont'd)

2. By specifying a file as the JavaScript source.
   Eg.
   ```
   <SCRIPT SRC="myscript.js">
   …
   </SCRIPT>
   ```

3. By specifying a JavaScript expression as the value of an HTML attribute.
   Eg.
   ```
   <HR WIDTH="&{barwidth};%" ALIGN="LEFT">
   ```
   where barwidth is a JavaScript variable

Embedding JavaScript in HTML (cont'd)

4. As event handlers within certain other HTML tags (mostly form elements).
   Eg.
   ```
   <INPUT TYPE="button"
     NAME="Button1" VALUE="Open Sesame!"
     onClick="window.open('mydoc.html', 'newWin')">
   ```

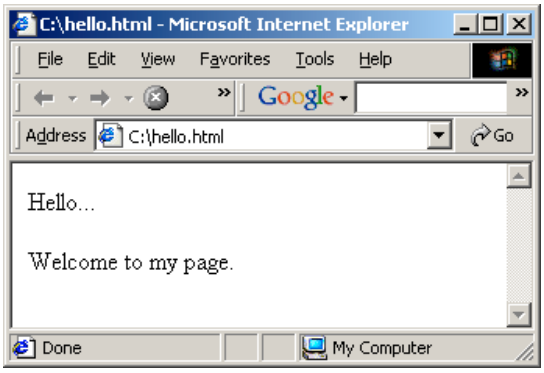

Common Use of JavaScript

- Write a message when the document first loads:

```
<html>
    <head>
        <script language="javascript">
            document.write("Hello...") ;
        </script>
    </head>

    <body>
        <p>Welcome to my page.</p>
    </body>
</html>
```

hello.html



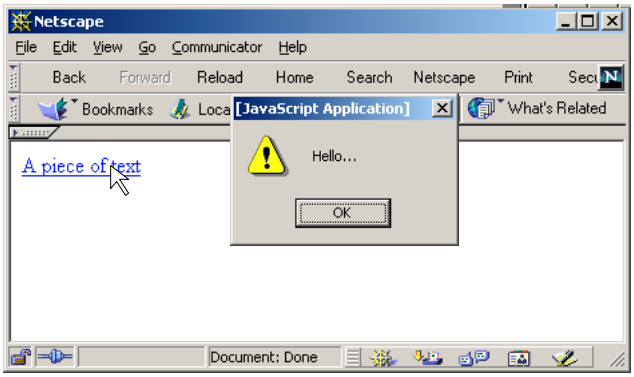
Common Use of JavaScript

- Pop up a message when the mouse moves over some text:

```
<html>
    <head>
        <script language="javascript">
            function testfunction( ) {
                window.alert("Hello...") ;
            }
        </script>
    </head>

    <body>
        <a href="#" onMouseover="testfunction( );">A piece of text</a>
    </body>
</html>
```

testfunction.html



Common Use of JavaScript

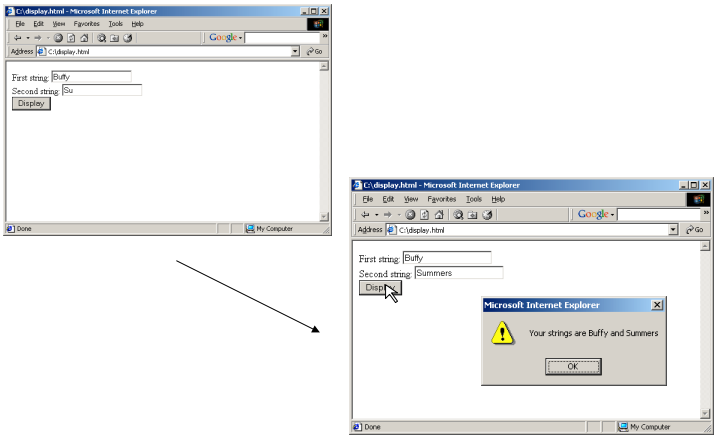
- Displaying 2 strings typed in by the user:

display.html

```
<html>
    <head>
        <script language="javascript">
            function display (Form) {
                window.alert ("Your strings are " + Form.FirstStr.value + " and " + Form.SecondStr.value) ;
            }
        </script>
    </head>

    <body>
        <form onSubmit="display(this) ;">
            <p>
                First string: <input type="text" name="FirstStr"><br>
                Second string: <input type="text" name="SecondStr"><br>
                <input type="submit" value="Display">
            </p>
        </form>
    </body>
</html>
```

The keyword **this** refers to the current element, in this case, the form.



Practical work

- I will not put up and explain significant sections of code in this lecture.
- We will go through actual full examples of JavaScript code from the textbook in week 5 and 6's lab.
- You can also learn by viewing the sources of web pages with JavaScript
  - But remember that different people (and different page generators) will implement JavaScript differently.
  - You can look at all the examples on pages in the B211 site if you want a set consistent with what I teach.

Browser Implementations

- The textbook presents JavaScript supposedly common to both Netscape Navigator and Microsoft Internet Explorer, but other browser implements them differently.
  - Eg. you can use a newline character to separate statements instead of a semicolon in IE.
- Always **rigorously** test your scripts on different browsers with different inputs to ensure they work properly.