CM0133 Internet Computing

Introduction to JavaScript

Program Code in Web Pages

There are many code types that can be included in web pages to increase interactivity:

- Java Applets
- Active X
- Flash
- VB Script
- ASP
- JSP
- PHP
- Perl
- JavaScript

How to decide for a language?

A posting from a programming newsgroup:

I have written a VBScript codes to develop my webpage. I have always tested it using IE. Today, I learned that when I am using Mozilla, none of my VBScipt codes are fired.

Can some please advice?

Server vs. Client

Server sided programming languages:

- generate traffic for each request
- the response can be slow
- other servers might be necessary to deal with the load

Client sided programming:

- Complex calculations take long time and depend on the client resources
- DB interaction is not possible

Proprietary vs. Standard

- ECMA-262 is the official JavaScript standard. The standard is based on JavaScript (Netscape) and JScript (Microsoft).
- The language was invented by Brendan Eich at Netscape (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.
- The development of ECMA-262 started in 1996, and the first edition of was adopted by the ECMA General Assembly in June 1997.
- The development of the standard is still in progress.

Javascript

- A programming tool (e.g. http://www.w3schools.com/js/default.asp)
- can put dynamic text into an HTML page - A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- can read and write HTML elements - A JavaScript can read and change the content of an HTML element
- JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

Example 1

```
<head>
 <title>Example 1</title>
 <script language="javascript">
 <!--
  function showMessage()
  {
   document.writeln("<h3>This is some text ...</h3>");
   document.writeln("Note the following:");
   document.writeln("<ul>");
   document.writeln(" <li>writeln is a ...</li>");
   document.writeln(" <li>The output can...</li>");
   document.writeln("</ul>");
  }
  //-->
 </script>
</head>
<body>
 <script language="javascript">
   showMessage();
 </script>
</body>
```

JavaScript

- JavaScript syntax is very similar to that of Java and C
- JavaScript has
  - variables, objects and functions (methods)
  - blocks of code {}
  - text output
  - input forms and dialog boxes
  - conditional structures (**`if...then...else`**)
  - iterative structures (**`while...`** and **`for...`**)
- JavaScript has events associated with HTML elements

Events

- Events can be used to run JavaScript functions in response to user actions, e.g.
  - to validate user input from a form
  - 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.

Events

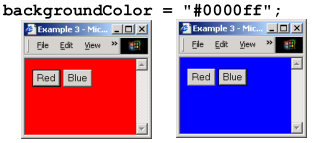
- Functions are often called in response to events associated with HTML elements
- `onLoad` is associated with the `body` element
- `onSubmit` is associated with the `form` element
- `onMouseOver`, `onMouseDown`, `onClick`, `onBlur`, `onFocus`, `...` are associated with mouse events

Selected event handlers

- `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).

Example 2

```
<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>
```



Document Object Model (DOM)

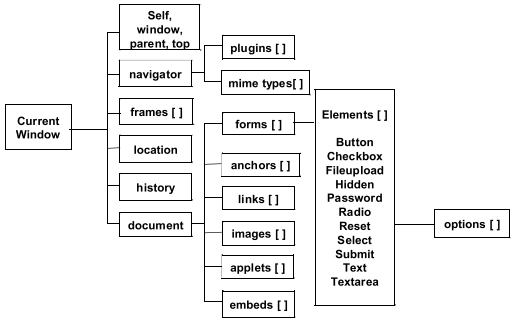
- JavaScript can manipulate objects associated with a web page. These objects have methods (such as **`writeln`**)
- The objects are arranged into a hierarchy of objects called the Document Object Model (DOM)
- At the top of the hierarchy is the window object. The document object is a child of the window object.
- Netscape and IE do not use exactly the same DOM
- In Example 1, the document object is assumed to belong to the currently open window.

Accessing document data

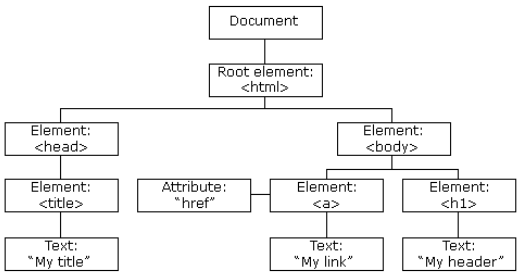
- There are many attributes associated with the objects specified in the DOM
- These correspond to the attributes of the HTML elements represented by the objects
- For example, if a form has an input element called **`address`**, the associated text can be obtained via the document object model using

```
document.forms[0].address.value
```


Summary of DOM



Document DOM



Arrays of objects

- The DOM specifies various arrays of objects
  - each array is indexed from 0
- The `anchors[]` array contains every anchor contained in the document (in the order in which they appear)
- The array elements can also be referenced by the value of their `name` attribute
  - For example, if the first form on the web page is called `"orderForm"` then
  - `forms[0]` is equivalent to `forms['orderForm']`

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">
```

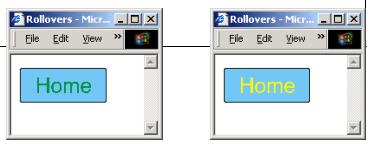
- The source of this image is stored in the object

```
document.homeImage.src
```

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

Rollovers

```
<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>
```



Demo

- Hello World!
- onload function as a simple example of an event
- alert command as a simple javascript function

Importing JavaScript

- Say you want to include the same piece of JavaScript in multiple pages.
- It is a bad idea to copy and paste this code into each page. Why do you think this is?
- A better solution is to write the JavaScript once, and then import it into each page.

```
<script language = "javascript"
  src="myscript.js"></script>
```

Where to put the code?

- Function definitions can be placed inside the `<script>` element within the **`head`** element
- These functions may then be called by JavaScript commands inside the **`body`** element.
- The contents of the **`script`** element should be enclosed in a HTML comment so that it can be ignored by older browsers
- The **`<script>`** tag should include the attribute-value pair `language="javascript"`
- The code can be contained in a file that is referenced by the page.

<noscript>

- Browsers that do not support JavaScript will display JavaScript as page content.
- To prevent them from doing this, and as a part of the JavaScript standard, the HTML comment tag can be used to "hide" the JavaScript. Just add an HTML comment tag <!-- before the first JavaScript statement, and a --> (end of comment) after the last JavaScript statement.
- <noscript>Your browser does not support JavaScript! </noscript>

Variables

- Variables are declared using the **`var`** keyword, values are assigned using the equals operator (=)

```
var name, age;
var today = "Tuesday";
```

- Variable names must begin with a letter, digit, or the underscore (_).
- Spaces are not allowed in variable names.
- Names are case-sensitive, so that fred, FRED and frED all refer to different variables.
- Reserved words are part of the JavaScript language and thus not allowed as variable names.

Lifetime of Variables

- When you declare a variable within a function, the variable can only be accessed within that function. When you exit the function, the variable is destroyed. These variables are called local variables. You can have local variables with the same name in different functions, because each is recognized only by the function in which it is declared.
- If you declare a variable outside a function, all the functions on your page can access it. The lifetime of these variables starts when they are declared, and ends when the page is closed.

Data Types

- **Numeric:** basic numbers can be integers (1,22,333) or floating point (-12.34, 3E45). There is no need to differentiate between them. Anything not used in a mathematical expression is not a number for javascript.
- **Strings:** collection of characters that are not numbers. The value of a string can contain spaces and may be totally made up of digits, e.g. “Internet Programming Class”, “Tuesday”, “1610-1700”. Use single quotes (') or double quotes with a backslash (\) to nest strings (just one nesting!), e.g.

  var aStr = “This is a 'bStr' in aStr”;
  var bStr = “This is a \”cStr\” in a bStr”;
- **boolean:** Are variable that hold either a true or false value.
- **null:** Variables that have not been given a value yet (it is NOT nil or zero !!!).

Arrays

- The Array object is used to store multiple values in a single variable.

```
var myColors=new Array("red","green","blue");
```

- To access and to set values inside an array, you must use the index numbers as follows:
  - myColors[0] is the first element
  - myColors[1] is the second element
  - myColors[2] is the third element

Array functions

- `join(sep)`
  - joins the array elements into a string (separated by **`sep`**)
- `pop()`
  - removes the last element of the array
- `push(element)`
  - adds an element to the end of the array
- `reverse()`
  - reverses the order of the array elements
- `shift()`
  - removes the first element of the array
- `sort()`
  - sorts the array into lexicographic (dictionary) order

String manipulation

- `charAt(index)`
  - returns the character at position **`index`** in the string
- `concat(string)`
  - concatenates two strings
- `length()`
  - returns the number of characters in the string
- `split(separator)`
  - breaks the string apart whenever it encounters the **`separator`** character (pieces returned in an array)
- `toUpperCase()/toLowerCase()`
  - converts the string to upper/lower case

String manipulation II

- **concat ("string"[,"string"[, ..."]])**
  - Serves to join strings, alternatively use the "+" operator, but this may lead to unexpected results as "+" is not explicit about what it is doing.
- **indexOf("search"[,offset])**
  - searches for the string in the first parameter and returns the start of the target string, returns -1 if unsuccessful
- **substr(index[,length])**
  - Returns a substring which starts at the character indicated by the index parameter
- **substring(index1[,index2])**
  - Returns the set of characters which start at index1 and continues up to index2-1.

Basics

- The syntax of control structures is almost identical to that of the Java language, including
  - `while` loops and `for` loops
  - `if` statements and `break` statements
- Arrays are easily defined (index starts at zero)

```
var days = ["Monday", "Tuesday", "Wednesday"];
```

- Mathematical functions are methods of the `Math` class

```
result = Math.sqrt(value);
```


Arrays

- Array declarations:

```
var values = new Array(100);
var nums = [3, 6, 66, 3, 8, 10, 99];
```

- Example

```
var maxval = 5;
var myArray = new Array(5);
for (j=0; j<maxval; j++) myArray[j] = 2*j;
for (j=0; j<maxval; j++) {
  document.writeln("value " + j + " is " + myArray[j]);
  document.writeln("<br>");
}
```


`for` loops

```
for (initialise counter; test condition; increment) {
   do something;
}
```

```
var i;
var myArray = [1,1,2,3,5,8,13];
for(i=0; i<myArray.length(); i++) {
  document.writeln("value is " + myArray[i]);
  document.writeln("<br>");
}
```

`while` loops and `do` loops

`while` (condition is true) {do something }

```
count=0;
while(count < maxval) {
  document.writeln("value is " + myArray[count]);
  document.writeln("<br>");
  count = count + 1;
}
```

`do` {something} `while` (condition is true);

```
count=0;
do {
  document.writeln("value is " + myArray[count]);

  document.writeln("<br>");
  count = count + 1;
} while (count < maxval);
```


`if` statements

`if` (condition) {do something}

```
if(scoreA > scoreB) {
  document.writeln("The winner is A")
}
```

```
if(scoreA > scoreB) {
  document.writeln("The winner is A")
}
else if (scoreA < scoreB) {
  document.writeln("The winner is B")
}
else {
  document.writeln("Everyone's a winner")
}
```


Functions

- A function contains code that will be executed by an event or by a call to that function.
- You may call a function from anywhere within the page (or even from other pages if the function is embedded in an external .js file).
- Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that the function is read/loaded by the browser before it is called, it could be wise to put it in the <head> section.

```
function functionname(var1,var2,...,varX)
{
   some code
}
```

Functions II

- The return statement is used to specify the value that is returned from the function.

```
function add(a,b)
{
   var sum = a+b;
   return sum;
}
```


Mathematical functions

- `abs(value)`
  - returns the absolute value (modulus)
- `sin(value)/cos(value)/tan(value)`
  - trigonometric functions
- `parseInt(value)/parseFloat(value)`
  - converts a string to an integer/floating point number
- `ceil(value)`
  - returns the smallest integer greater than **`value`**
- `max(value1,value2)`
  - returns the larger/smaller of its two arguments
- `random()`
  - returns a pseudorandom number between 0 and 1

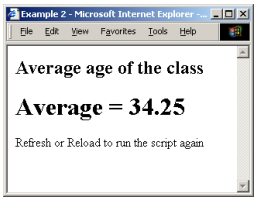
Example 3

```
<head>
 <script language="javascript">
 <!--
  var age, averageAge, counter, total, ageValue;
  total = 0; ageValue = 1; counter = 0;

  document.writeln("<h2>Average age of the class</h2>");
  while (ageValue > 0) {
    age = window.prompt("Enter age (0 to end):","0");
    ageValue = parseInt(age);
    if (ageValue > 0) {
      total = total + ageValue;
      counter++;
    }
  }
  averageAge = total/counter;
  document.writeln("<h2>Average = " + averageAge +"</h2>");
 //-->
 </script>
</head>
<body>
Refresh or Reload to run the script again
</body>
```

Example 3

- `var` declares variables
- `writeln` is a method of the `document` object
- `prompt` is a method of the `window` object
- `parseInt` converts a string to an integer
- `+` is used to include variables in strings



JavaScript

- Fragments of JavaScript code can be executed in response to events
  - main modules and exit statements are not necessary (unlike in standard Java..)
- JavaScript is an interpreted language
  - the script is compiled at run-time (unlike Java !)
- JavaScript has no strict data typing (unlike Java !)
  - It is not necessary to declare the type of a variable (except those of type `object`).
  - The interpreter will guess the data type (from the surrounding context)

Available Libraries

- Many simple and repetitious tasks have been programmed by others and are available for your use
- Javascript has a plethora of libraries available on the Web. They are likely to be faster and more efficient in lot of cases and often have been tested for cross browser compliancy.
- Examples: http://javascriptlibraries.com/

Info

- Information around the lecture and to coursework can be found on: http://users.cs.cf.ac.uk/F.A.Twaroch/teach.html
- Lecture on 18th March is cancelled !
- Come to the labs and train the theoretic contents in practice. Use also the Internet as source of information (google,yahoo), lots of examples and tutorials can be found there, e.g. : http://www.w3schools.com/
- A Texteditor that supports syntax highlighting maybe useful, e.g. Textpad, EditPad Lite, Crimson Text Editor (Windows), gedit, nedit, kate, quanta, bluefish (Linux) and many many more ...