CM0133 Internet Computing

JavaScript 2

Outline

- Recap – Javascript
- Verifying form input
  - using an alert box
  - using a message window
- Regular Expressions
- Location coordinates & Animation
- Inner Functions
- Browser sniffing
- Object Oriented Javascript
- jQuery – A Javascript Library
- Debugging

Recap - What is it used for ?

- Handling User Interaction
  - Doing small calculations
  - Checking for accuracy and appropriateness of data entry from forms
  - Doing small calculations/manipulations of forms input data
  - Search a small databased embedded in the downloaded page
  - Save data as cookie so it is there upon visiting the page
- Generating Dynamic HTML documents
- Examples
  - Bookmarklets
  - Google Maps
  - Google Suggest

Variables

- Variable type is not specified explicitly, but determined by assignment
- Numbers:

```
var valueA, valueB;
var valueC = 22, valueD = 64.8;
valueA = 22;
valueB = valueA + valueC;
```

- Strings:

```
var str1, str2;
var str3 = "hello there";
str1 = "I wonder";
str2 = str3 + " how are you";
```

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")
}
```

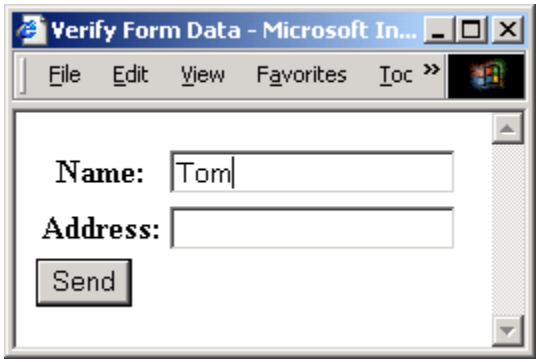
```
if(scoreA > scoreB) {
  document.writeln("The winner is A")
}
else if (scoreA < scoreB) {
  document.writeln("The winner is B")
}
else {
  document.writeln("Everyone's a winner")
}
```

Example 1: Verifying form input

- We use event handlers to process form data (supplied by the user) on the client side
- The `onSubmit` event handler to is called when the submit button is pressed
- The form data can be checked using JavaScript, and the user can be notified of any errors using an alert box or a message window
- The `onFocus` and `onBlur` mouse event handlers can be called when the mouse moves over the form elements, for example to provide help messages
- Form data can be accessed using the `name` attributes

Example 1

```
<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>
```



file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/verifyForm.html

Example 1

- Function **`verifyForm`** takes a form as a parameter
- **`username`** is the name of an input element
- The corresponding text is tested against an empty string
- If the text value is empty, an appropriate message is issued using an alert window.

```
onSubmit = "return verifyForm(this)"
```

- **`onSubmit`** is an event associated with forms
- **`this`** is the name of the current form object
- Form submission fails if **`onSubmit=false`**

Example 2

```
<form name="feedbackForm"
  onSubmit="readForm(this)">
<h2>Tell us what you think</h2>
Name: <input type="text" name="username"><br>
Address: <input type="text" name="address" size="35"><br>
<table><tr><td>
How did you find this web site?:<br>
Friend <input type="checkbox" name="how" value="friend"><br>
Google <input type="checkbox" name="how" value="google"><br>
Other  <input type="checkbox" name="how" value="other"><br>
</td><td>
How do you rate this site?
<select name="rating">
<option value="good">Good
<option value="bad">Bad
<option value="ugly">Ugly
</select>
</td></tr></table>
<h3>Thank you</h3>
<input type="submit" value="Send">
<input type="reset" value="Clear">
</form>
```

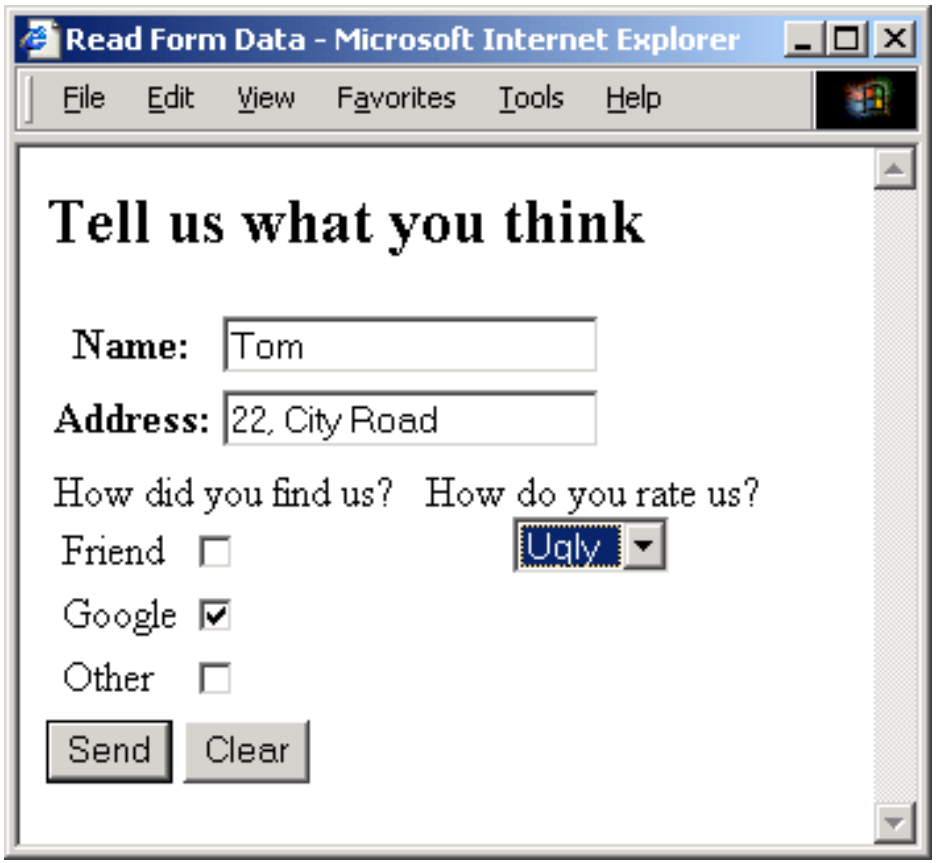
Example 2

```
function readForm(theForm)
{
  var i, optIdx, rating;
  win = window.open("","messageWin");
  win.document.writeln("<h2>The name is: "
                       +theForm.username.value+ "</h2>");
  win.document.writeln("<h2>The address is: "
                       +theForm.address.value+ "</h2>");

  for (i=0; i<3; i++) {
    if(theForm.how[i].checked) {
      win.document.writeln("<h2>You found us by: "
                        +theForm.how[i].value+ "</h2>");
    }
  }
  opIdx = theForm.rating.selectedIndex;
  rating = theForm.rating.options[opIdx].value;
  win.document.writeln("<h2>Rating: " +rating+ "</h2>");

  win.close;
}
```

Example 2



A message window created in response to user input

Regular expressions

- Regular expressions are used for text processing
- Regular expressions are string patterns
- We test whether or not a string matches a given pattern
- static: `pattern = /fish|fowl/;`
- dynamic: `pattern = new RegExp("fish|fowl");`

```
var pattern = new RegExp("fish|fowl");
var myString "Can you find the fish?";
var results = pattern.exec(myString)
```

- Regular expressions are an integral part of the Perl scripting language. Refer to lecture 7 for details.

Regular expressions

`^`	Match at the start of the string
`$`	Match at the end of the string
`*`	Match 0 or more times
`+`	Match 1 or more times
`?`	Match 0 or 1 time
`a|b`	Match `a` or `b`
`{n}`	Match the string `n` times
`\d`	Match a digit
`\D`	Match anything except digits
`\w`	Match any alphanumeric character or underscore
`\W`	Match anything except alphanumeric characters or underscores
`\s`	Match a whitespace character
`\S`	Match anything but a whitespace character
`[...]`	Match any character in the set (range defined using a hyphen: `[A-Z]`)
`[^...]`	Match any character not in the set

Class `string` functions

- **`match(pattern)`**
  - searches for a matching pattern. Returns an array holding the results (or null if no match is found)
- **`replace(pattern1, string1)`**
  - searches for pattern1. If the search is successful, pattern1 is replaced by string1
- **`search(pattern)`**
  - searches for matching pattern. If the search is successful, the index (offset) of the start of the pattern is returned
- **`split(pattern)`**
  - splits the string into parts based on the pattern (or regular expression) which is supplied as a parameter

Class `RegExp` functions

- **`exec(string)`**
  - executes a search for a matching pattern. Returns an array holding the result(s) of the operation
- **`test(string)`**
  - executes a search for a matching pattern. Returns true if a match is found, false otherwise

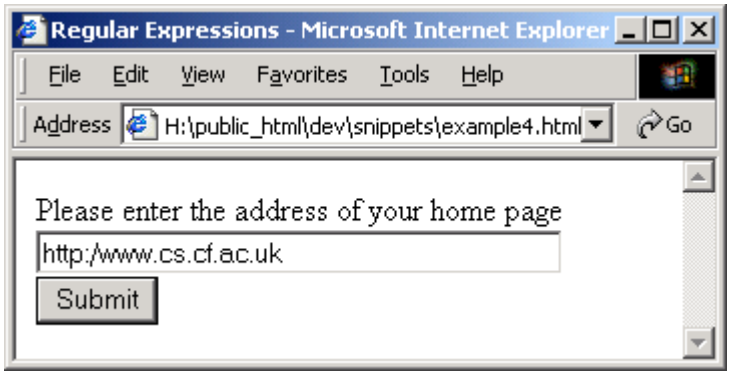
Flags

- `i`: ignore the case of the pattern and input string
- `m`: search of data which spans several input lines
- **`g`**: rather than stopping when a match is successful, this forces global matching across all of the input

Example 3

```
<script language="javascript">
<!--
  function checkURL()
  {
    var url = document.forms[0].homeURL.value;
    var url_pattern = new RegExp("^http:\/\/\\w+[.]\\w+");
    if(!url.match(url_pattern)) {
      alert("URL not valid");
      return false;
    }
    document.write("Thank you");
    return true;
  }
-->
</script>

<form method="POST" action="processing.php"
onSubmit="return checkURL()">
      <p>Please enter the address of your home page</p>
      <p><input type="text" name="homeURL" size="40"></p>
      <p><input type="submit" value="Submit"></p>
  </form>
 </body>
```



file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/regexp.html

Modifying the values of DOM objects

- The values of style attributes can be accessed as children of the style sub-object

```
document.elementName.style.left
```

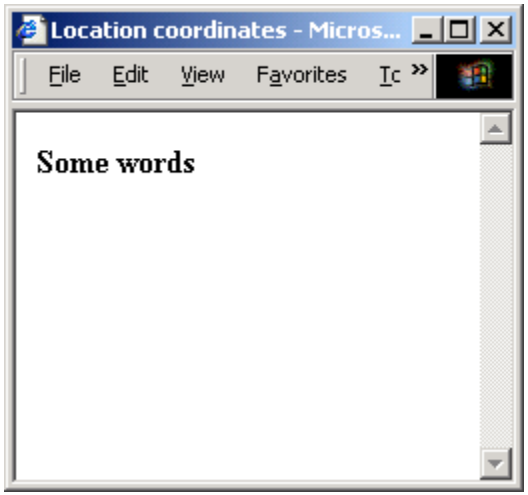
- This refers to the value of the left position attribute for the element named **`elementName`**
- The document content of an element can be accessed using **`innerHTML`**

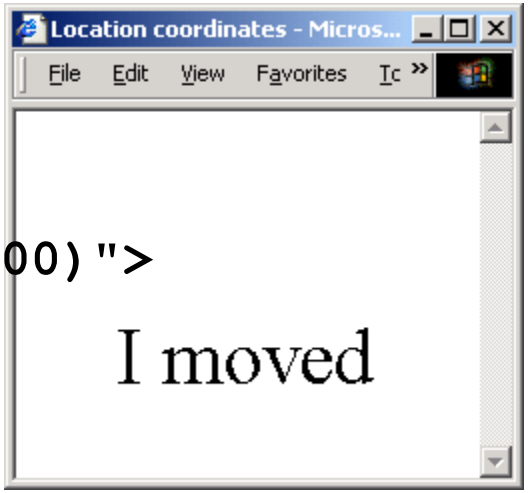
```
para1.innerHTML = "hello there"
```

- This changes the text in paragraph **`para1`**

Example 4

```
<html>
 <head>
  <title>Location</title>
  <style>
    #para1 {position: absolute}
  </style>
  <script language="javascript">
  <!--
    function move(elementid,x,y) {
      elementid.style.left = x;
      elementid.style.top = y;
      elementid.innerHTML="I moved";
      elementid.style.fontSize = 40;
    }
  -->
  </script>
 </head>
 <body>
  <p id="para1" onClick="move(this,50,100)">
    Some words
  </p>
 </body>
</html>
```





file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/location.html

Multiple Contents in One Page

```
<script>
<!--
function switchContent(id) {

        document.getElementById('content1').style.display ='none';
        document.getElementById('content2').style.display ='none';
        document.getElementById('content3').style.display ='none';

        document.getElementById(id).style.display ='block';



}
-->
</script>
```

file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/multiContent.html

`setInterval` and `clearInterval`

- Used to call JavaScript code repeatedly at a specified time interval

```
window.setInterval(code, interval)
```

  - **`code`** is a string of JavaScript code (in quotes)
  - **`interval`** is time in milliseconds

```
window.clearInterval(interval_Id)
```

  - **`interval_Id`** is the value returned by **`setInterval`**

- To delay the execution of code without repeat, use **`setTimeout`** and **`clearTimeout`** (same parameters)

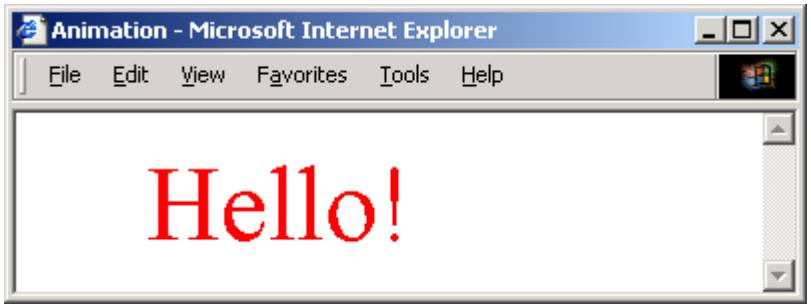
Animation

```
<style>
   #para1 {position: absolute;
     color : red;
     font-size : 40pt}
 </style>
 <script language="javascript">
   var x = 20, xshift = 1;
   function move(elementId) {
     x = x + xshift;
     elementId.style.left = x;
     if (x > 150) {
       xshift = -1;
       elementId.innerHTML = "Goodbye!";
       elementId.style.color = "blue";
     }
     if (x < 20 ) {
       xshift = 1;
       elementId.innerHTML = "Hello!";
       elementId.style.color = "red";
     }
   }
 </script>
```

Animation

```
<body onLoad="intervalId = setInterval('move(para1)',10)"
      onClick="clearInterval(intervalId)">

 <p id="para1" >
    Hello!
 </p>
</body>
```



file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/animation.html

Inner functions

- Functions do not all have to be defined at the top level (or left edge), Functions can be defined inside of other functions.
- An inner function has access to the variables and parameters of functions that it is contained within. This is known as Static Scoping or Lexical Scoping.
- The scope that an inner function enjoys continues even after the parent functions have returned. This is called *closure*.

Example

```
function fade(id) {
    var dom = document.getElementById(id),
        level = 1;
    function step () {
        var h = level.toString(16);
        dom.style.backgroundColor =
            '#FFFF' + h + h;
        if (level < 15) {
            level += 1;
            setTimeout(step, 100);
        }
    }
    setTimeout(step, 100);
}
```

/Dropbox/teaching/CM0133/examples/javascript/innerFunction.html

Browser sniffing

- The w3c have specified a standard document object model for web browsers (see `http://www.w3.org/DOM/`)
- The Netscape DOM and the Internet Explorer DOM do not comply with the standard DOM
- However, Internet Explorer 5+ and Netscape 6+ have agreed on new set of standards which is closer to the W3C DOM than previous versions
- We need to detect the type of browser that is accessing our page. This is called browser sniffing.
- Information about the browser is contained in the **`navigator`** object of the DOM

Browser sniffing

- Information about the browser is contained in the **navigator** object of the document object model
- Using the **appName** property

```
var browser = navigator.appName;
var Nav = (browser == "Netscape");
var IE = (browser == "Internet Explorer);
```

- Using the **appVersion** property

```
var Nav = navigator.appVersion.indexOf("Nav")>0;
var IE  = navigator.appVersion.indexOf("MSIE")>0;

var version = parseInt(navigator.appVersion);
var Nav4 =( Nav && version>=4);
var IE4 = (IE && version>=4);
```

Object detection

- Rather than explicitly detecting the client (then using that particular client's objects and methods), we simply check to see whether an object exists before using it.

```
var isNetscape = (document.layers);
var isIE = (document.all);
var isStandard = (document.getElementById);
```

- The existence of every object and method should be tested before they are used.
- It is risky to assume that if **`document.all`** exists, then the client is running on Internet Explorer (and hence that we can use all of IE's DOM, not just **`document.all`**)

Objects in Javascript

- Native objects are those objects supplied by JavaScript. Examples of these are String, Number, Array, Image, Date, Math, etc.
- Host objects are objects that are supplied to JavaScript by the browser environment. Examples of these are window, document, forms, etc.
- And, user-defined objects are those that are defined by you, the programmer.
- Javascript has developed into an object based language and large code frameworks, e.g. google maps, YUI (http://developer.yahoo.com/yui/) are written exploiting the object oriented programming paradigms.

Object Oriented Languages

- Encapsulation - Support for method calls on a JavaScript object as a member of a Class.
- Polymorphism - The ability for two classes to respond to the same (collection of) methods.
- Inheritance - The ability to define the behaviour of one object in terms of another by sub-classing.

http://mckoss.com/jscript/object.htm

Prototypes

- JavaScript uses prototypes instead of classes for inheritance. It is possible to simulate many class-based features with prototypes in JavaScript.
- The prototype object of JavaScript, introduced starting in JavaScript 1.1, is a prebuilt object that simplifies the process of adding custom properties/ methods to all instances of an object.

Objects in JavaScript

- It looks like as if we would deal with “classes”:

```
 function Person(myName, myAge) {
    this.name = myName;
    this.age  = myAge;
}

var someGuy = new Person(“Shawn”, 28);
```

Functions - Revisited

- Functions as object constructors
  - Functions double as object constructors along with their typical role. Prefixing a function call with **new** creates a new object and calls that function with its local this keyword bound to that object for that invocation. The constructor's prototype property determines the object used for the **new object's internal prototype**. JavaScript's built-in constructors, such as Array, also have prototypes that can be modified.
- Functions as methods
  - Unlike many object-oriented languages, there is **no distinction between a function definition and a method definition**. Rather, the distinction occurs during function calling; **a function can be called as a method**. When a function is called as a method of an object, the function's local **this keyword** is bound to that object for that invocation.

Objects in Javascript

- Accessing object's properties

```
var someGuy = new Person("Shawn", 28);
document.writeln('Name: ' + someGuy.name);
```

- Objects and Associative Arrays are in fact two interfaces to the same data structure
  - Which means you can access elements of someGuy like so: someGuy["age"] or someGuy["name"]

```
document.writeln('Name: ' + someGuy["name"]);
```

Objects in Javascript

- Object Functions / Methods
  - Functions are just properties like any other property of an object (name,age, etc…)

```
function displayName() {
document.writeln("I am " + this.name);
}
```

Then the constructor will become..

```
function Person(myName, myAge) {
    this.name = myName;
    this.age = myAge;
    this.displayMe = displayName;
}
```

Object Functions

- To call the function on the object:

```
var someGuy = new Person("Shawn", 28);
someGuy.displayMe();
var someOtherGuy = new Person("Tim", 18);
someOtherGuy.displayMe();
```

Object Functions

Alternatively you may declare the function inside the constructor:

```
function Person(myName, myAge) {
    this.name = myName;
    this.age = myAge;
    this.displayMe = function()
    { document.writeln("I am " + this.name);
    }
}
```

Inheritance in JavaScript

- No built-in inheritance
- Runtime Inheritance: Clone objects and add extra properties
- Assignment: Research at least one way of doing inheritance in JavaScript. Google is your friend!

```
function extend(subclass, superclass) {
  function Dummy(){}
  Dummy.prototype = superclass.prototype;
  subclass.prototype = new Dummy();
  subclass.prototype.constructor = subclass;
  subclass.superclass = superclass;
  subclass.superproto =
superclass.prototype;
}
```

http://peter.michaux.ca/articles/class-based-inheritance-in-javascript

file:///home/florian/Dropbox/teaching/CM0133/examples/javascript/polymorphism.html

typeof

The **`typeof`** prefix operator returns a string identifying the type of a value:

type	typeof
object	'object'
function	'function'
array	'object'
number	'number'
string	'string'
boolean	'boolean'
null	'object'
undefined	'undefined'

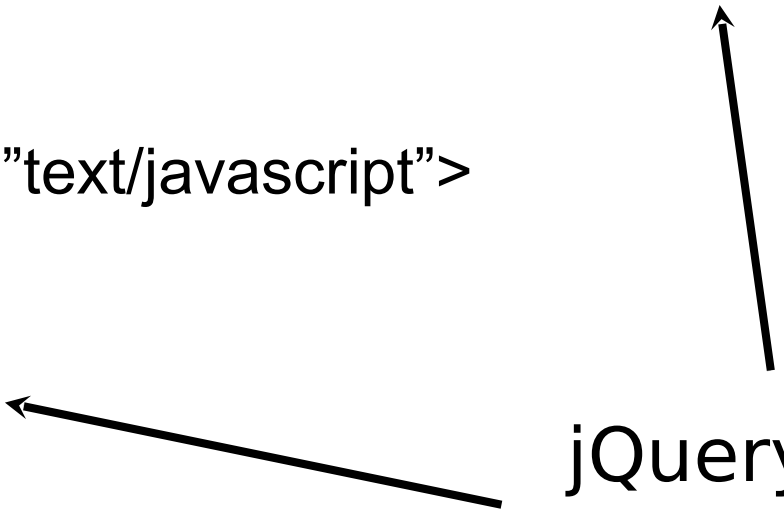
jQuery – A Javascript library

- Repetitive tasks should be centralized libraries.
- Writing your own library can be a time consuming process due to maintaining cross browser compatibility.
- Library frameworks that are extensively tested are available for you. But they come with a tradeoff you will have to learn their syntax and familiarize with their Application Programming Interface (API)
- jQuery is one of many javascript libraries : http://jquery.com/
- It got very popular for web programming and is widely used.

jQuery

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

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



jQuery library must be included before it is used anywhere in the page

jQuery

```
<script type=”text/javascript”>

$(‘#title’).hide();

$(‘a’).css(‘color’, ‘#00aa00’);

</script>
```

$ function returns a jQuery object that wraps the HTML elements

CSS selectors are used to specify which HTML element to get

Function specified is applied to all jQuery objects specified by the CSS selector

$ function can also return a list of elements

Effects

If the browser does not understand JavaScript then this link causes nothing to happen

onclick attribute contains JavaScript that is run when the link is clicked

```
<a href="#"
onclick="$('#title').hide(); return false;"
>Click to hide the title</a>
```

Causes the browser to not follow the link

Document Object Model

- Document Object Model (DOM) is how the web page looks to the JavaScript code
  - Every html tag is represented as an object
  - Tree-structure
  - Either
    - Linked to the document as rendered by the browser
    - Directly used to render the document
  - Changes to the DOM change the rendered document

Document Object Model

```
<script type=”text/javascript”>


$(document).ready(function() {

$(‘#js_warning’).remove();

});


</script>
```

Hook that is called when the document has completed loading

Anonymous function

This code is executed when the document has completed loading

Minification vs Obfuscation

- Reduce the amount of source code to reduce download time.
- Minification deletes whitespace and comments.
- Obfuscation also changes the names of things.
- Obfuscation can introduce bugs.
- Never use tools that cause bugs if you can avoid it.

http://www.crockford.com/javascript/jsmin.html

EXAMPLE: http://www.stunnix.com/prod/jo/

JSLint

- JSLint can help improve the robustness and portability of your programs.
- It enforces style rules.
- It can spot some errors that are very difficult to find in debugging.
- It can help eliminate implied globals.
- Commandline versions.
- In text editors and Eclipse.

`http://www.JSLint.com/`

Summary

- Javascript is a powerful scripting language that helps us to create
  - Dynamic contents
  - Validate data
  - Create Interactivity
- Syntactically it is very similar to Java or C++
- Javascript is Object Based rather than Object Oriented
- Don't build from scratch. Frameworks exist that help in rapid web development.
- Choose your tools carefully and learn to use them.
- Come to the labs :
  - IS Monday 3pm
  - CS Monday 10am, Tuesday 1pm and Friday 9am.