

# JavaScript Lecture I

<http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html>

## Introduction to JavaScript

- NOT Java
  - JavaScript was developed by Netscape
  - Java was developed by Sun
- Designed to 'plug a gap' in the techniques available for creating web-pages
  - Client-side dynamic content
- Interpreted

## JavaScript vs. Java

- Complementary
  - JavaScript
    - Cannot draw, multi-thread, network or do I/O
  - Java
    - Cannot interact with Browser or control content

## JavaScript vs. Java

- JavaScript is becoming what Java was originally intended to be
  - Java Applets are meant to be lightweight downloadable programs run within the browser for cross-platform compatibility
  - Java = Bloated
  - JavaScript is actually lightweight and accomplish most of what Applets do with a fraction of the resources

## What is it used for today?

- 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

## JavaScript Shell

- <http://www.squarefree.com/shell/>
- <http://www.squarefree.com/jsenv/>
- <http://www.mozilla.org/rhino/shell.html>

## JavaScript Syntax - Variables and Literals

- Declaration
  - Explicit: `var i = 12; // no 'var' in declaration`
  - Implicit: `i = 12;`
- Variable Scope
  - Global
    - Declared outside functions
    - Any variable *implicitly* defined
  - Local
    - *Explicit* declarations inside functions

## JavaScript Syntax - Variables and Literals

- Dynamic Typing - Variables can hold any valid type of value:
  - Number ... `var myInt = 7;`
  - Boolean ... `var myBool = true;`
  - Function ... [Discussed Later]
  - Object ... [Discussed Later]
  - Array ... `var myArr = new Array();`
  - String ... `var myString = "abc";`
  - ... and can hold values of different types at different times during execution

# JavaScript Syntax - Operators

- Key Comparison Operators

>	number on the left must be greater than the number on the right
<	number on the right must be greater than the number on the left
<=	number on the right must be greater than or equal to the number on the left
>=	number on the right must be less than or equal to the number on the left
!=	the numbers or objects or values must not be equal
==	the numbers or objects or values must be equal
!	Logical NOT
	Logical OR
&&	Logical AND

# JavaScript Syntax - Operators

- Key Assignment Operators

+	adds two numbers or appends two strings. If more than one type of variable is appended, including a string appended to a number or vice-versa, the result will be a string
-	subtracts the second number from the first
/	divides the first number by the second
*	multiplies two numbers
%	Modulus - divide the first number by the second and return the remainder
=	assigns the value on the right to the object on the left
+=	the object on the left = the object on the left + the value on the right this also works when appending strings
-=	the object on the left = the object on the left - the value on the right
++ / --	Increment / Decrement a number

## JavaScript Syntax – Control and Looping

- **'if' statement**

```
if ( boolean statement ) {  
    ...  
} else {  
    ...  
}
```

- **'switch' statement**

```
switch ( myVar ) {  
    case 1:  
        // if myVar is equal to 1 this is executed  
    case "two":  
        // if myVar is equal to "two" this is executed  
    case default:  
        // if none of the cases above are satisfied OR if no  
        // 'break' statement are used in the cases above,  
        // this will be executed  
}
```

## JavaScript Syntax – Control and Looping

- while, and do-while loops
- break and continue keywords
- Take a look at:  
<http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html>

## JavaScript Syntax - Functions

- Calling a function the same way you would in C

```
myFunc(arg1, arg2, ...);
```

- Declaring a function using the 'function' keyword

– No return type, nor typing of arguments

```
function add(numOne, numTwo) {  
    return numOne + numTwo;  
}
```

## JavaScript Output

- The document object allows printing directly into the browser page (amongst other things)
- window object is implied
- Writing in text or HTML with script

– No line-break

```
document.write("I am <B>BOLD</B>");
```

– With line-break

```
document.writeln("I am <U>underlined</U>");
```

## Code Example

- Variables, Loops, and Output

## Objects in JavaScript

- Not Object Oriented – but Object-Based
- Easy to declare and create JavaScript Objects

```
function Person(myName, myAge) {  
    this.name = myName;  
    this.age = myAge;  
}  
  
var someGuy = new Person("Shawn", 28);
```

## 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

– 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;  
}
```

# Objects in JavaScript

- Object Functions
  - Then to call the function on the object

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

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

- Alternatively you may declare the function inside the constructor:

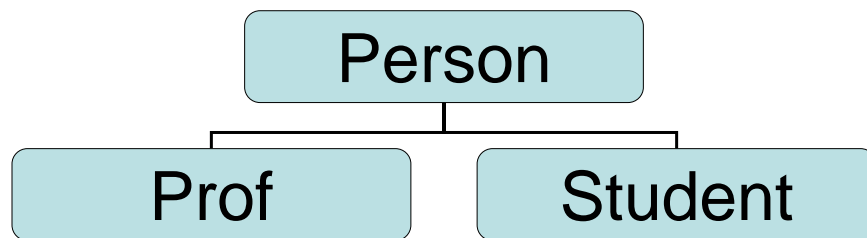
```
function Person(myName, myAge) {  
    this.name = myName;  
    this.age = myAge;  
    this.displayMe = function() { document.writeln("I am " +  
        this.name); }  
}
```

# Object Example

- Using objects

# Inheritance in JavaScript

- No built-in inheritance
- Runtime Inheritance: Clone objects and add extra properties
- For next week: Come prepared with at least one way of doing inheritance in JavaScript. Google is your friend.



# Built-in Objects

- Number, Boolean, String
  - Primitive types are automatically coerced into Objects when assigned to variables.
    - `var str = "abc";`
    - `var` is a String object
  - Number and Boolean are boring!
  - String has some helpful properties/functions:
    - `length`
    - `toUpperCase`
    - `substring`
    - `link`
- Date
  - No properties, but contains a bunch of methods for getting, setting and manipulating dates.
- Math
  - Calculate PI, find the SIN or COS of an angle, etc..

# Arrays

- Creating Arrays

```
var a = new Array(); // empty array
var b = new Array("dog", 3, 8.4);
var c = new Array(10); // array of size 10
var d = [2, 5, 'a', 'b'];
```

- Assigning values to Arrays

```
c[15] = "hello";
c.push("hello");
```

- Associative Arrays

```
var e = new Array();
e["key"] = "value";
e["numKey"] = 123;
```

# Arrays

- Properties and Methods

- length
- join()
- reverse()
- sort()
- concat()
- slice()
- splice()
- push() / pop()
- shift() / unshift()
- toString()
- .....

## Array Example

- General Usage

## Embedding in HTML

- Directly vs. Indirectly
  - Directly

```
<script type="text/javascript">
  ...code here...
</script>
```
  - Indirectly

```
<script type="text/javascript" src="test.js">
</script>
```
- Location: <HEAD> vs. <BODY>
  - Code directly or indirectly placed in the <HEAD> element is made available to be called later on in the document.
  - Code directly or indirectly placed in the <BODY> will be run when the document is parsed.
- Best to declare all functions in the HEAD and use the appropriate Event Handler to call the functions.

## Now we get into the cool stuff

The real power of JavaScript

Window Controls

Event Handlers

DOM

Cookies

And much much more...

## Window - Built-in Properties

- We have already seen the 'document' object – how we print to screen
- 'window' object – JavaScript representation of a browser window.
  - <http://www.comptechdoc.org/independent/web/cgi/javamanual/javawindow.html>
  - `closed` - A boolean value that indicates whether the window is closed.
  - `defaultStatus` - This is the default message that is loaded into the status bar when the window loads.  
`window.defaultStatus = "This is the status bar";`

## Window – Built-in Functions

- `alert("message")`
  - `window.close()`
  - `confirm("message")`
  - `focus()`
  - `open("URLname", "Windowname", ["options"])`
    - height, weight, alwaysRaised, location, menubar, etc..
- ```
open("http://google.com", "My Google",  
"toolbar=no,alwaysRaised=yes");
```

## Window Example

- Controlling one browser window from another

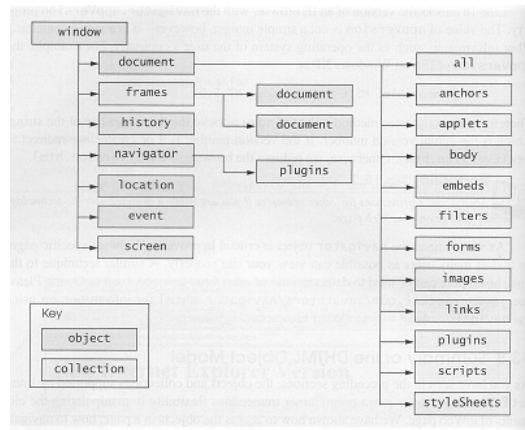
## Browser – Built-in Objects

- `window.location`
  - `href` represents a complete URL.
  - `hostname` represents the concatenation *host:port*.
  - `window.location.href="http://google.com";`
- `window.history`
  - `length` reflects the number of entries in the history list
  - `history.go(-1)`
  - `history.back()`

## Form Object

- Form objects can be accessed by:  
`window.document.myForm` OR  
`window.document.forms[0]`
- **Properties**
  - `action`, `target`, `length`, `method`, etc..
- **Functions**
  - `window.document.myForm.submit();`
  - `window.document.myForm.reset();`
- **Accessing Form Field Values**
  - `window.document.myForm.firstname.value`

## (D)HTML Object Hierarchy



## Event Handlers

- *Events* are actions that occur usually as a result of something the user does. For example, clicking a button is an event, as is changing a text field or moving the mouse over a hyperlink.
- Eg: Click, change, focus, load, mouseover, mouseout, reset, submit, select

## Event Handlers

- You can use *event handlers*, such as **onChange** and **onClick**, to make your script react to events.

```
<input type="button" onClick="javascript:doButton()>
```

```
<select onChange="javascript:doChange()">
```

```
<a onClick="javascript:doSomething()"> </a>
```

```
<form onSubmit="javascript:validate()">
```

```
<body onLoad="javascript:init()">
```

## Example

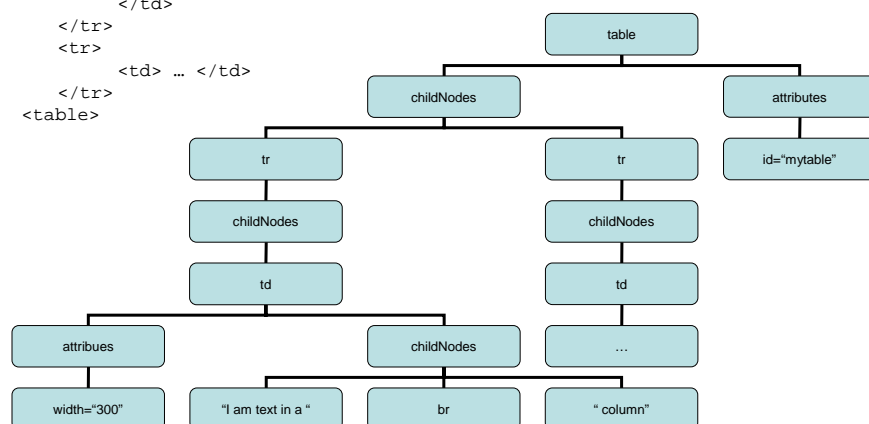
- Use event handlers, functions and document / form object to change information on the page

# DOM – Document Object Model

- W3C DOM, "The DOM", DOM Level (1/2/3)
- Method of accessing / modifying XML information on the page
- Tree structure of all HTML elements, including attributes and the text they contain.
- Contents can be modified or deleted
- New elements can be created and inserted into the DOM Tree
- The concept is used in other languages, such as Java (google: Java DOM)

## DOM Representation of HTML

```
<table id="mytable">
  <tr>
    <td width="300">
      I am text in a <BR> column
    </td>
  </tr>
  <tr>
    <td> ... </td>
  </tr>
</table>
```



## Document Object

- The `document` object is the JavaScript interface to the DOM of any HTML page.

- Accessing elements:

- By name

```
document.getElementsByTagName('td')[indexOfColumn]
```

- By ID

```
document.getElementById('id')
```

- Walk the DOM Tree

```
document.childNodes[0].childNodes[1].childNodes[4]
```

## Creating Time Delays

- Waits a specified amount of time, then calls a function.

```
window.setTimeout('myFunc()',timeInMillise  
conds);
```

- Waits a specified amount of time, then calls a function.  
Then after the initial delay, the function is called again

```
var myinterval =  
window.setInterval('myFunc()',timeInMillis  
econds);
```

- Wait / Call cycle is repeated until 'clearInterval' is called  
`window.clearInterval(myinterval);`

## Time Delay Example

## Regular Expressions

- You have seen this before!
- RegExp object

```
var beerSearch = /beer/  
var beerSearch = new RegExp("beer")
```
- Methods
  - search pattern and return results in a array  
`exec(str)`
  - returns true if match  
`test(str)`

```
if (beerSearch.test("beer tastes bad")) {  
  alert("I found a beer");  
}
```

# Regular Expressions

- Special Characters
  - \w Alphanumeric
  - \d Numerical digit
  - \s White spaces
  - . Anything other than newline
  - [abcde] Any of a,b,c,d,e
  - [a-e] Same as above
  - [^a-e] Anything but a to e
  - exp?,exp+,exp\* 0 or 1, 1 or more, 0 or more
  - exp{x,y} At least x but less than y
  - expA | expB expA or expB
- JavaScript RegEx Resources
  - <http://wsabstract.com/javatutors/redev.shtml>
  - [http://www.devguru.com/Technologies/ecmascript/quickref/regexp\\_special\\_characters.html](http://www.devguru.com/Technologies/ecmascript/quickref/regexp_special_characters.html)

# Regular Expression Examples

- Two examples
  - Postal Code
  - Money

## JavaScript Resources

- JavaScript Bible
- JavaScript: The Definitive Guide
- <http://www.w3schools.com/js/default.asp>
- <http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html>