305CDE Lab 1

JavaScript Language Basics

Colin Stephen

October 2014

Overview

- JavaScript Types
- Variables
- Operators
- Control Structures
  - Conditionals
  - Loops
- Arrays
- Functions
- Strict Mode
- `window.onload`
- `document.getElementById`

JavaScript Types

- Number: `42`, `0.3`, `1e4`
- String: `"hello"`
- Boolean: `true`, `false`
- Object: {}
  - Function: `function()`
  - Array: `[]`
  - *Date*
  - *RegExp*
- Null: `null`
- Undefined: `undefined`

The last two are subtle. Ones in italics won't be used much.

Variables

New variables in JavaScript are declared using the var keyword:

```
var a;
var name = "simon";
```

- If you declare a variable without assigning any value to it, its type is `undefined`.
- If a variable is defined using `var` in a compound statement (for example inside an `if` control structure), it will be visible to the entire function.

Operators

- Numeric: `+`, `-`, `*`, `/` and `%` (remainder)
- Assignment: `=`
  - Set the value of a variable as in `x = "hi"`
- Increment/decrement with assignment: `+=`, `-=`, `++`, `--`
- Concatenate: `+` between strings as in `"hello" + "world"`
- Comparison: `<`, `>`, `<=`, `>=`, `==`, `===`
- Negation: `!=`, `!==`
- Logical: `&&`, `||`
- Ternary: `... ? ... : ...` (see `if` statements below)

Control Structures - if

- if and if ... else
- can be chained together

```
var name = "kittens";
if (name == "puppies") {
  name += "!";
} else if (name == "kittens") {
  name += "!!";
} else {
  name = "!" + name;
}
name == "kittens!!"  // returns "true"

name == "joe" ? 42 : "blogs"  // returns "blogs"
```

Control Structures - `switch`

- `switch ... case ... break` is used for multiple branches based on a number or string

```
switch(food) {
    case 1: // fallthrough
    case 2:
        eatit(food);
        break;
    default:
        donothing();
}
```

Control Structures - `while` and `do ... while`

- `while` is good for basic looping
- `do-while` is good for loops where you wish to ensure that the body of the loop is executed at least once

```
while (true) {
  // an infinite loop!
}

var input;
do {
  input = get_input();
} while (inputIsNotValid(input))
```

Control Structures - `for`

- `for` loop the same as in C and Java

```
for (var i = 0; i < 5; i++) {
  // Will execute 5 times
}
```

Arrays

- Arrays are a special type of object
- They have a magic property called ‘length’. This is always one more than the highest index in the array.
- Created in two ways (prefer the second):

```
var a = new Array();
a[0] = "dog";
a[1] = "cat";
a[2] = "hen";
a.length  // returns "3"

var b = ["dog", "cat"];
b.length  // returns "2"
```

Array Iteration

- Can use arrays in `for` loops via their indices

```
var a = ["dog", "cat"];

for (var i = 0, len = a.length; i < len; i++) {
    // Do something with a[i]
}

for (var i in a) {
  // Do something with a[i]
}
```

Array Methods

- Arrays have useful built-in methods:

```
a.toString()
a.concat(item[, itemN])
a.join(sep)
a.pop()
a.push(item[, itemN])
a.reverse()
a.slice(start, end)
a.sort()  // with optional function
```

and several more

Functions

- The core component in understanding JavaScript (along with objects)
- Very powerful - discover more in later labs
- Basic definition very simple

```
function add(x, y) {
    var total = x + y;
    return total;
}
```

- can take 0 or more named parameters
- function body can contain as many statements as you like
- function can declare its own variables which are local to that function
- `return` statement can be used to return a value at any time, terminating the function
- if no return statement is used JavaScript returns `undefined`

Strict Mode

- Restricted “safer” variant of JS
- Applies to entire scripts or to individual functions:

```
"use strict";
var v = "Hi!  I’m a strict mode script!";
```

or

```
function strict(){
  'use strict';
  function nested() { return "And so am I!"; }
  return "Hi!  I’m a strict mode function!  " + nested();
}
function notStrict() { return "I’m not strict."; }
```

- Doesn't apply to block statements enclosed in {} braces
- See the MDN reference for details
- If in doubt, use it!

DOM Essentials

You saw both of these used in the examples.

`window.onload`

- an event triggered when the page has finished loading
- good to check for it to ensure the DOM is present before applying your JS

`document.getElementById`

- a DOM element selector
- finds the element based on the `id` attribute of the HTML tag