Introduction to JavaScript: Overview of the Language
Part Two

Spring 2017
Nathan Hull
Calling JavaScript Functions

- JavaScript can be placed ...
  - in the `<body>` ... `</body>` of a web page
  - in the `<head>` ... `</head>` of a web page
  - in an external file with the suffix of `.js`

- Current “best practices” uses the method of putting all JavaScript in separate script files.

- The syntax is:
  ```html
  <script type="text/javascript" src="javaScriptFile.js">
  </script>
  ```
JavaScript vs. Python

• There are some important syntax differences between JavaScript and Python which I will point out as we work through code samples. The two most important for now are:

  • Python blocks of code are set off by tabs (indenting the code); but JavaScript uses curly braces: `{ ... }`
  • Python statements terminate typically at the end of a line but JavaScript requires a semi-colon (`;`) to terminate a statement.
  • JavaScript functions return one variable (although there are “work arounds”)
Variables in JavaScript

• JavaScript is case-sensitive.
• Variable names cannot contain spaces, punctuation or start with a number.
• Variable names cannot be one of the reserved words (e.g. for, if, while ... etc)
Examples of Data Types in JavaScript

• Data types:
  - number - any numeric value
    ```javascript
    var x = 3.14159;
    ```
  - String - text
    ```javascript
    var word = “elephant”;
    ```
  - Boolean - true or false
    ```javascript
    var isPrime = true;
    ```
## Operators in JavaScript

<table>
<thead>
<tr>
<th>Operator</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x + y ) (for numbers)</td>
<td>addition</td>
</tr>
<tr>
<td>( x + y ) (for strings)</td>
<td>concatenation</td>
</tr>
<tr>
<td>( x - y )</td>
<td>subtraction</td>
</tr>
<tr>
<td>( x \times y )</td>
<td>multiplication</td>
</tr>
<tr>
<td>( x / y )</td>
<td>division</td>
</tr>
<tr>
<td>( x % y )</td>
<td>mod (for integer arithmetic)</td>
</tr>
<tr>
<td>( x++, ++x )</td>
<td>increments by 1</td>
</tr>
<tr>
<td>( x--, --x )</td>
<td>decrements by 1</td>
</tr>
<tr>
<td>(-x)</td>
<td>reverses the sign</td>
</tr>
</tbody>
</table>
## Assignments in JavaScript

<table>
<thead>
<tr>
<th>Assignment</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = y</code></td>
<td>assignment</td>
</tr>
<tr>
<td><code>x += y</code></td>
<td>same as <code>x = x+y</code></td>
</tr>
<tr>
<td><code>x -= y</code></td>
<td>same as <code>x = x-y</code></td>
</tr>
<tr>
<td><code>x *= y</code></td>
<td>same as <code>x = x * y</code></td>
</tr>
<tr>
<td><code>x /= y</code></td>
<td>same as <code>x = x / y</code></td>
</tr>
<tr>
<td><code>x %= y</code></td>
<td>same as <code>x = x % y</code></td>
</tr>
</tbody>
</table>
Comparison Operators in JavaScript

<table>
<thead>
<tr>
<th>Comparison</th>
<th>what it does ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x == y$</td>
<td>returns true if $x$ and $y$ are equal</td>
</tr>
<tr>
<td>$x != y$</td>
<td>returns true if $x$ and $y$ are not equal</td>
</tr>
<tr>
<td>$x &gt; y$</td>
<td>returns true if $x$ is greater than $y$</td>
</tr>
<tr>
<td>$x &gt;= y$</td>
<td>returns true if $x$ is greater than or equal to $y$</td>
</tr>
<tr>
<td>$x &lt; y$</td>
<td>returns true if $x$ is less than $y$</td>
</tr>
<tr>
<td>$x &lt;= y$</td>
<td>returns true if $x$ is less than or equal to $y$</td>
</tr>
<tr>
<td>$x &amp;&amp; y$</td>
<td>returns true if both $x$ and $y$ are true</td>
</tr>
<tr>
<td>$x</td>
<td></td>
</tr>
<tr>
<td>$!x$</td>
<td>returns true if $x$ is false</td>
</tr>
</tbody>
</table>
Programming / Syntax Example

• statements; declaring and assigning variables; primitive data types
• selection statements: if, if/else, switch
• iteration statements: for, while, do/while
• functions
• arrays (similar to lists in Python)
• events (e.g. "onClick" when a button is clicked)
Primitive Data Type

• Some of the primitive data types are
  - String
  - Number
  - Boolean

• Variable declarations:
  ```
  var number = 7;
  var flower = "rose";
  var isPrime = true;
  ```
Casting (data type conversions)

var strLuckyNumber = String(luckyNumber);
This will result in a String data type.
if and else statement

Note that each expression is surrounded by parentheses ...
... and that each condition is surrounded by braces.

(Python users: indentation is not effective here!)

    (expression ) {
    ....   }
else {
    .... }


switch (x) {
    case 1:
        outputString2 += "one . Sorry, You lose ... </p>";
        break;
    case 2:
        outputString2 += "two or a three . Good bye!! </p>";
        break;
    ...
    default:
        outputString2 += "nothing. This is an error. </p>";
    }

switch: a conditional statement
for loop

Syntax: The declaration, assignment, and test are all in the same line:

```javascript
for (var j = 0; j < 10; j++) {
    ...
}
```
while loop

• Bart Simpson has to write the same sentence over and over in class ...


• The structure of a while loop:

```java
while (test is true) {
    ...
    ...
    ...
    ... update
}
```
do/while loop

• Note that for do/while loops ... that the test is at the end, so the loop will execute at least once:

    do {
      ...
      update
    } while (test)
Function

• Syntax:
  ```
  function test ( ... ) {
      ...
      ...
      return ...
  }
  ```

• Functions do not require return values.
• Functions can accept zero, one or more parameters.
Functions and Recursion

• JavaScript supports recursion:

```javascript
function factorial(n) {
    if (n == 0) // Stopping condition
        return 1;
    else
        return n * factorial(n - 1); // Call factorial recursively
}
```
Scope: Local and Global Variable

- The keyword `var` in JavaScript is used to confirm that a variable is local.
- Without `var` ... the new variable could be ambiguous as it “behaves” like a global variable.
- Variables that are declared outside of any function are global variables.
Avoid or be very careful with Global Variables in JavaScript

• Without `var`, declaring a variable can become ambiguous if that variable name is used elsewhere in the application.

• Since so many JavaScript applications make use of JavaScript libraries (e.g. jQuery, etc), it is important to use only local variables as one cannot be sure of all of other variable names.
The **return** keyword

• Use **return** to return a value from a function:

```javascript
function function1() {
    var x = 10;
    var y = 20;
    var z = function2(x, y);
}

function function2(number1, number2) {
    var number3 = number1 * number2;
    return number3;
}
```
Array

• For those of you who studied Python, arrays are similar to lists.
• For those of you who have studied Java, arrays in JavaScript are similar to arrays in Java.
• Arrays are widely used in JavaScript as a way to keep track of images for example and other objects.
• They are typically used with for loops as they are in other computer languages.
Arrays, continued

• There are several ways to declare an array. For example:
  ```javascript
  var dieRoll = new Array(7);
  ```
• Note the keyword `new` because in JavaScript, an array is an object (as it is in Java)
• You can also declare and initialize an array in one step:
  ```javascript
  var animalsArray = new Array('lions', 'tigers', 'bears');
  ```
Arrays, continued

• The importance of an array ... is that one object name (which in this context behaves like a "regular" variable) ... can hold multiple values.

• Each value can be uniquely identified.

```javascript
var threeColors = new Array(3);
threeColors[0] = 'red';
threeColors[1] = 'blue';
threeColors[2] = 'green';
```
multi-dimensional array

• JavaScript can handle multi-dimensional arrays.
• For example, a 3x3 grid can be set up of one array ... of three elements ... and each element in turn is made up of an array of 3 elements.

    var grid = new Array(3);
    grid[0] = new Array('1', '2', '3');
    grid[1] = new Array('4', '5', '6');
    grid[2] = new Array('7', '8', '9');
Handling Events

- Events are actions that the user performs.
- JavaScript provides event handlers to manage the user’s actions such as:
  - onerror
  - onfocus
  - onload
  - onmouseover
  - onmouseout
  - onselect
  - onsubmit
  - etc.
Event

• We will also work with JavaScript events.
• JavaScript events allow us to define what the browser should do next given a user's action (such as clicking on a button, having the mouse hover over a hyperlink, etc).
• For example, you might want to execute a specific function when a button is clicked:

  <input name="blue-" value=" -50 " onclick="downblue(50)" type="button" />