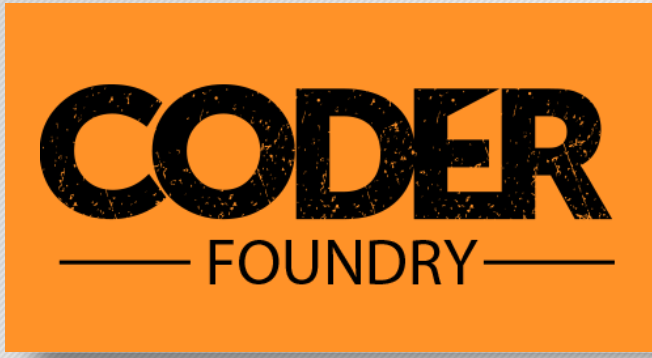
JavaScript



JavaScript Basics

- Borrows most of its syntax from Java (inherits from others…)
- Basic concepts:
  - Everything is case-sensitive
  - Variables are loosely typed
    - Use the var keyword
    - Variables don't have to be declared before being used
  - End-of-line semicolons are optional
    - var test1 = "red"
    - var test2 = "blue"; //do this to avoid confusion
  - Comments are the same as in Java, C, and Perl
  - Braces indicate code blocks (as in Java, C, etc)

Keywords and Reserved Words

CODER FOUNDRY

- Keywords and reserved words cannot be used as variables or function names
- Keywords
  - break, case, catch, continue, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with
- Reserved Words
  - abstract, boolean, byte, char, class, const, debugger, double, enum, export, extends, final, float, goto, implements, import, int, interface, long, native, package, private, protected, public, short, static, super, Synchronized, throws, transient, volatile

Primitive and Reference Values

CODER FOUNDRY

- Primitive Values
  - Primitive values are simple pieces of data that are stored on the stack
  - Value is stored directly in the memory location the variable accesses
  - The value is one of the JavaScript primitive types:
    - Undefined, Null, Boolean, Number, or String
  - Many languages consider strings as a reference type and not a primitive type, JavaScript breaks from this tradition
- Reference Values
  - objects stored in the heap
  - the value stored in the variable location is a pointer to a location in memory where the object is stored

Primitive Types

CODER FOUNDRY

- JavaScript has five primitive types:
  - Undefined
    - The Undefined type has only one value, undefined
  - Null
    - The Null type has only one value, null
  - Boolean
    - The Boolean type has two values, true and false
  - Number
    - 32-bit integer and 64-bit floating-point values
  - String
    - Using either double quotes(”) or single quote(’)
    - No character type

The typeof Operator

- To determine if a value is in the range of values for a particular type, use the typeof operator
- Returns “object” for a value that is null.
  - An error in the original JavaScript implementation.
  - Today, it is rationalized that null is considered a placeholder for an object.

Value	Typeof
Boolean	boolean
Number	number
String	string
Undefined	undefined
Null	object
Object	object
Array	object
Function	function

Object Class

- The Object class in JavaScript is similar to java.lang.Object in Java
- Each of properties and methods are designed to be overridden by other classes
- Properties:
  - constructor – A reference value (pointer) to the function that created the object
  - prototype – A reference value to the object prototype for this object
- Methods:
  - hasOwnProperty(property)
  - isPrototypeOf(object)
  - propertyIsEnumerable(property)
  - toString()
  - valueOf()

Primitive Type Wrapper Classes

CODER FOUNDRY

- There are wrapper classes (upper case versions) for each of the primitive types – for example:
  - number = Number
  - boolean = Boolean
  - string = String
- Whenever possible, use primitives rather than objects

Operators

- Unary
  - delete, void, Prefix ++/--, Postfix ++/--, Unary +/-
- Bitwise
  - ~, &, |, ^, <<, >>, >>>
- Boolean
  - !, &&, ||
- Arithmetic
  - +, -, *, /, %
- Assignment
  - =, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=
- Comparison
  - >, >=, <, <=, **==**, **!=**, **===**, **!==**
- Conditional
  - variable = *boolean_expression* ? *true_value* : *false_value*;
- Comma
  - var iNum = 1, iNum=2;

Statements

- if...else
- switch
- while
- do...while
- for
- for...in
  - Used to enumerate the properties of an object (all objects have method propertyIsEnumerable())
  - for (*property* in *expression*) *statement*
- with
  - A very slow segment. Avoid using it.
- Label, break and continue
- try...catch...finally
- throw

Functions

- Collection of statements that can be run anywhere at anytime
- The *function* keyword
  - function functionName(arg0, arg1,…, argN)
    { functionBody }
  - var functionName = function(arg1, arg2,…, argN)
    { functionBody }
- Functions that have no return value actually return *undefined*
- Unlike true OOP languages Functions cannot be overloaded
  - The last function becomes the one that is used
- The Function Class
  - Functions are actually full-fledged objects
  - var functionName = new Function(arg1, arg2,…, argN, functionBody);

Object Oriented Terminology

CODER FOUNDRY

- A class is a kind of recipe for an object
- An object is a particular instance of a class
- If a member of an object is a function, it is a method; otherwise, the member is a property
- JavaScript supports (to some degree anyway) the requirements of object-oriented languages
  - Encapsulation
  - Inheritance
  - Polymorphism

Class-based vs. Prototype-Based

CODER FOUNDRY

- Class-based Programming
  - Inheritance is achieved by defining classes of objects, as apposed to the objects themselves
  - The most popular and developed model of OOP
  - Java, C++, C#, etc
- Prototype-based Programming
  - Classes are not present
  - Behavior reuse (aka. inheritance) is accomplished through a process of cloning existing objects which serve as prototypes
  - Class-less, prototype-oriented, or instance-based programming
  - JavaScript, etc.

Early Binding vs. Late Binding

CODER FOUNDRY

- Early binding
  - properties and methods are defined for an object (via its class) before it is instantiated
  - compiler/interpreter assembles the machine code at compilation time
  - Java, C++, C#, etc. (IntelliSense)
- Late binding
  - compiler/interpreter doesn't know what type of object is being held in a particular variable until runtime
  - JavaScript, etc. (no IntelliSense)
- Due to late binding, JavaScript allows a large amount of object manipulation to occur without penalty

Objects in JavaScript

CODER FOUNDRY

- In JavaScript, objects are implemented as a collection of named properties
- The most basic objects in JavaScript act as hashtables or dictionaries
- Objects can be created directly through object literal notation:
  - var myDog = {
    age: 3,
    color: “black”,
    bark: function() { alert(“Woof!”); }
    }
  - Object’s properties and methods are defined as comma-separated name/value pairs inside curly braces
  - Each member introduced by name, followed by a colon and definition
  - Methods are created by assigning an anonymous function

Debugging

- No debugging in Visual Studio
- Debugging in browser only
  - F12 in Google Chrome brings up developer console
  - Full and detailed debugging environment
- No need to restart app execution from VS on code change – refresh browser window after save

Interacting with HTML

- DOM vs. BOM
  - The document object
  - .getElementById
  - .getElementsByName
- jQuery
  - $(selector).action()
    - $ sign to define/access jQuery
    - (selector) to "query (or find)" HTML elements
    - jQuery action() to be performed on the element(s)
  - Examples:
    - $(this).hide() - hides the current element.
    - $("p").hide() - hides all <p> elements.
    - $(".test").hide() - hides all elements with class="test"
    - $("#test").hide() - hides the element with id="test"

Just a few basics, now some practice…