JavaScript for C# Developers

Module 2: JavaScript Functions

Shawn Wildermuth
Wilder Minds
wilderminds.com

pluralsight
hardcore developer training

Agenda

- **JavaScript Functions**
  - Function Parameters
  - Functions that Return Values
  - Function as an Object
  - What is 'this'
  - Closures and Scope
  - Namespaces

Function Parameters

- **Looks like C#, but not…**

```
// JavaScript
function foo(one, two, three) {
  alert(one);
  alert(two);
  alert(three);
}

foo(1); // two and three are undefined
```

Function Parameters

- **Looks like C#, but not…**

```
// JavaScript
function foo(one, two, three) {
  alert(one);
  if (two) alert(two);
  if (three) alert(three);
}

foo(1); // two and three are undefined
```

Function Parameters

- **Overloading Functions?**

```
// JavaScript
function foo(one) {
  alert("first");
}

function foo(one, two) {
  alert("second");
}

foo(1); // "second"
```

Function Parameters

- **`arguments` object**
  - Available Inside Function Body Only

```
// JavaScript
function foo(one, two, three) {
  alert(arguments.length);
}

foo(1);       // 1
foo(1, 2);    // 2
foo(1, 2, 3); // 3
```

Function Parameters

- **`arguments` object**
  - Declared Parameters Do Not Matter

```
// JavaScript
function foo() {
  alert(arguments.length);
}

foo(1);       // 1
foo(1, 2);    // 2
foo(1, 2, 3); // 3
```

Function Parameters

- **`arguments` object**
  - Accessing the Values

```
// JavaScript
function foo() {
  for (var x = 0; x < arguments.length; x++) {
    alert(arguments[x]);
  }
}

foo(1);       // 1
foo(1, 2);    // 1,2
foo(1, 2, 3); // 1,2,3
```

Function Parameters

- **Parameters are for mapping to arguments**

```
// JavaScript
function foo(one, two, three) {
  alert(one);
}

foo(1,"hello", new Date());
```

Function with Return Value

- **All Functions Return a value**
  - If not defined it's 'undefined'

```
// JavaScript
function foo() {
}

var x = foo(); // typeof "undefined"
```

Function with Return Value

- **All Functions Return a value**
  - If not defined it's 'undefined'

```
// JavaScript
function foo() {
  return;
}

var x = foo(); // typeof "undefined"
```

Function with Return Value

- **All Functions Return a value**
  - If not defined it's 'undefined'

```
// JavaScript
function foo() {
  return "";
}

var x = foo(); // typeof "string"
```

Function Object

- **Just an Object**
  - Has properties and member functions

```
// JavaScript
function log(s) { alert("yup"); }

var x = log.length;     // 1 parameter

var y = log.name;       // "log" (Non-Standard)

var z = log.toString(); // "function log(s) { alert("yup"); }"
```

Function Object

- **Can Store as Variable**
  - "Anonymous Delegate"

```
// JavaScript
var f = function(s) { alert("yup"); };

var x = f.length;     // 1 parameter

var y = f.name;       // "" (Non-Standard)

var z = f.toString(); // "function(s) { alert("yup"); }"

f(1);                 // Calling like a function
                      // (or delegate)
```

What is 'this'?

- In C#, this represents the instance of the class

```
// C#
class Foo
{
  string _name;

  void Run(string newName)
  {
    this._name = newName;
  }
}
```

What is 'this'?

- **Function Body Variable**
  - "this" applies to the owner of the function

```
// JavaScript
var f = function() {
  alert(this);
};

f();  // [Object Window] (huh?)
```

What is 'this'?

- **'this' is Owner**

```
// JavaScript
var obj = {
  name: "myObj",
  myFunc: function() {
    log(this.name);
  }
};

obj.myFunc();  // "myObj"
```

What is 'this'?

- **'this' is Owner**
  - bind() lets you change the owner

```
// JavaScript
var obj = {
  name: "myObj",
  myFunc: function() {
    log(this);
  }
};

obj.myFunc();                    // this == obj

var f = obj.myFunc.bind(this); // Copy Function with global

f();                             // this == global object
```

Closures

- **References outside of function are accessible in function**
  - Regardless of lifetime

```
// JavaScript
var x = 1;

function someFunction() {
  // Works as it wraps 'x' with a closure
  var y = x;
}

// Much Later
someFunction();
```

Scoping

- **C# is different than JavaScript**

```
// C#
var a = "Hello";

if (true)
{
  // This works
  var b = a;
}

// This doesn't work
var c = b;
```

Scoping

- **C# is different than JavaScript**

```
// JavaScript
var a = "Hello";

if (true) {
  // This works
  var b = a;
}

// This works too
var c = b;
```

Scoping

- **C# is different than JavaScript**

```
// JavaScript
var a = "Hello";

function () {
  // This works (closure)
  var b = a;
}

// This doesn't (functions define scope)
var c = b;
```

Polluting the Global Scope

- **Name Collision Problematic in Large Projects**

```
// JavaScript
var appName = "foo";
var compileTime = new Date();

function printAppInfo() {
  return appName + " : " + compileTime;
}

console.log(printAppInfo()); // Works because of closures
```

Polluting the Global Scope

- **Anonymous Self-Executing Functions**
  - Protects the global namespace by function scope

```
// JavaScript
function () {
  var appName = "foo";
  var compileTime = new Date();

  function printAppInfo() {
    return appName + " : " + compileTime;
  }
} // Hides it all but no way to execute it

console.log(printAppInfo()); // Doesn't work
                             // out of scope
```

"Namespaces"

- **JavaScript lacks real namespaces**
  - Can create with objects

```
// JavaScript

// Construct or Import Namespace
var WilderMinds = WilderMinds || {};

// Add function to namespace
WilderMinds.currentTime = function () {
  return new Date();
};
```

"Namespaces"

- **JavaScript lacks real namespaces**
  - Can create with objects

```
// JavaScript
var WilderMinds = WilderMinds || {};
WilderMinds.Models = WilderMinds.Models || {};

// Add function to namespace
WilderMinds.Models.Customer = function () {
  // ...
};
```

All Together Now!

- **Namespaces and Anonymous Self-Executing Functions**
  - Handles the global pollution and scoping of functionality

```
// JavaScript
(function(ns) {

  var currentDate = new Date();

  // Add function to WilderMinds namespace
  ns.currentTime = function () {
    return currentDate;
  };

})(window.WilderMinds = window.WilderMinds || {});
```

Summary

- **Functions**
  - The Center of your world in JavaScript
  - Big Differences between C# Methods and JavaScript Functions
  - Closures, Scoping, 'this' and namespaces can tame JavaScript