JavaScript Design Patterns

Aaron Powell

http://www.aaron-powell.com

http://twitter.com/slace

Common Object Patterns

Aaron Powell

http://readify.net | http://www.aaron-powell.com

Function arguments

Magic arguments

```
function myFunc(a, b, c) {
    return a + b + c;
}

console.log(myFunc(1, 2, 3));
console.log(myFunc(1, 2, 3, 4));
console.log(myFunc(1, 2));
```

Magic arguments

- **Arguments aren’t required like .NET languages**
- **Like all variables in JavaScript arguments are untyped**
- **Unspecified arguments become undefined**
- **Arguments available through the `arguments` object**
  - It’s array-like but not really an array

```
function myFunc() {
   var x = 0;
   for (var i = 0; i < arguments.length; i++) {
      x = x + arguments[i];
   }
   return x;
}
```

Chaining

Introduction

- **Useful for creating fluent APIs for working against a mutable object**
- **Designed around returning the source object**
- **Popularized by jQuery**
  - Common from a .NET perspective in LINQ

```
jQuery(‘.foo’)
        .addClass(‘bar’)
        .fadeIn(‘slow’)
        .html(‘Hello World’);
```

Example a chained API

```
new Calc(0)
   .add(1)
   .add(2)
   .multiply(3)
   .equals(function (result) {
       console.log(result); //logs 9
   });
```

Demo

Creating a chained API

Recap

- **Return the object you want to chain**
    - The `this` object
    - Capture the `this` object if the caller isn’t trusted

Observable Properties

Introduction

- **How can you react to a value changing on an object?**
  - .NET we can have properties with method bodies
  - INotifyPropertyChanging/ INotifyPropertyChanged available in .NET
- **JavaScript properties are really just public fields**
  - So how do we address the lack of method body?
  - By using methods-as-properties
    - This is what Knockout.js uses

Demo

Implementing methods-as-properties

Note on ECMAScript 5

- **In ES5 properties can have method bodies**
  - Similar to how .NET properties look
  - Very powerful
- **Only available in current generation browsers**

Using ES5 properties

```
function Book () {
   var name = '';
   Object.defineProperty(this, 'name', {
      get: function () {
          return name;
     },
     set: function (val) {
       console.log(val);
       name = val;
     }
   });
}
```

Recap

- Properties implemented as methods
- Check the incoming value and decide if you want to update
- Return private variable
- Store event handlers in an array
- Utilise return values to abort the updating process