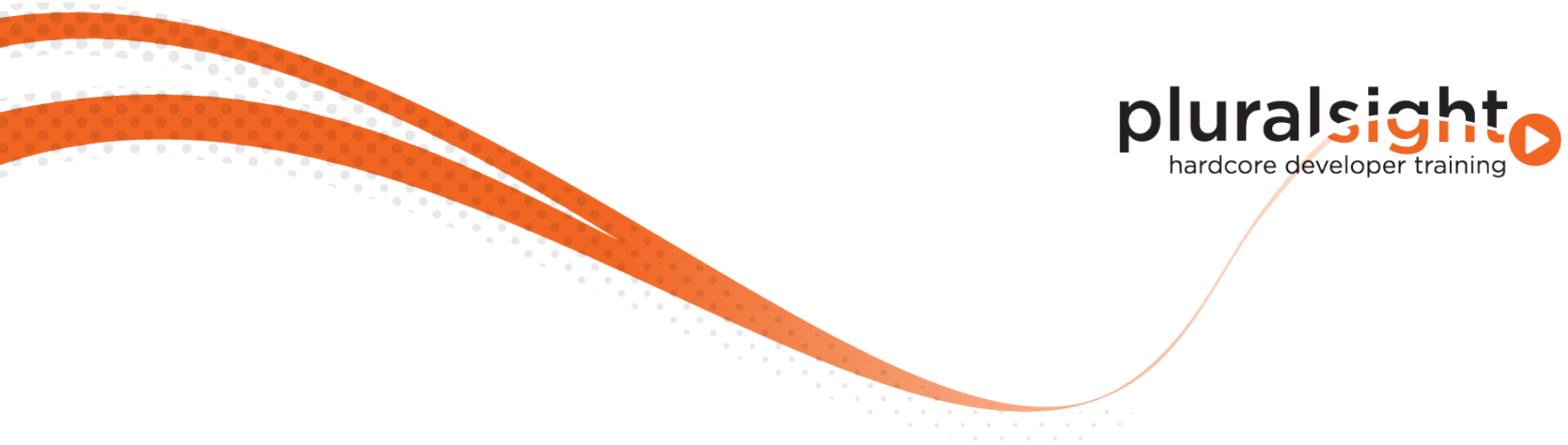
# JavaScript Patterns: Spaghetti to Ravioli

John Papa

@john_papa

http://johnpapa.net



pluralsight

hardcore developer training

# Outline

- Spaghetti and Ravioli
  - Separation Patterns
  - Avoiding Globals
- **Object Literals**
- **Module Pattern**
  - Anonymous Closures
  - Private/Public Members
  - Immediate Invocation
- **Revealing Module Pattern**
  - Refinements to Module Pattern

# Function Spaghetti Code

- Wikipedia: http://jpapa.me/spaghetticode

# Problems with Spaghetti Code

- Mixing of Concerns
- No clear separation of functionality or purpose
- Variables/functions added into global scope
- Potential for duplicate function names
- Not modular
- Not easy to maintain
- No sense of a “container”

# Some Examples of Spaghetti Code with JavaScript

- **Script all over the page**
- **Objects are extended in many places in no discernible pattern**
- **Everything is a global function**
- **Functions are called in odd places**
- **Everything is a global**
- **Heavy JavaScript logic inside HTML attributes**
  - Obtrusive JavaScript
  - http://en.wikipedia.org/wiki/Unobtrusive_JavaScript

# Advantages of Ravioli Code

- **Objects encapsulate and decouple code**
- **Easy to extend and maintain**
- **Separation of Concerns**
  - Variables/functions are scoped
  - Functionality in closures

# Namespaces

- Encapsulate your code under a namespace
- Avoid collisions
- First and easy step towards Ravioli's

```
var my = my || {};
```

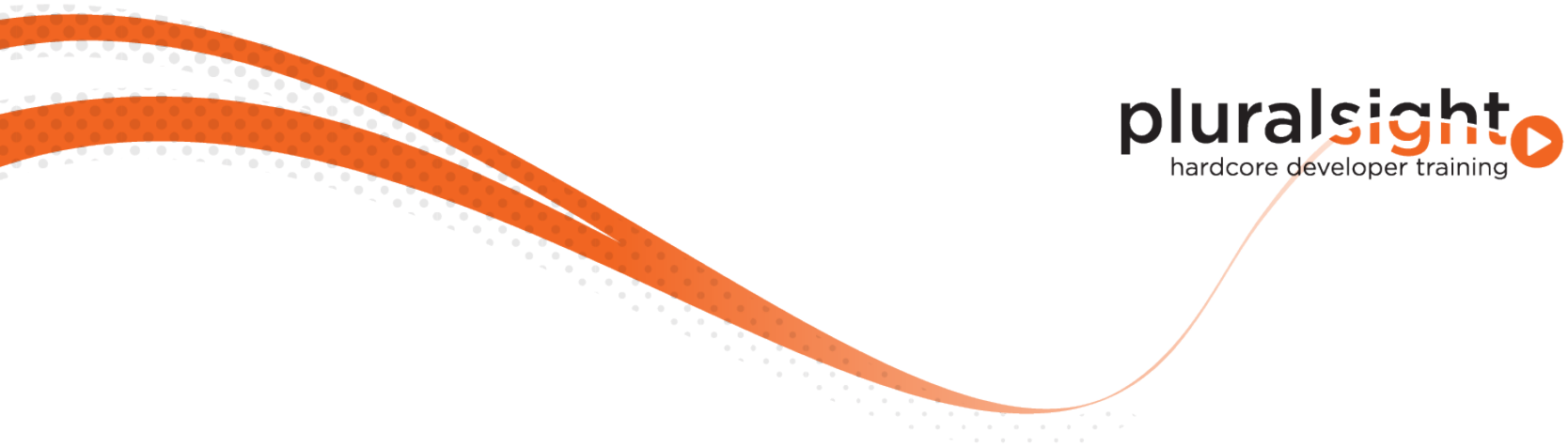
```
my.viewmodel = function(){



}
```

# DEMO

Namespaces and Separation



pluralsight
hardcore developer training

# Outline

- Spaghetti and Ravioli
- Object Literals
- Module Pattern
- Revealing Module Pattern

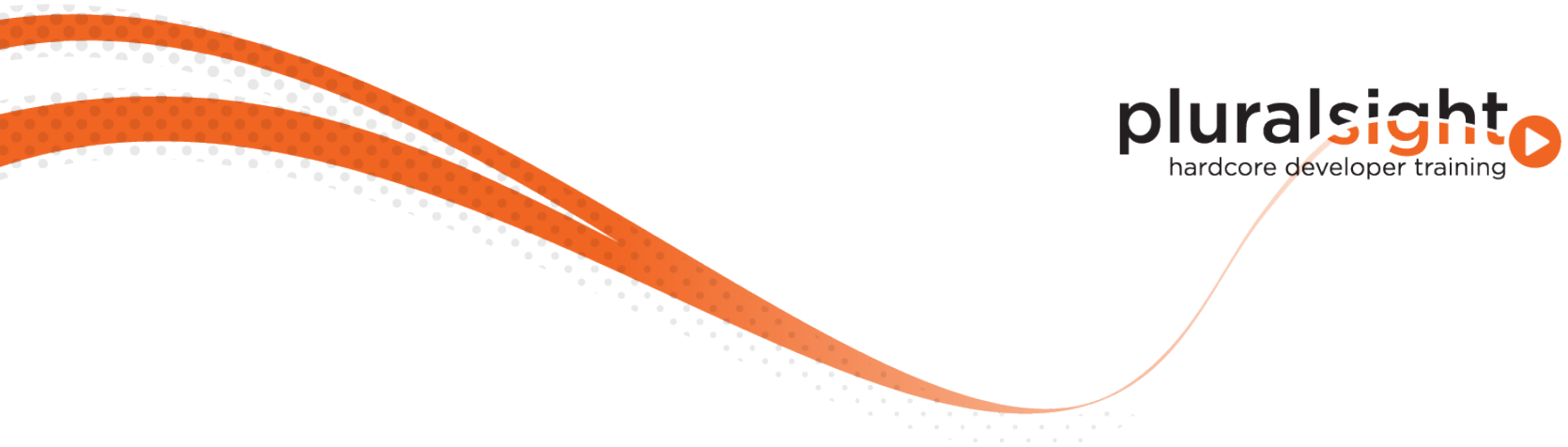
# Object Literals

- **Benefits**
  - Quick and easy
  - All members are available
- **Challenges**
  - "this" problems
  - Best suited for simple view models and data

```
my.viewmodel = {
   name: ko.observable(),
   price: function(x, y){
       return x + y;
   }
};
```

# DEMO

ViewModel as an Object Literal



pluralsight
hardcore developer training

# Outline

- Spaghetti and Ravioli
- Object Literals
- Module Pattern
- Revealing Module Pattern

# Module Pattern

- **Anonymous Closures**
  - Functions for encapsulation
- **Immediate function invocation**
- **Private and public members**

# Anonymous Closure

- **Function expression instead of function definition**
  - Wrapped in parens
- **Scoped**
  - All vars and functions are enclosed

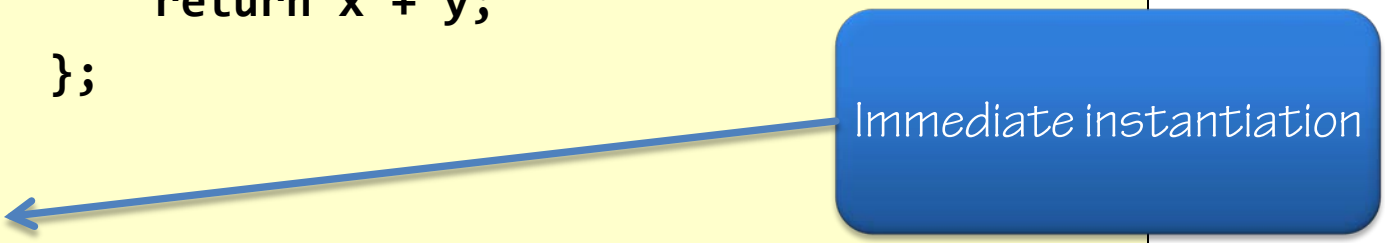
```
(function () {

}());
```

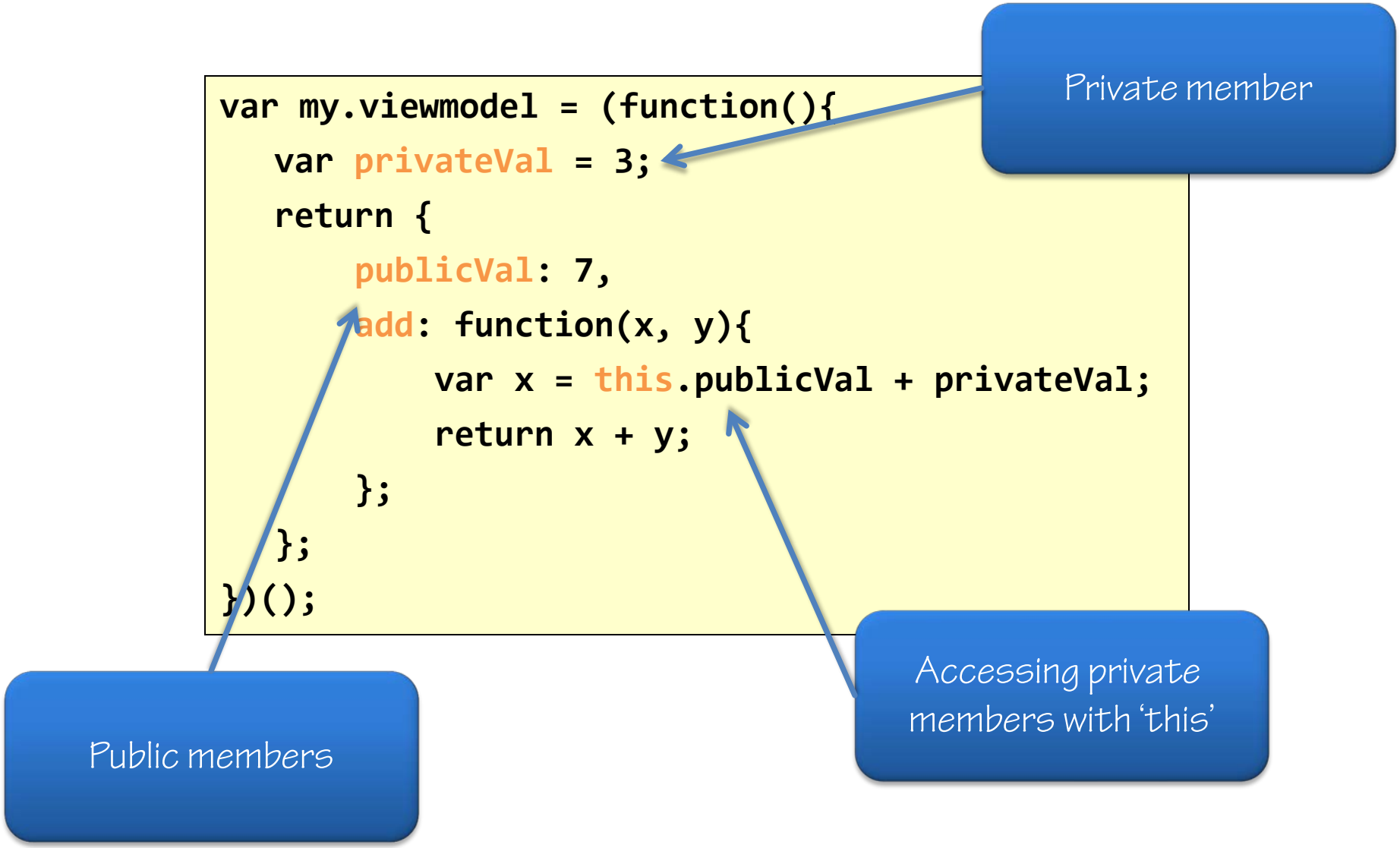
# Immediate Function Invocation

- Create a module
- Immediately available

```
my.viewmodel = (function(){
   var tempValue = 1;
   return {
       someVal: "john",
       add: function(x, y){
           this.tempValue = x;
           return x + y;
       };
   };
})();
```



# Private/Public Members



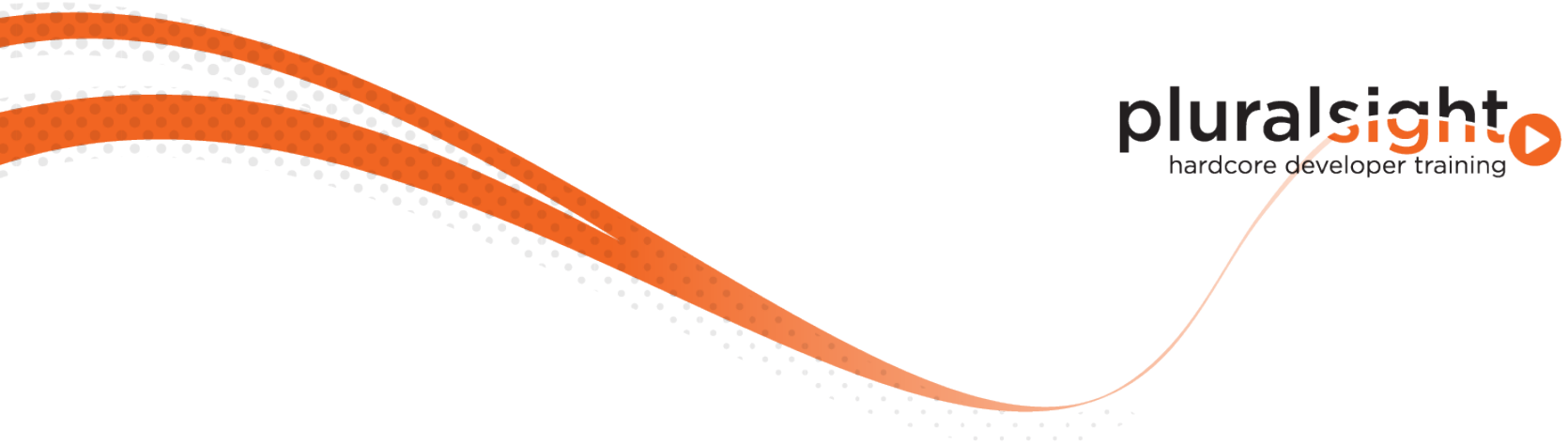
# The Module Pattern

- **Benefits:**
  - Modularize code into re-useable objects
  - Variables/functions taken out of global namespace
  - Expose only public members
  - Hide plumbing code

- **Challenges:**
  - Access public and private members differently

# DEMO

ViewModel as a Module



pluralsight
hardcore developer training

# Outline

- Spaghetti and Ravioli
- Object Literals
- Module Pattern
- Revealing Module Pattern

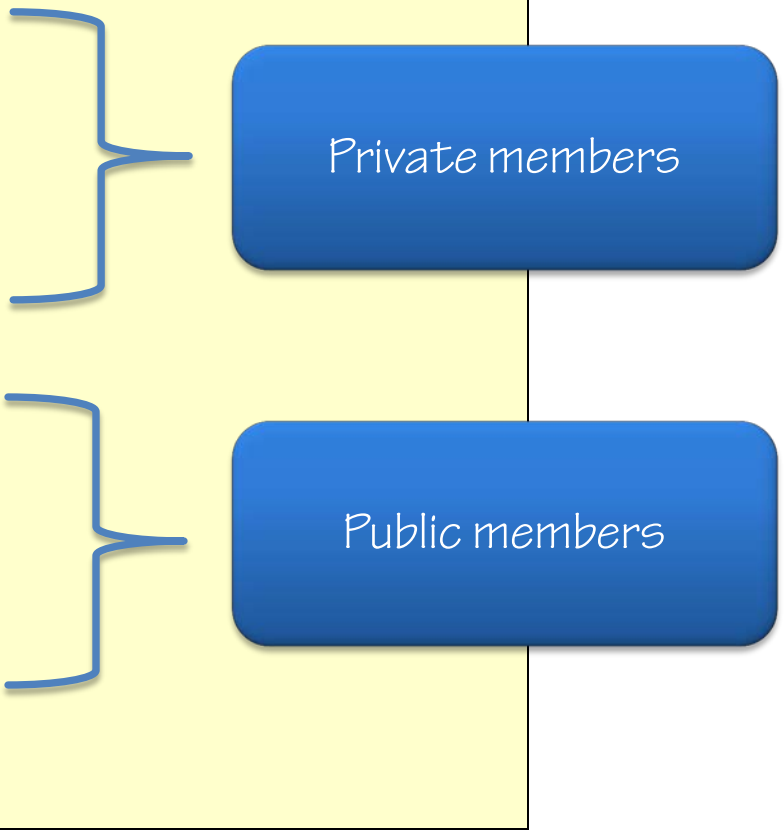
# The Revealing Module Pattern

- **All the Benefits of the Module Patterns +**
  - Makes it clear what is public vs private
  - Helps clarify "this"
  - Reveal private functions with different names

# Revealing

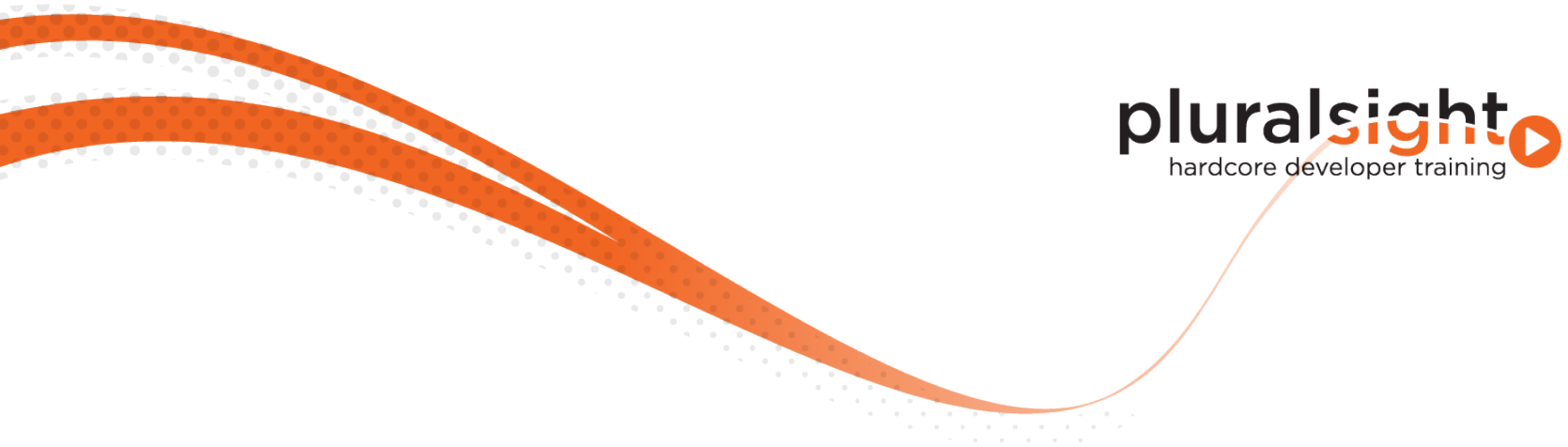
```
my.viewmodel = (function(){
    var privateVal = 3,
        add: function(x, y){
            return x + y;
        };

    return {
        someVal: privateVal,
        add: add
        };
    };
})();
```



# DEMO

ViewModel as a Revealing Module



pluralsight
hardcore developer training

# Summary

- **Spaghetti and Ravioli**
  - Avoid globals
  - Avoid function spaghetti code
  - Separation of presentation, structure, and behavior
- **3 ViewModel Patterns**
  - Object Literals
  - Module Pattern
  - Revealing Module Pattern

**Spaghetti to Ravioli**