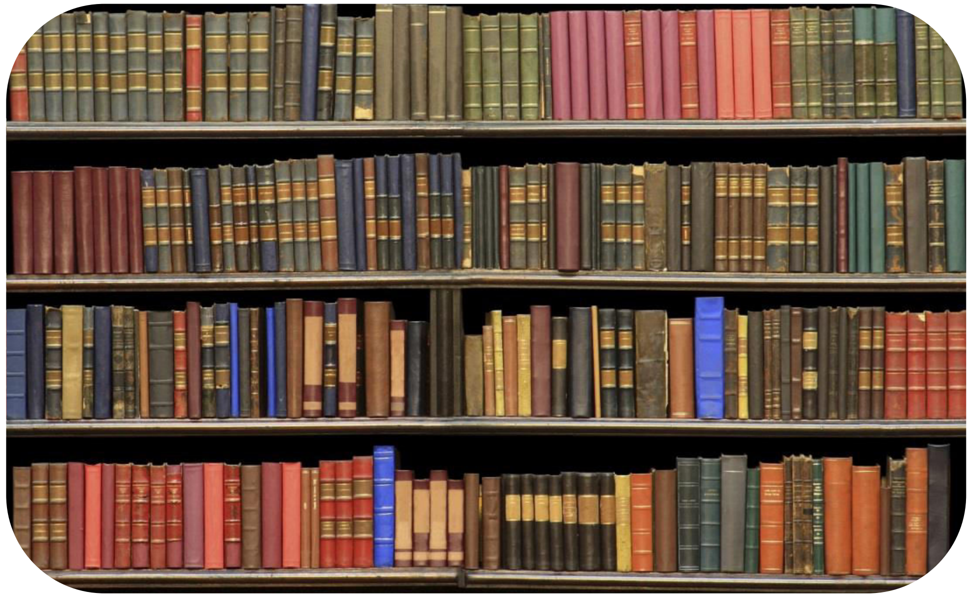
Telerik®
Develop experiences

JavaScript Modules and Patterns

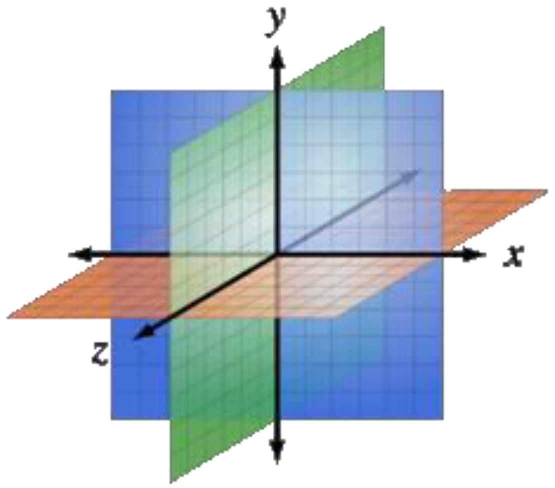
Telerik Software Academy
http://academy.telerik.com

Table of Contents



Public/Private fields

Using the function scope



Public/Private Fields

- Each variable is defined:
  - In the global scope (Public)
  - In a function scope (Private)

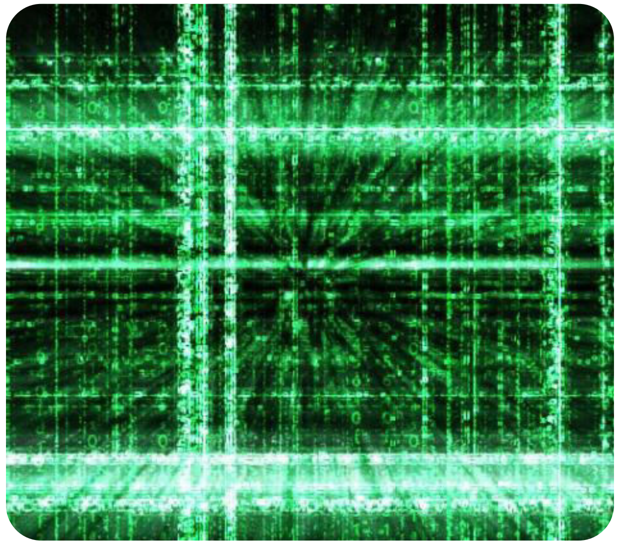
```
var global = 5;

function myFunction() {
  var private = global;

  function innerFunction(){
    var innerPrivate = private;
  }
}
```

Public/Private fields

Live Demo



The Module Pattern

Hide members

Pros and Cons

- Pros:
  - “Modularize” code into re-useable objects
  - Variables/functions not in global namespace
  - Expose only public members
- Cons:
  - Not easy to extend
  - Some complain about debugging

Telerik

Module Pattern: Structure

```
var module = (function() {
	//private variables
	//private functions

	return {
		//public members
		someFunc: function() {…},
		anotherFunc: function() {…}
	};
}());
```

Module Pattern: Summary

- Module pattern provides encapsulation of variables and functions
- Provides a way to add visibility (public versus private) to members
- Each object instance creates new copies of functions in memory

Module Pattern

Live Demo

The Revealing Module Pattern

Reveal the most interesting members

Revealing Module Pattern: Pros and Cons

- Pros:
  - “Modularize” code into re-useable objects
  - Variables/functions taken out of global namespace
  - Expose only visible members
  - "Cleaner" way to expose members
  - Easy to change members privacy
- Cons:
  - Not easy to extend
  - Some complain about debugging
  - Hard to mock hidden objects for testing

Telerik

Revealing Module Pattern: Structure

```
var module = (function() {
          //hidden variables
          //hidden functions

          return {
            //visible members
                    someFunc: referenceToFunction
                    anotherFunc: referenceToOtherFunction
          };
}());
```

Revealing Module Pattern: Summary

- Module pattern provides encapsulation of variables and functions
- Provides a way to add visibility (public versus private) to members
- Extending objects can be difficult since no prototyping is used

Telerik

Revealing Module Pattern

Live Demo

The Revealing Prototype Pattern

Reveal the most interesting members (again)

Revealing Prototype Pattern: Pros and Cons

- Pros:
  - “Modularize” code into re-useable objects
  - Variables/functions taken out of global namespace
  - Expose only public members
  - Functions are loaded into memory once
  - Extensible
- Cons:
  - "this" can be tricky
  - Constructor is separated from prototype

Telerik

Revealing Prototype Pattern: Structure

```
var Constructor = function () {
        //constructor defined here
}

Constructor.prototype = (function() {
        //hidden variables
        //hidden functions

        return {
          //exposed members
                someFunc: pointerToSomeFunc
                anotherFunc: pointerToAnotherFunc
        };
}());
```

Telerik

Revealing Prototype Pattern: Summary

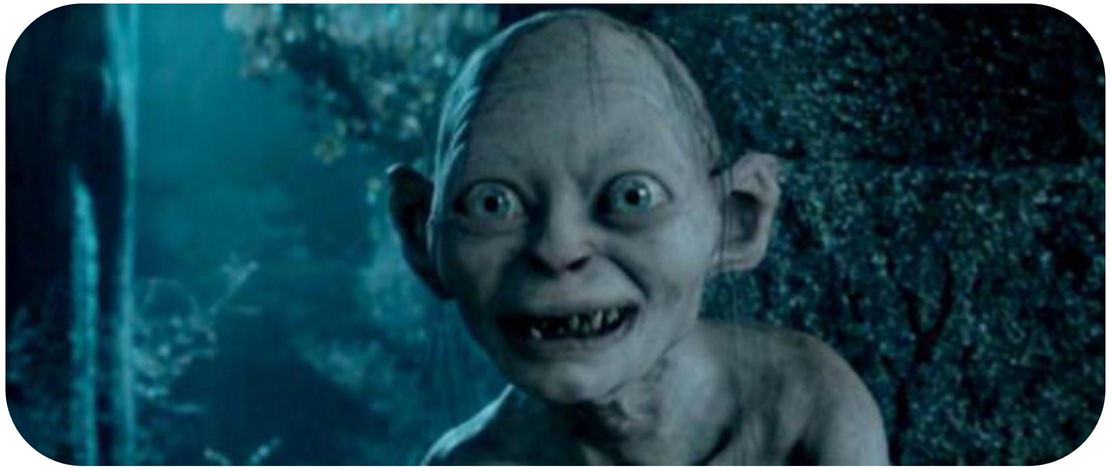
- Module pattern provides encapsulation of variables and functions
- Provides a way to add visibility (exposed versus hidden) to members
- Provides extension capabilities

Revealing Prototype Pattern

Live Demo

Singleton Pattern

One object to rule them all!



Singleton Pattern: Structure

```
var module = function() {
  var instance, getInstance;

  return {
    getInstance: function(){
      if(!instance){
        instance = new Instance();
      }
      return instance;
    }
  };
}();
```

Singleton Pattern

Live Demo

Augmenting Modules

Live Demo

JavaScript Modules and Patterns

Questions?

http://academy.telerik.com