Modules

Objectives

- Understand the module pattern in JavaScript

Constructing objects

- Many ways to create JavaScript objects

```
var user = {
    name: 'Kevin'
};

function User(){}

var kevin = new User();
```

Constructing objects

- Using 'new' can have issues
  - Each instance has own copies of functions
- Use prototype instead for shared functions

```
function User(){
};

User.prototype.setName = function(){
};

var kevin = new User();
kevin.setName("Kevin");
var terry = new User();
terry.setName("Terry");
```

Module Patterns

- Modules are a common way of managing JavaScript code
  - Allow for encapsulation
  - Good tool support (require, CJS, AMD)
  - Often created using Immediate Functions

```
(function(){...})();
```

Why the Module Pattern?

- The major benefit is encapsulation
  - Can pass needed dependencies to the module …
  - … scoped within the module
  - Variables are scoped within the module
- Other benefit is what you return …

(Revealing) Module Pattern

- Return what you need from the module
  - Object
  - Constructor

Returning an object

- Pass in jQuery reference
  - scoped to module
- Return an object that exposes functionality

```
var authn = (function ($) {
    var email = "";

    var vm = {
        email: email,
        signIn: signIn
    };

    function initialize(params) {}
    function signIn() {
        $.post("")…
    };

    return vm;
})(jQuery);
```

Returning a constructor

- Can now create instances

```
function blogPost () {
    var shared;
    function BlogPost(item) {
        var title;
        if (item != null) {
            this = item.title;
        }
        this.getTitle = function () {
            return title;
        };
    }
    return BlogPost;
};

var ctor = blogPost();
var post = new ctor({title: 'Title'});
```

Augmenting module

- **Add methods to an existing module**

```
var MODULE = (function (my) {
  my.anotherMethod = function(){}
  return my;
})(MODULE);
```

Loose Augmenting module

- **Add methods to an existing module**
  - **weird MODULE || {} checks if module exists in global namespace**

```
var MODULE = (function (my) {
  my.anotherMethod = function(){}
  return my;
})(MODULE || {});
```

Tight Augmenting module

- **Add methods to an existing module**

```
var MODULE = (function (my) {
  var old_moduleMethod = my.moduleMethod;

   my.moduleMethod = function () {
    // method override, has access to old through
    // old_moduleMethod...
  return my;
})(MODULE || {});
```

Summary

- Module pattern is powerful
- Used for encapsulation
- Can be used as an extension mechanism