**Encapsulation: Part 2**

**The Module Pattern**

The module pattern overcomes some of the limitations of the object literal, offering privacy for variables and functions while exposing a public API if desired.

**Example 10.3. The module pattern**

var feature =(function() {

// private variables and functions

var privateThing = 'secret',

publicThing = 'not secret',

changePrivateThing = function() {

privateThing = 'super secret';

},

sayPrivateThing = function() {

console.log(privateThing);

changePrivateThing();

};

// public API

return {

publicThing : publicThing,

sayPrivateThing : sayPrivateThing

}

})();

feature.publicThing; // 'not secret'

feature.sayPrivateThing();

// logs 'secret' and changes the value

// of privateThing

In the example above, we self-execute an anonymous function that returns an object. Inside of the function, we define some variables. Because the variables are defined inside of the function, we don't have access to them outside of the function unless we put them in the return object. This means that no code outside of the function has access to the privateThing variable or to the changePrivateThing function. However, sayPrivateThing does have access to privateThing and changePrivateThing, because both were defined in the same scope as sayPrivateThing.

This pattern is powerful because, as you can gather from the variable names, it can give you private variables and functions while exposing a limited API consisting of the returned object's properties and methods.

Below is a revised version of the previous example, showing how we could create the same feature using the module pattern while only exposing one public method of the module, showItemByIndex().

**Example 10.4. Using the module pattern for a jQuery feature**

$(document).ready(function() {

var feature = (function() {

var $items = $('#myFeature li'),

$container = $('<div class="container"></div>'),

$currentItem,

urlBase = '/foo.php?item=',

createContainer = function() {

var $i = $(this),

$c = $container.clone().appendTo($i);

$i.data('container', $c);

},

buildUrl = function() {

return urlBase + $currentItem.attr('id');

},

showItem = function() {

var $currentItem = $(this);

getContent(showContent);

},

showItemByIndex = function(idx) {

$.proxy(showItem, $items.get(idx));

},

getContent = function(callback) {

$currentItem.data('container').load(buildUrl(), callback);

},

showContent = function() {

$currentItem.data('container').show();

hideContent();

},

hideContent = function() {

$currentItem.siblings()

.each(function() {

$(this).data('container').hide();

});

};

$items

.each(createContainer)

.click(showItem);

return { showItemByIndex : showItemByIndex };

})();

feature.showItemByIndex(0);

});