**Managing Dependencies: Part 2**

**Creating Reusable Modules with RequireJS**

RequireJS makes it easy to define reusable modules via require.def(). A RequireJS module can have dependencies that can be used to define a module, and a RequireJS module can return a value — an object, a function, whatever — that can then be consumed by yet other modules.

If your module does not have any dependencies, then just specify the name of the module as the first argument to require.def(). The second argument is just an object literal that defines the module's properties. For example:

**Example 10.7. Defining a RequireJS module that has no dependencies**

require.def("my/simpleshirt",

{

color: "black",

size: "unisize"

}

);

This example would be stored in a my/simpleshirt.js file.

If your module has dependencies, you can specify the dependencies as the second argument to require.def() (as an array) and then pass a function as the third argument. The function will be called to define the module once all dependencies have loaded. The function receives the values returned by the dependencies as its arguments (in the same order they were required in the array), and the function should return an object that defines the module.

**Example 10.8. Defining a RequireJS module with dependencies**

require.def("my/shirt",

["my/cart", "my/inventory"],

function(cart, inventory) {

//return an object to define the "my/shirt" module.

return {

color: "blue",

size: "large"

addToCart: function() {

inventory.decrement(this);

cart.add(this);

}

}

}

);

In this example, a my/shirt module is created. It depends on my/cart and my/inventory. On disk, the files are structured like this:

my/cart.js

my/inventory.js

my/shirt.js

The function that defines my/shirt is not called until the my/cart and my/inventory modules have been loaded, and the function receives the modules as the cart and inventory arguments. The order of the function arguments must match the order in which the dependencies were required in the dependencies array. The object returned by the function call defines the my/shirt module. Be defining modules in this way, my/shirt does not exist as a global object. Modules that define globals are explicitly discouraged, so multiple versions of a module can exist in a page at a time.

Modules do not have to return objects; any valid return value from a function is allowed.

**Example 10.9. Defining a RequireJS module that returns a function**

require.def("my/title",

["my/dependency1", "my/dependency2"],

function(dep1, dep2) {

//return a function to define "my/title". It gets or sets

//the window title.

return function(title) {

return title ? (window.title = title) : window.title;

}

}

);

Only one module should be required per JavaScript file.