Managing resuable modules with components over separate files in AngularJs

by Benny Bottema - Saturday, January 11, 2014

http://www.bennybottema.com/2014/01/11/managing-resuable-modules-with-components-over-separate-files-in-angularjs/

Here's a trick I'm using to easily define reusable components over multiple files, contained within a specific module (which will function as a collection of components).

First, know that calling angular.module('MyModule') **references an existing module**, whereas angular.module('MyModule', []) **always creates a new one** (overwriting the old one)! So this poses the question: Where do we create the generic module? The answer is: either you include a javascript file that does this, or you do it manually for each app.

For example, let's define an *MainApp* module and a *GenericComponents* module, with components spread across files. Here's one way to do that:

```
// directive-foo.js:
angular.module('GenericComponents').directive('DirectiveFoo', ...)
// controller-moo.js:
angular.module('GenericComponents').controller('MooController', ...)
// providers.js:
angular.module('GenericComponents').directive('MyConfigProvider', ...)
angular.module('GenericComponents').directive('MyFirstProvider', ...)
// services.js:
angular.module('GenericComponents').directive('GenericDataService', ..
.)
angular.module('GenericComponents').directive('someOtherservice', ...)
```

And the main application script:

```
// main-app.js:
var app = angular.module('MainApp', ['GenericComponents'])
app.service('MainAppFooService', ...)
(..)
```

Now this won’t run, because AngularJs will complain that it can’t find the module called “GenericComponents”. This makes sense since we aren’t actually creating it, only referencing it. To fix this here’s the trick I’m currently using:

```
angular.module('GenericComponents', []);

// main-app.js:
var app = angular.module('MainApp', ['GenericComponents'])
app.service('MainAppFooService', ...)
(..)
```

I’m creating an empty module “GenericComponents” which will be populated during Angular’s config phase. The implementation-neutral way is to include a *generic-components.js* file that contains that one line of code, but I find that to be useful to purists only, unless you need module specific (global) configuration that you want to do in one place. In that case it makes sense to keep that code encapsulated in its own script.
