Using require.js

Relevant Links

- require.js home page[1]
- require.js API[2]
- AMD Configuration Options[3]
- AMD Loader Plugins[4]

Notes

In this section we will discuss setting up an AMD-based application with require.js. Other AMD loaders will work in similar ways. You should consult the require.js documentation page for details.

We will use as a model our WebAppsTodo application[5].

Application Structure

This is by no means the only way to structure the application, but it is a common way:

```
Project Directory/
    index.html          <-- base html file that starts it all
    require.js          <-- the require.js file. Sets things up
    build.js            <-- file responsible for creating an optimized build
    app/                <-- contains your application code
        main.js         <-- loaded from require.js. Kickstarts app
        helper/         <-- contains helper modules (e.g. data structures)
            util.js
            mixin.js
        helper.js       <-- container for the various helper files
        otherFolders/   <-- to organize your app's pieces
    lib/                <-- for other people's modules
        jquery.js
    build/              <-- the results of an optimized build process
    test/               <-- tests
        test1.spec.js
        test2.spec.js
```

A common pattern that you will see often is to use a container file for all parts of a module or subfolder. For instance in the above made up example, there are two files in the helper folder. But we can also create a helper.js, which simply puts them all together:

[1] http://requirejs.org/
[2] http://requirejs.org/docs/api.html
[3] https://github.com/amdjs/amdjs-api/blob/master/CommonConfig.html
[4] https://github.com/amdjs/amdjs-api/blob/master/LoaderPlugins.html
[5] https://github.com/skiadas/WebAppsTodo

```
define(
    ["./helper/util", "./helper/mixin"],
    function(util, mixin) {
        return {
            util: util,
            mixin: mixin
        };
});
```

So a single file that exports as an object all the contents of the folder. Then other parts of the application can use "helper/mixin" if they want for instance only one of the parts, or "helper" if they want to get all subparts at once.

Configuration Options

There are a number of configuration options that require.js supports[6]. They are not officially part of the AMD standard yet, but most AMD loaders should support them. However, tread carefully.

These will typically be added in a require.config call or something similar.

baseUrl[7] a string indicating the root to be used for path resolutions. Paths are relative to the current working directory. For instance you can use something like "./app" if all your files are to reside inside app.

paths[8] an object of paths to be used for given module prefixes. This is useful for modules that do not reside under the baseUrl. For example it would be customary here to associate "jquery" with the path to your jquery installation.

Some implementations allow for an array of paths as the value associated to a module prefix. These paths will be accessed in order until one succeeds. For example for jquery we can use an array consisting of a CDN link followed by a local fallback.

packages[9] used to load packages that are in a CommonJS structure (e.g. Node packages). That structure specifies a package.json file with information about the package, the location of a main file etc. Look at the corresponding documentation[10] for details (also the packages specification[11]).

config[12] The configuration object may itself have a config property. You can use that property to pass configuration options to modules. Modules access these through their module property, as in module.config.bar.

shim[13] used for linking to files that define a global value (rather than using AMD). Useful for incorporating legacy code. See the require.js documentation (and also the AMD specification[14]) for more details.

[6] http://requirejs.org/docs/api.html#config-paths
[10] http://requirejs.org/docs/api.html#packages
[11] http://wiki.commonjs.org/wiki/Packages/1.1
[14] https://github.com/amdjs/amdjs-api/blob/master/CommonConfig.html

Loader Plugins

require.js and other AMD loaders provide support for plugins (see require.js[15] and AMD[16] pages).

Plugins are essentially module files of a specific form, and loaded in a specific way. A lot of plugins[17] already exist. The most standard of those is the "text" plugin for inclusion of templates, and the "domReady" plugin for specifying actions to be taken after the page has loaded.

"Modules" that should be handled by a plugin rather than be treated as normal Javascript files are specified by prepending the plugin, followed by an exclamation mark, to the module name. So something like "text!path/to/text/or/html/file" would interpret the contents of that path as a text file and return it as a string.

[15] http://requirejs.org/docs/plugins.html
[16] https://github.com/amdjs/amdjs-api/blob/master/LoaderPlugins.html
[17] http://requirejs.org/docs/api.html#plugins