The goal of This Proposal Is to define the scope for FuseBox to tackle multi-packaging and lazy loading packages.

The targeted environment is both Browser and NodeJs.

Today, with http://fuse-box.org/#making-many-bundles-at-once and http://fuse-box.org/#loader-api, we can already do

some pieces of requested features. However, doing this requires a manual work and well-known Information Architecture (IA) of your application.

Let's summarize what ingredients we already have in place:

1. Require(…), import ES6 and FuseBox.import
2. FuseBox events
3. FuseBox wildcards import

The point 1) is the most used syntax for everyone, this should be kept as it is because it allows easily for everyone to adopt the concept,

The point 2) allows to listen any requests from point 1), it can act as a sentinel,

The point 3) allows to import modules by using pattern in a configuration

Combining these 3 points allows us to define a well-known configuration to achieve our feature, this configuration describes all packages that we would like to bundle. One important thing that we need to have in this configuration is an entry point, let's call it « fuse-runtime.js ». This package is mainly our entry point and might contain all shareable packages. This file name can be whatever we want from the configuration.

The file « fuse-runtime.js » needs to be inserted in the script tag. However, loading package is conditional and per context of execution, for instance, in a Browser, this will be most probably the « route »:

1) If you browse to Home page with (/),

2) If you browse to About us with (/about-us),

Regarding the context of the route, we should know what appropriate bundle need to be loaded, it can be one or more packages. How we know that? We need to provide such metadata or manifest to achieve that.

There are 3 kinds of packages:

1) From your application, itself
2) From external npm package
3) From FuseBox bundler ☺

The good part here is that points 1) and 3) can take benefits of FuseBox feature « Wildcard » import. This can be useful to dispatch modules to package for bundling. However, point 2) will be based on the name of the npm package.

This being said, the tips to generate our bundles is to have a sort of « Reverse Proxy » feature to listen to what is being imported, and this is done by the FuseBox event.

Let's say that we want to import:

1- import * as moment from 'moment' → P1.js

2- let c = require(`./libs/c`) → P2.js

3- import {B} from './libs/b' → P3.js

4- let a1 = require('./libs/a') → fuse-runtime.js

In our example, we are going to have 4 bundles with FuseBox.

- **fuse-runtime.js**
- *P1.js*
- *P2.js*
- *P3.js*

We need to be able to generate the manifest that allow us to map which modules belong to which package, so that we can load then the package, this can be applied on lazy loading but not when you are navigating in a route.

So, we need to handle the route with some key to look up the manifest with the package to load. This means that a route can be used to make a direct load of the package associated to it.

Think of the route as a unique key to load packages, it would be interesting that a Front-End framework can implement their own strategy.

The configuration required to generate the bundles should be a simple json file like below:

1. fuse-runtime.js => ***index.ts + [./libs.c]***
2. P1.js => ***moment***
3. P2.js => ***./libs/c***
4. P3.js => ***./libs/b***

The good part of this json file is that it leverages already what already exist from multiple bundles.

The tips of the route are very important to contextualize how packages are loaded. Also, this information can be tracked by FuseBox events so it might give us insights to better design our packaging when our application is in large scale.

Let’s say that fuse-runtime.js is loaded on every page, and this fuse-runtime contains the minimal features to run your application. P1, P2 and P3 are loaded only per route request:

1. /about-us → P1
2. /our-products → P2
3. /creating-shared-value → P3

Building an engine based on rule with « **path-to-regexp** » module will simplify how to load packages based on route versus an explicit lazy load package.

When a route is matched, it gives back what packages is loaded immediately.

At the end, this will give us a chance to empower fuse to made an « **Adaptive bundling** » feature.