Create 3rdparty libs for egret

Friday, April 13, 2018 6:28 PM

This guide will use puremvc-typescript-multicore library as an example.
The original libraries contains 3 files if you use npm install puremvc-typescript-multicore.

```
<puremvc-typescript-multicore>
|___dist
      |____puremvc-typescript-multicore-1.1.d.ts
      |____puremvc-typescript-multicore-1.1.js
      |____puremvc-typescript-multicore-1.1-min.js
```

I. Use egret-cli to create an empty library

Execute this command in a directory outside of your project, e.g. Create a 3rdparty-libs directory and then:

```
$> egret create_lib puremvc
```

```
<workspace>
|
|___3rdparty-libs
|    |
|    |__puremvc
|         |____package.json
|         |____tsconfig.json
|
|____egret-project1
|
|____egret-project2
```

II. Create src folder & config.json file

Create them in the previously created 'puremvc' folder, then move the source files downloaded via npm into 'src':

```
3rdparty-libs
|
|___puremvc
    |
    |___config.json
    |
    |___src
    |    |___puremvc-typescript-multicore-1.1.d.ts
    |    |___puremvc-typescript-multicore-1.1.js
    |    |___puremvc-typescript-multicore-1.1-min.js
    |
    |___package.json
    |
```

|___tsconfig.json

III. Edit config.json

Edit the file with below content, this will be used as uRequire config:

```
{
    "path": "src",
    "dstPath": "build",
    "filez": [
        "**/*"
    ],
    "noLoaderUMD": true,
    "bundle": {
        "rootExports": {
            "puremvc": "puremvc"
        }
    }
}
```

IV. Install urequire-cli AND urequire local module

Execute commands in puremvc folder:

```
$> npm install urequire-cli -g
$> npm install urequire
```

V. Edit source files to export globals

Edit puremvc-typescript-multicore-1.1.js & puremvc-typescript-multicore-1.1-min.js,
and insert an object literal on the top of these file:
({urequire: { rootExports: 'puremvc' } });

----------Refer to urequire docs below----------
uRequire can generate the rootExports boilerplate. You can declaratively export one (or more) global variables from your UMD/AMD module on the web side.

You simply include an object literal on the top of your (source) module file like this:

({urequire: { rootExports: 'uBerscore' } });
Or use an array :

({urequire: { rootExports: ['uBerscore', '_B']}});
in case you want many global vars.

These globals be created as keys on 'root' (e.g. window on browsers, global on nodejs), with the module as the value (possibly overwriting existing keys).

Alternatively, if you don't want to pollute your modules with exporting info, you can handle it in the config, in the bundle.dependencies.exports.root key.

From <http://www.urequire.org/exporting-modules>

VI. Execute uRequire command

```
$> urequire config config.json
```

Now your folder structure will be look like this:

Name	Date modified	Type
build	4/18/2018 10:01 AM	File folder
node_modules	4/18/2018 10:02 AM	File folder
src	4/18/2018 10:01 AM	File folder
config.json	4/18/2018 10:02 AM	JSON Source File
package.json	4/17/2018 6:31 PM	JSON Source File
package-lock.json	4/18/2018 10:02 AM	JSON Source File
tsconfig.json	4/17/2018 6:31 PM	JSON Source File

|__build ----your urequire output files here
|__node_modules ---your other local tools, include urequire packages
|__src -------your source files

VII. Configure tsconfig.json

```
{
    "compilerOptions": {
        "target": "es5",
        "noImplicitAny": false,
        "sourceMap": false,
        "outFile": "bin/puremvc.js",
        "allowJs": true
    },
    "files": [
        "build/puremvc-typescript-multicore-1.1-min.js",
        "build/puremvc-typescript-multicore-1.1.js",
        "src/puremvc-typescript-multicore-1.1.d.ts"
    ],
    "include": [
        "build"
    ]
}
```

VIII. Execute egret build

Execute command in puremvc folder:
```
$> egret build
```

From <https://www.jianshu.com/p/c3cb7548a4a3>

IX. Edit egretProperties.json

Now open your primary egret project, and configure your 3rdparty lib in your egretProperties.json. Since we have created puremvc lib in a directory outside the project root folder, "../" symbol is needed

to find the place:

egretProperties.json

```
{
    "native": {
        "path_ignore": []
    },
    "publish": {…
    },
    "egret_version": "5.1.9",
    "eui": {…
    },
    "modules": [
        {
            "name": "egret"
        },
        {
            "name": "eui"
        },
        {
            "name": "assetsmanager"
        },
        {
            "name": "tween"
        },
        {
            "name": "game"
        },
        {
            "name": "promise"
        },
        {
            "name": "puremvc",
            "path": "../3rdparty-libs/puremvc"
        }
    ]
}
```

Refer to egret docs, 3rdparty libs will be copied to /libs/modules automatically when build project, if we put them outside our project directory. This is an expected behavior.