JavaScript Package Management

Steve Hartzog

1. What is a *JavaScript* package?

1. What is a *JavaScript* package manager?

1. Where are packages hosted?

What is a JavaScript package?

It’s a 3rd party library…

inside a downloadable unit.

And there is a specification.

CommonJS Package Spec 1.0

Designed to distribute CommonJS programs
Specifies a package descriptor format
Must use package.json

package.json fields

Required Fields:

Name
Description
Version
Keywords
Maintainers
Contributors
Bugs
Licenses
Repositories
Dependencies

Optional Fields:

home page
os
cpu
engine
builtin
directories
implements
scripts {
“dep”: “jspm update”
}

package.json sample

```
{
        “name”: “travel-channel”,
        “version”: “0.1.0”,  ~~//> currently remains static
        “dependencies”: {
                “time-grunt”: “~0.3.1”
        }
        “devDependencies”: {
                “grunt”: “0.4.4”,
                “grunt-shell”: “~0.6.4”,
                “bower”: “~1.3.3”,
                “load-grunt-tasks”: “~0.5.0”
        }
}
```

What is a JS Package *Manager*?

It’s a way to:

search for a package...

download it...

extract it…

AND update it!

Why use a package manager?

1. Easily update 3rd party dependencies

1. No need to add 3rd party code to our source

1. We *should* load from a public CDN
   … but we can’t always do this
   … or we need for our build only

What is out there?

npm
2.10.1

Started January 2010, now at

jamjs
0.2.17

Started in June 2012, now at

… and now dead.

bower
1.4.1

Started April 2013, now at

jspm

Started September 2013, now

npm

Used for node (server-side) packages

… packages do not *normally* run in browser

Uses the Package 1.0 spec from CommonJS

… and uses package.json for config

Default output is ./node_modules

Packages hosted at npmjs.org

jamjs

Uses the Package 1.0 spec from CommonJS

… and config done in npm's package.json

Default output is ./jam

RequireJS "compatible"

… and includes/recommends require.js

Uses semver

Packages hosted at jamjs.org

Installed with npm

bower

Default output is ./bower_components
Config done in .bowerrc and bower.json
Packages often include the entire github repo
… which requires us to bowercopy out
… although we shouldn’t be distributing
Packages hosted on github ***only***

bower.json

```
{
        “name”: “TC Dependencies”,
        “version”: “0.0.2”,
        “devDependencies”: {
                “requirejs”: “https://github…”
        }
}
```

jspm

Uses the Package 1.0 spec from CommonJS

… and config is done in npm's package.json

Default output is ./jspm_packages

Packages hosted at npm, github, ***locally*** or a **CDN**

Designed for SystemJS (more shortly)

Go watch this: http://guybedford.com/jspm-introduction!

package.json (with jspm)

```
{
        “name”: “travel-channel”, “version”: “0.1.0”,
        “dependencies”: { … }, “devDependencies”: { … },
        “jspm”: {
                “dependencies”: {
                        “bootstrap”: “github:twbs/bootstrap@^3.3.4”,
                        “font-awesome”: “npm:font-awesome@^4.3.0”
                },
                “devDependencies”: {
                        “less”: “github:less/less@^2.5.0”,
                }
        }
}
```

Bower vs. JSPM.io

Bower uses a separate file

… jspm uses existing package.json

Bower supports package load from github

… and from either npm, github or a CDN

Also, JSPM packages show the version in the file name and the folder name.

More? http://stackoverflow.com/questions/25416813/package-manager-bower-vs-jspm

Wait… what was SystemJS?

An async package (module) loader
Supports AMD, CommonJS and ES6 modules
Uses HTTP/2 (SPDY) to combine requests
Is faster than browserify
    … which must unpack/transpile it's bundles in browser
Automatically Transpiles ES6 modules
    … using Traceur or BabelJS
Supports IE8+

Go watch this: http://guybedford.com/jspm-introduction!

Recommendation?

Use **jspm** instead of bower

- Integrates FE dep directly in package.json
- No copy needed

Load modules using **SystemJS,** not RequireJS

- Bundle modules until Akamai supports HTTP/2
- Loads existing AMD modules today
- CommonJS / ES6 modules starting now
- ES6 features starting today

Questions?