JS Monorepos: a Study of Engineering Systems

kchau@microsoft.com

- A practical introduction of JS Monorepos
- What problems exists for engineering systems
- Best solutions we have discovered or built

Tools

Almost all of our tools are written in node.js

- frontend developers can participate in maintaining these tools
- very low friction in publishing and consuming public libraries
- enormous and open ecosystem of re-usable packages

Tech stack of a Node.js tool chain

V8

- C++, parses & runs JS
- Garbage collector
- Bindings for JS to access C++ libraries

Node.js

- modularity with require() and module.exports
- event loop that allows single-threaded JS to have async capability
- a standard library (e.g. for I/O)

Package manager (npm, yarn)

- a package is a folder with a `package.json` file
- registry as a glorified CDN
- authed API for publishing packages
- CLI tool to download package and their dependencies
- CLI to run npm "lifecycle" scripts (e.g.: build, test, bundle, lint)

Node resolution algorithm

```
// located in /project/path/main.js
require("mymodule");
```

Search paths:

- /project/path/**node_modules**/mymodule/**index.js**
- /project/**node_modules**/mymodule/**index.js**
- /**node_modules**/mymodule/**index.js**
- /**node_modules**/mymodule/**index.js**
- $HOME/.node_modules/mymodule/**index.js**
- $HOME/.node_libraries/mymodule/**index.js**
- $PREFIX/lib/node/mymodule/**index.js**

Monorepos

A single git repo that contains many related packages:

```
root:
  package.json
  packages/
    app/
      package.json
    businesslogic/
      package.json
    ui-components/
      package.json
    utils/
      package.json
```

Importing Code

Importing internal and external code are consistent:

```
import React from "react";
import { Button } from "ui-components";
```

Monorepo Management Stack

1. Workspace-enabled package manager

- **Installs externals dependencies** for the repository
- **Links internal packages** to satisfy the node resolution algorithm
- Optional: **hoisting**, **strictness** enforcement (phantom deps)
- On the market: `yarn`, `pnpm`, `rush`, `lerna + npm`

Problem 1.1: Dopplegangers

Dealing with transitive deps of different versions

```
a/
  node_modules/
    b/
      node_modules/
        react/
          package.json  <-- react@2.0.0
    c/
      node_modules/
        react/
          package.json  <-- react@2.0.0
    react/
      package.json      <-- react@1.0.0
```

> more info: https://rushjs.io/pages/advanced/npm_doppelgangers/

Problem 1.2: Phantom Dependencies

`yarn` v1 will "hoist" all the deps up to the top level

```
node_modules/
  react/
    package.json          <-- react@1.0.0
b/
  package.json            <-- does not depend on react
a/
  package.json            <-- depend react
```

Now modules inside "b" can **accidentally** be allowed to depend on react

> more info: https://rushjs.io/pages/advanced/phantom_deps/

Solutions

- `yarn` v2,
- `midgard-yarn` (for perf, v1 behavior)
- `pnpm` workspace
- `npm` v7+
- No silver bullet, sorry!

2. Task scheduler & runner

- **Runs npm scripts** for all packages
- **Optimize** task run speeds at the dev machine and CI
- Optionally in **topological** order or in **parallel**
- On the market: `lerna`, `wsrun`, `rush`, `pnpm recursive`, `lage`

Problem Statement

Optimizes package tasks in a monorepo for a single machine

Solution

A task runner that is:

- **Open sourced**
  - easily shared, public development demands **polish**
  - **easily contributed** to by many groups
- Works with all workspace implementations
- Easy setup
- Minimize idle CPU cores
- **Sublinear increase** in build time per package

... doesn't exist out there, so we built one...

Lage

> *v. to make (Norwegian); pr. LAH-geh*

- Open sourced: https://github.com/microsoft/lage
- Easy to integrate with existing codebase
- Scales up with pipelining
- Scales out with caching and scoping
- Contributions from: OXO, FAST, ODSP, Bohemia (Fluid Exp)

How does it work?

https://microsoft.github.io/lage/guide/levels.html

How to try it at home?

https://microsoft.github.io/lage/guide/getting-started.html

1. npm scripts (build, test, lint) are at package level
2. `npx lage init`
    - creates a `lage.config.js` - configure it
    - adds `lage` as a dep
3. `yarn lage build` or `npm run lage build`

3. Package publish tool

- Automated management of **semver** based on a change description
- **Validation** of description of changes
- **Synchronize versions** between npm registry and git repository
- Automated management of **changelog**
- On the market: `rush`, `lerna`, `semantic-release`, `beachball`

Solution

Use an automated package versioning manager

1. Make a new branch, edit code
2. Run a CLI tool to create a "change file"

```
> beachball change


Please describe the changes for: beachball
? Change type » - Use arrow-keys. Return to submit.
>   Patch       - bug fixes; no backwards incompatible changes.
    Minor       - small feature; backwards compatible changes.
    None        - this change does not affect the published package in any way.
```

3. Push branch & create a PR with complete with change file

Change Files

> An example PR


```
{
  "type": "minor",
  "comment": "Adds the ability to create and publish canary packages",
  "packageName": "beachball",
  "email": "kchau@microsoft.com",
  "dependentChangeType": "patch",
  "date": "2020-09-11T23:37:59.115Z"
}
```

Publishing to npm registry and push to git origin

An example publish

```
> beachball publish

...
Publishing - beachball@1.36.0
publish command: publish --registry https://registry.npmjs.org/ --tag latest --loglevel warn --//registry.npmjs.org/:_authToken=***
Published!
...
Pushing to https://github.com/microsoft/beachball
POST git-receive-pack (1116 bytes)
remote:
remote: GitHub found 6 vulnerabilities on microsoft/beachball's default branch (2 high, 2 moderate, 2 low). To find out more, visit:
remote:      https://github.com/microsoft/beachball/network/alerts
remote:
To https://github.com/microsoft/beachball
   72fce75..f9fa782  HEAD -> master
 * [new tag]         beachball_v1.36.0 -> beachball_v1.36.0
updating local tracking ref 'refs/remotes/origin/master'
```

Where to find everything we mentioned here:

- Flywheel team (OXO web engineering): flywheel-team@microsoft.com
- midgard-yarn - yarn, but faster
- lage repo, docs
- beachball repo, docs
- One JavaScript wiki: https://aka.ms/1js

Appendix

Node.js modularity

The current standard is called "CommonJS"

- 1 JS file is a "module"
- modules can import and export code

Imports and Exports

```
// speech.js
module.exports = {
    sayHello: (name) => console.log(`hello ${name}`);
}

// main.js
const { sayHello } = require('./speech');
sayHello("humans!")
```

Module state

require() caches the module exports as well as the state

```
// count.js - Keeping count
let count = 1;
module.exports.inc = () => count++;

// main.js
const { inc } = require("./count");
console.log(inc()); // displays: 1
console.log(inc()); // displays: 2
```

> advanced mode: you can clear this cache to reset state & "hot reload" the module