My current frontend setup is pretty neat

2017-01-28

From time to time, I see someone on Github forking or starring the Angular boilerplate I made and was using 2 years ago. As everyone knows, time in JavaScript-land tends to operate in a different scale: something done 2 years ago is prehistoric and a generation or two of frameworks went by.

I have been using React for about a year now, mainly for the frontend of Proppy and some experiments on the side. While the churn for the libraries during that year reached a level I had never seen before, it seems to have stabilised a bit. That or I stopped trying to follow JS news, who knows.

Since many people might suffer from analysis paralysis, I decided to document my own choices to help. It will also provide a good laugh for people find that article in a few months/years, missing the date and thinking "people are STILL using X?".

Package manager

Yes, you can even choose your package manager in JavaScript!

`npm` is fine as long as you remember to shrinkwrap and don't mind slow installs.

I settled on yarn, which does the right thing by default: a lock file and deterministic installs. Installing packages is also faster than `npm`, which is a nice bonus. Despite how recent it is, I haven't run into bugs yet.

Module bundler

I use Webpack. Not much to say here, once you understand how to configure it - and it might take a while - it is easy to get going (AKA `cp ../previous_project/webpack.config.js .`).

I also like to use html-webpack-plugin.

Language

I use TypeScript. If you are using plain Javascript or just Babel, I strongly encourage you to try something that adds types, be it Elm, Flow, TypeScript or something else like ScalaJS.

Some people view types as hindrance but they make me so much more productive:

- Object shapes and function arguments are documented and is enforced by the compiler
- I can do sweeping changes in a codebase without any trouble
- Fewer errors: trying to call a function that only takes a `string` with a potentially `undefined` argument? Nope

Elm catch phrase is "no runtime exceptions" but I'd argue it is also valid for anything sufficiently typed.

The big advantage TypeScript has over the alternatives mentioned is the huge community and type definitions are available for tons of libraries.

If you are planning to use TypeScript, make sure you start your project with a very strict config. I use the following for new projects:

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "jsx": "react",
    "experimentalDecorators": true,
    "noImplicitAny": true,
    "strictNullChecks": true,
    "noUnusedParameters": true,
    "noUnusedLocals": true,
    "sourceMap": true
  },
  "exclude": [
    "node_modules"
  ]
}
```

If you are converting a large app to TS, `noImplicitAny` might be a pain so only set it to `true` when done.

I also use tslint for linting, which comes with a default good config.

View

As mentioned in the introduction, I use React. I've also been following/contributing a bit to Inferno as an alternative but it doesn't have great type definitions, which is a deal breaker for me.

I would also recommend to mostly write your own components if design is an important factor and you have time for it instead of using third-party components. Unless it's something like an admin or a private site, then go ahead and use a material library or any UI library.

State management

I've settled on MobX and am pretty happy about it. I wrote about MobX and going from Redux before so I'm not going to repeat myself here.

In short: very simple to use, fully typed and not verbose at all. A pleasure after using Redux.

Testing

I'm currently using Mocha/Chai with Karma in Proppy and it's working well. I have heard good things about recent versions of Jest but haven't had the time to look at it yet, it is the next item on the TO TRY list.

CSS

I am using Sass with some postcss plugins like Lost or Autoprefixer. If you are targeting evergreen browsers, `autoprefixer` won't be needed.

I haven't seen a single CSS-in-JS library that convinced me. The closest was TypeStyle but is ultimately failing because it has a runtime overhead. I'll address that in the "What am I missing" section later on.

Prototyping

I recently discovered react-storybook. If you are using React and didn't hear about it or tried it, stop reading this article and go play with it. It's that glorious.

Rather than explaining how it works, have a look at Airbnb date picker storybook.

Storybook allows you to easily develop components and see all their potential usages in one place, as well as showing an always up-to-date style guide.

What am I missing

I'm still missing a few things and welcome any input.

Compile-time styling

As mentioned before, TypeStyle is really close to a good solution for styling but for me, a styling tool should have no runtime cost. CSS is working well and having JavaScript do its job doesn't really make sense.

I raised an issue for that but it seems rewriting AST is not available in TypeScript yet so a TS plugin that extracts those call is not doable for now.

Animations

I'm still unsure how to *easily* add animations. animate looks like a good candidate though.

TL;DR

My starter `package.json`:

```
{
  "name": "frontend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "clean": "rimraf -- dist",
    "dev": "webpack-dev-server",
    "build": "webpack",
    "storybook": "start-storybook -p 6006",
    "build-storybook": "build-storybook"
  },
  "devDependencies": {
    "@kadira/react-storybook-decorator-centered": "^1.0.0",
    "@kadira/storybook": "^2.21.0",
    "@types/classnames": "^0.0.32",
    "@types/react": "^15.0.0",
    "@types/react-dom": "^0.14.20",
    "autoprefixer": "^6.6.0",
    "css-loader": "^0.26.1",
    "html-webpack-plugin": "^2.24.1",
    "node-sass": "^4.1.1",
    "rimraf": "^2.5.4",
    "sass-loader": "^4.1.1"
    "sass-loader": "^4.1.1",
    "ts-loader": "^2.0.0",
    "tslint": "^4.2.0",
    "tslint-loader": "^3.3.0",
    "typescript": "^2.1.4",
    "webpack": "2.2.0-rc.4",
    "webpack-dev-server": "2.2.0-rc.0"
  },
  "dependencies": {
    "classnames": "^2.2.5",
    "mobx": "^3.0.0",
    "mobx-react": "^4.1.0",
    "react": "^15.4.1",
    "react-dom": "^15.4.1"
  }
}
```