# Practical ReactJS

Web Dev, Spring 2021

# Detour: NodeJS

An infrastructure for programming in Javascript like you would in Python

- not tied to browser
- interactive shell
- program spread over multiple files (each implementing a module)
- can use libraries managed by an external tool

# A NodeJS project

If no libraries used, can just create source files

- test.js
- helper.js

Files can be scripts or  modules (like in Python)

NodeJS by  default uses CommonJS modules

```
const r = require('something')
```

The file "required" must explicitly export something

```
exports.foo = foo
```

# Libraries

If you want to use libraries, you need a package manager to install and track dependencies

`npm init`

- creates package.json

package.json has multiple roles depending on what you do, but it tracks dependencies

`npm install library`

- adds dependency to package.json
- installs the library in `node_modules/`
- can use `require('library')` in files

# Why the detour?

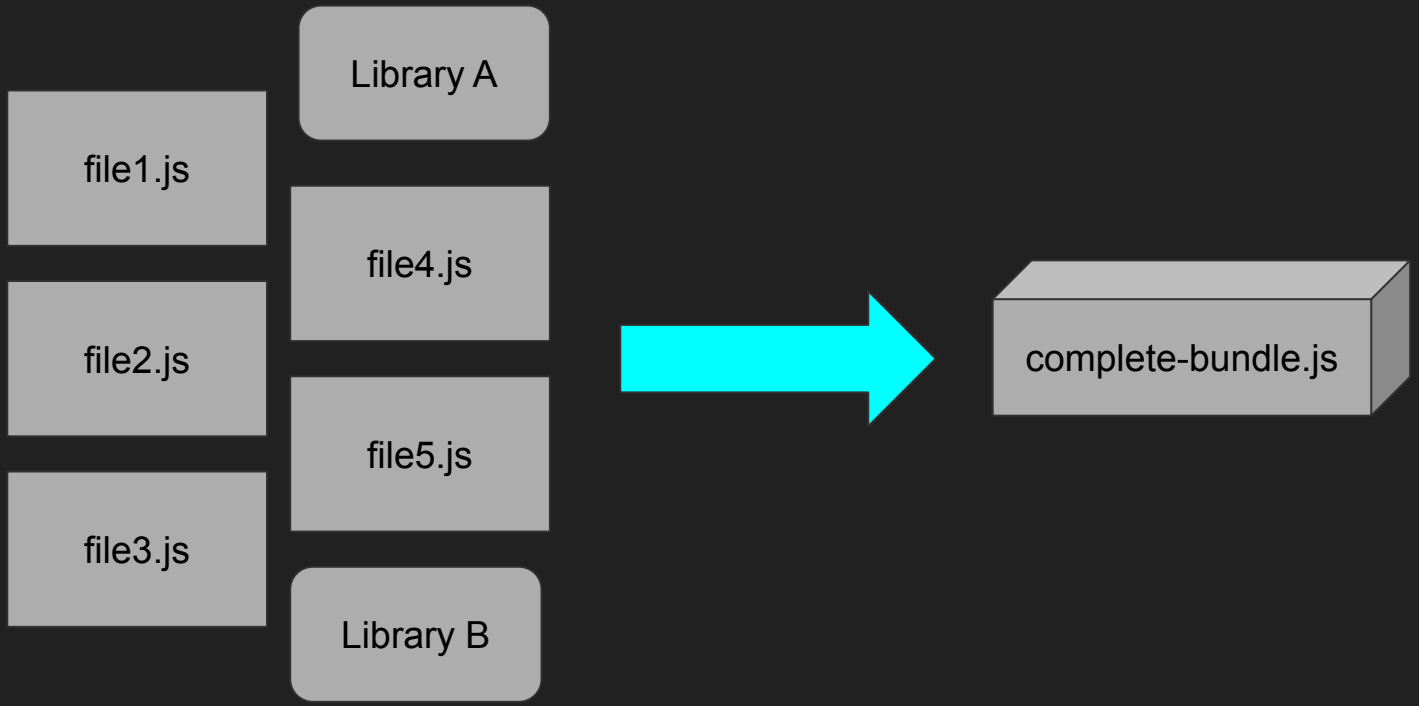
Modern front-end development tools tend to reuse the architecture set down by NodeJS (npm, package.json)

- can spread frontend code over multiple files in different modules
- can use libraries installed via npm
- react is such a library

But:

- frontend code needs to be ultimately run in the browser,
- browser don't know (much) about modules and libraries
- need to bundle all source into a "single" .js file for deployment
    - bundlers: browserify, webpack, rollup, parcel, ...

# Bundling


Library A
file1.js
file4.js
file2.js
file5.js
file3.js
Library B
complete-bundle.js

# create-react-app

It all gets super complicated super fast — and it changes every 6 months

People have developed tools to help manage this complexity

E.g., create-react-app:

NodeJS app to create a package.json, install the basic react libraries, adds a bundler, and a way to run the bundler to create a distributable frontend

# Demo

- create-react-app
- sortable with buttons
- tabbing
- functional components