Client-Side Web Programming: JavaScript (Part 5)

Robert M. Dondero, Ph.D.

Princeton University

Objectives

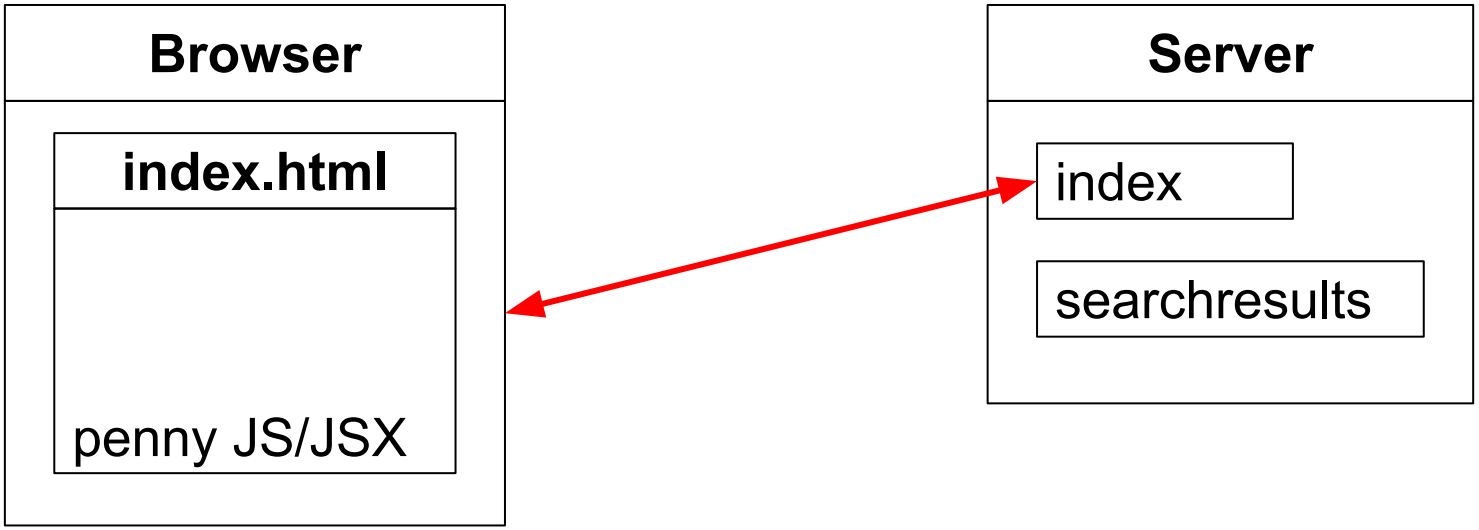
- We will cover:
  - Bundled React
  - Bundled React via Vite
  - React commentary

Agenda

- **Bundled React: motivation**
- Bundled React
- Bundled React: Vite
- React commentary

Bundled React: Motivation

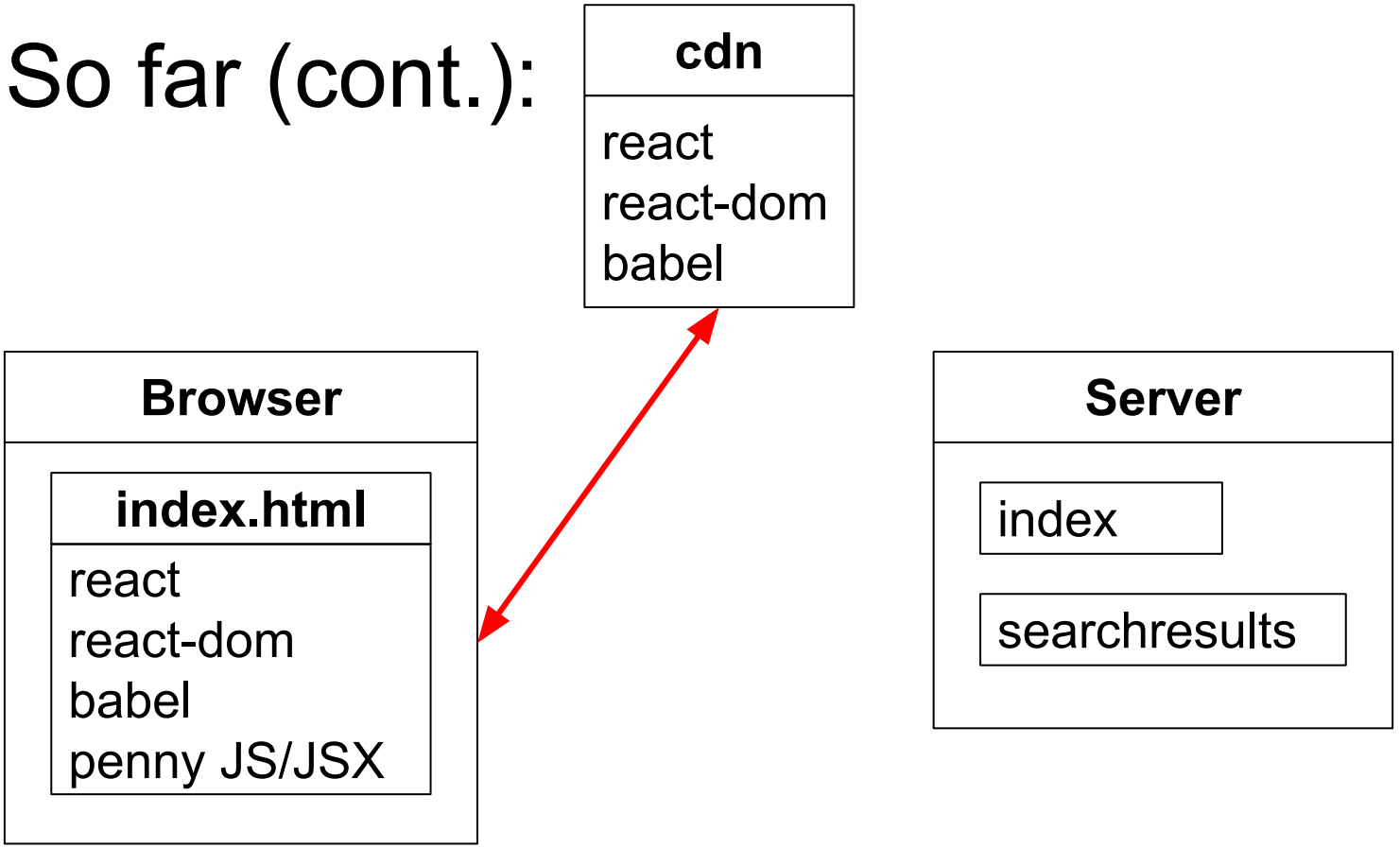
So far:



Browser requests and receives index.html

Bundled React: Motivation

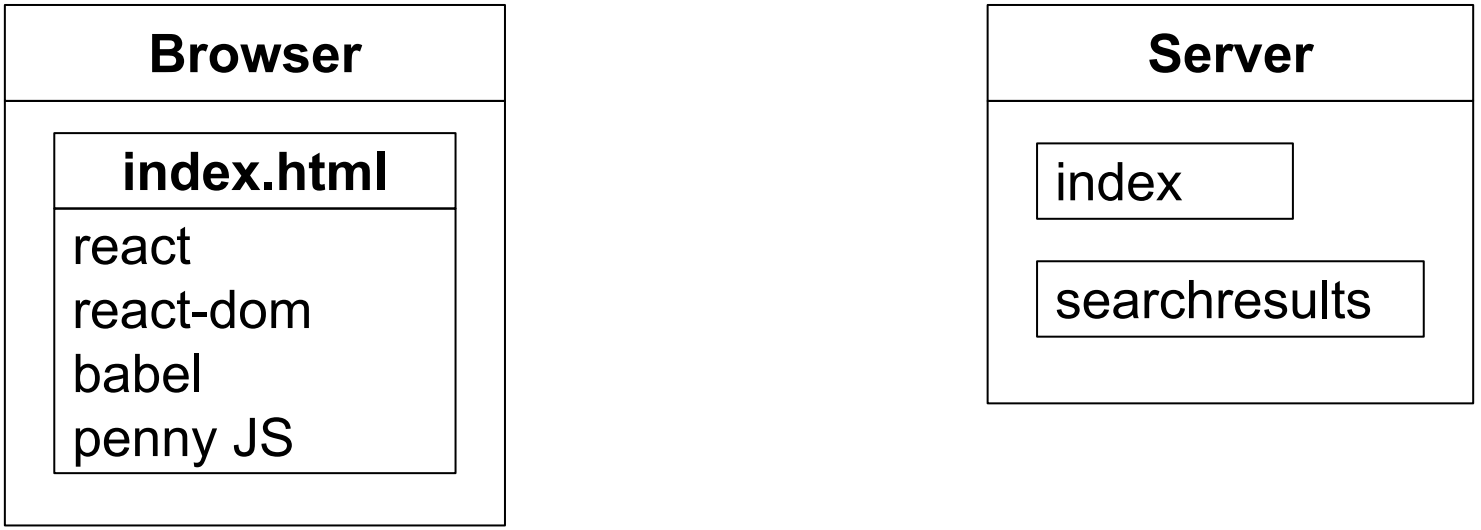
So far (cont.):



Browser requests and receives react, react-dom, and babel

Bundled React: Motivation

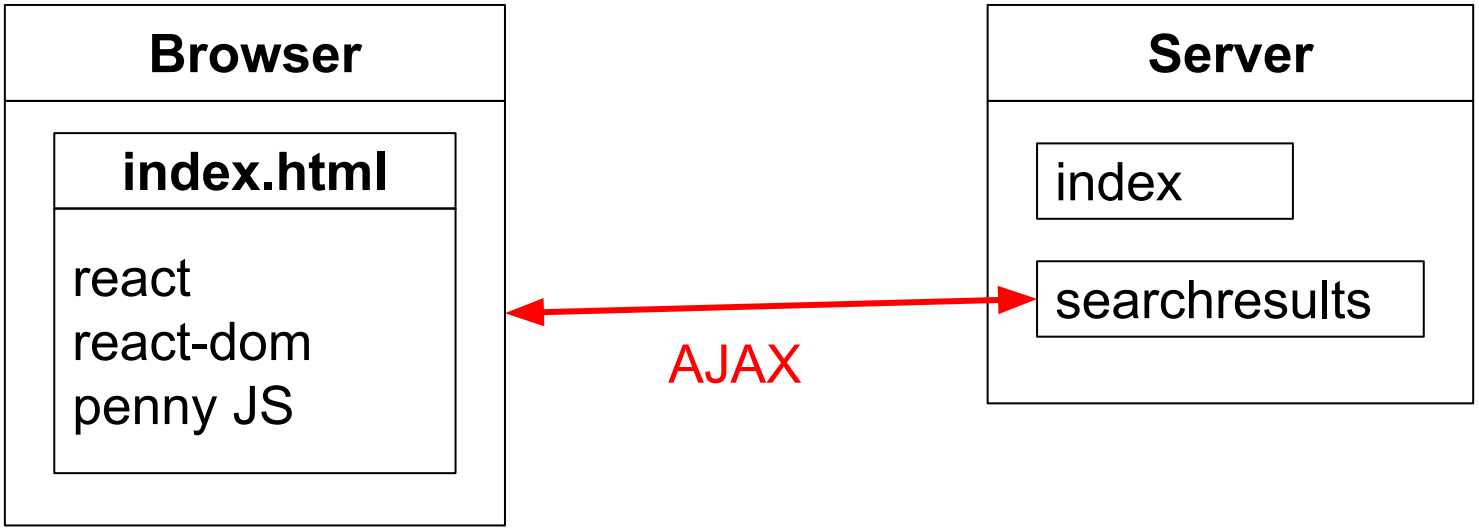
So far (cont.):



Browser uses babel to convert JSX code to JS code

Bundled React: Motivation

So far (cont.):



Browser requests and receives book info

Bundled React: Motivation

- **Problem**
  - At load-time:
    - Browser fetches index.html page, and then…
    - Browser fetches react
    - Browser fetches react-dom
    - Browser fetches babel
    - Browser uses babel to convert your JSX code to JavaScript code
    - Browser executes your JavaScript code

Blue => load-time overhead

Agenda

- Bundled React: motivation
- **Bundled React**
- Bundled React: Vite
- React commentary

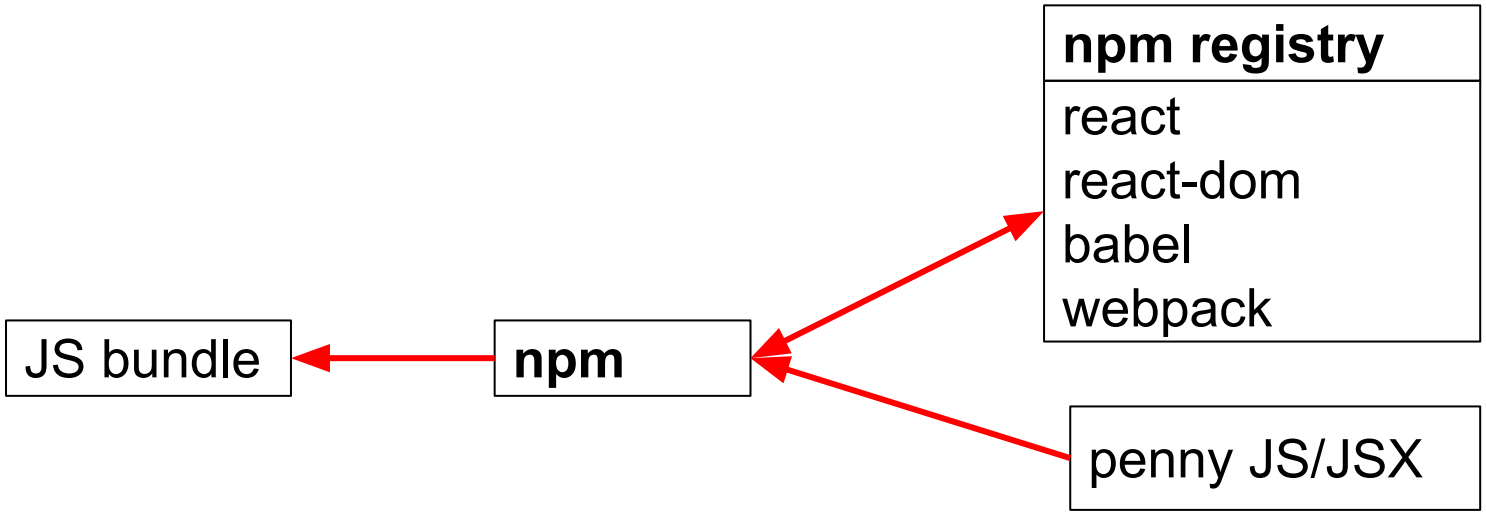
Bundled React

- Preliminary note:
  - **Don’t bundle your Assignment 4 solution!!!**

Bundled React

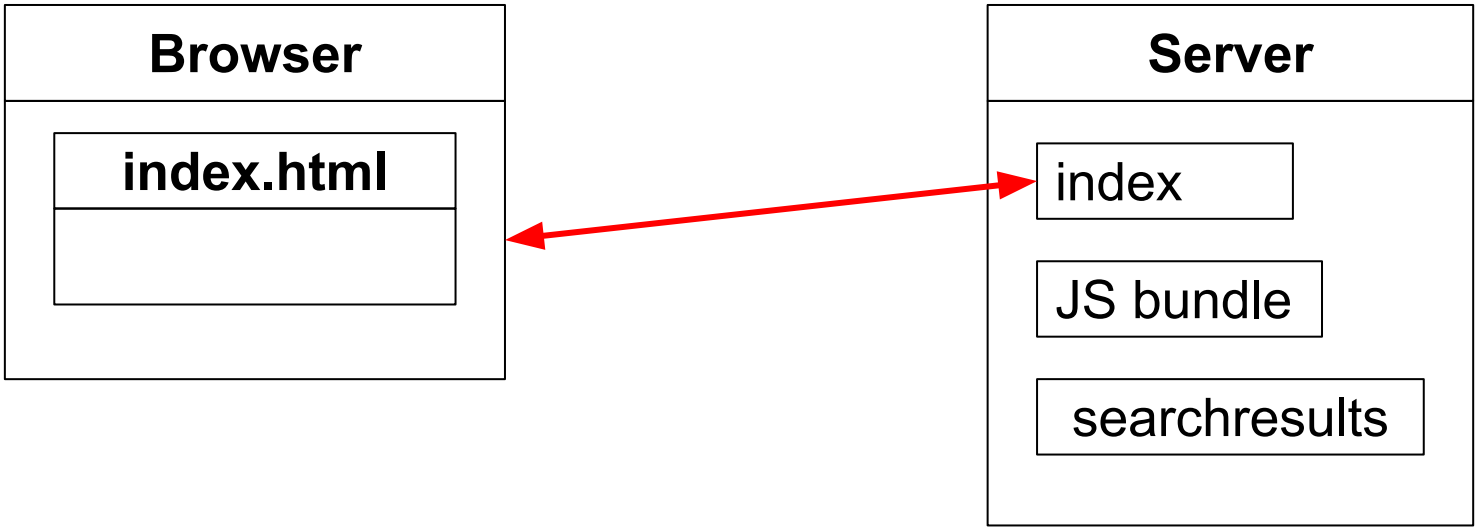
- **Solution**
  - Before load-time:
    - Use **babel** to convert your JSX code to JavaScript code
    - Use a bundling program (e.g., **webpack)** to place react, react-dom, and your JavaScript code in a JavaScript ***bundle***
  - At load-time:
    - Browser fetches your index.html page
    - Browser fetches your JavaScript bundle
    - Browser executes your JavaScript code

Bundled React



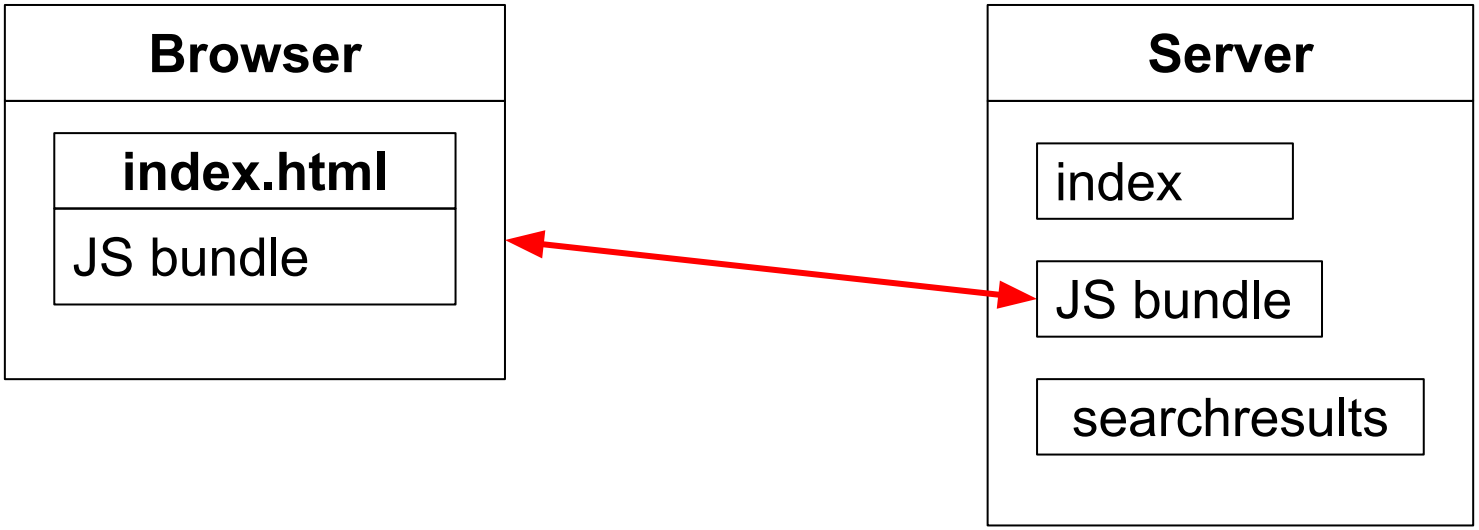
npm requests and receives react, react-dom, babel, webpack
npm uses webpack and babel to create a JS bundle containing react, react-dom, and penny JS

Bundled React



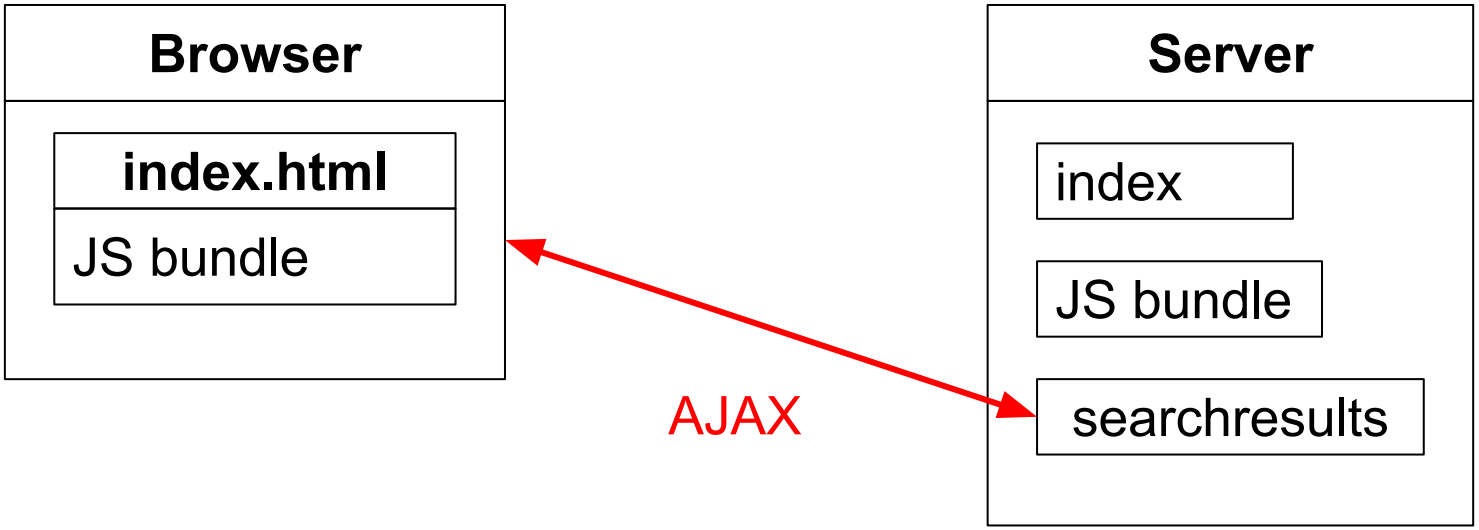
Browser requests and receives index page

Bundled React



Browser requests and receives JS bundle

Bundled React



Browser requests and receives book info

Bundled React

- Detailed instructions…

Bundled React

- Thanks, in part, to Lucas Manning (‘20)…
- See **PennyReactWebpack** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py (same)**
  - **PennyHeader.jsx, PennyFooter.jsx, PennySearch.jsx, App.jsx**
  - **main.js**
  - **index.html**

Bundled React

- See **PennyReactWebpack** app (cont.)
  - **package.json**
    - Configures npm
  - **webpack.config.js**
    - Configures webpack

Bundled React

- To give it a try:
  - Install node.js
  - Install dependencies
    - `npm install`
      - Examines `package.json`
      - (Recursively) installs dependencies into `node_modules` directory
      - Creates `package-lock.json` file
        - » Summary of contents of `node_modules` directory

Bundled React

- To give it a try (cont.):
  - Build the bundle
    - `npm run build`
      - Runs **webpack**
        - » Examines `webpack.config.js`
        - » Uses **babel** to convert JSX to JavaScript, and transpile JavaScript to ES5
        - » Packs all ES5 JavaScript code into one large bundle (static/app.bundle.js)

Bundled React

```
$ cd PennyReactWebpack
$ npm run build

> pennyreactwebpack@1.0.0 build
> webpack

asset app.bundle.js 15.6 KiB [compared for emit] [minimized] (name: main) 1
related asset
orphan modules 9.25 KiB [orphan] 4 modules
cacheable modules 34.3 KiB
  modules by path ./node_modules/react/ 16.6 KiB
    ./node_modules/react/index.js 186 bytes [built] [code generated]
    ./node_modules/react/cjs/react.production.js 16.5 KiB [built] [code
generated]
  modules by path ./node_modules/react-dom/ 7.83 KiB
    ./node_modules/react-dom/index.js 1.33 KiB [built] [code generated]
    ./node_modules/react-dom/cjs/react-dom.production.js 6.5 KiB [built]
[code generated]
  ./main.js + 4 modules 9.8 KiB [built] [code generated]
webpack 5.97.1 compiled successfully in 420 ms
$
```

Bundled React

- To give it a try (cont.):
  - Run the app
    - `python runserver.py 55555`

Bundled React

```
$ cd PennyReactWebpack
$ python runserver.py 55555
 * Serving Flask app 'penny'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a
production deployment. Use a production WSGI server
instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:55555
 * Running on http://192.168.1.10:55555
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 957-120-414
```

Bundled React

- To give it a try (cont.):
  - Browse to `http://localhost:55555`

Agenda

- Bundled React: motivation
- Bundled React
- **Bundled React: Vite**
- React commentary

Bundled React: Vite

- **Problem**
  - Bundling a large app can be slow
  - Using webpack (as shown) to repeatedly generate bundles can be slow during development of a large app

Bundled React: Vite

- **Solution 1**
  - Configure webpack to allow development without bundling
    - Bundles created at your command
  - Configure webpack to do ***hot module reloading***
    - Change JavaScript code => browser reloads it
- **Solution 2**
  - Use a high-level **React development environment**

Bundled React: Vite

- **High-level React development environments**
  - ***create-react-app***
    - Popular but deprecated
  - ***Next.js***
    - Popular but complicated
  - ***Vite***
    - Popular and (relatively) simple
  - Several others

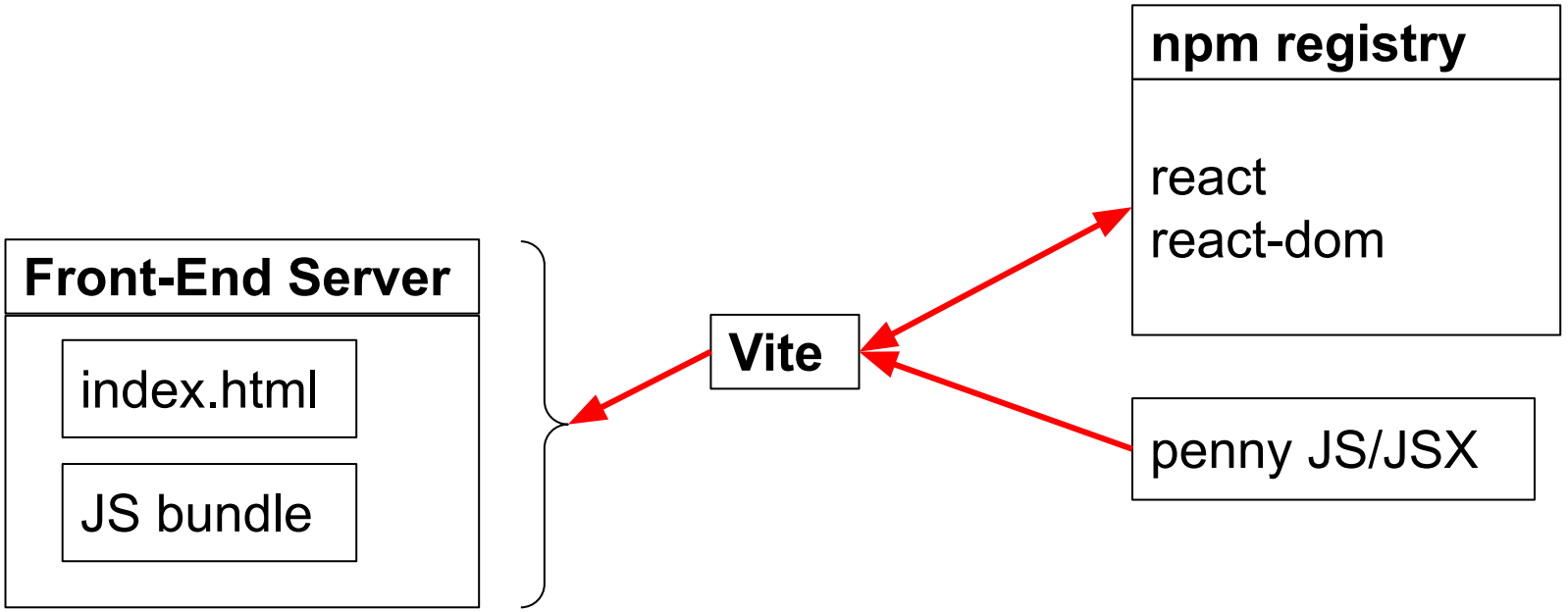
Bundled React: Vite

- ***Vite***
  - A popular React web development environment
  - Recognized for its speed

Bundled React: Vite

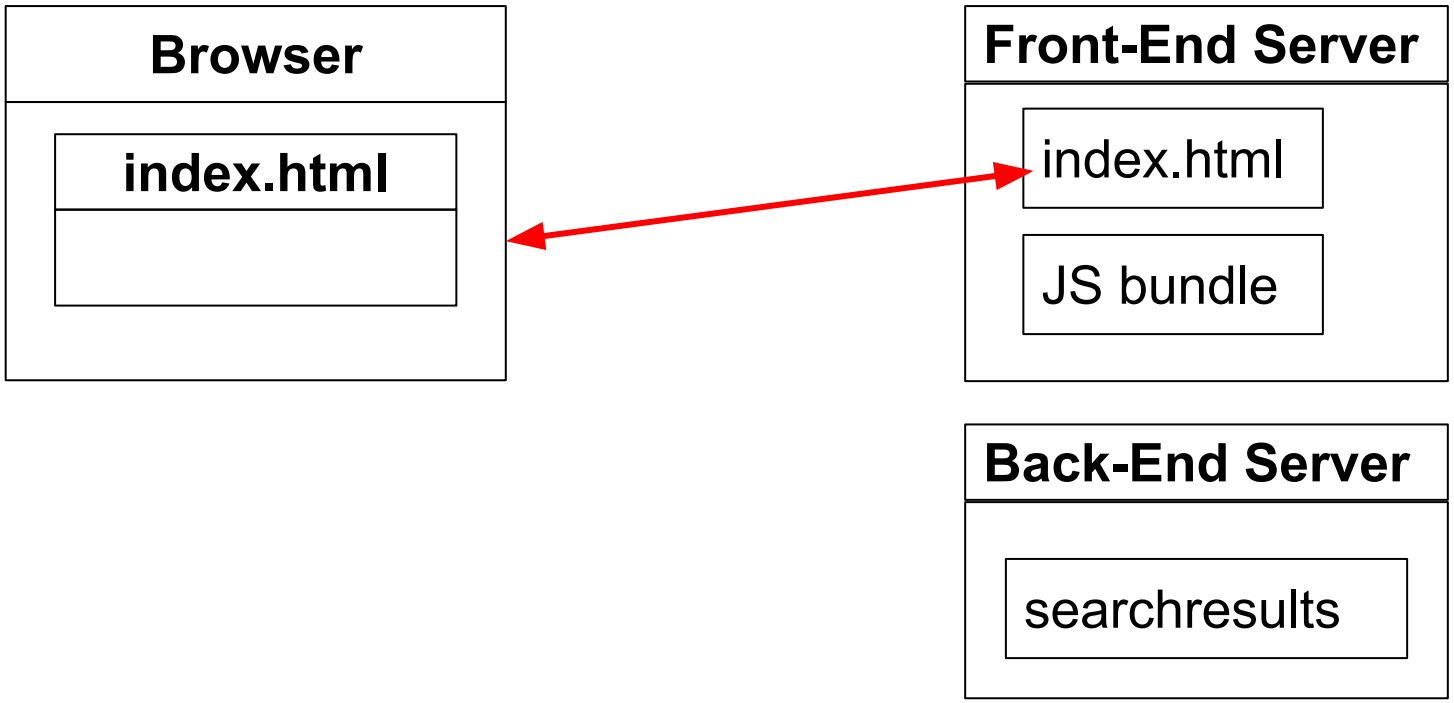
- General approach
  - Through Vite, create a ***front-end server***
    - Delivers index.html and JS bundle to browser
  - Independent of Vite, create a ***back-end server***
    - Written in Python/Flask/Jinja2 (or whatever!)
    - Provides services (API) to React app
    - Interacts with DB

Bundled React: Vite



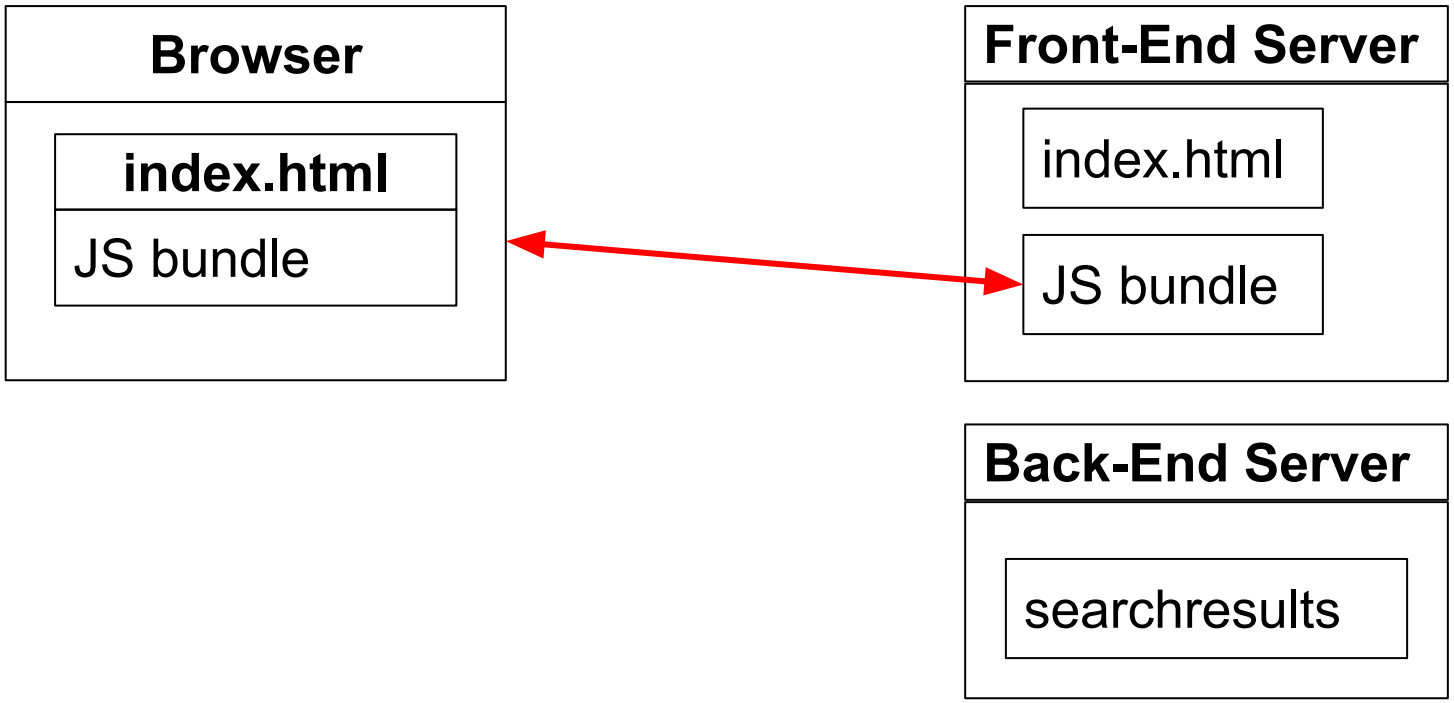
Vite requests and receives react, react-dom
Vite creates JS bundle containing those libraries and penny JS

Bundled React: Vite



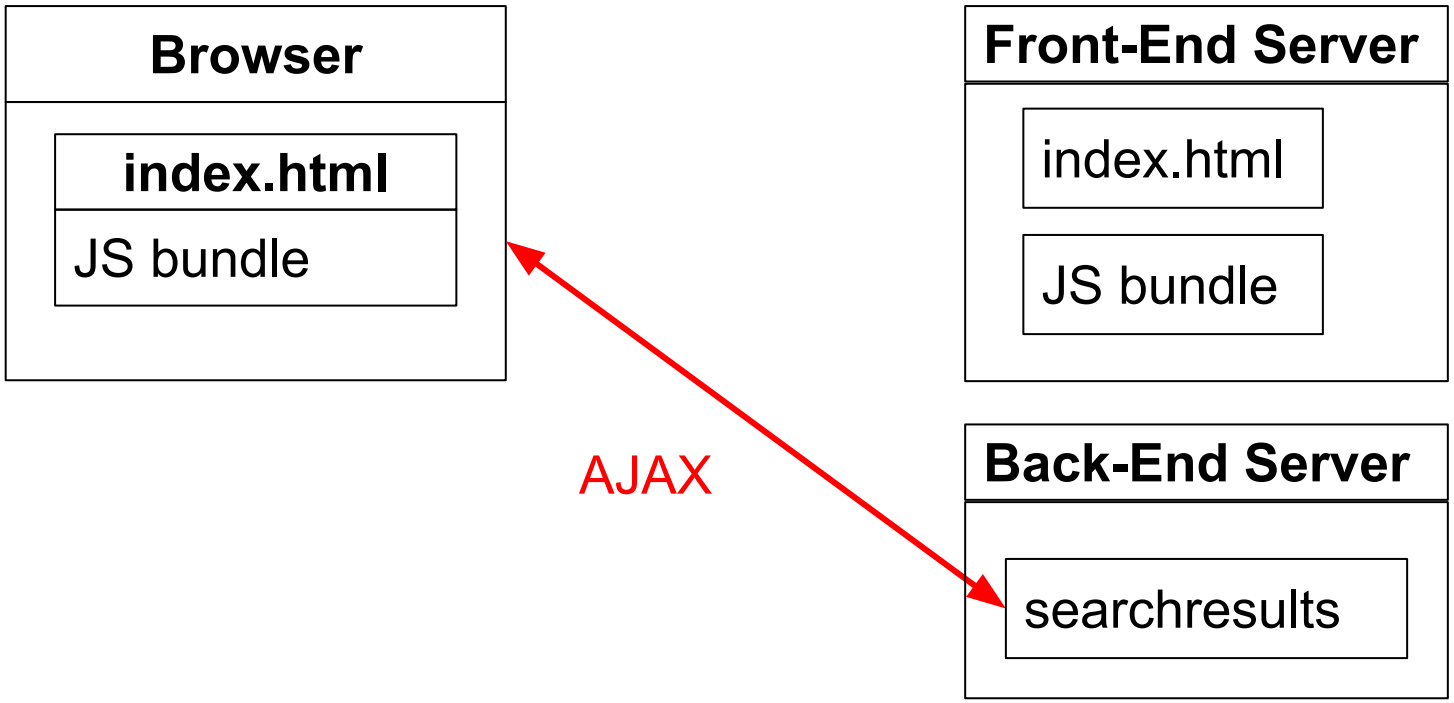
Run the front-end and back-end servers
Browser requests and receives index page

Bundled React: Vite



Browser requests and receives JS bundle

Bundled React: Vite



Browser requests and receives book info

Bundled React: Vite

- **Problem**: ***Cross-Origin Resource Sharing (CORS)***
  - Browser loads JS code from front-end server
  - JS code sends AJAX requests to back-end server
  - Back-end server notes that JS code is not from back-end server (has a different origin)
  - Back-end server refuses to respond

Bundled React: Vite

- **Solution**:
  - Override CORS
  - Command back-end server to allow AJAX requests from JavaScript code that the browser loaded from the front-end server

Bundled React: Vite

- Detailed instructions…

Bundled React: Vite

- **Step 1**: Create the back end
  - Assuming that you've created a proper Python virtual env and have installed Flask…

Bundled React: Vite

- **Step 1.1**: Create a PennyReactViteBackend directory anywhere in your file system

Bundled React: Vite

- **Step 1.2**: Place in the PennyReactViteBackend directory these files:
  - **runserver.py**
  - **penny.sql**
  - penny.sqlite
  - **database.py**
  - **penny.py**
  - **requirements.txt**

Bundled React: Vite

- **Step 2**: Create the front end
  - Assuming that you've installed node.js…

Bundled React: Vite

- **Step 2.1**: Create a PennyReactViteFrontend directory containing a default app anywhere in your file system

```
$ npm create vite@latest \
    PennyReactViteFrontend -- \
    --template react
```

Enter pennyreactvitefrontend as the Package name

Bundled React: Vite

- **Step 2.2**: Delete all files from the PennyReactViteFrontend/public directory

```
$ cd PennyReactViteFrontend/public
$ rm *
```

Bundled React: Vite

- **Step 2.3**: Delete all files from the PennyReactViteFrontend/src directory

```
$ cd PennyReactViteFrontend/src
$ rm -r *
```

Bundled React: Vite

- **Step 2.4**: In the PennyReactViteFrontend/src directory add these files:
  - **main.jsx**
  - **App.jsx**
  - **PennyHeader.jsx**,
  - **PennyFooter.jsx**
  - **PennySearch.jsx**

Bundled React: Vite

- **Step 2.5**: In the PennyReactViteFrontend directory add/overwrite these files:
  - **.env.development**
  - **.env.production**
  - **vite.config.js**

Bundled React: Vite

- **Step 2.6**: In the PennyReactViteFrontend directory edit index.html
  - Change this
    - `<title>Vite + react</title>`
  - to this:
    - `<title>Penny.com</title>`

Bundled React: Vite

- **Step 2.7**: Install dependencies
  - Installs dependencies into the node_modules directory

```
$ cd PennyReactViteFrontend
$ npm install
```

Bundled React: Vite

- **Step 3**: Run the app in development mode

Bundled React: Vite

- **Step 3.1**: Run PennyReactViteBackend
  - Starts back-end test server on localhost at port 5000

```
$ cd PennyReactViteBackend
$ export FRONTEND_URL=http://localhost:5173
$ python runserver.py
```

Bundled React: Vite

- **Step 3.2**: Run PennyReactViteFrontend
  - Starts front-end test server on localhost at port 5173

```
$ cd PennyReactViteFrontend
$ npm run dev
```

Bundled React: Vite

- **Step 3.3**: Browse to http://localhost:5173

Bundled React: Vite

- **Step 4**: Run the app in production mode

Bundled React: Vite

- **Step 4.1**: Run PennyReactViteBackend
  - Starts back-end test server on localhost at port 5000

```
$ cd PennyReactViteBackend
$ export FRONTEND_URL=http://localhost:4173
$ python runserver.py
```

Bundled React: Vite

- **Step 4.2**: Build PennyReactViteFrontend
  - Builds React bundle

```
cd PennyReactViteFrontend
npm run build
```

Bundled React: Vite

- **Step 4.3**: Run PennyReactVite
  - Starts front-end test server on localhost at port 4173

```
cd PennyReactViteFrontend
npm run preview
```

Bundled React: Vite

- **Step 4.4**: Browse to http://localhost:4173

Aside: Deployment

- Deploying to Render/Heroku
  - Deploy front-end server as a *static site*
  - Deploy back-end server as a *web service*

Aside: Adding Authentication

- Adding authentication (CAS or Google) to:
  - A **one-server** bundled React app
    - (Such as is created by webpack)
    - Not too difficult
  - A **two-server** bundled React app
    - (Such as is created by Vite)
    - Very difficult
    - Alternative: hack Vite so it generates a one-server app

More React

- There is **much** more to React…

- Recommended starter book:
  - *The Road to React* (Robin Weiruch)

Agenda

- Bundled React: motivation
- Bundled React
- Bundled React: Vite
- **React commentary**

React Commentary

- **jQuery**
  - HTML code contains JavaScript code
  - Modularity by **technologies**
- **React**
  - HTML code is generated by JavaScript code
  - Modularity by **components**

React Commentary

- Commentary:
  - Should you use React for:
    - The “hello” application?
    - The “echo” application?
    - The “datetime” application?
    - The Penny application?
    - The Assignment 4 application?
    - Your project application?

React Commentary

- Commentary:
  - Use React iff it's appropriate to do so!
    - Large web application
    - Web application with a component that's repeated many times
    - Web application that benefits from using existing React components

Summary

- We have covered:
  - Bundled React apps
  - Bundled React apps: Vite

Summary

- We have covered:
  - Client-side web programming using JavaScript
    - The browser DOM
    - AJAX
    - jQuery
    - React
- See also:
    - **Appendix 1**: Arrow Functions

Appendix 1:
Arrow Functions

Arrow Functions

- Recall from JavaScript lectures…
- **Question**: How is `this` bound within a function `f()`?
- **Answer**: Depends upon how `f()` is called:

Function Call	Binding of `this`
`f()`	In `f()`, `this` is `undefined`
`o.f()`	In `f()`, `this` is bound to `o`
`new f()`	In `f()`, `this` is bound to a new empty object

Arrow Functions

- Some terms for this lecture:
  - **Ordinary function**: a non-arrow function
  - **Ordinary variable**: a non-`this` variable

Arrow Functions

- ***Arrow function def expressions***
  - Informally ***arrow functions***
  - Arrow functions vs ordinary functions:
    - More succinct
    - Same semantics - **mostly!!!**

Aside: setInterval & setTimeout

In browsers:

```
window.setInterval(f, ms);
// Call f every ms milliseconds
```

We have seen

```
window.setTimeout(f, ms);
// Call f after ms milliseconds
```

We have seen

In Node.js:

```
setInterval(f, ms);
// Call f every ms milliseconds
```

```
setTimeout(f, ms);
// Call f after ms milliseconds
```

We'll use now

Arrow Functions

- **Fact 1**: In an ordinary function…
  - The value of `this` is determined **dynamically**
    - Based upon the call
    - `o.f()`
      - In the function `this` is bound to `o`
    - `f()`
      - In the function `this` is undefined

Arrow Functions

```
$ node arrow1.js
undefined
$
```

- See **arrow1.js**
  - Notes:
    - Global code calls `main()`
    - `main()` calls `blueCar.writeColor()`
    - `blueCar.writeColor()` calls `setTimeout()`
    - `setTimeout()` calls given ordinary function
      - As `f()`, not as `o.f()`
    - In ordinary function, `this` is undefined

Arrow Functions

- **Fact 2**: In an ordinary function…
  - The value of an ordinary variable is determined **statically**
    - Based upon program block structure

Arrow Functions

```
$ node arrow2.js
blue
$
```

- See **arrow2.js**
  - Notes:
    - Global code calls `main()`
    - `main()` calls `blueCar.writeColor()`
    - `blueCar.writeColor()` calls `setTimeout()`
    - `setTimeout()` calls given ordinary function
      - As `f()`, not as `o.f()`
    - In ordinary function, `this` is undefined
      - But the ordinary function doesn't use `this`!

Arrow Functions

- **Fact 3**: In an arrow function…
  - The value of `this` (and any ordinary variable) is determined **statically**
    - Based upon program block structure

Arrow Functions

```
$ node arrow3.js
blue
$
```

- See **arrow3.js**
  - Notes:
    - Global code calls `main()`
    - `main()` calls `blueCar.writeColor()`
    - `blueCar.writeColor()` calls `setTimeout()`
    - `setTimeout()` calls given arrow function
      - As `f()`, not as `o.f()`
    - In arrow function, `this` is bound to `blueCar`

Arrow Functions

- **Question**: Why use arrow functions?
- **Answer 1**: They're often more succinct
- **Answer 2**: `this` is defined statically

- Arrow functions often are appropriate as callback functions