Client-Side Web Programming: JavaScript (Part 5)

Robert M. Dondero, Ph.D.

Princeton University

Objectives

- We will cover:
  - A realistic React app
  - Bundled React via webpack
  - Bundled React via Vite
  - React commentary

Agenda

- **React: realistic example**
- Bundled React: webpack
- Bundled React: Vite
- React commentary

Aside: JavaScript map Function

Iteration over an array using the `for` statement:

```
for (let word of words) {
   process.stdout.write(word);
}
```

Iteration over an array using the `map()` function:

```
words.map(
   (word) => {process.stdout.write(word)}
);
```

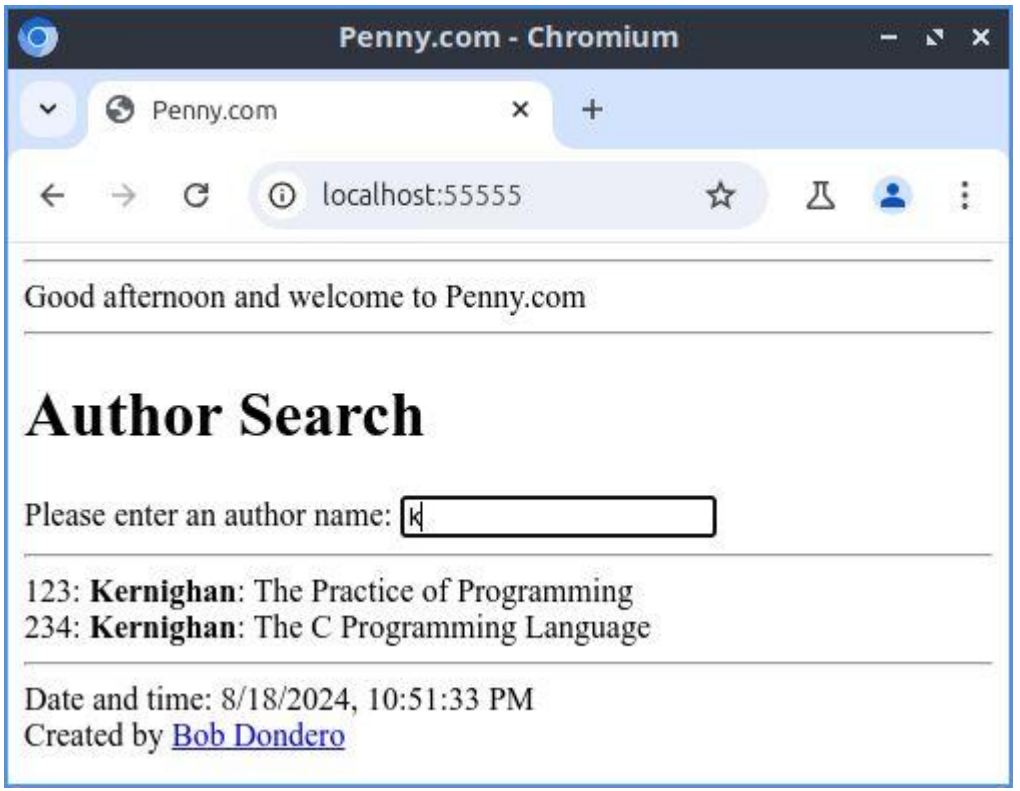
React: Realistic Example

- See **PennyReact** app

Thanks, in part, to Liam Esparraguara ('24)

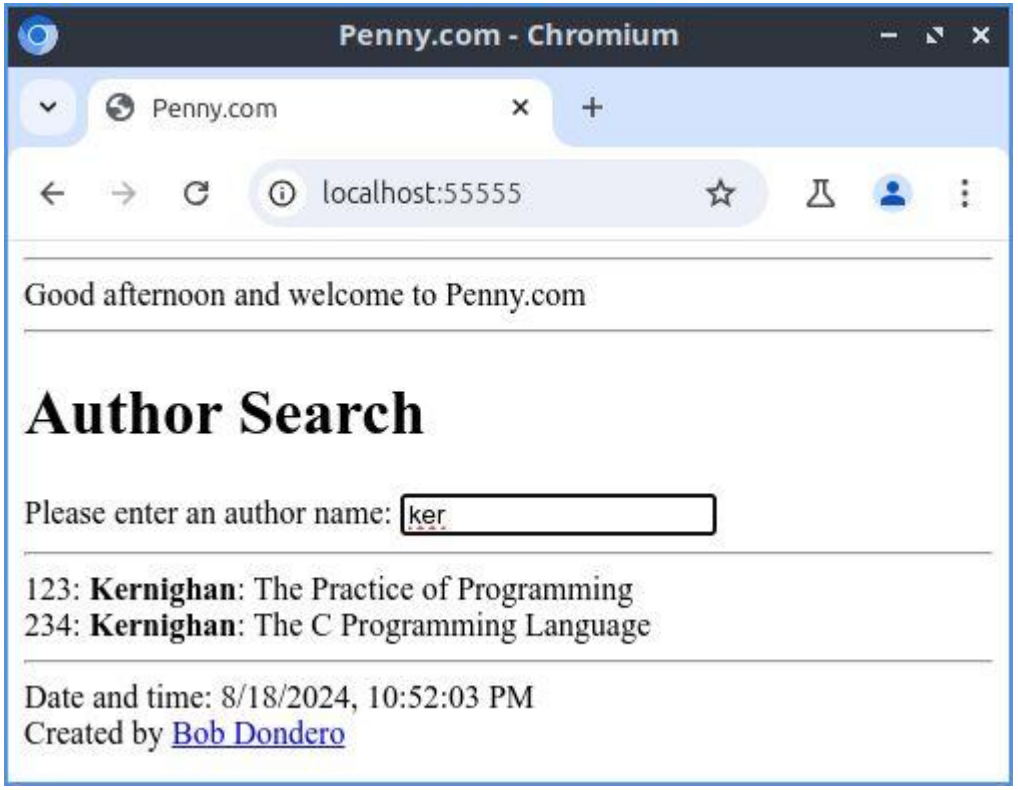
React: Realistic Example

- See **PennyReact** app (cont.)



React: Realistic Example

- See **PennyReact** app (cont.)



React: Realistic Example

- See **PennyReact** app (cont.)

React: Realistic Example

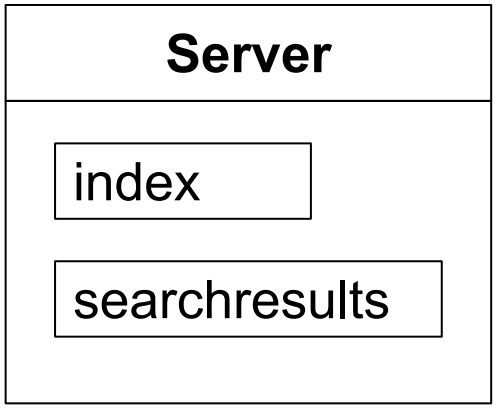
- See **PennyReact** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - **penny.py**
  - **index.html**

Agenda

- React: realistic example
- **Bundled React: webpack**
- Bundled React: Vite
- React commentary

Bundled React: webpack

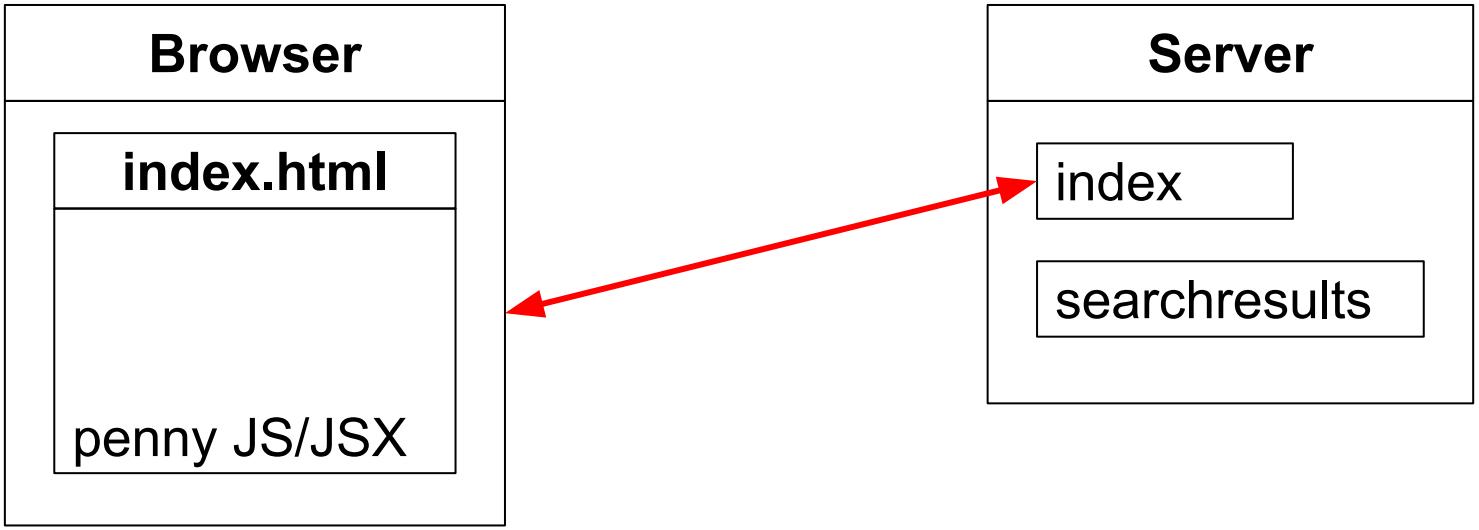
So far:



Run the server.

Bundled React: webpack

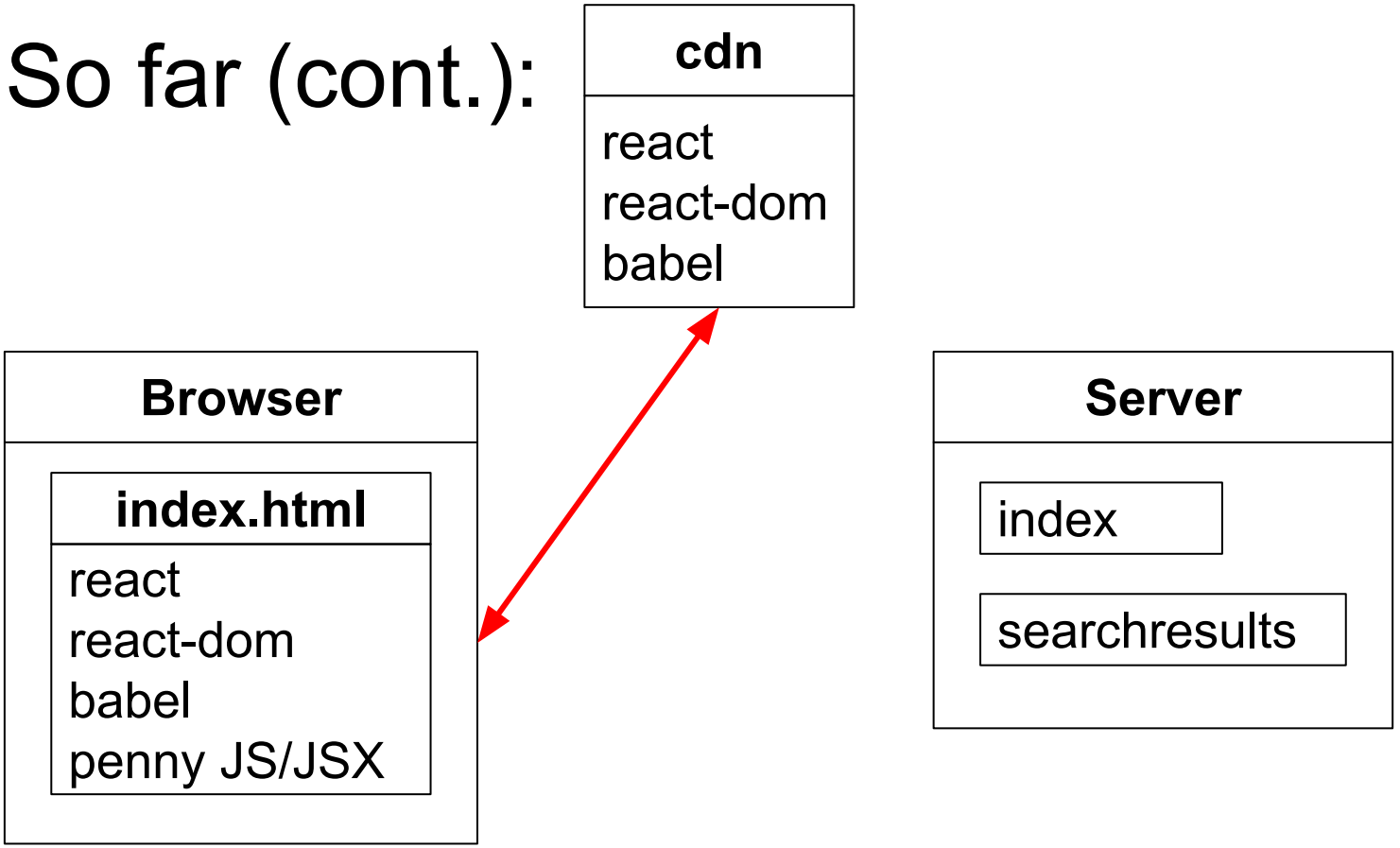
So far:



Browser requests and receives index.html.

Bundled React: webpack

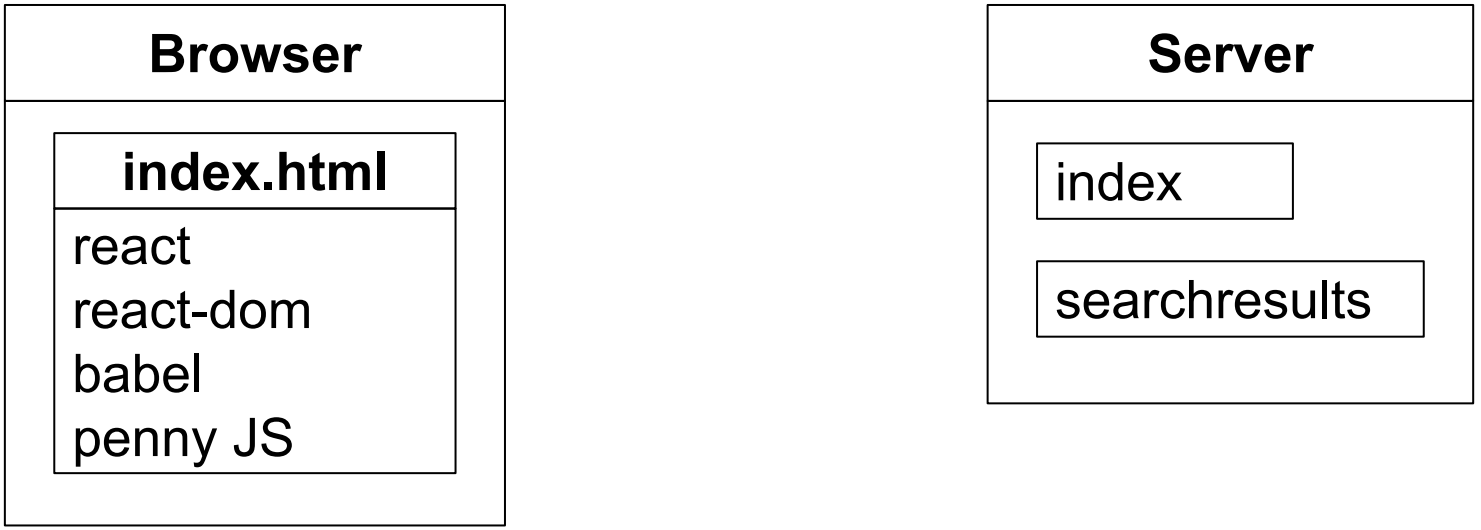
So far (cont.):



Browser requests and receives react, react-dom, and babel from cdn website.

Bundled React: webpack

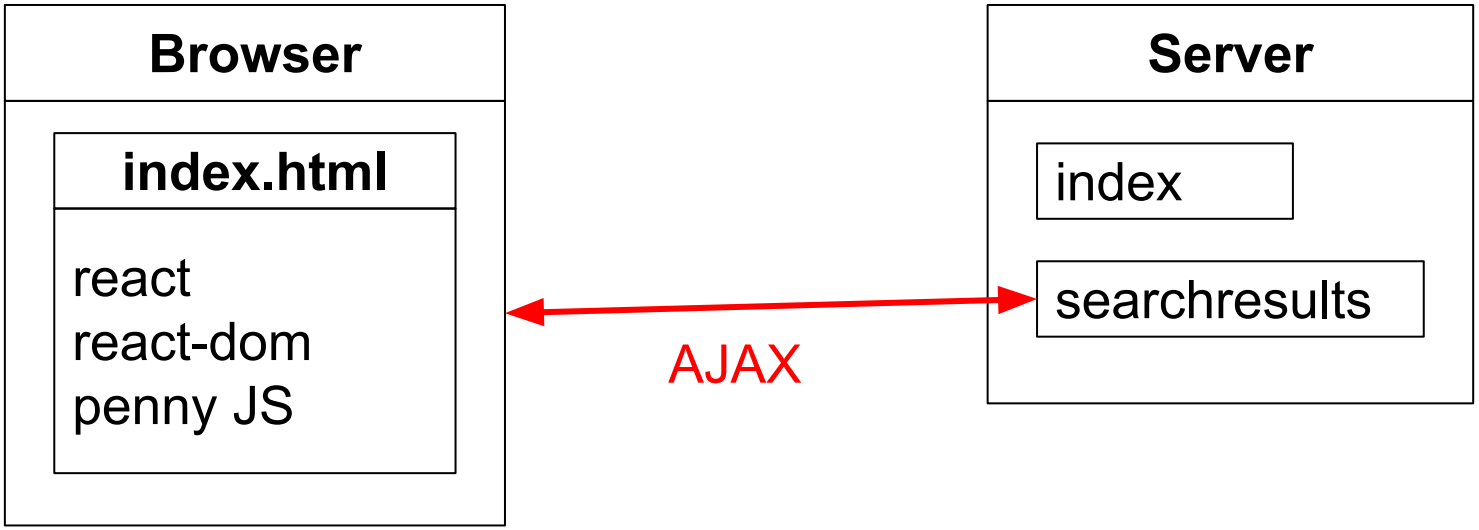
So far (cont.):



Browser uses babel to convert JSX code to JS code.

Bundled React: webpack

So far (cont.):



Browser repeatedly requests and receives book info.

Bundled React: webpack

- **Problem**
  - At run-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

Run-time overhead

Bundled React: webpack

- **Solution**
  - Before run-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***

Bundled React: webpack

- Bonus…
- webpack does ***tree shaking***
  - Places in the bundle only the react and react-dom code that is used by your JavaScript code

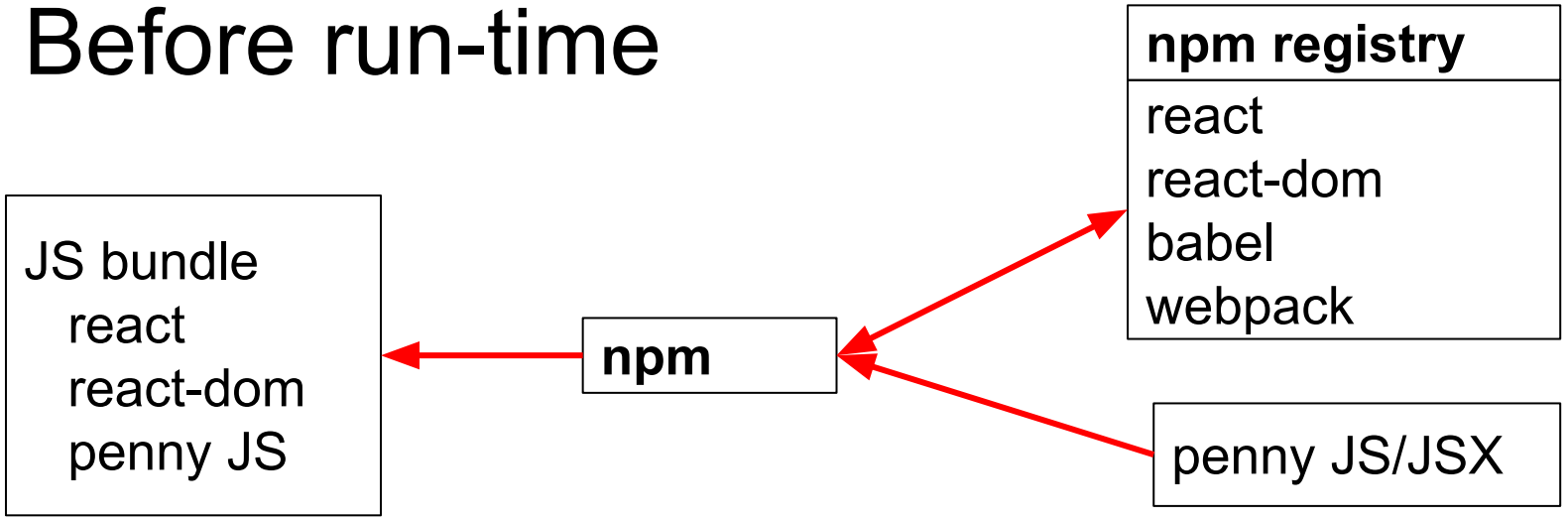
Thanks to Luke Sanborn ‘28 for the term

Bundled React: webpack

- **Solution** (cont.)
  - At run-time:
    - Browser fetches your index.html page
    - Browser fetches your JavaScript bundle
    - Browser executes your JavaScript code

Bundled React: webpack

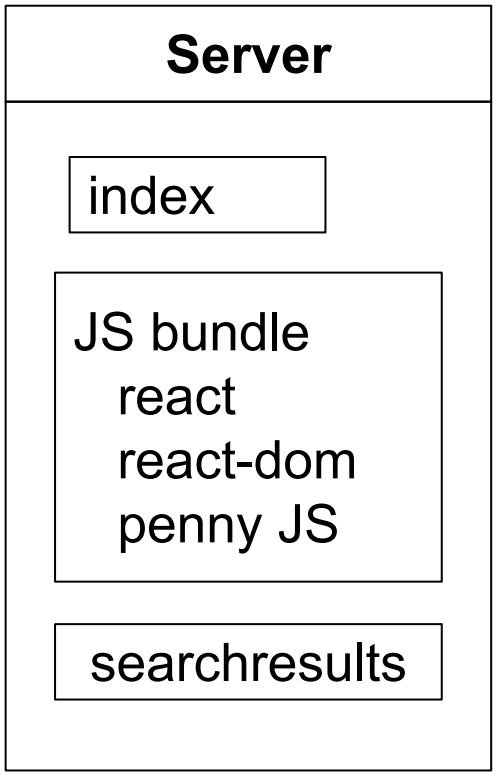
Before run-time



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

Bundled React: webpack

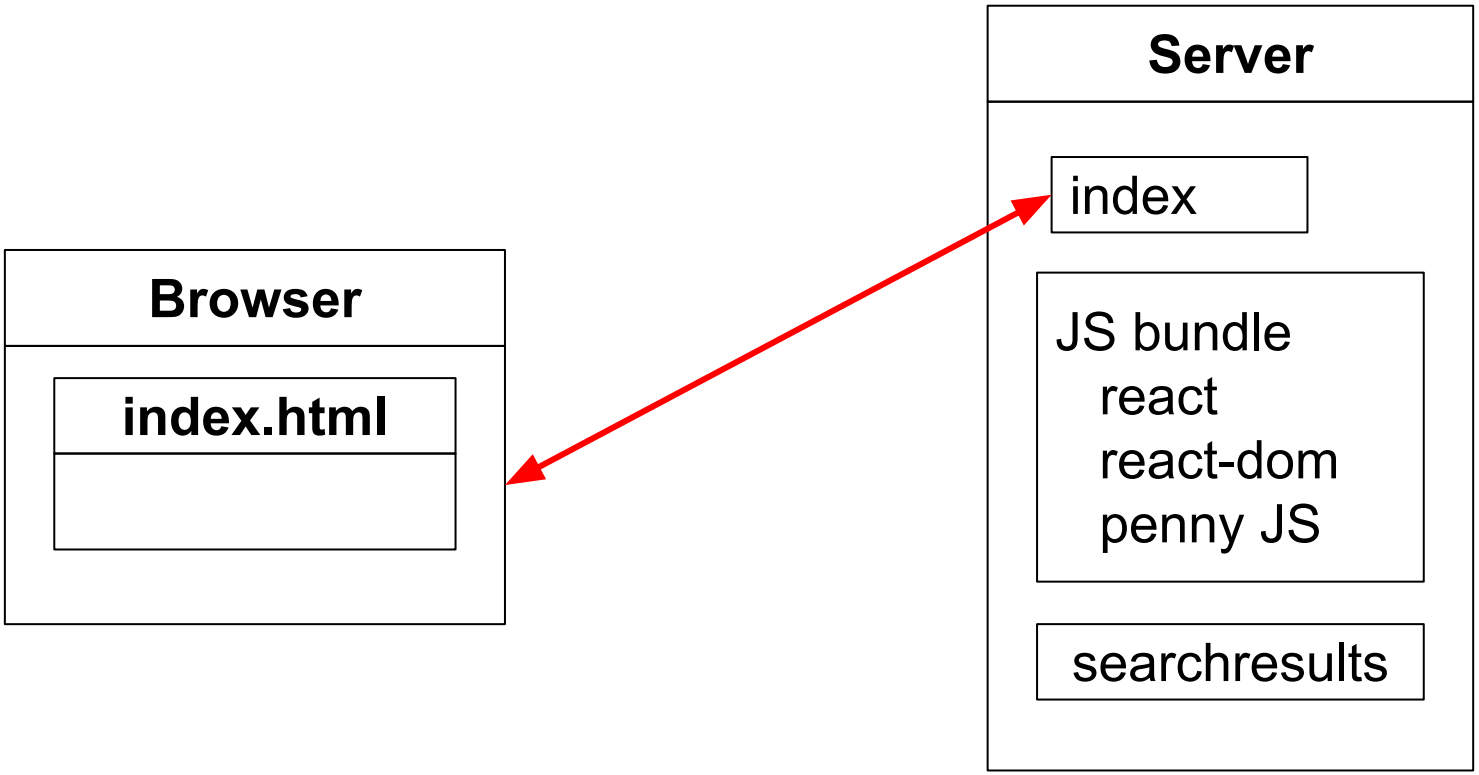
At run-time



Run the server.

Bundled React: webpack

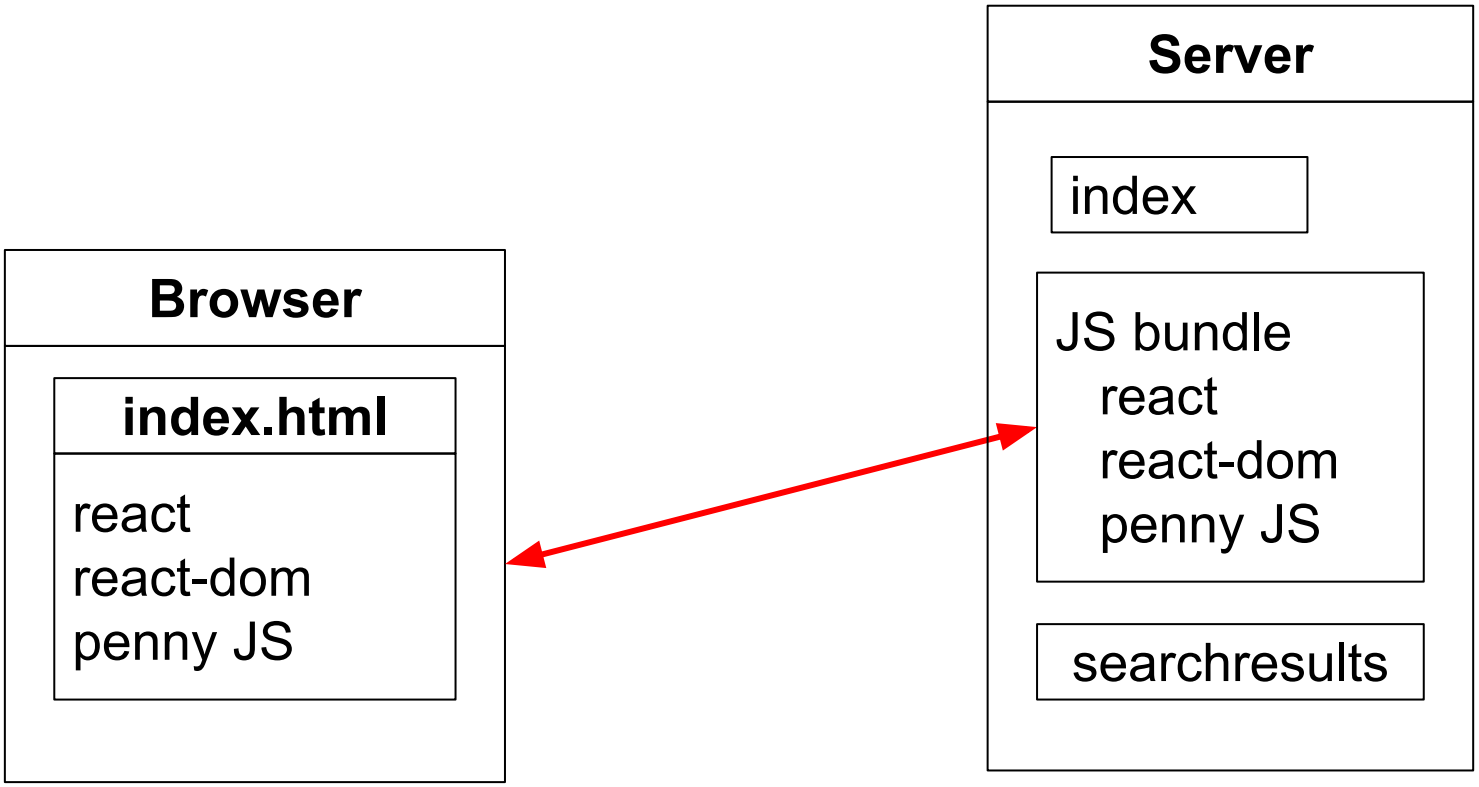
At run-time



Browser requests and receives index page.

Bundled React: webpack

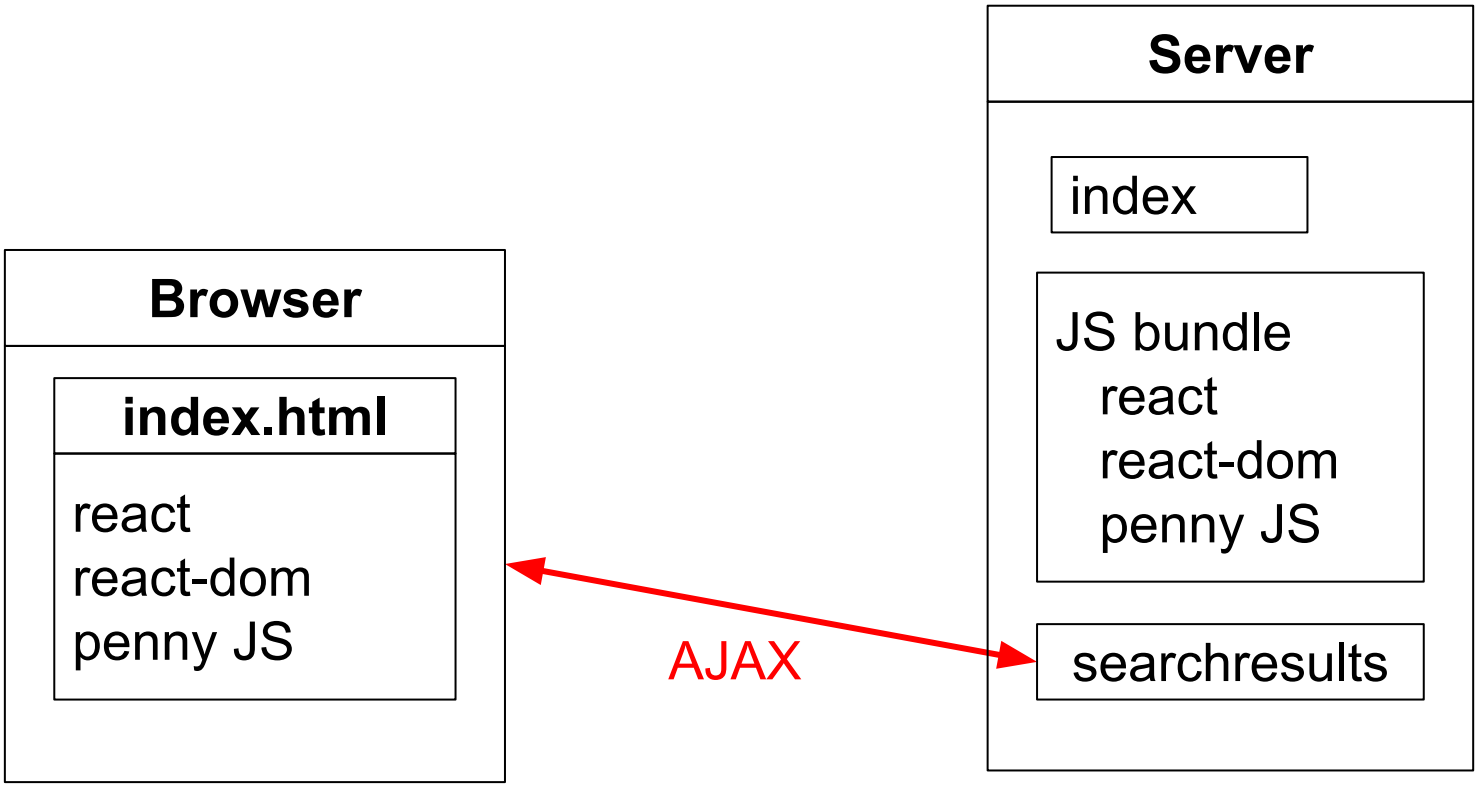
At run-time



Browser requests and receives JS bundle.

Bundled React: webpack

At run-time



Browser repeatedly requests and receives book info.

Bundled React: webpack

- Detailed instructions…

Bundled React: webpack

- See **PennyReactWebpack** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlite
  - database.py
  - penny.py
  - **pennyheader.jsx, pennyfooter.jsx, pennysearch.jsx, app.jsx**
  - **index.jsx, index.html**

Thanks, in part, to Lucas Manning ‘20

Bundled React: webpack

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

Bundled React: webpack

- 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: webpack

- To give it a try (cont.):
  - Build the bundle
    - **`npm run builddev`**
      - Runs **webpack**
        - » Examines `webpack.config.js`
        - » Uses **babel** to convert JSX to JavaScript, and transpile JavaScript to ES5
          - **Does not compress the JavaScript code**
        - » Packs all ES5 JavaScript code into one large bundle (static/index.bundle.js)

Bundled React: webpack

```
$ cd PennyReactWebpack
$ npm run builddev

> pennyreactwebpack@1.0.0 builddev
> webpack --mode=development

asset index.bundle.js 1.1 MiB [emitted] (name: index) 1 related asset
runtime modules 1.04 KiB 5 modules
cacheable modules 1.09 MiB
  modules by path ./node_modules/ 1.08 MiB
    modules by path ./node_modules/react/ 85.7 KiB 2 modules
    modules by path ./node_modules/react-dom/ 1010 KiB 2 modules
    modules by path ./node_modules/scheduler/ 17.3 KiB 2 modules
  modules by path ./*.jsx 9.8 KiB
    ./index.jsx 561 bytes [built] [code generated]
    ./app.jsx 746 bytes [built] [code generated]
    ./pennyheader.jsx 2.54 KiB [built] [code generated]
    ./pennysearch.jsx 3.4 KiB [built] [code generated]
    ./pennyfooter.jsx 2.58 KiB [built] [code generated]
webpack 5.97.1 compiled successfully in 1912 ms
```

Bundled React: webpack

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

Bundled React: webpack

```
$ cd PennyReactWebpack
$ npm run buildprod

> pennyreactwebpack@1.0.0 buildprod
> webpack --mode=production

asset index.bundle.js 141 KiB [emitted] [minimized] (name: index) 1 related
asset
orphan modules 9.26 KiB [orphan] 4 modules
modules by path ./node_modules/ 141 KiB
  modules by path ./node_modules/react/ 6.95 KiB
    ./node_modules/react/index.js 190 bytes [built] [code generated]
    ./node_modules/react/cjs/react.production.min.js 6.77 KiB [built] [code
generated]
  modules by path ./node_modules/react-dom/ 130 KiB
    ./node_modules/react-dom/index.js 1.33 KiB [built] [code generated]
    ./node_modules/react-dom/cjs/react-dom.production.min.js 129 KiB
[built] [code generated]
  modules by path ./node_modules/scheduler/ 4.33 KiB
    ./node_modules/scheduler/index.js 198 bytes [built] [code generated]
    ./node_modules/scheduler/cjs/scheduler.production.min.js 4.14 KiB
[built] [code generated]
./index.jsx + 4 modules 9.8 KiB [built] [code generated]
webpack 5.97.1 compiled successfully in 5013 ms
```

Bundled React: webpack

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

Bundled React: webpack

```
$ 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: webpack

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

Agenda

- React: realistic example
- Bundled React: webpack
- **Bundled React: Vite**
- React commentary

Bundled React: Vite

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

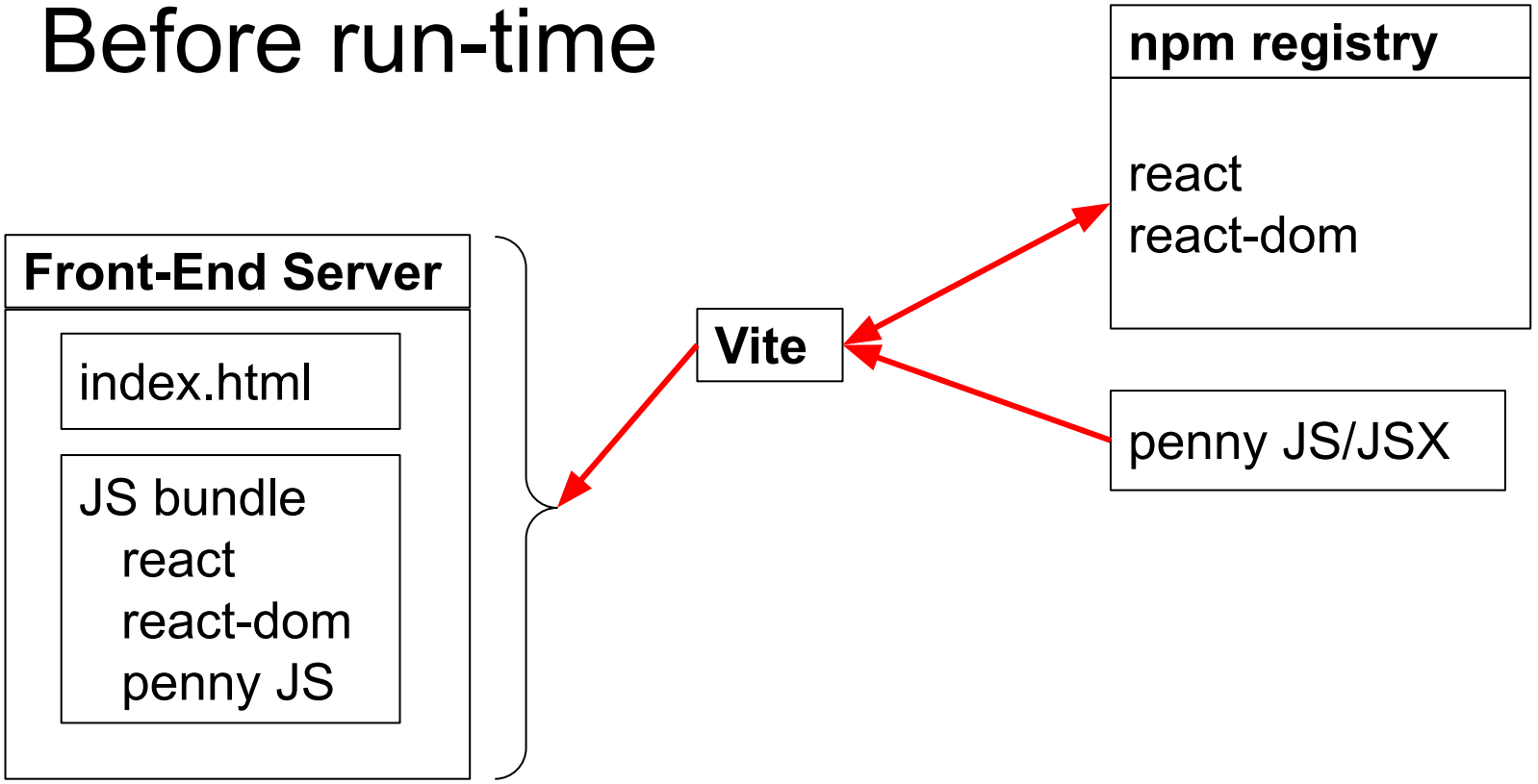
- ***Vite***
  - A popular React web development environment
  - Recognized for its speed

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 (or whatever!)
    - Provides services (API) to React app
    - Interacts with DB

Bundled React: Vite

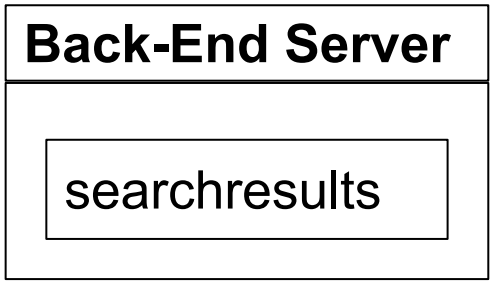
Before run-time



Vite requests and receives react, react-dom from npm registry.
Vite creates JS bundle containing those libraries and penny JS.
Vite creates a front-end server.

Bundled React: Vite

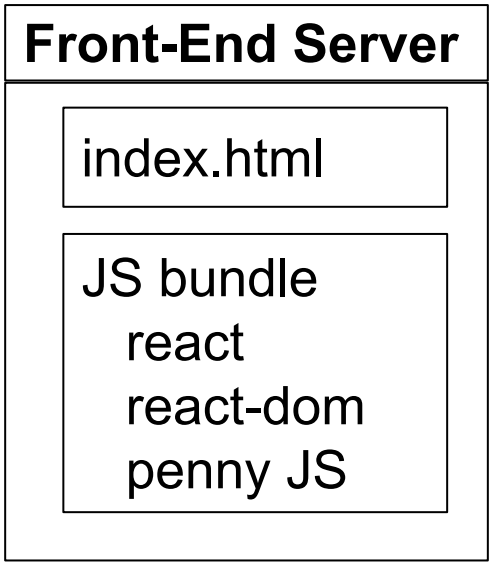
Before run-time



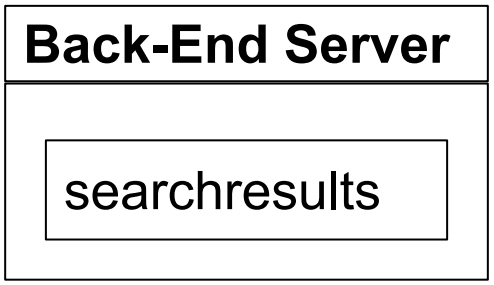
Independently, you create a back-end server (using Python/Flask or whatever).

Bundled React: Vite

At run-time

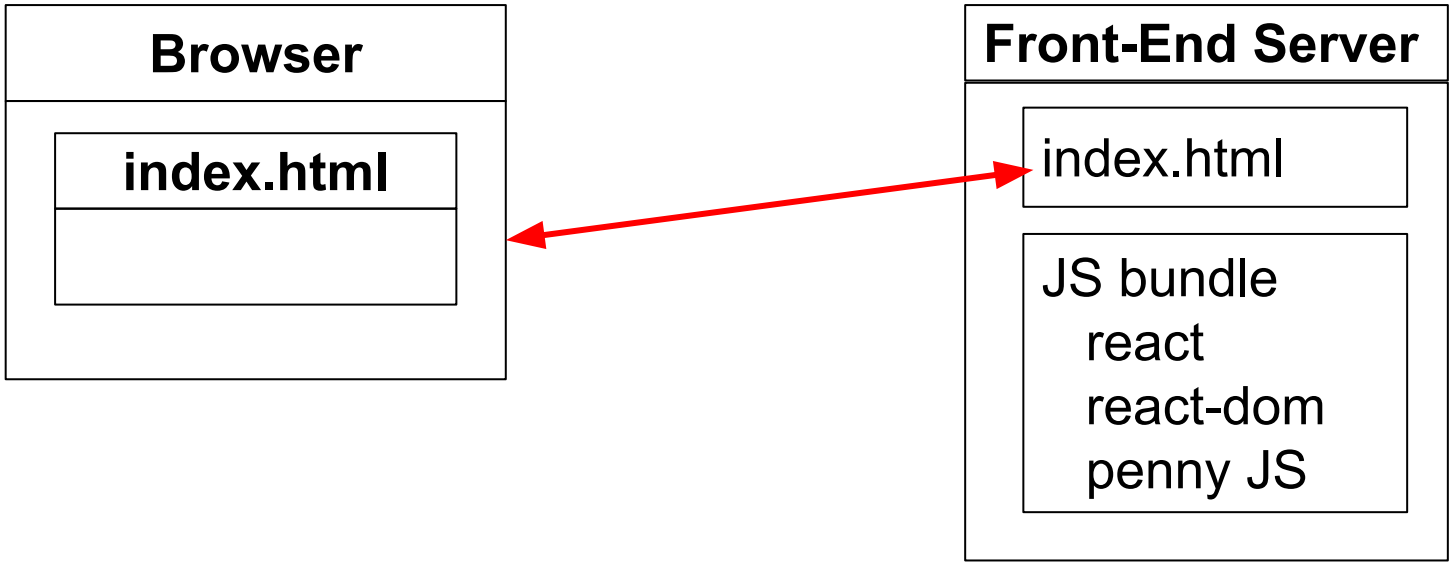


Run the back-end server.
Run the front-end server.

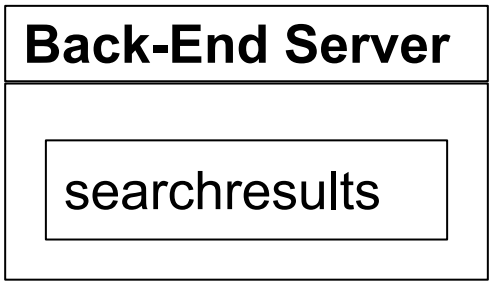


Bundled React: Vite

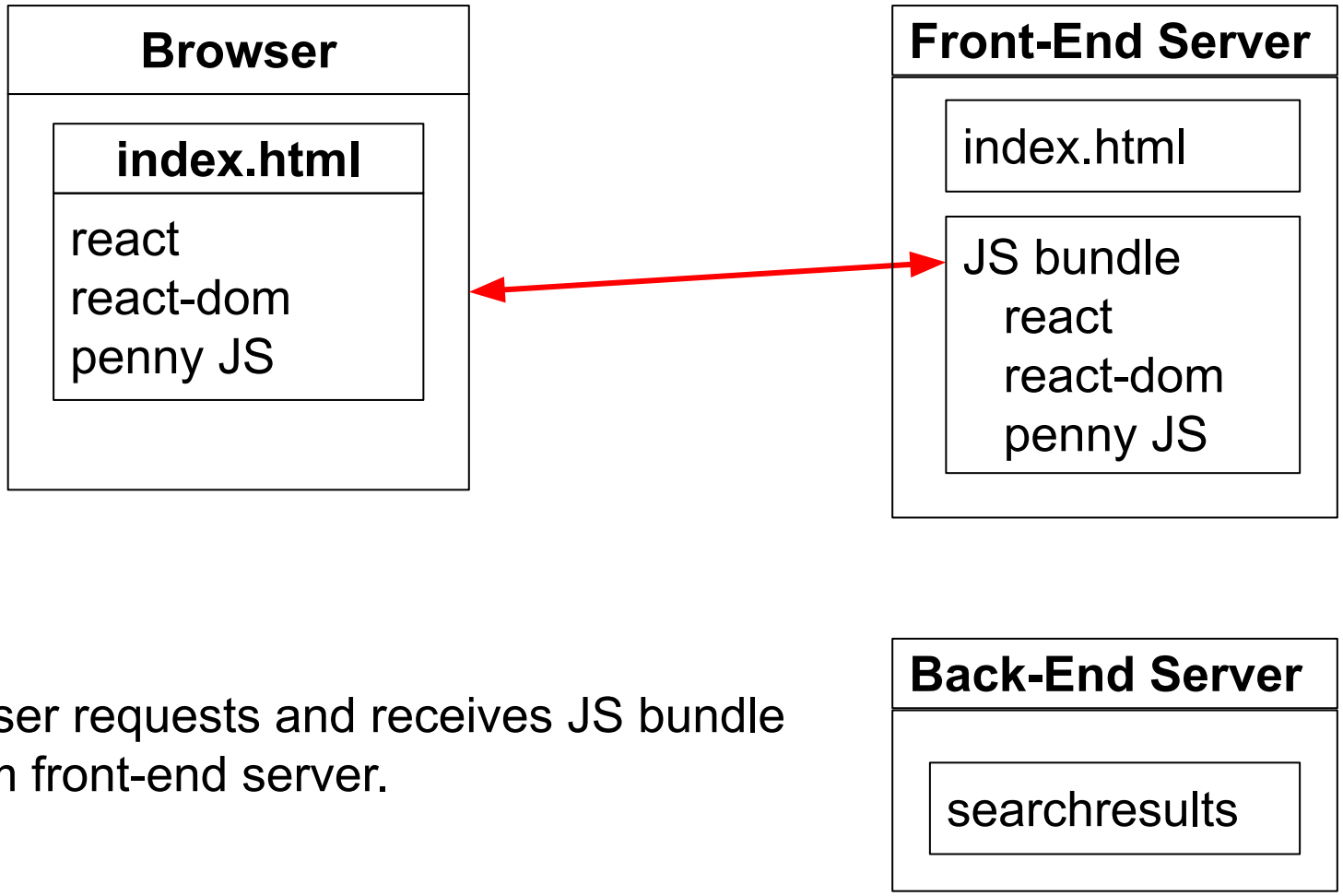
At run-time



Browser requests and receives index page from front-end server.

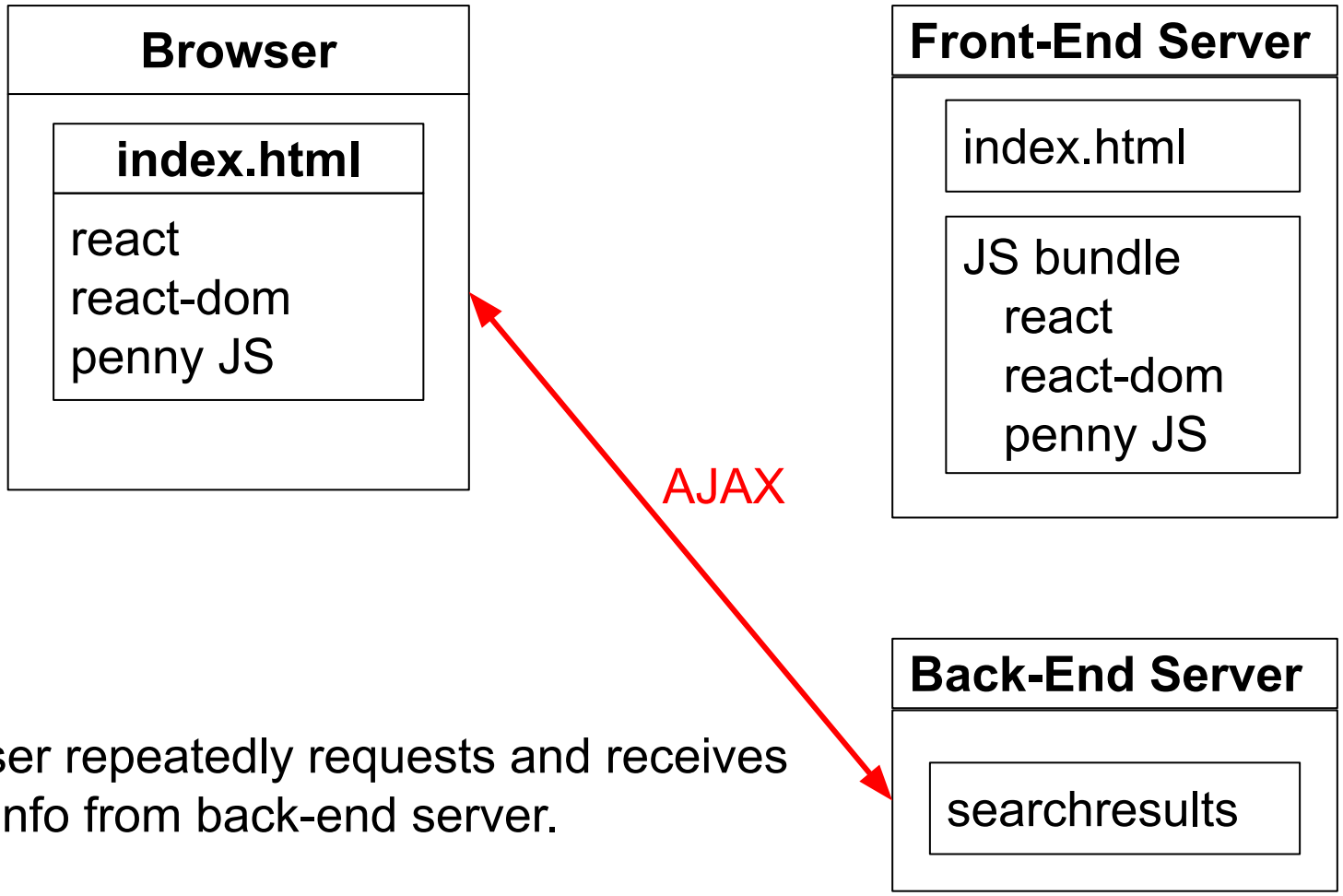


Bundled React: Vite



Browser requests and receives JS bundle from front-end server.

Bundled React: Vite



Browser repeatedly requests and receives book info from back-end server.

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
    - Was not delivered to the browser by the back-end server
    - Has a different origin
  - Back-end server refuses to respond

Bundled React: Vite

- **Solution**: Selectively override CORS
  - Command back-end server to allow AJAX requests from JavaScript code that was delivered to the browser by the front-end server
  - In penny.py:

```
…
import flask_cors
…
flask_cors.CORS(app,
    resources={r'/api/*': {'origins': FRONTEND_URL}})
…
```

Bundled React: Vite

- Detailed instructions…
  - See **Appendix 1**

Aside: Adding Authentication

- Adding authentication (CAS or EntraID or Google) to:
  - A **one-server** bundled React app
    - (Such as is created by webpack)
    - Straightforward
  - A **two-server** bundled React app
    - (Such as is created by Vite)
    - Difficult
    - Alternative: hack Vite so it generates a one-server app

Aside: React in COS 333

- **Assignment 4**
  - If you use React, then:
    - You're *required* to use **unbundled** React

Aside: React in COS 333

- **Project**
  - If you use React, then:
    - I *recommend* that you **bundle**
    - I *recommend* that you create a **one-server app** (via webpack) rather than a **two-server app** (via Vite)

More React

- There is **much** more to React…

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

Agenda

- React: realistic example
- Bundled React: webpack
- 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

- Use React when it's appropriate to do so!
  - When React **fits the application**
    - For a **large web application**
    - For a web application with a **component that's repeated many times**
    - For a web application that benefits from using **existing React components**
  - When React **fits the development team**

React Commentary

- When developing a new app:
  - Don't use the library that **you like best**
  - Don't use the library that is **the hottest**
  - Instead, choose the library in a **principled way**:
    - (1) The library that best fits the application
    - (2) The library that best fits the development team

Lecture Summary

- In this lecture we covered:
  - A realistic React app
  - Bundled React via webpack
  - Bundled React via Vite

Lecture Series Summary

- In this lecture series we covered:
  - Client-side web programming using JavaScript
    - The browser DOM
    - AJAX
    - jQuery
    - React
- See also:
    - **Appendix 1**: Vite Detailed Instructions

More Information

- The COS 333 *Lectures* web page provides references to supplementary information

Appendix 1:
Vite Detailed Instructions

Vite Detailed Instructions

- Instructions to use React and Vite to create a **PennyVite** application consisting of front-end and back-end servers…

Vite Detailed Instructions

- **Step 1**: Create and run the back-end server

Vite Detailed Instructions

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

Vite Detailed Instructions

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

Vite Detailed Instructions

- **Step 1.3**: Create and activate a proper Python virtual environment

```
$ cd PennyReactViteBackend
$ python -m venv myvenv
$ source myvenv/bin/activate
$ python -m pip install -r requirements.txt
```

Vite Detailed Instructions

- **Step 1.4**: Run PennyReactViteBackend
  - Start the back-end server on localhost at port 5000

```
$ cd PennyReactViteBackend
$ python runserver.py
```

Vite Detailed Instructions

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

Vite Detailed Instructions

- **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

Vite Detailed Instructions

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

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

Vite Detailed Instructions

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

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

Vite Detailed Instructions

- **Step 2.4**: In the PennyReactViteFrontend/src directory add these files:
  - **main.jsx**
  - **app.jsx**
  - **pennyheader.jsx**,
  - **pennyfooter.jsx**
  - **pennysearch.jsx**

Vite Detailed Instructions

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

Vite Detailed Instructions

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

Vite Detailed Instructions

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

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

Vite Detailed Instructions

- **Step 3**: Run the front-end server in development mode

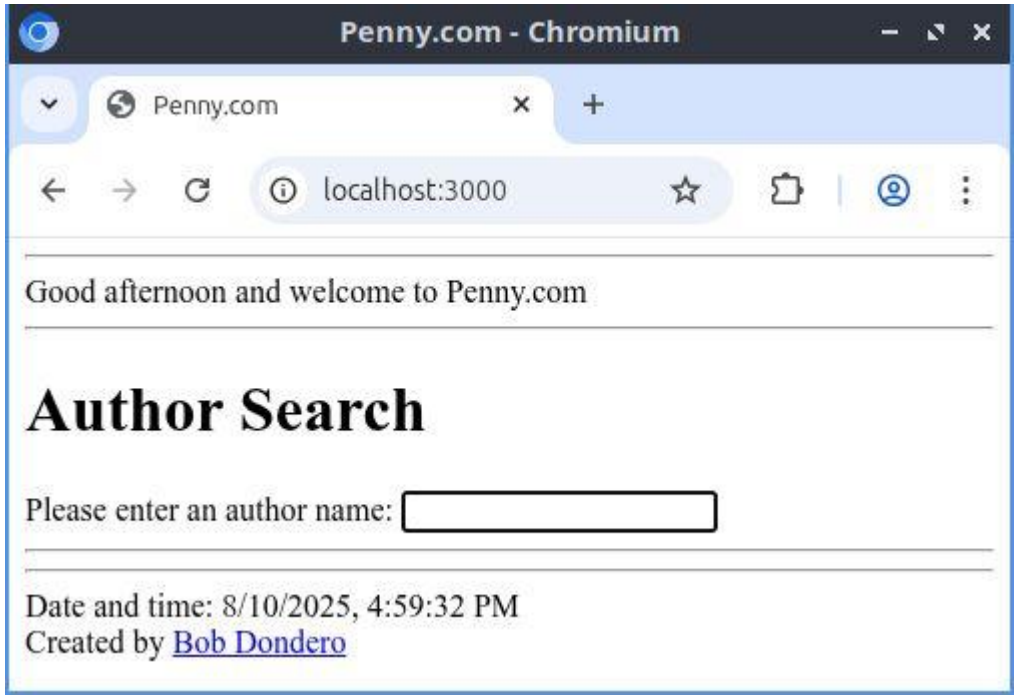
Vite Detailed Instructions

- **Step 3.1**: Run PennyReactViteFrontend
  - Start the front-end server on localhost at port 3000

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

Vite Detailed Instructions

- **Step 3.2**: Browse to http://localhost:3000



Vite Detailed Instructions

- **Step 4**: Run the front-end server in production mode

Vite Detailed Instructions

- **Step 4.1**: Build PennyReactViteFrontend
  - Build the React bundle

```
cd PennyReactViteFrontend
npm run build
```

Vite Detailed Instructions

- **Step 4.2**: Run PennyReactVite
  - Start the front-end server on localhost at port 3000

```
cd PennyReactViteFrontend
npm run preview
```

Vite Detailed Instructions

- **Step 4.3**: Browse to http://localhost:3000