Client-Side Web Programming: JavaScript (Part 4)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

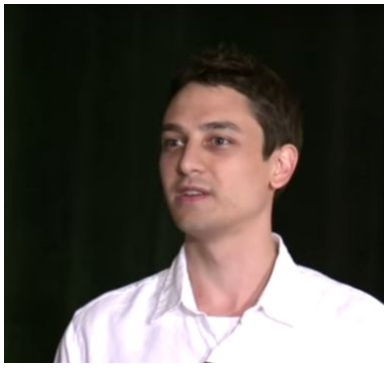
- We will cover:
  - React
  - React with JSX
  - Bundled React apps

Agenda

- **React**
- React with JSX
- Bundled React apps
- jQuery vs. React

React

- **Who**: Jordan Walke
- **When**: 2011
- **Where**: Facebook
- **Why**: Develop user interfaces in web apps



Note: **React == React.js == ReactJS**

React: Overview

- There are two styles of React programming:
  - Class-based
  - Functional
- We'll cover class-based…

5

React: Overview

[see slide]

I believe that object-based will be more comfortable for you

React: Overview

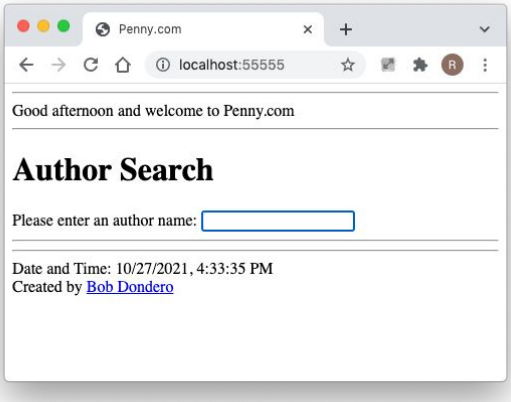
- Pgmmer defines custom ES6 classes, extending React `Component` class
- Each custom `Component` object
  - Can have properties (props)
  - Can have state
  - Renders its HTML

React: Overview

- React ***virtual DOM***
  - In-memory data structure
  - With each event:
    - Your `Component`s update parts of virtual DOM
    - React *reconciles* virtual DOM and browser DOM
      - Updates parts of browser DOM that need to be updates
      - Avoids incremental re-rendering
  - Major performance boost

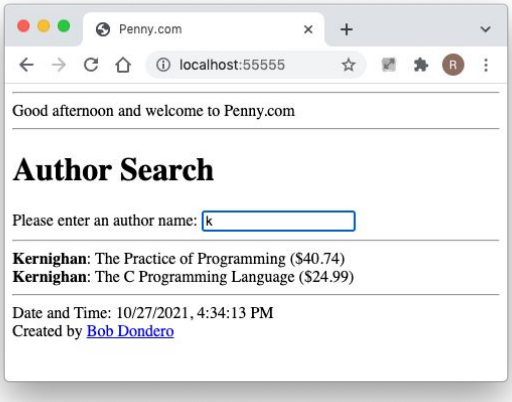
React: Example

- See **PennyReact** app



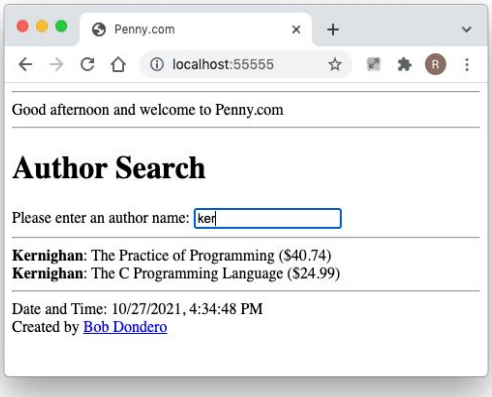
React: Example

- See **PennyReact** app



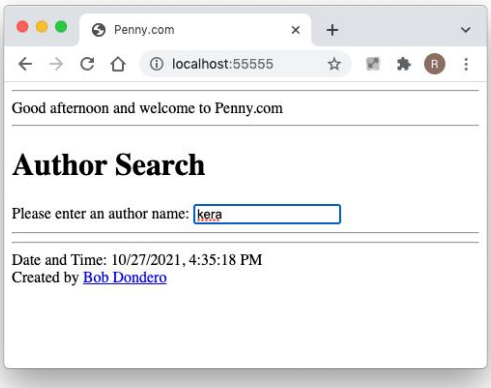
React: Example

- See **PennyReact** app



React: Example

- See **PennyReact** app



React: Example

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

12

React: Example

[see slide]

Code notes: index.html

Commands browser to fetch react
Commands browser to fetch react-dom
    Defines objects that React uses to interact with the browser DOM
Defines div element with id root
    Essentially all other HTML code is generated via JavaScript

[see subsequent slides for descriptions of classes and global code]

React: Example

- See **PennyReact** app (cont.)
  - `PennyHeader` class
    - A `PennyHeader` object…
    - Defines `PennyHeader` element
    - Has `state` object with `datetime` property
    - Defines `render()` function
      - Uses `state` object
      - Calls `React.createElement()` to create partial virtual DOM tree
      - Returns it

React: Example

- See **PennyReact** app (cont.)
  - `PennyHeader` class (cont.)
    - Via timer, `state` changes every 1 second

React: Example

- See **PennyReact** app (cont.)
  - `PennyHeader` class (cont.)
    - At page load, and whenever `PennyHeader` object's state changes…
    - React calls `PennyHeader` object's `render()`
      - Uses `state` to create & return a partial virtual DOM tree
    - React updates virtual DOM tree
    - React reconciles virtual DOM tree with browser DOM tree

React: Example

- See **PennyReact** app (cont.)
  - `PennyFooter` class
    - A `PennyFooter` object…
    - Defines `PennyFooter` element
    - Has `state` object with `datetime` property
    - Defines `render()` function
      - Uses `state` object
      - Calls `React.createElement()` to create partial virtual DOM tree
      - Returns it

React: Example

- See **PennyReact** app (cont.)
  - `PennyFooter` class (cont.)
    - Via timer, `state` changes every 1 second

React: Example

- See **PennyReact** app (cont.)
  - `PennyFooter` class (cont.)
    - At page load, and whenever `PennyFooter` object's state changes…
    - React calls `PennyFooter` object's `render()`
      - Uses `state` to create & return a partial virtual DOM tree
    - React updates virtual DOM tree
    - React reconciles virtual DOM tree with browser DOM tree

React: Example

- See **PennyReact** app (cont.)
  - `PennySearch` class
    - A `PennySearch` object…
    - Defines `PennySearch` element
    - Has `state` object with `books` and `inputValue` properties
    - Defines `render()` function
      - Uses `state` object
      - Calls `React.createElement()` to create partial virtual DOM tree
      - Must call special function to insert HTML into virtual DOM tree; makes programmer aware of XSS attacks (described later in course)
      - Returns it

React: Example

- See **PennyReact** app (cont.)
  - `PennySearch` class (cont.)
    - When contents of `inputElement` change
      - `getResults()` executes
      - `getResults()` does AJAX request using `fetch`
      - `getResults()` changes state

React: Example

- See **PennyReact** app (cont.)
  - `PennySearch` class (cont.)
    - At page load, and whenever `PennySearch` object's state changes…
    - React calls `PennySearch` object's `render()`
      - Uses `state` to create & return a partial virtual DOM tree
    - React updates virtual DOM tree
    - React reconciles virtual DOM tree with browser DOM tree

React: Example

- Global code:
  - Generates `div` element containing:
    - `PennyHeader` element
    - `PennySearch` element
    - `PennyFooter` element
  - Calls `ReactDom.render()` function
    - Renders `div` element into virtual DOM within element whose `id` is `root`
  - React reconciles virtual DOM tree with browser DOM tree

PennyReact App

- But that's not how React apps really look…

Agenda

- React
- **React with JSX**
- Bundled React apps
- jQuery vs. React

React with JSX

- ***JSX (JavaScript XML)***
  - Allows embedding of HTML-like code (actually, XML code) in JavaScript code
  - No explicit calls of `React.createElement()`

React with JSX: Example

- See **PennyReactJsx** app
  - runserver.py
  - penny.sql, penny.sqlite
  - book.py, database.py
  - penny.py
  - **index.html**

26

React with JSX: Example

[see slide]

Code notes: index.html

Commands browser to fetch JSX

Uses JSX instead of React.createComponent()

React with JSX

- But that's still not how React apps really look…

Agenda

- React
- React with JSX
- **Bundled React apps**
- jQuery vs. React

Bundled React Apps

- The general idea:
  - Before run-time:
    - Place React and all JavaScript files comprising your app in a JavaScript bundle
  - At run-time:
    - Browser requests “index” page; server delivers it
    - Index page commands browser to fetch bundle
    - Browser requests bundle; server delivers it
    - Browser interprets bundle
    - Browser incrementally requests book lists; server incrementally delivers them

Bundled React Apps: Example

- Thanks, in part, to Lucas Manning ('20)…
- See **PennyReactBundled** app
  - runserver.py
  - penny.sql, penny.sqlite
  - book.py, database.py
  - penny.py
  - **PennyHeader.jsx, PennyFooter.jsx, PennySearch.jsx, main.js**
  - **index.html**

30

Bundled React Apps: Example

[see slide]

PennyHeader.jsx, PennyFooter.jsx, PennySearch.jsx, main.js

Each component is defined in a distinct file

Each uses the ES6 module mechanism (vs. the Node.js module mechanism)

index.html

Minimal HTML page

Causes browser to fetch static/app.bundle.js script

Browser execution of static/app.bundle.js populates div element whose id is root

Aside: Node.js Revisited

- Node.js
  - Provides tools to help with development of React client-side
    - For example: Babel, Webpack
    - Via ***npm***, the Node.js package manager

Bundled React Apps: Example

- See **PennyReactBundled** app (cont.)
  - **package.json**
  - **webpack.config.js**

32

Bundled React Apps: Example

[see slide]

Code notes: package.json

(Similar to Java pom.xml file)

Specifies modules upon which project depends

One approach: search npm repo to determine latest stable versions

Specifies build script

Code notes: webpack.config.js

Configures **Webpack** to pack all JavaScript code into one large bundle (static/app.bundle.js)

Recall: index.html causes browser to fetch/execute that bundle

Bundled React Apps: Example

- To give it a try:
  - Install the node.js environment from https://nodejs.org/en/
  - `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 Apps: Example

- To give it a try (cont.):
  - `npm run build`
    - Examines `webpack.config.js`
    - Uses **Babel** to convert JSX to JavaScript, and transpile JavaScript from ES6 to ES5
    - Uses **Webpack** to pack all ES5 JavaScript code into one large bundle (static/app.bundle.js)
  - `python runserver.py 55555`
  - Browse to `http://localhost:55555`

Bundled React Apps: Example

- At run-time:
  - Browser requests index.html; server delivers it
  - Browser requests `app.bundle.js`; server delivers it
  - Browser interprets `app.bundle.js`
  - Browser incrementally requests book lists; server incrementally delivers them

React Conclusion

- There is **much** more to React
- Next steps:
  - Study some richer examples
    - Especially examples that involve props and nested components
  - Learn about React with the ***FLUX*** pattern
    - As implemented by the ***Redux*** library
  - Learn about `create-react-app`
    - See Appendices

Agenda

- React
- React with JSX
- Bundled React apps
- **jQuery vs. React**

jQuery vs. React

- **jQuery**
  - HTML is primary, JavaScript is secondary
  - Typically in distinct files: HTML, JavaScript
    - Modularity by *technologies*
- **React**
  - JavaScript is primary, HTML is secondary
  - Typically in distinct files: components
    - Modularity by *components*

jQuery vs. React

- Commentary:
  - Is jQuery code more maintainable?
    - Because of better separation of concerns?
  - Is React code more reusable?
    - Because of encapsulated components?

Summary

- We have covered:
  - React
  - React with JSX
  - Bundled React apps

40

Summary

In this lecture we covered…

[see slide]

Summary

- We have covered:
  - JavaScript for client-side web pgmming
  - AJAX: XMLHttpRequest, fetch
  - JavaScript arrow functions
  - JavaScript libraries: jQuery, React
- See also:
  - **Appendix 1**: Two-Server Apps via create-react-app
  - **Appendix 2**: One-Server Apps via create-react-app

41

Summary

In this sequence of lectures on JavaScript Client-Side Web Programming, we have covered…

[see slide]

Appendix 1:
Two-Server Apps via create-react-app

Create-react-app

- ***create-react-app***
  - Node.js program
  - Latest way to create a React app
  - By default generates…

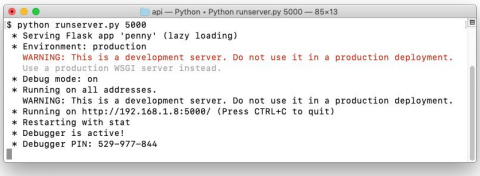
Create-react-app

- ***Two-server app***
  - Server 1 (JavaScript+Express)
    - Delivers HTML page containing React code
    - Forwards data requests to server 2
  - Server 2 (Python+Flask)
    - Delivers data (typically XML or JSON) in response to AJAX requests

PennyReactTwoServer App

- See **PennyReactTwoServer** app

Launch server 2:

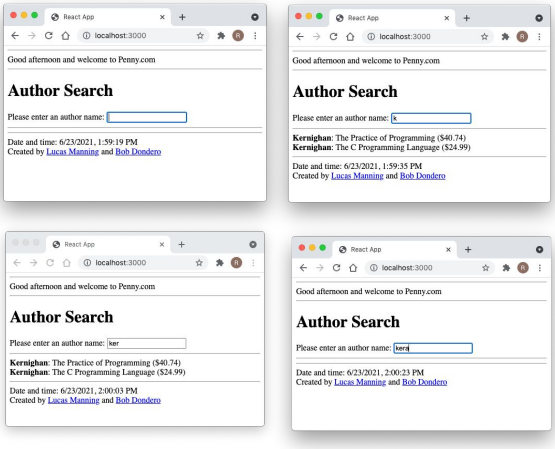


Launch server 1:

`npm start`

PennyReactTwoServer App

- See **PennyReactTwoServer** app



PennyReactTwoServer App

- Steps to create your own version:
- (1) Install `npm`
  - Instructions are OS specific
- (2) Use `npm` to install create-react-app
  - `mkdir PennyReact`
  - `cd PennyReact`
  - `npm install create-react-app`
- (3) Use create-react-app to create an app named pennyreact
  - `npx create-react-app pennyreact`

PennyReactTwoServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        .git/
        .gitignore
        node_modules/
            …
        package-lock.json
        package.json
        public/
            favicon.ico
            index.html
            logo182.png
            logo512.png
            manifest.json
            robots.txt
```

```
PennyReact/
    pennyreact/
        README.md
        src/
            App.css
            App.js
            App.test.js
            index.css
            index.js
            logo.svg
            reportWebVitals.js
            setupTests.js
```

PennyReactTwoServer App

- (4) Create an api directory below pennyreact and place **runserver.py**, **book.py**, **database.py**, **penny.py**, **penny.sqlite**, and **penny.sql** in that directory
  - Note: penny.py has only 1 endpoint

PennyReactTwoServer App

- (5) Delete all files from the src directory, and place **PennyHeader.jsx**, **PennySearch.jsx**, **PennyFooter.jsx** and **index.js** in that directory
  - Same as in PennyReactBundled app

PennyReactTwoServer App

- (6) Delete all files from the public directory, and place **index.html** in that directory
  - Essentially same as in PennyReactBundled app

PennyReactTwoServer App

- (7) Add line to pennyreact/package.json:

```
{
  "name": "pennyreact",
  "version": "0.1.0",
  "private": true,
  "proxy": "http://localhost:5000",
  "dependencies": {
   …
```

  - (Don't forget trailing comma)
  - Thereafter server 1 will forward any HTTP request that it cannot handle to server 2

PennyReactTwoServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        api/
            book.py
            database.py
            penny.py
            penny.sql
            penny.sqlite
            runserver.py
        node_modules/
            …
```

```
PennyReact/
    pennyreact/
        package-lock.json
        package.json
        public/
            index.html
        README.md
        src/
            PennyFooter.jsx
            PennyHeader.jsx
            PennySearch.jsx
            index.js
```

PennyReactTwoServer App

- (8) Build the React bundle
  - `cd …/PennyReact/pennyreact`
  - `npm run build`

PennyReactTwoServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        api/
            book.py
            database.py
            penny.py
            penny.sql
            penny.sqlite
            runserver.py
        node_modules/
            …
```

```
PennyReact/
    pennyreact/
        build/
            asset-manifest.json
            index.html
            static/
                js/
                    …
        package-lock.json
        package.json
        public/
            index.html
        README.md
        src/
            PennyFooter.jsx
            PennyHeader.jsx
            PennySearch.jsx
            index.js
```

PennyReactTwoServer App

- (9) Launch server 2
  - `cd …/PennyReact/pennyreact/api`
  - `python runserver.py 5000`

PennyReactTwoServer App

- (10) Launch server 1
  - `cd …/PennyReact/pennyreact`
  - `npm start`
- Browser renders application

Create-react-app Commentary

- Two-server app (vs. one-server app) commentary
  - (pro) More modularity
  - (pro) More separation of concerns
  - (con) Harder to maintain server-side state
    - Cookies, sessions, CAS?
  - (con) Can yield CORS (cross-origin-resource sharing) errors
  - (con) Harder to deploy (to Heroku)

Appendix 2:
One-Server Apps via create-react-app

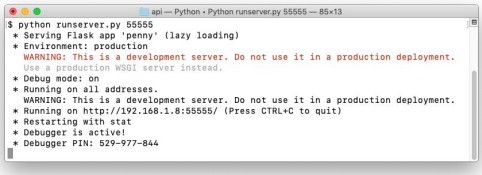
Create-react-app

- ***One-server app***
  - Server 1 (Python+Flask)
    - Delivers HTML page containing React code
    - Delivers data (typically XML or JSON) in response to AJAX requests
  - Structurally similar to PennyReactBundled app

PennyReactOneServer App

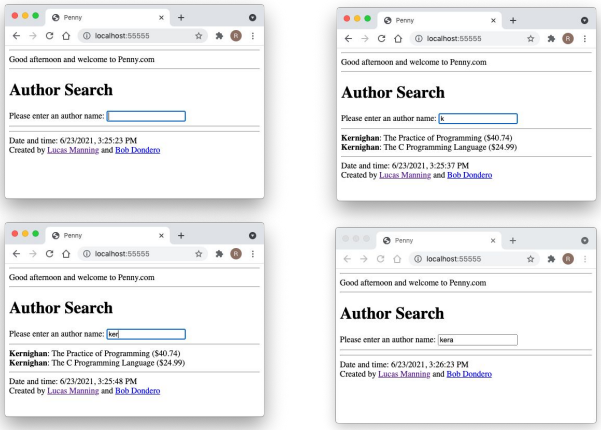
- See **PennyReactOneServer** app

Launch server 2:



PennyReactOneServer App

- See **PennyReactOneServer** app



PennyReactOneServer App

- Steps to create your own version:
- (1) Install `npm`
  - Instructions are OS specific
- (2) Use `npm` to install create-react-app
  - `mkdir PennyReact`
  - `cd PennyReact`
  - `npm install create-react-app`
- (3) Use create-react-app to create an app named pennyreact
  - `npx create-react-app pennyreact`

PennyReactOneServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        .git/
        .gitignore
        node_modules/
            …
        package-lock.json
        package.json
        public/
            favicon.ico
            index.html
            logo182.png
            logo512.png
            manifest.json
            robots.txt
```

```
PennyReact/
    pennyreact/
        README.md
        src/
            App.css
            App.js
            App.test.js
            index.css
            index.js
            logo.svg
            reportWebVitals.js
            setupTests.js
```

PennyReactOneServer App

- (4) Create an api directory below pennyreact and place **runserver.py**, **book.py**, **database.py**, **penny.py**, **penny.sqlite**, and **penny.sql** in that directory
  - Note: penny.py has 2 endpoints
    - First delivers index.html
    - Second delivers books in response to AJAX calls

PennyReactOneServer App

- (5) Delete all files from the src directory, and place **PennyHeader.jsx**, **PennySearch.jsx**, **PennyFooter.jsx** and **index.js** in that directory
  - Same as in PennyReactBundled app

PennyReactOneServer App

- (6) Delete all files from the public directory, and place **index.html** in that directory
  - Essentially same as in PennyReactBundled app

PennyReactOneServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        api/
            book.py
            database.py
            penny.py
            penny.sql
            penny.sqlite
            runserver.py
        node_modules/
            …
```

```
PennyReact/
    pennyreact/
        package-lock.json
        package.json
        public/
            index.html
        README.md
        src/
            PennyFooter.jsx
            PennyHeader.jsx
            PennySearch.jsx
            index.js
```

PennyReactOneServer App

- (7) Build the React bundle
  - `cd …/PennyReact/pennyreact`
  - `npm run build`

PennyReactOneServer App

- Resulting directory structure:

```
PennyReact/
    node_modules/
        …
    package-lock.json
    pennyreact/
        api/
            book.py
            database.py
            penny.py
            penny.sql
            penny.sqlite
            runserver.py
        node_modules/
            …
```

```
PennyReact/
    pennyreact/
        build/
            asset-manifest.json
            index.html
            static/
                js/
                    …
        package-lock.json
        package.json
        public/
            index.html
        README.md
        src/
            PennyFooter.jsx
            PennyHeader.jsx
            PennySearch.jsx
            index.js
```

PennyReactOneServer App

- (8) Launch the server
  - `cd …/PennyReact/pennyreact/api`
  - `python runserver.py 55555`
- Browse to http://localhost:55555

Create-react-app Commentary

- One-server app (vs. two-server app) commentary
  - (con) Less modularity
  - (con) Less separation of concerns
  - (pro) Easier to maintain server-side state
    - Cookies, sessions, CAS
  - (pro) Avoids CORS (cross-origin resource sharing) errors
  - (pro) Easier to deploy (to Heroku)