CS5220 Advanced Topics in Web Programming

React for Building Single-Page Applications (SPA)

Chengyu Sun

California State University, Los Angeles

Create React App

- The recommended way to create a new single-page app

```
npx create-react-app <name>
```

- See `package.json` for build commands
- Use `npm run eject` to see the tooling underneath

Basic App Structure

$$\begin{tikzcd}
 & \text{<div id="root"></div>} \\
\text{index.html} \arrow[ur] \arrow[r] \arrow[d, Rightarrow] & \text{Bundled scripts added by the build process} \\
\text{index.js} \arrow[r] \arrow[d, Rightarrow] & \text{Loads the React libraries and renders the App component} \\
\text{App.js} \arrow[r] & \text{The App component}
\end{tikzcd}$$

Imports

- ES6 modules
- Webpack imports

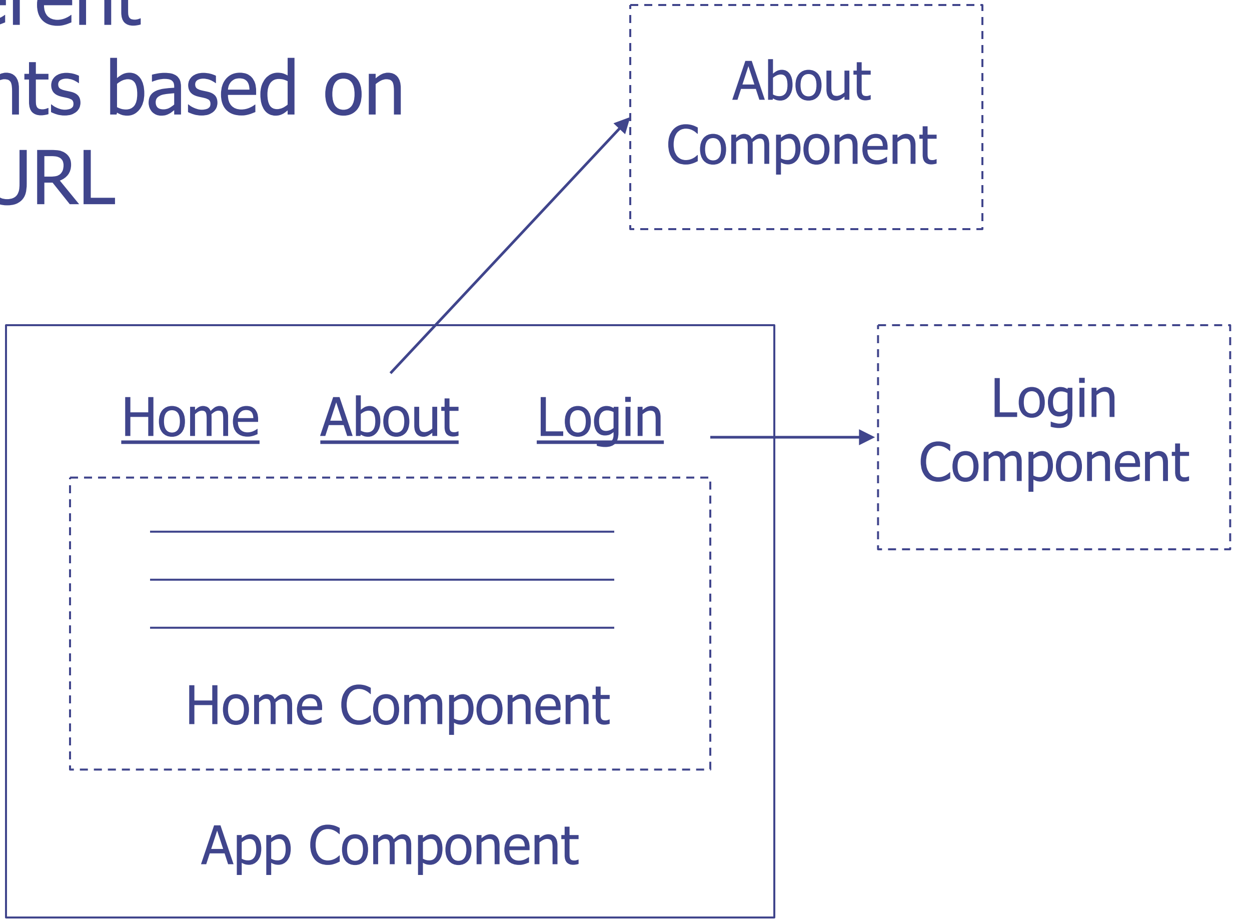
The Need for Routing

- Usually we have different pages in a web application
- How do we have different "pages" in a SPA?

Home
About
Login

Routing

- Load different components based on different URL



React Router

- React Router is the most popular routing solution for React

```
npm install react-router-dom
```

Basic Usage of React Router …

```
<Router>
   <ul>
      <li><Link to="/">Home</Link></li>
      <li><Link to="/about">About</Link></li>
  </ul>
   <div id="content">
      <Switch>
         <Route path="/about"><About /><Route>
         <Route path="/"><Home /></Route>
      </Switch>
   </div>
</Router>
```

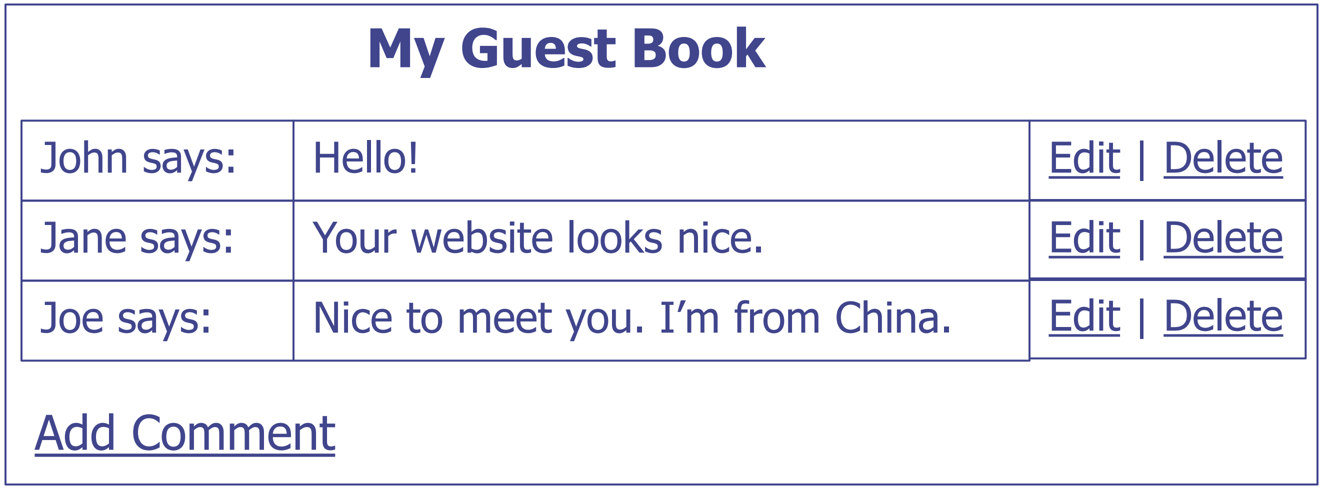
… Basic Usage of React Router

- `<Router>` component listens to URL changes and loads/unloads components
- `<Link>` replaces `<a>` so the requests do not go back to the server
- `<Route>` maps a path to a component
  - `exact` attribute matches the full path instead of just the prefix
- `<Switch>` renders the first matched `<Route>` in its children

Other React Router Components

- `<BrowserRouter>` **vs** `<HashRouter>`
  - BrowserRouter relies on HTML 5 History API which may not be available on older browsers
  - BrowserRouter also relies on the web server configured to serve index.html regardless of the path
  - HashRouter has # in URLs which is aesthetically unpleasing
- `<Link>` **vs** `<NavLink>`
  - NavLink allow more styling (e.g. activated link)

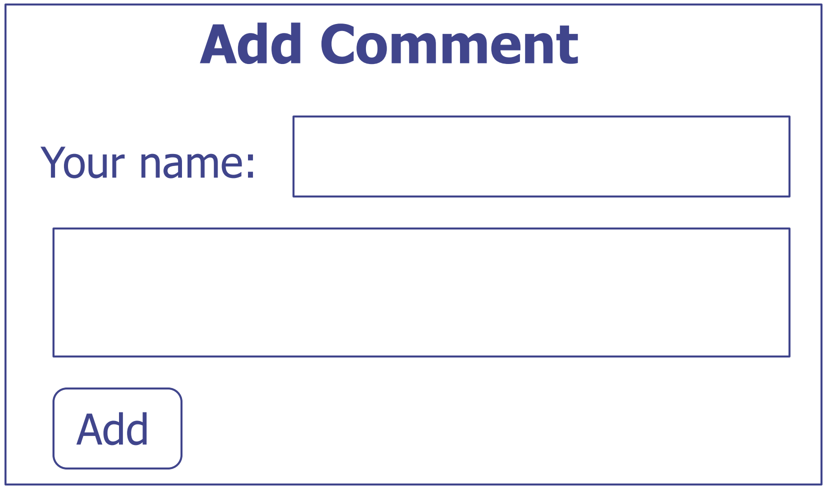
Guest Book Example



My Guest Book

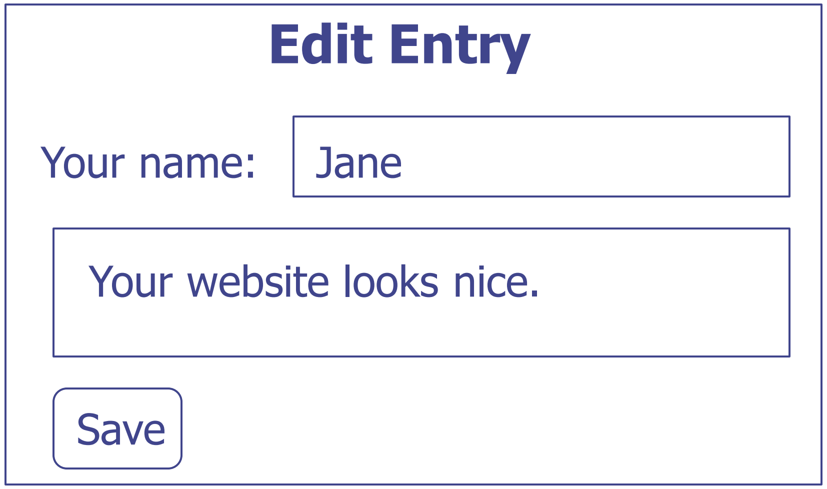
John says:	Hello!	Edit | Delete
Jane says:	Your website looks nice.	Edit | Delete
Joe says:	Nice to meet you. I'm from China.	Edit | Delete

Add Comment



Add Comment

Your name:



Edit Entry

Your name: Jane

Your website looks nice.

GuestBook REST API

- https://github.com/cysun/guestbook-node-api

List Entries

- Fragment: `<></>`
- Keep entries in React states
- Map entries to an array of <tr>
- Load data from server using `useEffect()` and axios

Map GuestBook Entries to Rows

Array map() method converts one array into another

When a list of elements are rendered, each element should a string `key` attribute

JavaScript expression

```
<tbody>
{
   entries.map(entry => (
      <tr key={entry.id}>
         <td>{entry.name}</td>
         <td>{entry.message}</td>
      </tr>
   ))
}
</tbody>
```

useEffect()

- A "side effect" is something a function does that's not part of what it returns
- `useEffect(func)`: func will be called after the component is rendered
- A second argument can be used to control when `func` is called, e.g. when some states change
  - `[]` to run func only once
  - More in How the useEffect Hook Works

Fetch vs Axios

- Both are Promise-based
- Fetch is a standard [Web API]
  - May not be available on older browsers
- [Axios] is a popular JavaScript HTTP client library
  - API is more use friendly
  - Treat error status codes as error so it can be handled in `catch()` instead of `then()`

Add Entry

- Routing
  - `<Link>`
  - Redirect after the entry is added
    - useHistory() and history
- Form handling

Forms Are Special …

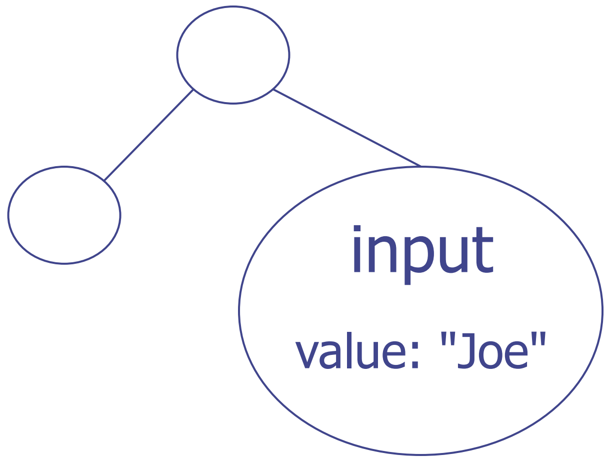
```
const [name, setName] = useState("Joe");

<input type="text" value={name} />
```

React

state: { name: "Joe" }

DOM



… Forms Are Special

- Form elements keep their internal states in DOM (e.g. `value` for `<input>`)
- The data kind of goes from view → state instead of the state → view
- React only allows *one-way binding*

Three Ways to Deal with Forms

- Use ref to reference DOM elements and get their values
- Use Controlled Components (recommended)
  - Handle `onChange` event
  - Set state in event handler
- Use custom hooks (for large forms)
  - See Chapter 6 of Learning React (2nd Ed)

Submit a Form

- Handle the `onSubmit` event of the form, e.g.

```
<form onSubmit={
    event => {
        event.preventDefault();
        axios.post("/", {
            name,
            message
        })
        .then( ()=>history.push("/") );
    }
}>
```

Event handler

Prevent browser from submitting the form

"Redirect" to /

A hook provide by React Router

Edit and Delete

- Use URL parameters, e.g. `/edit/:id`
  - useParams()

References

- React Router