Single Page Applications

Mendel Rosenblum

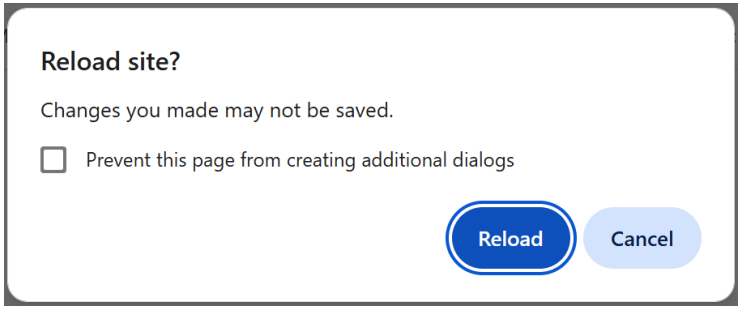
Web Apps and Browsers

- Web apps run in browsers (by definition)
- Users are used to browsing in browsers



  - Browser maintains a history of URLs visited
    - Back button - Go back in history to previous URL
    - Forward button - Go forward in history to next URL
  - Can move to a different page
    - Typing into location bar or forward/back buttons
    - Selecting a bookmarked URL
    - Page refresh operation
- Browser tosses the current JavaScript environment when navigating to a different page (except bfcache (back/forward cache))
  - Problematic for JavaScript frameworks: URL (and cookies) are the main shareable state

Problem with some web apps



- Initial: pages served from web server
  - Each page had a URL and app switched between pages served by web server
- Early JavaScript apps: Website a single page/URL with the JavaScript

  Problem: Restart web app on navigation (Can lose a lot of work!)

  ```
  addEventListener('beforeunload', e => e.preventDefault())
  ```

- Users expect app in browser to do the right thing:
  - Navigate with forward and back buttons, browser page history
  - Navigate away and come back to the app
  - Bookmark a place in the app
  - Copy the URL from the location bar and share it with someone
  - Push the page refresh button on the browser

Changing URL without page refresh

- Can change hash fragment in URL without reload

  `http://example.com` `http://example.com#fragment`

  `http://example.com?id=3535`
  `http://example.com?id=3535#fragment`

- HTML5 give JavaScript control of page reload
  - Browser History API - `window.history` - Change URL without reloading page
  - `history.pushState(state, unused, url)` – adds an entry to the browser's session history stack.

Deep linking

- Concept: the URL should capture the web app's context so that directing the browser to the URL will result the app's execution to that context
  - Bookmarks
  - Sharing
- Context is defined by the user interface designer!

  Consider: Viewing information of entity and have an edit dialog open

  Should the link point to the entity view or to the entity & dialog?

  Does it matter if I'm bookmarking for self or sharing with others?

  How about navigating away and back or browser refresh?

Deep linking in Single Page Apps

Two approaches:

1. Maintain the app's context state in the URL
   + Works for browser navigation and refresh
   + User can copy URL from location bar
2. Provide a share button to generate deep linking URL
   + Allows user to explicitly fetch a URL based on need
   + Can keep URL in location bar pretty

Either way web app needs to be able to initialize self from deep linked URL

Ugly URLs

http://www.example.org/dirmod?sid=789AB8&type=gen&mod=Core+Pages&gid=A6CD4967199

versus

http://www.example.org/show/A6CD4967199

What is that ugly thing in the location bar above my beautiful web application?

https://www.flickr.com/photos/jarnasen/24593000826/in/explore-2016-01-26/

React support for SPA

- ReactJS has no opinion! Need 3rd party module.
- Example: React Router v8 reactrouter.com
  - Idea: Use URL to control conditional rendering
  - Three modes: declarative (shown in lecture), data, framework
- Various ways of encoding information in URL
  - HTML5 History API (default): `BrowserRouter`
  - Fragment part of URL (fallback for static hosting): `HashRouter`
- Import as a module:

```
import {BrowserRouter, Routes, Route, Link, Outlet} from 'react-router';
```

Example React Router v8

```
<BrowserRouter>
  <Routes>
    <Route path="/" element={<App />}>
      <Route path="states" element={<States />} />
    </Route>
  </Routes>
</BrowserRouter>

// Inside App:
<Link to="/states">States</Link>
<Outlet />
```

- `<Routes>` renders the `<Route>` whose path matches the URL
- A nested route renders where the parent puts `<Outlet/>`
- Use `Link` to generate a hyperlink:
  `<a href="/states">`
  `States</a>`

Passing parameters with React Router

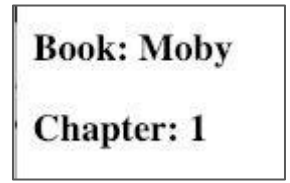
- Parameter passing in URL

```
<Route path="/Book/:book/ch/:chapter"
  element={<BookChapter />} />
```

- Read parameters with the `useParams()` hook

```
function BookChapter() {
  const { book, chapter } = useParams();
  return (<div>
    <h3>Book: {book}</h3>
    <h3>Chapter: {chapter}</h3>
  </div>);
}
```

```
<Link to="/Book/Moby/ch/1">
     Moby
</Link>
```



Route: element, nested routes

- `element={<BookChapter />}`
  - Mounts on match, unmounts when the URL no longer matches
  - Reads URL state with hooks: `useParams()`, `useLocation()`
- `<Route>` inside `<Route>`: nested routes
  - Parent layout always renders
  - Matching child renders in the parent's `<Outlet/>`
- `<Routes>` renders the most specific matching route
  - Priority: static (`/states/new`) > `:param` (`/states/:id`) > `*`
  - Order of routes in the code does not matter
- Loading data before render (`loader`) needs data mode: `createBrowserRouter`

Example

```
example.com/users?q=ann&sort=salary&dir=desc&page=2&size=10
```

Search: **ann**

Name	Email	Salary ▼
Ann Lee	ann.lee@mail.com	$103,999
Annika Ross	annika@mail.com	$86,740
Joanna Hart	jhart@mail.com	$63,736

‹ Page **2** of 5 › 10 per page

URL → UI state

`q` → search box

`sort, dir` → sorted column

`page, size` → pager

What to keep in URL: table length, viewport in table, search box, sort column, etc.
Is it different for bookmark or share? Nav away and back?

Example: Not everything goes in URL

