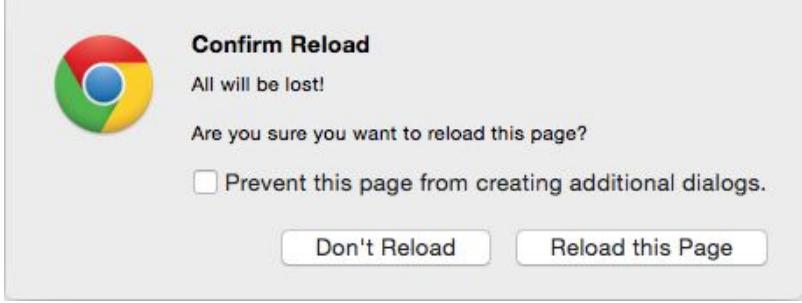
Single Page Applications

Mendel Rosenblum

Web Apps and Browsers

- Web apps run in browsers (by definition)
- Users are use 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
  - Problematic for JavaScript frameworks: URL (and cookies) are the only information preserved

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!)

  ```
  window.onbeforeunload = function(e) { return 'All will be lost!'; }
  ```
- 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

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/
```

ReactJS support for SPA

- ReactJS has no opinion! Need 3rd party module.
- Example: React Router Version 5 https://v5.reactrouter.com/
  - Idea: Use URL to control conditional rendering
  - Newer version 6 is available using same concepts as v5 but slightly different syntax
- Various ways of encoding information in URL
  - In fragment part of the URL: `HashRouter`
  - Use HTML5 URL handler: `BrowserRouter`
- Import as a module:

```
import {HashRouter, Route, Link, Redirect} from 'react-router-dom';
```

Example React Router V5

```
<HashRouter>
   <div>
     ...
     <Route path="/states" component={States} />
     ...
     <Link to="/states">States</Link>
      ...
    </div>
 </HashRouter>
```

- JSX block controlled by URL enclosed in `HashRouter`
- `Route` will render the component if URL matches.
- Use `Link` component to generated hyperlink:

```
<a href="#/states">
  States
</a>
```

Passing parameters with React Router

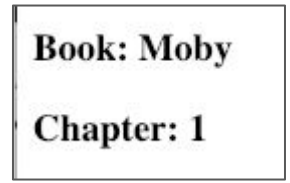
- Parameter passing in URL

```
<Route
    path="/Book/:book/ch/:chapter"
    component={BookChapterComponent}
/>
```

- Parameters put in `prop.match` of the component

```
function BookChapterComponent({ match }) {
  return ( <div>
             <h3>Book: {match.params.book}</h3>
             <h3>Chapter: {match.params.chapter}</h3>
           </div> );
}
```

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



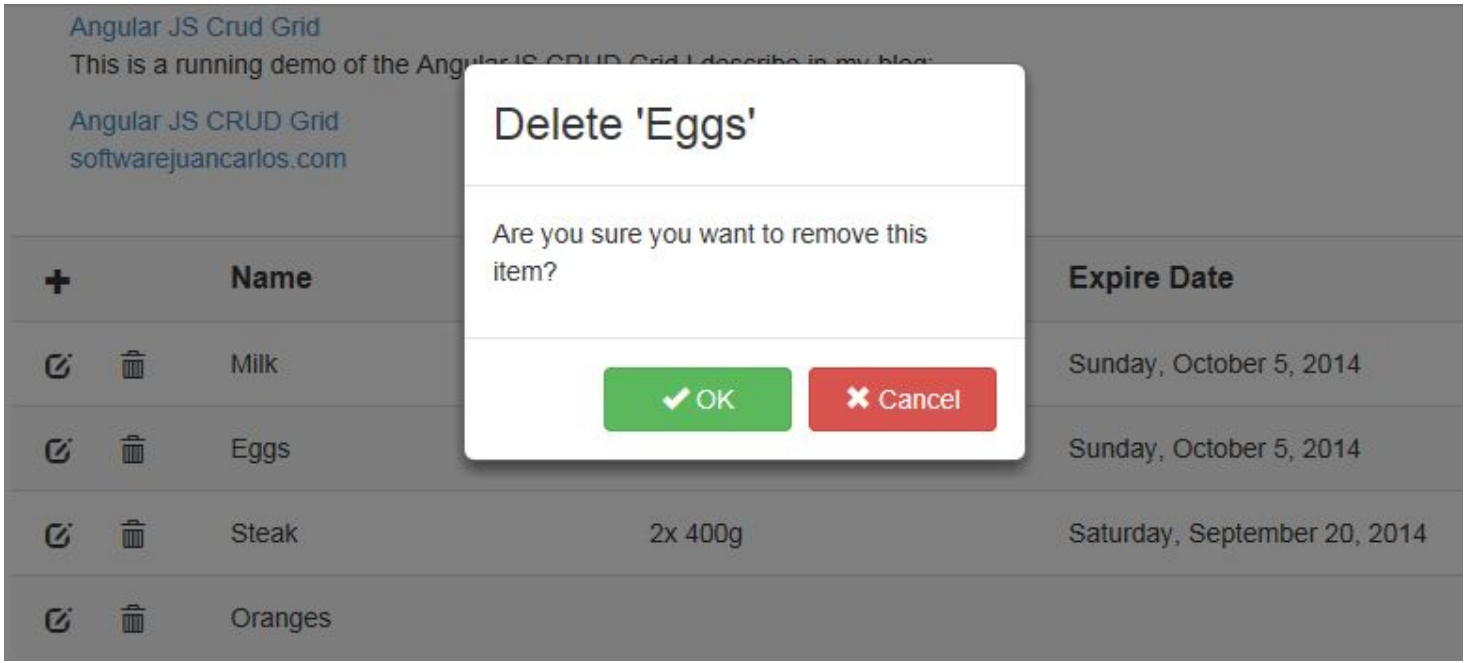
Route: `component=`, `render=`, `children=`

- `component={BookChapterComponent}`
  - Mounts components on match (unmounts on URL change)
  - Passes `match` object with: `params`, `url`, `history`
- `render={props => <BookChapterComponent book={props.match.params.book} chapter={props.match.params.chapter} />}`
  - Calls function with props having `match` object from above.
  - Doesn't mount/unmount component (does update it)
- `children=` - Like render= except is called regardless of the match
  - match will be `null` if URL doesn't match
  - Useful if you want to have something always render but only active on matching URL.

Multiple route matches have precedence order: component, render, children

`Switch` is useful with multiple Route - Renders the first matching one

Example

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