Deeplinking

- **Deeplinking** is SPA urls for specific contents
  - Even though it is all the same html page
- Two options:
  - **hash-based urls**
  - **path-based urls**
- Both require:
  - Navigating SPA "pages" changes browser url
  - JS reads URL on page load and sets app state
  - Set app state on back/forward button

Routing Libraries are normal solution

- Deeplinking has lots of subtleties
    - Libraries have solved those
        - But you CAN do it "the hard way"
    - You are not expected to do so for this course
- BUT
    - You must understand UX impacts of options

Hash-based URLs

- The urls for your app all use `#`
  - Often with a path-like string after it
  - Ex: `#/`, `#/about`, `#/privacy`
- Browser using these URLs loads same page
  - Hash fragments in urls are NOT sent to server
- JS checks URL before `<App/>` renders
  - Sets initial app state
  - Conditionally Renders based on THAT state
- "Navigation" inside app sets URL
  - uses `history.pushState()` (a whole thing)

Path-based URLs

- The urls for your app all use different paths
    - Like actual files
    - Usually without file extensions
    - Ex: `/`, `/about`, `/privacy`
- Browser using these URLs loads same page
    - Server MUST be configured to do this!
- JS checks URL before `<App/>` renders
    - Sets initial app state
    - Conditionally Renders based on THAT state
- "Navigation" inside app sets URL
    - Using JS to set without a page load

Notes about Hash-based Routing

- Easier to write for developer
    - No special server configuration required
- Search Engines may not index pages of app
    - All URLs indicate same page!
- Server logs can't track which links are used
    - All URLs are same according to server

Notes about Path-based Routing

- Better for logging
- Better for Search Engines
- Requires Server configuration
    - Always return same page for multiple urls
- The CRA dev server (`npm start`) already has this!
- BUT the server you deploy to may not!
    - `npx serve` does not, for example!