# Deeplinking and State

- A site/app URL is requested by a browser
- Server sends back the HTML page

A MPA (site) has a different url for each page

- server might decide a url isn't allowed in current "state"

A SPA doesn't have this behavior naturally

- All one page = all the same URL
- Adding behavior is called "deeplinking"
    - linking to a particular state

## Deeplinking Examples

All for SPAs only

- Linking to a product
- Linking to a profile page
- Linking to a search page
- Linking to a particular view

Anything that isn't the default start

## Options for "Views"

- Conditional
  - Have state and decide what to show based on it
- A routing library
  - Does conditional work + deeplinking
  - react-router is most common choice for React

## Conditional rendering

```
function App() {
  const [page, setPage] = useState('Home');

  return (
    <div className="app">
      <Nav page={page} onClick={setPage}/>
      { page === 'Home' && <Home/> }
      { page === 'About' && <About/> }
      { page === 'Privacy' && <Privacy/> }
    </div>
  );
}
```

## Changing Pages

Following a normal `<a>` tag link will have two effects:

- Loads a different url
- Is a page load

## Two Approaches

- Link using hash fragment
  - Ex: `http://example.com/#search`

OR

- Link using different url path
  - Ex: `http://example.com/search`

## Using the Hash Fragment

First, need to understand the hash fragment

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

## Hash Fragment in URL

- Last part of url
    - AFTER path
    - AFTER any query params (`?q=cats`)
- starts with `#` and runs to end

## Hash Fragment is Client Side Only

The fragment is used by the browser

- not included in url sent to server!
- server can't record it, can't react to it

## Why client only?

Original intention was to bookmark a part of the document

- Ex: Wikipedia shows hash fragments used to jump to a section
- **https://simple.wikipedia.org/wiki/Cat#Behaviour**
- Scrolls to element with matching id

## SPAs cheat with hash fragments

As with many things on web:

- intended use not always actual use

SPA can check url on load

- set initial state
- needs no element with that id

## Using the URL Path

Usually for static pages

- path = path in document root to html

Not true for dynamic pages

- Also not true for SPAs using paths for deeplinking

## The Path can be a lie

Server with Deeplinking SPA

- returns same page for MULTIPLE paths
- Usually everything from a given root
    - that doesn't match something else
    - Ex: /index.html, /styles.css, /cat.png
        - Everything else gives index.html

on load, SPA sets initial state based on path

## What to deeplink?

You rarely put your entire state in the deeplink

Focus on what a user would want to share/return to

- A given view/page choice
    - "search page", specific product, etc
- Any essential inputs
    - search input, product id, etc

## Does URL UX matter?

Should a URL be human-understandable?

- Old timers like me think so
- A name, a title, AND an address
- We still share by voice and/or copy by hand

Not a resolved issue

- How often do you look at urls?
- Many are unrepeatable gibberish
    - But still "work"
    - And most never notice

## Include Deeplinks in initial design

When you define behaviors

- Include deeplink plans
- Impacts state model
    - and models are HUGE
- Impacts navigation

Deeplinking is not a casual UX impact

## Deeplink Design Example

Imagine: Building a mini-Canvas for the class

Functions:

- Pending assignments
- Next class
- Links to recordings
- Links to notes
- Syllabus
- Email instructor/TA

Which of these should have deeplink?

## Navigation vs Data

State Changes

- Sometimes are data (entered or chosen)
- Sometimes are navigation (mode/view change)
    - Explicit (click a link to change view)
    - Implicit (action that results in new view)

## Deeplinked State

Deeplinked state may be data OR navigation

- More commonly view
- Consider search input though!
- Hard to do complex data
    - Ex: Interest calculator
- Not impossible to do complex data
    - Ex: game "build" calculators
    - Usually has dense "code" that represents many choices

## Client State vs Server

"State" can be

- From client (in browser memory only)
- From server (stored on server, send to client)

Usually a mix of both

Deeplinking is about preserving client state only

- But can reference server data
- Example: a "profile" page link
    - requires login to show YOUR profile
    - deeplink url sets UI state to be on that view

## Navigation in SPA

If you reload a SPA

- your state is lost

Deeplinking will get you a new state

- but is a lousy way to change views
    - loses all other state
    - reloads all html/JS/CSS/images/etc
- better only for bookmarks or sharing

## How to navigate?

How to have links that change view/mode?

- Option 1: use local hash fragment links
    - Ex: `<a href="#profile">`
    - Doesn't trigger a reload
    - Does change url
        - Best to only change url for deeplinks
        - But you can ignore this advice
- Option 2: preventDefault on links

## What to href

Advice: Don't use `<a href="#">`

- give a short, meaningful name (a hint)
    - be kind to the next coder
        - it may even be future you!
    - shows up in browser hover
        - UX!
- preventDefault() to keep out of browser url
    - is bare "#" in url "ugly"?
    - deeplinks SHOULD go to browser url

# Link vs Button

If not a deeplink, should you use a link?

- `<a>` vs `<button>`
- Navigation vs "control"
- You can change each to look like the other

Exact line being debated!

## Deeplinking with hash

- Navigation (HTML)
  - `<a href="#profile">`
- Controls (JS)
  - `document.location.hash = '#profile';`
- On load (JS)
  - `const hash = document.location.hash;`
  - Use to set initial state
    - Change the value passed to `useState`
    - NOT calling setter if you can avoid it
      - triggers a re-render

## Hash Deeplinks: Pro vs Con

Pro:

- Easy to set url w/navigation links
- No server configuration required
    - CRA works everywhere!

Con:

- Confusion with on-page ids
- No server log info for analysis
- Server can't redirect if links later change
    - UX :(
- Some consider "ugly"

# Deeplinking with path

(!) Requires server configuration (!)

- not shown here

Your CRA dev server will work this way

- But files in `build/` from `npm run build`
  - will run on a different server
  - That server may or may not have that config
  - That server may or may not be ABLE to

## Deeplinking with path

Navigation (HTML)

- must preventDefault() even on "correct" links
    - otherwise reload, source of confusing bugs

Controls (JS)

- `window.history.pushState({}, '', '/profile');`
    - first has options, second is historical

On Load (JS)

- `const path = document.location.pathname;`
- Use to set initial state (useState)

## Path Deeplinks: Pro vs Con

Pro:

- "clean" urls
- Server can redirect in future
- Server can analyze logs

Con:

- Requires server configuration
- All links must preventDefault()
  - Even "correct" ones

## Deeplinks and the Back button

SPAs hate the "Back Button"

- generally lose entire state
- rarely what user expected

Deeplinks help, but don't resolve

- only "back" through url changes
- not all state changes
- user can over do it, then all state is lost

Dealing with this is annoyingly complex!

- Many sites elect not to deal with it

## Back button isn't page load

If "back" is same page

- not a page load
- doesn't give useState() initial state

Deeplinks don't auto-work with back button

## Noticing when Back is used

`window` will fire `popstate` event

```
// Vanilla JS
window.addEventListener('popstate', (e) => {
  // Can read window.location.hash
  // or window.location.pathname
  // Note: document.location MAY not be updated yet
  // and update state
});
```

React needs the same listener

- but you want to avoid adding it repeatedly!
- useEffect() clean-up function
    - remove and re-add if component removed

# popstate can be unintended

- Fires on back
- Fires on forward
- MAY fire on page load (browser dependent)
- Fires if you change `document.location.hash`
- Fires if you follow `<a href="#profile">`

hash deeplinks may want to use pushState()

- and preventDefault()
- to avoid triggering event on navigation
- like path deeplinks

## Summary - Deeplinking

- different urls
    - load SPA in different state
- behavior must be added
- back button behavior is extra effort

## Summary - hash vs path urls

- pros/cons each
- big one: path requires server config
    - automatic in CRA *development*
    - but not auto for *deployed* CRA app

## Summary - Deeplink changes

- During design
    - What state to deeplink?
- On load
    - set state based on url
- On in-app navigation
    - prevent reload
- With in-app state changes
    - update url
- On Back
    - notice event
    - set state