Pages and Views on a SPA

An MPA has multiple-pages

- In the name

A SPA has a single-page

- Still in the name

How do we present multiple "views" for the user?

How do we handle "views" when

- User "reloads" in browser?
- User shares/bookmarks "page"?

Options for "Views"

- Conditional
  - Decide what to show based on state
- A **Routing library**
  - Does conditional work + deeplinking
  - `react-router` is a common choice for React
    - May be changing, ex: `@tanstack/router`
- Let's understand the concepts without a library
  - Use libraries once not magical

Conditional rendering

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

  return (
    <>
      <Nav setPage={setPage}/>
      { page === 'Home' && <Home/> }
      { page === 'About' && <About/> }
      { page === 'Privacy' && <Privacy/> }
    </>
  );
}
```

`<Nav>` would call `setPage()` to change current page

- And `preventDefault()` on any ACTUAL navigation
- Other components would have to call a passed `setPage()` to navigate within app

Changing Pages

Following a normal `<a>` tag link:

- Loads a different url
- Is a page load
    - Loses JS-based state!

Changing URLs

Changing URLs is a big deal!

- **Usually a different page (not the SPA)**

You can have issues here!

- Vite dev server still shows SPA
    - Only used in development
- But a static file server will not
    - Unless configured to

Page Load Means new JS State

Even if the server returns the same SPA

- You've done a page load
- Like hitting "reload/refresh"
- All state is reset to initial state

A common source of "weird" bugs

- You don't want this behavior in a SPA

Links in a SPA

If you have a SPA and a `<a>` tag link

- `e.preventDefault()`
    - Don't navigate
- Update state instead

You may recall a similar issue with `<form>` submit

Dealing with the Confusion: Deeplinking

Deeplinking is having a url

- Reflects the content shown to user
- Shows the current state when (re)loaded
- Works with Back/Forward

The Browser History API

- Allows us to change the "history stack" in browser
- Also changes current page URL in browser
    - Without triggering a reload/navigation!
- Back/Forward over added entries
    - Allows Back/Forward
    - Will not trigger a reload/navigation!
    - We will have to detect and change content

window.history.pushState() to add to history stack

- First param is a `state`
    - Whatever content we want
    - Only works with back/forward
        - We will use url only here
    - We send `null` or `{}` and ignore
- Second param was a mistake
    - Use `''` (MDN says it is "safe")
- Third param is url string
    - Absolute or relative path

Modifying our App

- Tell Browser to change URL
    - When "page" changes
- Set "page" in state
    - On Page Load
- Confirm it all works

Changing the URL using History API

```
function Nav({ setPage }) {
  function changePage(e) {
    e.preventDefault(); // stop actual browser navigation
    // change url to match path from link
    window.history.pushState(null, '', e.target.pathname);
    // Update our state to show different content
    setPage(e.target.pathname);
  }

  return (
    <nav className="nav">
      <a href="/" onClick={ changePage } >Home</a>
      <a href="/about" onClick={ changePage } >About</a>
    </nav>
  );
}
```

Behavior after setting History on page change

- 😻 App can change views ("pages")
- 😻 URL DOES change on view change
- 😿 JS State resets on reload
    - Page State may not match URL

Back/Forward over pushed history entries

- 😻 Does NOT leave the app unexpectedly
- 😻 Does NOT cause a page load
- 😿 Doesn't yet change our page state

Setting page state on page load

- Read url and set page state!
- But when?
    - Easy option: First time App() renders
        - useEffect!
        - App WILL render "wrong" once
            - Consider if/how that is a problem
            - Actual Routing libraries solve this

Modifying App.jsx

App.jsx

```
function App() {
  const [ page, setPage ] = ''; // No content initially

  useEffect( () => {  // Set state on page load
    setPage(document.location.pathname); // From browser url
  }, []); // Important to have empty dependency array!

  return (
    <>
      <Nav setPage={setPage}/>
      { page === '/' && <Home/> }
      { page === '/about' && <About/> }
    </>
  );
}
```

After changes on page load

- 😻 App can change views ("pages")
- 😻 URL DOES change on view change
- 😻 JS State matches URL on load/reload

Back/Forward over pushed history entries

- 😻 Does NOT leave the app unexpectedly
- 😻 Does NOT cause a page load
- 😿 Doesn't yet change our page state

`popstate` event when Back/Forward

Back/Forward over pushed history entries

- Does NOT yet change state
- Will fire a `popstate` event
    - on `window`
    - `window` is not controlled by React

We need to add an eventListener to window

- Outside of React
- When? On Page Load
    - `useEffect()`

Adding popstate listener

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

  useEffect( () => {
    setPage(document.location.pathname); // on page load

    window.addEventListener('popstate', () => {
      console.log('changing state');
      setPage(document.location.pathname); // on back/forward
    });
  }, []);

  return (
    <>
      <Nav setPage={setPage}/>
      { page === '/' && <Home/> }
      { page === '/about' && <About/> }
    </>
  );
}
```

Listener is added twice

- Double the effect
- Not REALLY a problem
    - But good practice to notice and fix
    - With **cleanup function**
- Removing event listeners is a bit weird
    - `.removeEventListener()`
    - With same param
    - Easiest to have named handler callback

Cleanup popstate event listener

```
useEffect( () => {

  function onPageChange() {
    setPage(document.location.pathname);
  }

  onPageChange(); // On page load

  // Back/Forward
  window.addEventListener('popstate', onPageChange);

  return () => { // useEffect cleanup
    window.removeEventListener('popstate', onPageChange);
  }
}, []);
```

After adding popstate listener

- 😻 App can change views ("pages")
- 😻 URL DOES change on view change
- 😻 JS State matches URL on load/reload

Back/Forward over pushed history entries

- 😻 Does NOT leave the app unexpectedly
- 😻 Does NOT cause a page load
- 😻 DOES change our page state

DOES require server config to always load `index.html`

Deeplinking in production

- We have all the parts working, right?
    - All happy cats!

EXCEPT...

- When we do `npm run build`
- And run with only the express server
    - It only PARTIALLY works
- Navigation, back/forward, all work
    - But reloading a url like `/about` FAILS

Server SPA Configuration

Vite Dev Server returns SPA HTML/CSS/JS

- For most requests
- Including for `/about`
    - Loads SPA
    - SPA runs `useEffect` callback
    - Sets page state

Express server looks in `./dist` for file named `"about"`

- Finds none
- Returns 404

We need a special configuration

This wasn't a problem for actual static files

- Files with matching filenames existed

Wasn't a problem for dynamic server-generated HTML

- Server routes matching filenames/paths existed

We need a special configuration to return `index.html`

Config for an express server

- Other servers use their language/conventions
- Should be LAST entry before `app.listen()`
    - So other options get chance to match request

```
app.get('*', (req, res) => { // Default to sending index.html
    res.sendFile(path.join(__dirname, "dist", "index.html"));
});
```

- Any GET request (`*`) will send dist/index.html
- Last choice, so more specific matches happen instead

Summary - Views

- SPA is all one page
- "Navigation" is a state change
    - not actual browser navigation
- preventDefault on links
    - update state instead

Summary - Deeplinking

- SPAs will use initial state when loaded
    - Making it hard to share an app with a state
- Deeplinks used by SPA to load alternate state
    - Rarely all state, just select "page"
- Requires SPA changes
    - On load: set initial state
    - On mode change: update url
- Requires server config
    - If using paths in url
- Usually done with a **routing library**