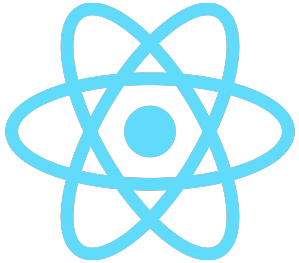


React.js

Lecture 5

Agenda

- Recap last lecture points
- Context API
- Redux vs Context
- Apply context instead of redux
- General Topics
- Open discussion and questions

Context

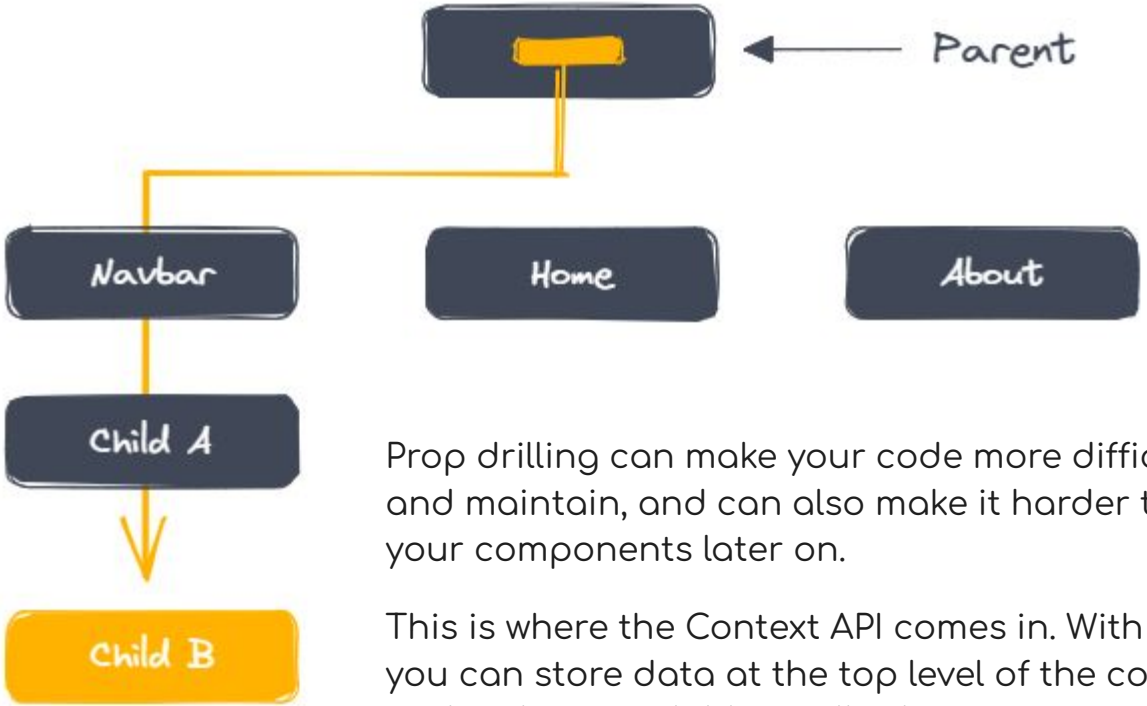
Context provides a way to pass data through the component tree without having to pass props down manually at every level.

When to Use Context

Context is designed to share data that can be considered “global” for a tree of React components, such as the current authenticated user, theme, or preferred language.

https://www.freecodecamp.org/news/context-api-in-react/

Passing Props ..



Prop drilling can make your code more difficult to read and maintain, and can also make it harder to refactor your components later on.

This is where the Context API comes in. With Context API, you can store data at the top level of the component tree and make it available to all other components that need it without passing props.

Context VS Redux

- If you are using Redux only to avoid passing props down to deeply nested components, then you could replace Redux with the Context API. It is exactly intended for this use case.

- On the other hand, if you are using Redux for everything else (handling your application's logic outside of your components, centralizing your application's state, using Redux DevTools to track when, why, and how your application's state changed, or using plugins such as Redux Form, Redux Thunk, etc...), then there is absolutely no reason for you to abandon Redux.

  The Context API doesn't provide any of this.

Context : Create context

Create context file config :

```
import React from "react";

export const ContextFeature = React.createContext();
```

Context : Use context

Wrap the components that needs to have the provider values with context provider:

```
const [value, setValue] = useState(initial value);

<Context.Provider  value={{ value, setValue}}>
        <Component />
</Context.Provider>
```

Context : Use context

useContext Hook

In child component you can use and set context value using useContext hook

const {value , setValue} = useContext(contextFeature)

Then you can use the value and its setter function to update context value

Check the following for class components with context : Link

Extra Topics

Code Splitting

Bundling is great, but as your app grows, your bundle will grow too. Especially if you are including large third-party libraries. You need to keep an eye on the code you are including in your bundle so that you don’t accidentally make it so large that your app takes a long time to load.

To avoid winding up with a large bundle, it’s good to get ahead of the problem and start “splitting” your bundle

https://dev.to/franklin030601/code-splitting-in-react-js-4o2g

Code Splitting

import()

The best way to introduce code-splitting into your app is through the dynamic `import()` syntax.

React.lazy

The `React.lazy` function lets you render a dynamic import as a regular component.

Code Splitting

Before:

```
import OtherComponent from './OtherComponent';
```

After:

```
const OtherComponent = React.lazy(() => import('./OtherComponent'));
```

This will automatically load the bundle containing the `OtherComponent` when this component is first rendered.`React.lazy` takes a function that must call a dynamic `import()`. This must return a `Promise` which resolves to a module with a `default` export containing a React component.

Code Splitting

The lazy component should then be rendered inside a Suspense component, which allows us to show some fallback content (such as a loading indicator) while we're waiting for the lazy component to load.

Example:

```
import { Suspense } from 'react';

<Suspense fallback={<div>Loading...</div>}>
    <OtherComponent />
</Suspense>
```

Env variables

https://create-react-app.dev/docs/adding-custom-environment-variables/

- Create .env file at the same level of src folder
- Any environment variable should start with REACT_APP then the key name like
    - REACT_APP_BASE_URL=http://test.com
- You can use the env variable in you code using the following
    - {process.env.REACT_APP_BASE_URL}

Remember to restart the server after change anything in env file.

Deployment

https://create-react-app.dev/docs/production-build

npm run build

creates a `build` directory with a production build of your app. Inside the `build/static` directory will be your JavaScript and CSS files. Each filename inside of `build/static` will contain a unique hash of the file contents. This hash in the file name enables long term caching techniques.

Resources

Resources

- Crash Course:
  - https://www.youtube.com/watch?v=4baq00tHfmA&t=24s
  - https://www.youtube.com/watch?v=u6gSSpfsoOQ&t=123s
  - https://www.youtube.com/watch?v=SqcY0GlETPk
  - https://www.youtube.com/watch?v=2-crBg6wpp0

- Interactive Learning Course:
  - https://scrimba.com/learn/learnreact

- React Hooks
  - https://www.youtube.com/watch?v=0riHps91AzE
  - https://www.youtube.com/watch?v=TNhaISOUy6Q&t=26s
  - https://www.youtube.com/playlist?list=PLtxOBbrOOPH4ro6EXTNHrIvmoNaOcPAwe

- Forms in deep
  - https://www.youtube.com/watch?v=a94FOvaBomQ&list=PLC3y8-rFHvwiPmFbtzEWjESkqBVDbdgGu
  - https://www.youtube.com/watch?v=KejZXxFCe2k&list=PLC3y8-rFHvwjmgBr1327BA5bVXoQH-w5s

Resources

- Redux
  - https://www.youtube.com/watch?v=u3KlatzB7GM&list=PL0Zuz27SZ-6M1J5I1w2-uZx36Qp6qhjKo
  - https://www.youtube.com/watch?v=_shA5Xwe8_4&t=1s

- Chat with firebase
  - https://www.youtube.com/watch?v=zQyrwxMPm88

- React V18
  - https://www.youtube.com/watch?v=N0DhCV_-Qbg
  - https://www.youtube.com/watch?v=uybSanJSQng

- React router V6
  - https://www.youtube.com/watch?v=zEQiNFAwDGo&t=84s
  - https://www.youtube.com/watch?v=L2kzUg6IzxM&t=136s
  - https://scrimba.com/learn/reactrouter6

Resources

- Project Structure Patterns
  - https://itnext.io/top-6-best-folder-structures-for-react-ultimate-comparison-effc29ae5045

- Translation (Format.js)
  - https://formatjs.io/
  - https://blog.logrocket.com/translate-react-app-using-format-js/

- Redux vs Context
  - https://betterprogramming.pub/my-frustrations-with-the-context-api-in-react-26189fcd5371

- UseRef Array
  - https://www.codemzy.com/blog/reactjs-useref-array

Thank you

Lap

Movie App

Task 1

Create a dropdown in the navbar with languages like (ar , en) and save the language value in a context when change the app language you should change the app direction based on the current lang

- If lang is ‘ar’ => direction would be RTL
- If lang is ‘en’ => direction would be LTR

[Bonus]

You can call the api of movies and send a query param with the current language to get data with en or ar for example :

https://api.themoviedb.org/3/movie/popular?api_key={apiKey}&language=ar

Movie App

Task 2

Apply code splitting for pages.

Task 3

Add api base url to env file