REACT

ABOVE AND BEYOND

NICK LAURYSSEN

- Frontend developer at Katoen Natie
- Axxes'er since october 2015
- 🎸 🐶 🎾

WHO ARE YOU?

SLIDO.COM

#2000

DAY 1

8:30 - 9:30: RENDERING, COMPONENTS AND PROPS

15 min break

9:45 - 10:45: STATE AND EVENTS

15 min break

11:00 - 12:15: HOOKS

DAY 1

13:15 - 14:15: CONDITIONAL RENDERING AND LISTS

15 min break

14:30 - 15:30: THINKING IN REACT

15 min break

15:45 - 17:00: ROUTING

DAY 2

8:30 - 9:30: CONTEXT

15 min break

9:45 - 10:45: STYLING AND PERFORMANCE

15 min break

11:00 - 12:15: FINAL EXERCISES

REACT?

A JAVASCRIPT LIBRARY TO BUILD USER INTERFACES BASED ON COMPONENT COMPOSITION

IN OTHER WORDS

A LIBRARY THAT HELPS YOU TO JUST RENDER SOMETHING IN YOUR BROWSER

LEARNING CURVE

REACT BASICS: 😄

THINKING IN REACT AND COMPONENTS: 😅

STUFF YOU CAN DO WITH REACT: 😳

REACT ECOSYSTEM

ENTIRELY POSSIBLE TO BUILD A REACT APP WITH NO OTHER DEPENDENCIES BUT MOST APPS DON'T

REACT ECOSYSTEM

👍 YOU CAN PICK EXACTLY THE LIBRARIES YOU NEED FOR YOUR USE CASES

👎 YOU NEED TO PICK EXACTLY THE LIBRARIES YOU NEED FOR YOUR USE CASES

REACT ECOSYSTEM

- Axios
- React-router
- React-redux
- use-hook-form
- Styled-components
- Material-UI
- Webpack
- ...

REPOSITORY

- git clone git@bitbucket.org:axxesit/traineeship-react-2022.git
- checkout main branch

SETUP FOR TODAY

- All instructions in REQUIREMENTS.md

HOW TO START THE APP

- yarn install
- yarn start

USEFUL TOOL

- React Developer Tools (chrome extension)

MATERIAL-UI DOCS

MDN JAVSCRIPT DOCS

REACT DOCS

QUIZ: INTRODUCTION

GOAL

RENDERING

Demo

VANILLA JS AND DOM

```
const rootElement = document.getElementById("root");
const element = document.createElement("div");

element.className = "container";
element.textContent = "Hello Traineeship";

rootElement.appendChild(element);
```

WITH REACT

```
const rootElement = document.getElementById("root");
const element = React.createElement("div", {
  className: "container",
  children: "Hello traineeship"
});

ReactDOM.render(element, rootElement);
```

BUT STILL NOT THE MOST ERGONOMIC API

TO WRITE UI CODE

JSX

```
const rootElement = document.getElementById("root");
const element = <div>Hello traineeship</div>;

ReactDOM.render(element, rootElement);
```

JSX

AN HTML-LIKE SYNTAX THAT EXTENDS JAVASCRIPT

USES BABEL TO COMPILE

JSX BEFORE TRANSPILATION WITH BABEL

```
const element = <div className="beautiful">Hello traineeship</div>;
```

JAVASCRIPT AFTER TRANSPILATION WITH BABEL

```
const element = React.CreateElement(
  "div",
  { className: "beautiful" },
  "Hello traineeship"
);
```

Demo

I Am Devloper
@iamdevloper

Following

Consensus: “You shouldn’t mix your HTML and JS together”,

Facebook: “You should mix your HTML and JS together”,

...

Consensus: “We should”.

RETWEETS 558 FAVORITES 427

5:32 AM - 13 May 2015

JSX, BUT WHY?

- Rendering logic coupled with other UI logic
- State changes
- Event handling
- Async data handling
- Conditional rendering
- Reduces context switching
- FULL JavaScript power for rendering logic
- ...

JSX, A FEW RULES

- Use className instead of class
- React component names must be uppercase vs lowercase HTML elements
- Elements must return something or 'null'
- Use { ... } to escape from JSX and do some pure javascript

```
const rootElement = document.getElementById("root");
const children = "Hello traineeship";
const className = "container";

const element = <div className={className}>{children}</div>;

ReactDOM.render(element, rootElement);
```

QUIZ: RENDERING AND JSX

VIRTUAL DOM

REACT ONLY RENDERS WHAT IS NECESSARY

Hello, world!

It is 12:26:46 PM.

Console Sources Network Timeline

```
▼<div id="root">
  ▼<div data-reactroot>
      <h1>Hello, world!</h1>
    ▼<h2>
        <!-- react-text: 4 -->
        "It is "
        <!-- /react-text -->
        <!-- react-text: 5 -->
        "12:26:46 PM"
        <!-- /react-text -->
        <!-- react-text: 6 -->
        "."
        <!-- /react-text -->
      </h2>
  </div>
</div>
```

VIRTUAL DOM != SHADOW DOM

VIRTUAL DOM

- Copy of the real DOM
- Actually just an object in javascript
- Diffs changes with the real DOM
- Direct DOM manipulation is very slow 👎

COMPONENTS

CONSIST OF A GROUP OF REACT ELEMENTS

COMPONENTS

JUST LIKE FUNCTIONS

- They accept inputs (=prop)
- They return React elements to describe the UI
- Start with a capital letter

DEFINING COMPONENTS

1. Functional components
2. Class components

FUNCTIONAL COMPONENT

```jsx
const HelloWorld = () => {
  return <div>Hello World!</div>;
};
```

CLASS COMPONENT

```
class HelloWorld extends React.Component {
  render() {
    return <div>Hello world!</div>;
  }
}
```

RENDERING A COMPONENT

```
const Welcome = () => {
  return <h1>Hello, Nick</h1>;
};

const element = <Welcome />;

ReactDOM.render(element, document.getElementById("root"));
```

PROPS

```
const Welcome = (props) => {
  return <h1>Hello, {props.name}</h1>;
};

const element = <Welcome name="Nick" />;

ReactDOM.render(element, document.getElementById("root"));
```

PROPS

- Attributes passed through in a single object
- Props are read-only 👆

PROPTYPES

- Anything can be passed as a prop
- Provide runtime type checking environment
- Can serve as documentation

MOST COMMON PROPTYPES

- bool
- string
- array
- func
- number

List of all Proptypes

ONE GOLDEN RULE

ALL REACT COMPONENTS MUST ACT LIKE PURE FUNCTIONS WITH RESPECT TO THEIR PROPS

EXERCISE 1

CREATE A LABEL COMPONENT THAT RENDERS TEXT IN A PARAGRAPH

- use a functional component
- use a prop 'text'

```
<Label text="Welcome to the traineeship" />
```

GIT CHECKOUT EX1

EXERCISE 1 SOLUTION

```
GIT CHECKOUT EX1-SOLUTION
```

CHILDREN

```
const Label = ({ children }) => {
  <p>{children}</p>;
};

const element = <Label>Welcome to the traineeship</Label>;
```

COMPOSING COMPONENTS

```
const ComposedComponent = () => {
  return (
    <div>
      <Welcome name="Laura" />
      <Welcome name="Tom" />
      <Welcome name="Jake" />
    </div>
  );
};
```

COMPOSING COMPONENTS

Fragments let you group a list of children without adding extra nodes to the DOM.

```
const ComposedComponent = () => {
  return (
    <React.Fragment>
      <Welcome name="Laura" />
      <Welcome name="Tom" />
      <Welcome name="Jake" />
    </React.Fragment>
  );
};
```

QUIZ: VIRTUAL DOM AND COMPONENTS

EVENTS

HANDLING EVENTS IN REACT IS QUITE SIMILAR AS WITH DOM ELEMENTS

EVENTS

- Event name is camelCase instead of lowercase
- Gets assigned a function instead of a string

EVENTS

HTML

```
<button onclick="handleClick()">Click me</button>
```

React

```
<button onClick={handleClick}>Click me</button>
```

STATE

STATE IS SIMILAR TO PROPS, IT'S ONLY PRIVATE AND FULLY CONTROLLED

CLASSES VS FUNCTIONS

HOOKS

USESTATE

INITIALIZE STATE

```
const Card = () => {
  const [ expanded, setExpanded ] = useState(false)

  return (
    ...
  )
}
```

UPDATING STATE BY VALUE

```
const Card = () => {
  const [ expanded, setExpanded ] = useState(false)

  const handleClick = () => {
    setExpanded(true)
  }

  return (
    <button onClick={handleClick}>Expand</button>
  )
}
```

UPDATING STATE BY FUNCTION

```
const Card = () => {
  const [ expanded, setExpanded ] = useState(false)

  const handleClick = () => {
    setExpanded((expanded) => !expanded)
  }

  return (
    <button onClick={handleClick}>Toggle</button>
  )
}
```

NEVER UPDATE STATE DIRECTLY

```
const [ expanded, setExpanded ] = useState(false)
expanded = true 😭
```

```
const [ expanded, setExpanded ] = useState(false)
setExpanded(true) 👍
```

USE STATE

```
const [state, setState] = useState(null);

setState((prevState) => {
  return { ...prevState, ...updatedValues };
});
```

EXERCISE 2

MAKE SURE OUR NAVIGATION DRAWER CAN BE SHOWN/HIDDEN

```
GIT CHECKOUT EX2
```

EXERCISE 2 SOLUTION

```
GIT CHECKOUT EX2-SOLUTION
```

BUILT-IN HOOKS

- useState
- useEffect
- ...

USE EFFECT

- data fetching
- subscriptions
- DOM updates

```
useEffect(() => {
  document.title = `This is the React traineeship`;
});
```

BY DEFAULT RUNS AFTER EVERY RERENDER

BUT YOU CAN PASS A SECOND ARGUMENT TO LIMIT THIS

```
useEffect(() => {
  ...
},[]) // Execute only on mount/unmount
```

```
useEffect(() => {
...
}, [someVariable]) //Execute only when someVariable changes
```

SIDE EFFECTS WITH CLEANUP

```
useEffect(() => {
  window.addEventListener("keydown", myHandler);

  return () => {
    window.removeEventListener("keydown", myHandler);
  };
}, []);
```

TIPS

- Use multiple effects to seperate concerns
- Optimize performance by skipping effects

RULES OF HOOKS

- Only call Hooks at the top level
- Only call Hooks from React function components or your own custom hooks
- Hooks start with “use”

EXERCISE 3

CREATE A TIMER COMPONENT

```
<Timer time="60" start={true} />
```

GIT CHECKOUT EX3

EXERCISE 3 SOLUTION

```
GIT CHECKOUT EX3-SOLUTION
```

YOU CAN ALSO BUILD YOUR OWN HOOKS!

CUSTOM HOOK

A CUSTOM HOOK IS A GREAT WAY TO CREATE REUSABLE LOGIC

```jsx
import { useTheme } from "@mui/material/styles";

const PrimaryText = ({ children }) => {
  const theme = useTheme();

  return <p style={{ color: theme.palette.primary.main }}>{children}</p>;
};
```

CONDITIONAL RENDERING

```jsx
function UserGreeting(props) {
  return <h1>Welcome back!</h1>;
}

function GuestGreeting(props) {
  return <h1>Please sign up.</h1>;
}
```

```jsx
function Greeting({ isLoggedIn }) {
  return <div>{isLoggedIn ? <UserGreeting /> : <GuestGreeting />}</div>;
}

ReactDOM.render(
  <Greeting isLoggedIn={false} />,
  document.getElementById("root")
);
```

EXERCISE 4

WHEN OUR TIMER ENDS INSTEAD OF SHOWING '0' SHOW A TEXT MESSAGE

`GIT CHECKOUT EX4`

EXERCISE 4 SOLUTION

```
GIT CHECKOUT EX4-SOLUTION
```

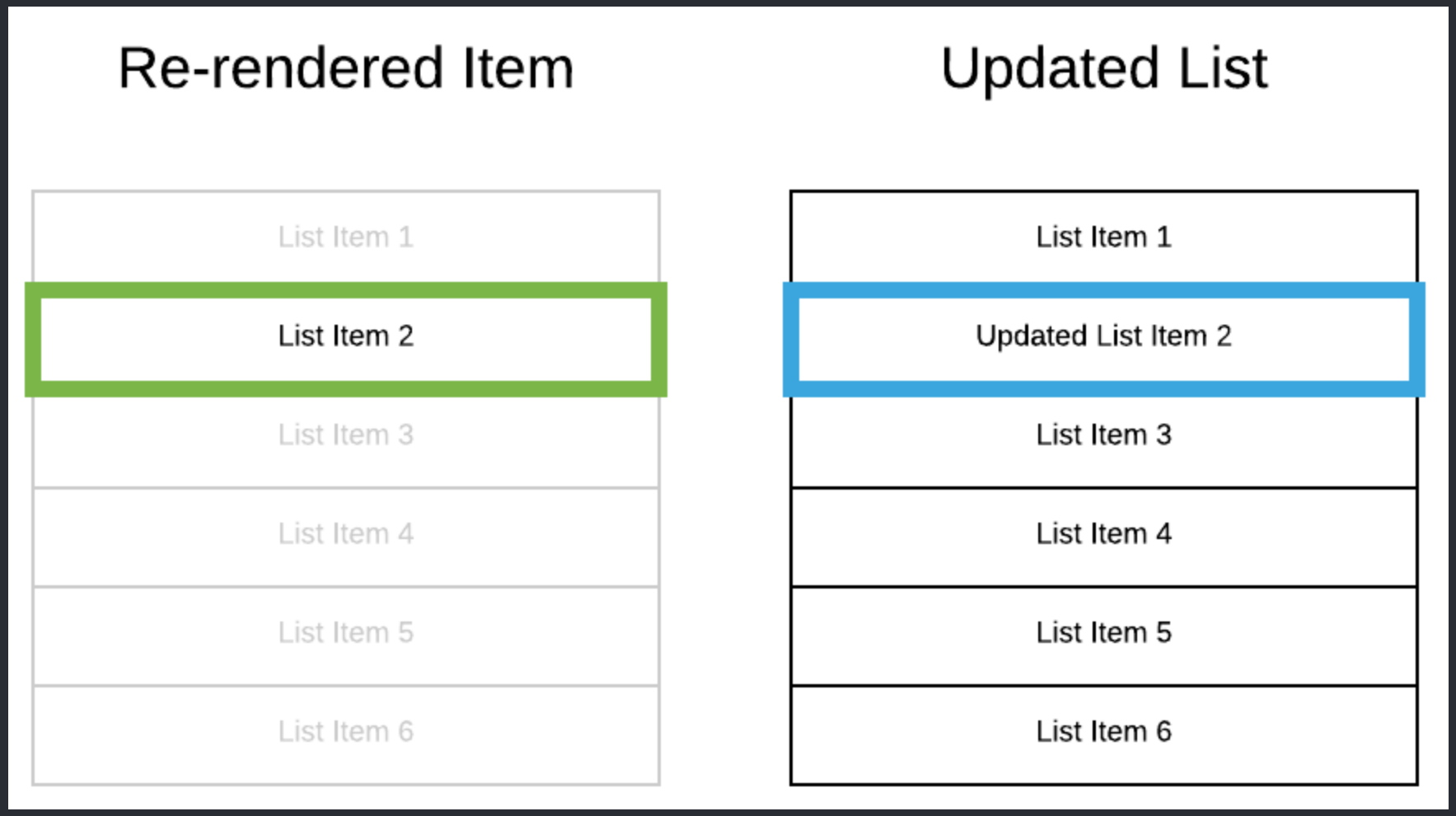
LISTS

- A unique key must be provided to each list element

```
class TodoList extends PureComponent {
  render() {
    const { todos } = this.props;

    return (
      <React.Fragment>
        {todos.map((todo) => {
          <li key={todo.id}>{todo.description}</li>;
        })}
      </React.Fragment>
    );
  }
}
```

LISTS


Re-rendered Item
Updated List
List Item 1
List Item 2
List Item 3
List Item 4
List Item 5
List Item 6
List Item 1
Updated List Item 2
List Item 3
List Item 4
List Item 5
List Item 6

EXCERCISE 5

CREATE A COMPONENT THAT RENDERS A LIST OF MEETINGS

REQUIREMENTS

- use the MeetingList component
- adjust the code to make it work

`GIT CHECKOUT EX5`

EXERCISE 5 SOLUTION

```
GIT CHECKOUT EX5-SOLUTION
```

QUIZ: CONDITIONAL RENDERING AND LISTS

THINKING IN REACT

SINGLE RESPONSIBILITY RULE

EACH COMPONENT SHOULD ONLY DO ONE THING

Search...

Only show products in stock

Name	Price
Sporting Goods	
Football	$49.99
Baseball	$9.99
Basketball	$29.99
Electronics	
iPod Touch	$99.99
iPhone 5	$399.99
Nexus 7	$199.99

Search...

Only show products in stock

Name	Price
Sporting Goods	
Football	$49.99
Baseball	$9.99
Basketball	$29.99
Electronics	
iPod Touch	$99.99
iPhone 5	$399.99
Nexus 7	$199.99

COMPONENT HIERARCHY

```
> FilterableProductTable
  > SearchBar
  > ProductTable
    > ProductCategoryRow
    > ProductRow
```

EXERCISE 6

SPLIT UP THE MEETINGLIST COMPONENT

`GIT CHECKOUT EX6`

EXERCISE 6 SOLUTION

```
GIT CHECKOUT EX6-SOLUTION
```

EXERCISE 7

USE AXIOS TO GET DATA FROM OUR LOCAL SERVER

- endpoint: http://localhost:3002/meetings
- axios documentation: https://axios-http.com/docs/example

`GIT CHECKOUT EX7`

EXCERCISE 7 SOLUTION

```
GIT CHECKOUT EX7-SOLUTION
```

ROUTING WITH REACT ROUTER

```jsx
import ReactDOM from "react-dom/client";
import {
  BrowserRouter,
  Routes,
  Route,
} from "react-router-dom";
// import your route components too

const root = ReactDOM.createRoot(
  document.getElementById("root")
);
root.render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />}>
        <Route index element={<Home />} />
        <Route path="teams" element={<Teams />}>
          <Route path=":teamId" element={<Team />} />
          <Route path="new" element={<NewTeamForm />} />
          <Route index element={<LeagueStandings />} />
        </Route>
      </Route>
    </Routes>
  </BrowserRouter>
);
```

EXERCISE 8

ADD ROUTING TO OUR PROJECT

- /meetings (MeetingOverview component)
- / should redirect to /meetings
- Unknown routes should show the `NotFound` component

HTTPS://REACTROUTER.COM/EN/MAIN/GETTING-STARTED/OVERVIEW

GIT CHECKOUT EX8

EXERCISE 8 SOLUTION

```
GIT CHECKOUT EX8-SOLUTION
```

CONTEXT

PROVIDES A WAY TO PASS DATA THROUGH THE COMPONENT TREE WITHOUT HAVING TO PASS PROPS DOWN AT EVERY LEVEL

WHY?

PREVENT PROP-DRILLING

```
const App = ({theme, ...rest}) => {
  return (
    <Layout theme={theme} />
  )
}

const Layout = ({props}) => {
  return (
    <CardList theme={props.theme} />
  )
}

const CardList = ({props}) => {
  return (
    <Card theme{props.theme} />
  )
}
```

WITH CONTEXT

```
const ThemeContext = React.createContext("light");

function Layout() {
  return (
    <ThemeContext.Consumer>{(theme) => <CardList />}</ThemeContext.Consumer>
  );
}

function App({ theme, ...rest }) {
  return (
    <ThemeContext.Provider value="dark">
      <Layout theme={theme} />
    </ThemeContext.Provider>
  );
}
```

EXERCISE 9

BE ABLE TO SWITCH BETWEEN USERS IN THE APPBAR THROUGH CONTEXT

REQUIREMENTS

- use a provider which keeps the state of the current user
- use the `useContext` hook
- show a `create meeting` button in the meeting overview if the current user is `employee`

EXERCISE 9 SOLUTION

```
GIT CHECKOUT EX9-SOLUTION
```

QUIZ: CONTEXT

PERFORMANCE TIPS

MOST OPTMIZATIONS ARE PREMATURE

1. USING ARROW FUNCTIONS DIRECTLY AS PROPS

- Creates a new reference on each render
- Unnecessary re-render of child components 😭

```
<button onClick={(e) => handleClick(e)}>Click me</button>
```

1. USING ARROW FUNCTIONS DIRECTLY AS PROPS

- Instead declare the function and memoize it using useMemo

```
const Container = () => {
  const handleClick = useMemo(() => {
    ...
  }, [])

  return (
    <button onClick={handleClick}>
      Click me
    </button>
  )
}
```

2. BE CAREFUL WHEN DERIVING DATA

- New reference on each render
- Unnecessary re-render of child components

```
const Container = ({posts}) => {
  const topTen = posts.sort((a, b) => b.likes - a.likes).slice(0, 9)

  return ...
}
```

2. BE CAREFUL WHEN DERIVING DATA

MDN DOCS FOR SLICE METHOD

Return value

A new array containing the extracted elements.

2. DO NOT DERIVE DATA IN RENDER METHOD

```
import { useMemo } from "react";

const Container = ({ posts }) => {
  const topTen = useMemo(
    (posts) => {
      return posts.sort((a, b) => b.likes - a.likes).slice(0, 9);
    },
    [posts]
  );
};
```

3. REACT.MEMO

```
const Container = ({posts}) => {
  ...
}

export default React.memo(Container)
```

CHECKS FOR PROP CHANGES

QUIZ: PERFORMANCE OPTIMIZATIONS

STYLING

STYLING

CLASSNAMES CAN BE ADDED THROUGH A **CLASSNAME** PROP

```
const Container = () => {
  return <span className='menu--active'>Menu</span>
}
```

INLINE STYLES

- Pass as object
- CSS properties are camelcase
- Generally not recommended

```
const Container = () => {
  return <span style={{backgroundColor: 'tomato'}}>Menu</span>
}
```

CSS MODULES

- CSS files in which all classnames and animation names are scoped locally by default
- Uses a custom webpack loader

CSS MODULES

```css
.button {
  background-color: tomato;
}
```

```jsx
import styles from './style.css'
import React from 'react'

const Container = () => {
  return <button className={styles.button}>My awesome button</button>
}
```

CSS-IN-JS

```jsx
import React from 'react';
import { makeStyles } from '@material-ui/core/styles';
import Button from '@material-ui/core/Button';

const useStyles = makeStyles({
  root: {
    background: 'linear-gradient(45deg, #FE6B8B 30%, #FF8E53 90%)',
    border: 0,
    borderRadius: 3,
    boxShadow: '0 3px 5px 2px rgba(255, 105, 135, .3)',
    color: 'white',
    height: 48,
    padding: '0 30px',
  },
});

export default function Hook() {
  const classes = useStyles();
  return <Button className={classes.root}>Hook</Button>;
}
```

CSS-IN-JS

- Pattern where CSS is composed using JavaScript instead of defined in css files

https://github.com/MicheleBertoli/css-in-js

CSS-IN-JS

- emotion
- styled-components
- ...

EXERCISE 10

ADD FUNCTIONALITY SO WE CAN CREATE A MEETING

REQUIREMENTS

- create a new meeting page
- use react hook form (https://react-hook-form.com/get-started)
- make page available on `/meetings/new`
- use the create meeting button to navigate
- use a POST to `http://localhost:3002/meetings` to submit the form

EXERCISE 10 SOLUTION

EXERCISE 11

ADD FUNCTIONALITY SO WE CAN VIEW A MEETING DETAIL

REQUIREMENTS

- add a status to your mock data (db.json)
- create a meeting detail page
- navigate to `/meetings/{id}` by clicking on the card
- get the meeting id from your url params
- get the meeting details by id like this `http://localhost:3002/meetings/7`
- only an admin can click on these cards = use the useUser hook

EXERCISE 11 SOLUTION

WEB COMPONENTS

A SET BROWSER STANDARDS THAT ALLOWS US, DEVELOPERS, TO BUILD NATIVE BROWSER COMPONENTS

WEB COMPONENTS

- HTML templates
- shadow DOM
- custom elements

CAN THEY BE USED IN REACT?

YES!

WHAT ABOUT SHADOW DOM ENCAPSULATION?

REACT 18 WILL ALLOW BETTER SUPPORT

Q&A

Your feedback is appreciated!

THANKS!