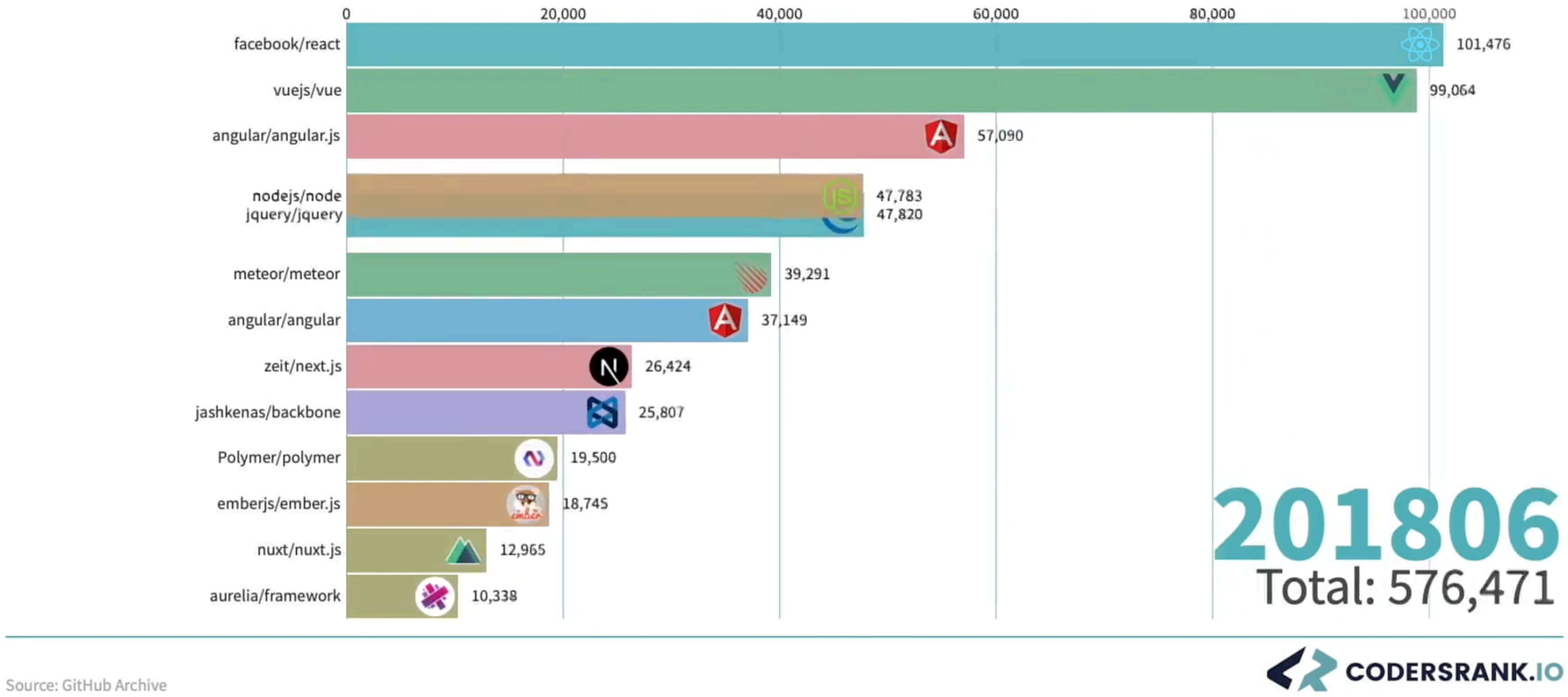
REMINDER 🙋


Most popular JavaScript frameworks
2012 - 2019
0
20,000
40,000
60,000
80,000
100,000
facebook/react
101,476
vuejs/vue
99,064
angular/angular.js
57,090
nodejs/node
47,783
jquery/jquery
47,820
meteor/meteor
39,291
angular/angular
37,149
zeit/next.js
26,424
jashkenas/backbone
25,807
Polymer/polymer
19,500
emberjs/ember.js
18,745
nuxt/nuxt.js
12,965
aurelia/framework
10,338
201806
Total: 576,471
Source: GitHub Archive
CODERSRANK.IO

React, Angular, Vue.js, jQuery,

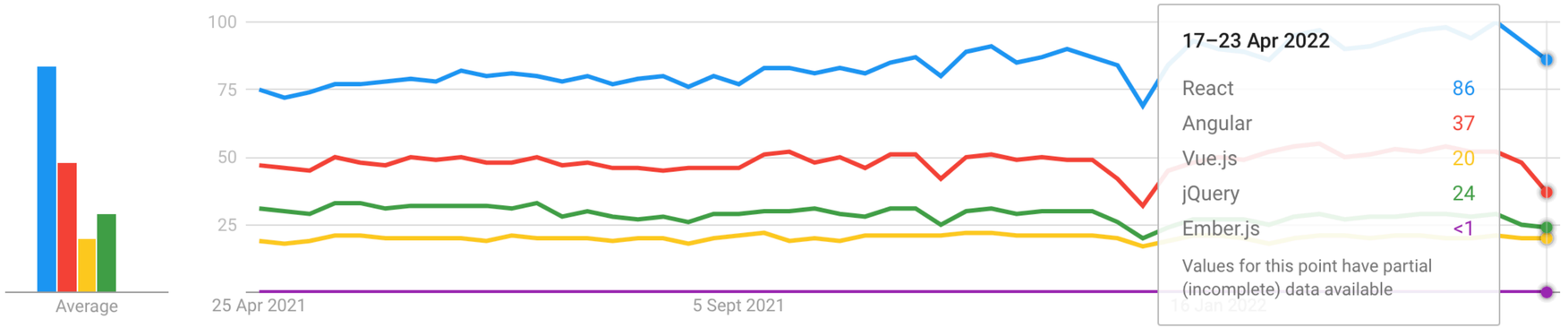
https://trends.google.com/trends/explore?q=%2Fm%2F012l1vxv,%2Fg%2F11c6w0ddw9,%2Fg%2F11c0vmgx5d,%2Fm%2F0268g...

Google Trends | Compare

- React — JavaScript library
- Angular — Web framework
- Vue.js — Topic
- jQuery — Software
- Ember.js — Topic

Worldwide | Past 12 months | All categories | Web Search

Interest over time



React & Async JS

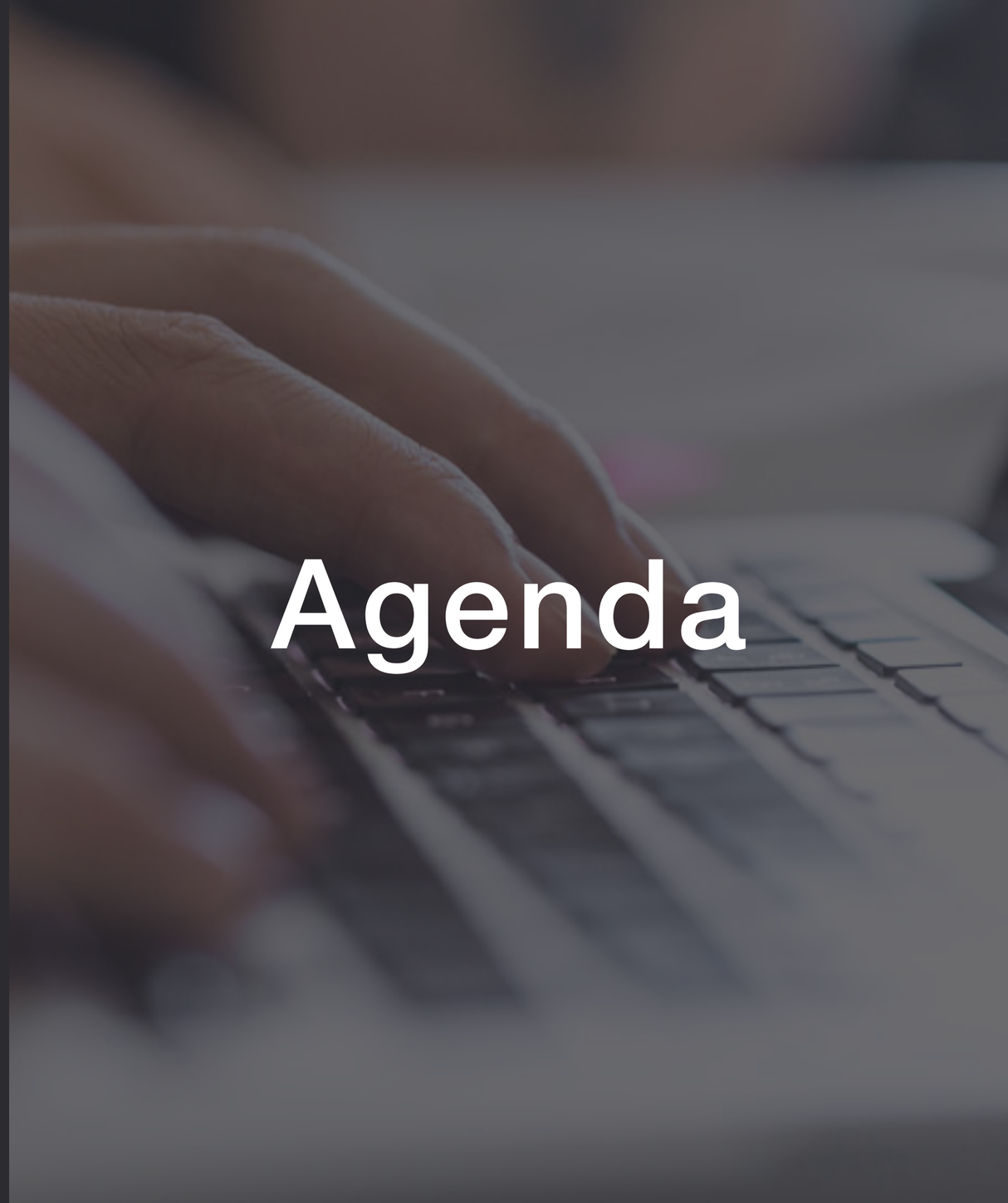
Frontend Programming

Purpose

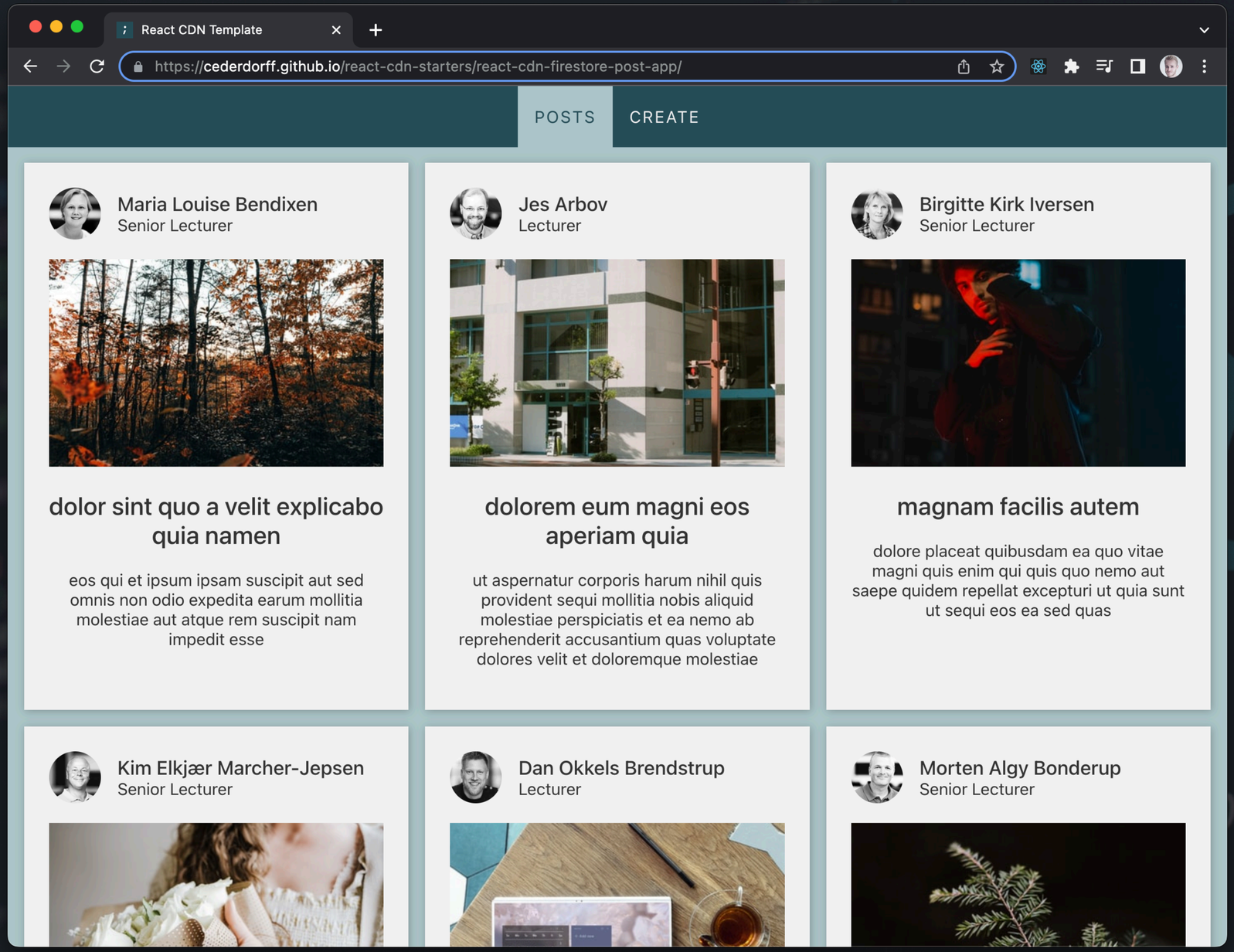
- Gain a deeper understanding of React Core Concepts.
- Improve your knowledge about fetch & Async JS.
- Single Page Apps & Routing

Agenda

- React Core Concepts
- React Hooks
- Async JS in React
- CLI, Build Tools & create-react-app
- React Router & SPA



- Tryout useState
- React Fetch Posts
- Download & install Node.js & npm
- Generate create-react-app
- Create React SPA



cederdorff.github.io/react-cdn-starters/react-cdn-firestore-post-app/

Make sure you have a basic (JS) knowledge ...

HTML & CSS Basics, CSS Layouts, Grids & Flexbox, Selectors
Variables, Objects, Arrays, Loops, Functions, DOM-Manipulation, Template String, ES6+ Features, fetch, API, JSON, Web Apps, Single Page Apps, Routing, CRUD, Forms & Events

Computer science student

Senior developer, 10+ years experience

https://www.instagram.com/p/BxWAgatgSmn/

I’m not kidding! It’s an important skill!

linkedin.com/posts/javascript-developer_activity-6922443466585088001-z1Fl/

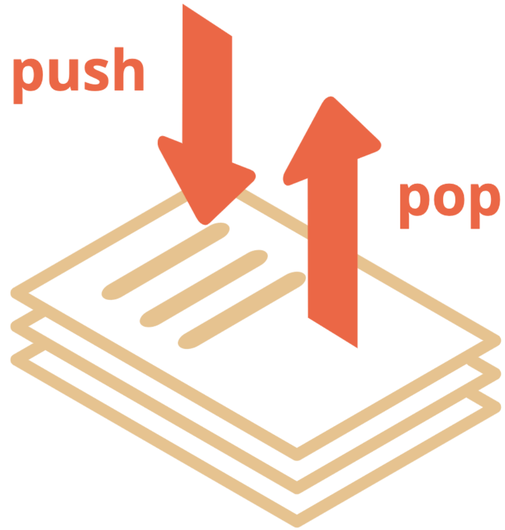
OBJECTS

Objects are used to store keyed collections of various data

Containers for named values called properties. A property is a “`key: value`” pair

ARRAYS

Arrays are ordered collections



An array is a way to hold more than one value at a time we have a 1st, a 2nd, a 3rd, a 4th element and so on.

Array Methods to know

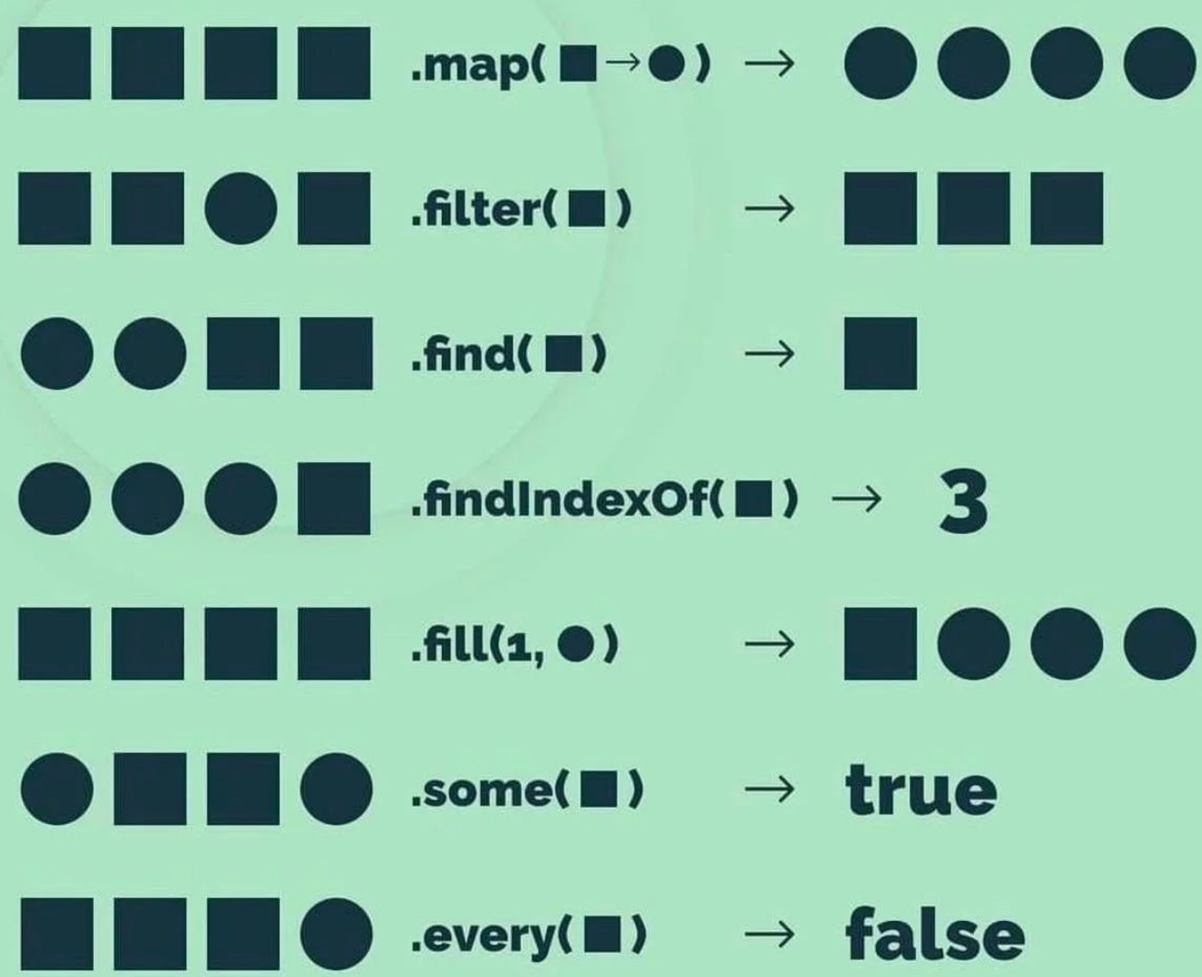
.push(…)

.map(…)

.filter(…)

.find(…)

.sort(…)



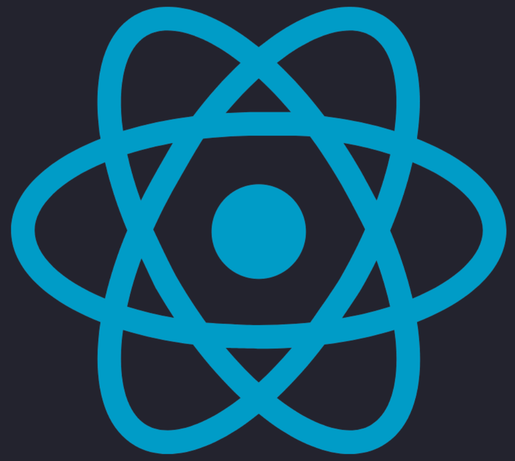
https://javascript.info/array-methods
https://medium.com/@mandeepkaur1/a-list-of-javascript-array-methods-145d09dd19a0

Array.map(...)

...iterate over an array and modify each element.
Array.map(...) calls a callback function for each element in the array.

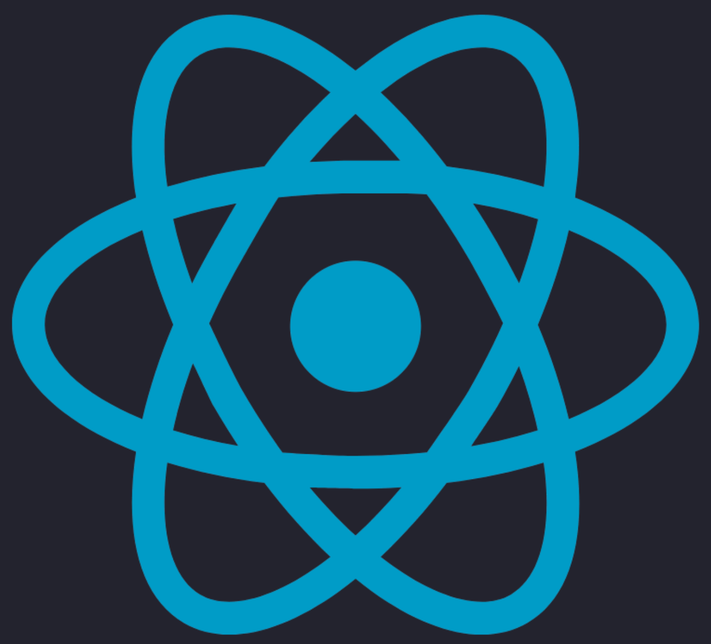
```
const persons = [
    { firstname: "Birgitte", lastname: "Iversen" },
    { firstname: "Lykke", lastname: "Dahlen" },
    { firstname: "Rasmus", lastname: "Cederdorff" }
];

const mapped = persons.map(person => {
    return {
        name: `${person.firstname} ${person.lastname}`
    };
});

console.log(mapped);
```

```
▼ (3) [{…}, {…}, {…}] i
  ▶ 0: {name: 'Birgitte Iversen'}
  ▶ 1: {name: 'Lykke Dahlen'}
  ▶ 2: {name: 'Rasmus Cederdorff'}
    length: 3
```



React

Core Concepts



What is React?

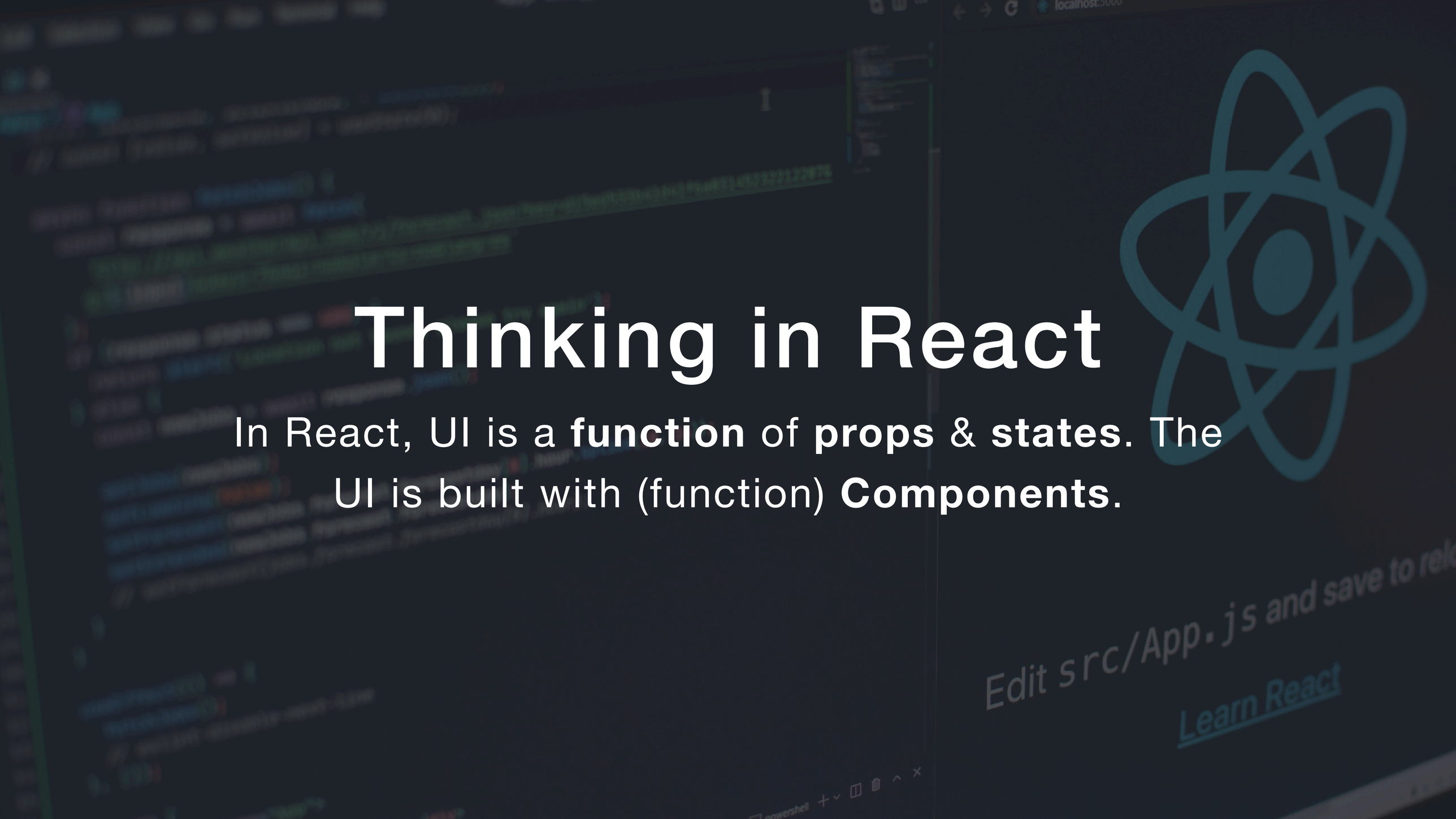
- A JavaScript library for building User Interfaces.
- Allow us to create reusable UI Components.
- Single Page Apps

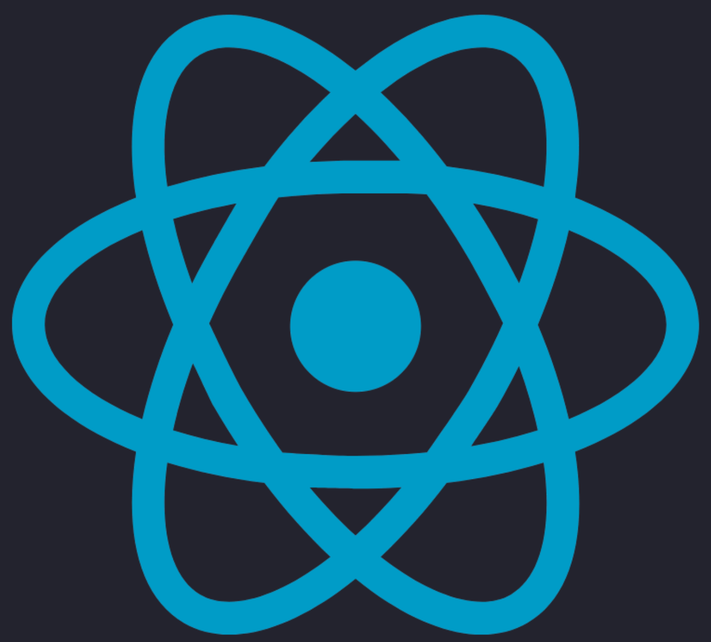
Components, Props & States

The Core Concepts of React

Thinking in React

In React, UI is a **function** of **props** & **states**. The UI is built with (function) **Components**.

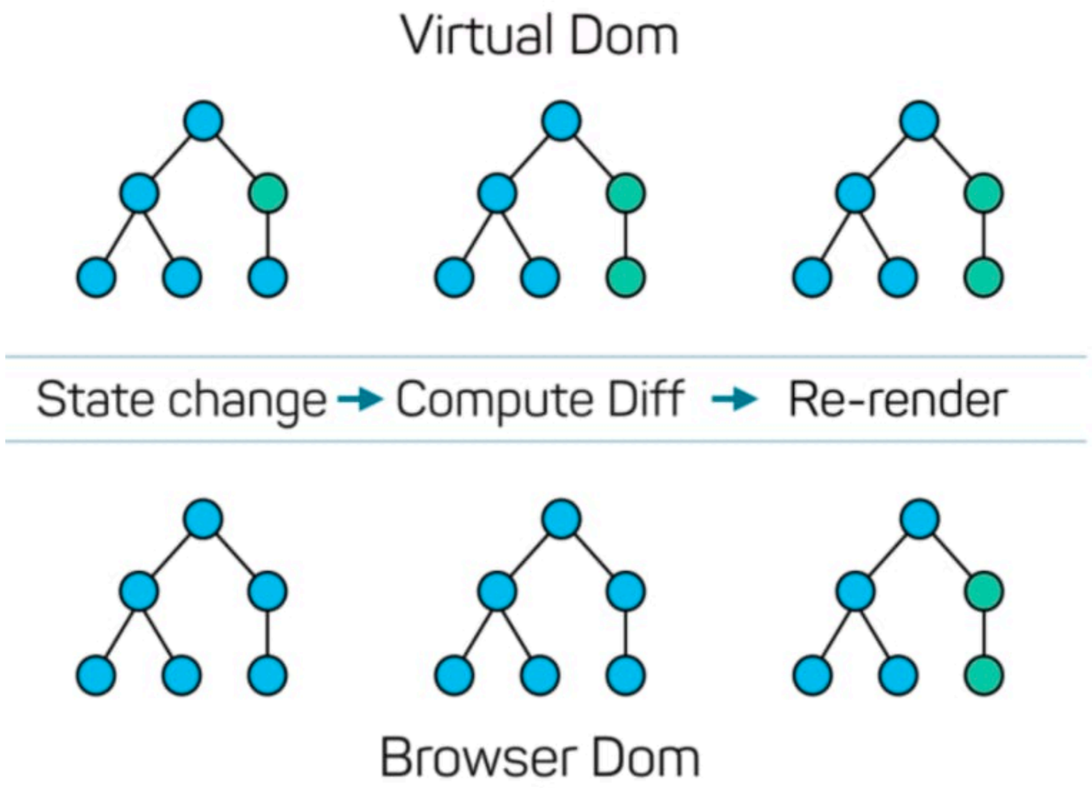




But how?

- React creates a Virtual DOM instead of manipulating directly with the browser's DOM.
- React makes the changes in Virtual DOM and compares it to the browser DOM.
- Then React detects what’s needs to be changed in the browser DOM and changes **only** what needs to be changed!

React Virtual DOM

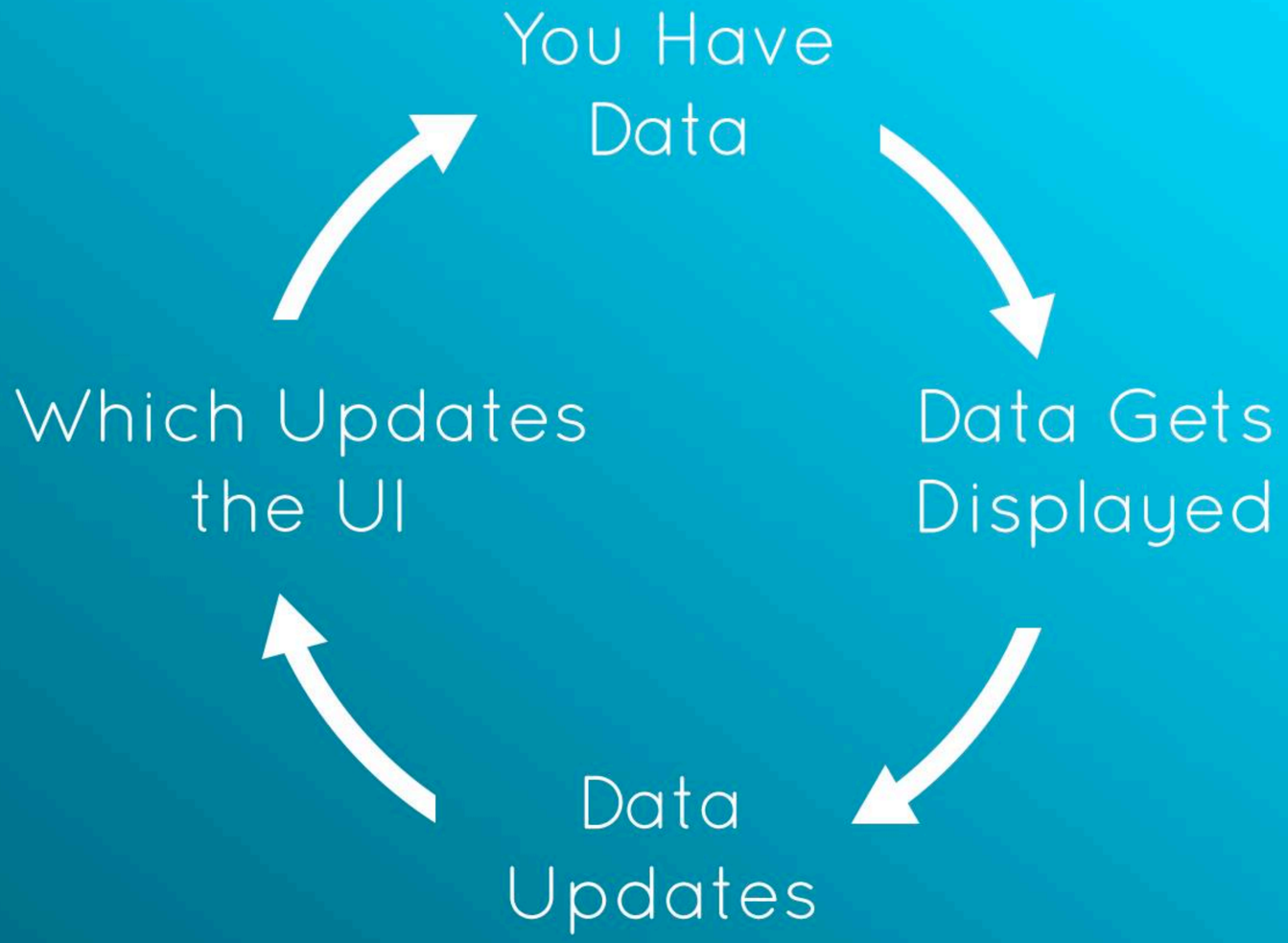


“The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM.”

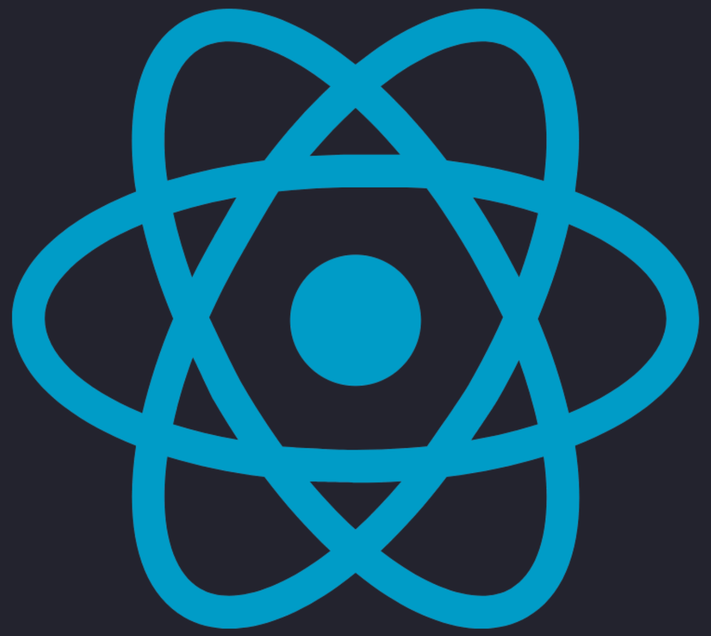
“You tell React what state you want the UI to be in, and it makes sure the DOM matches that state. This abstracts out the attribute manipulation, event handling, and manual DOM updating […]”.

https://reactjs.org/docs/faq-internals.html

The React UI Cycle



This happens AUTOMATICALLY once you define components

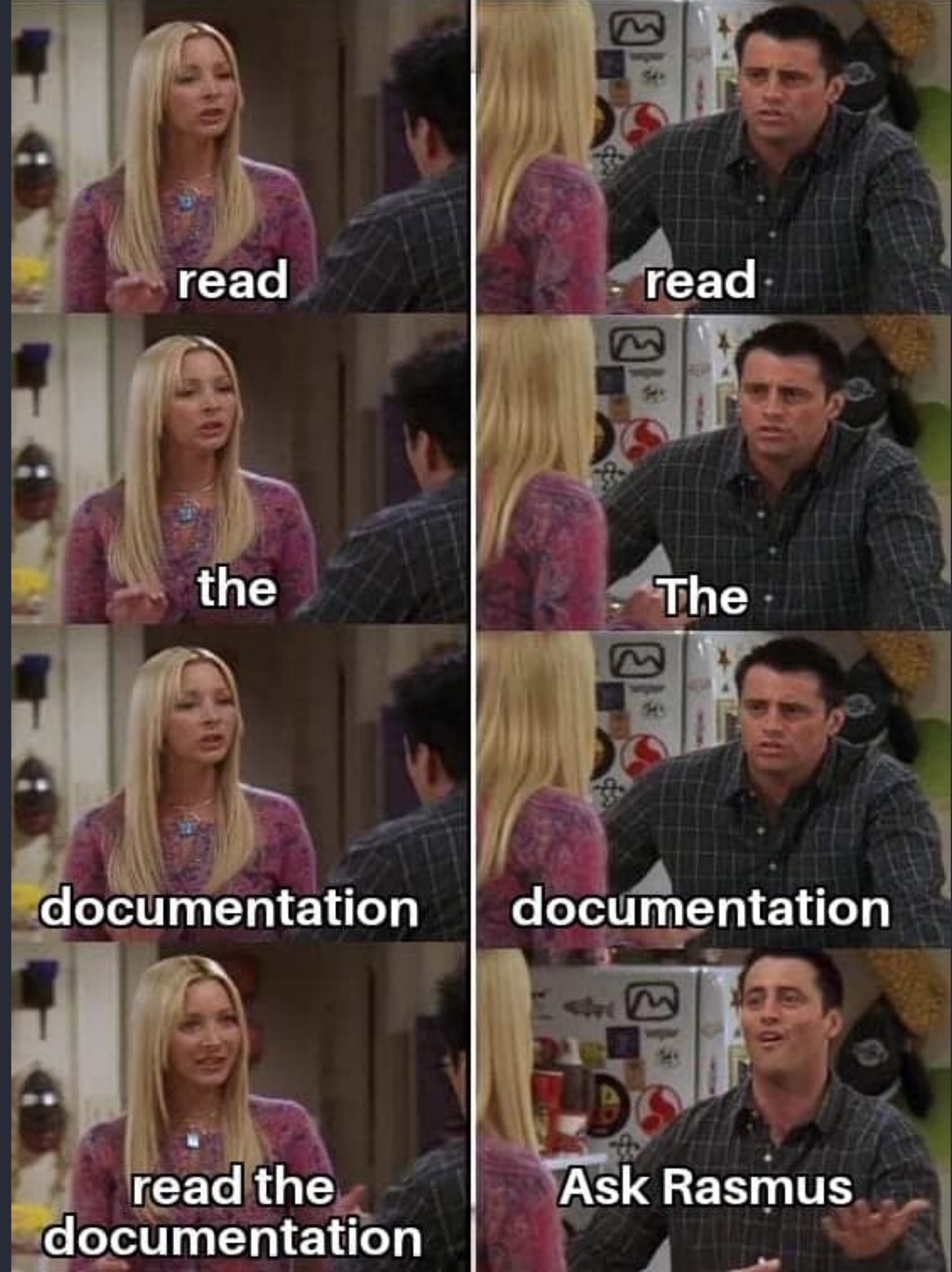


But why?

- It’s so hard to work with the DOM and DOM manipulation is slow.
- You must make sure the DOM is constantly updated and synchronised with the state of your data.
- React *reacts* to **changes** and then updates the DOM automatically and performant without rerendering all elements - only the needed elements.

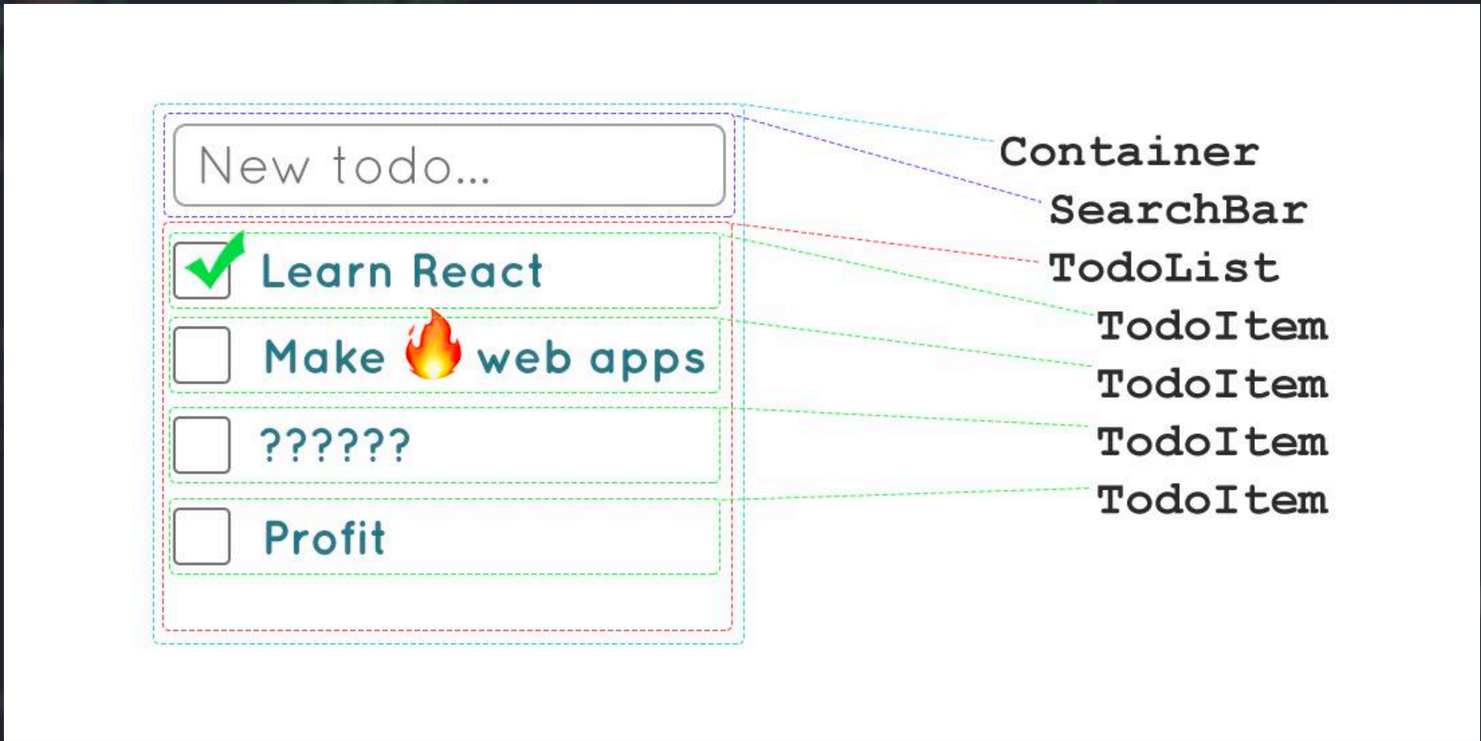
But RACE, how do you know all that stuff?

- React Resources
- React Self-study
- Guides & Tutorials
- From Vanilla JavaScript to React Developer



How to Think in React

Review slides & note questions



Function Component

A Component is simply just a function that returns HTML (JSX).

Components are independent blocks of code.

Components MUST return a single element. We can wrap in a parent element or use <></>.

```
function MyComponent() {
    const name = "RACE";

    return (
        <section>
            <h1>Hello, {name}</h1>
        </section>
    );
}
```

JSX

Java**S**cript **X**ML

- Allow us to write HTML in JavaScript.
- Makes it easier to write and add HTML in React.
- We can create our own tags.
- We MUST close all tags.

```
// With JSX

const myelement = <h1>I Love JSX!</h1>;

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

```
// Without JSX

const myelement = React.createElement("h1", {}, "I do not use JSX!");

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

Function Component

https://twitter.com/chrisachard/status/1175022111758442497

Function Component

```
function Greeting() {
  return (
    <div>
      <h1>Hello, React!</h1>
    </div>
  );
}
```

Define the component “Greeting”

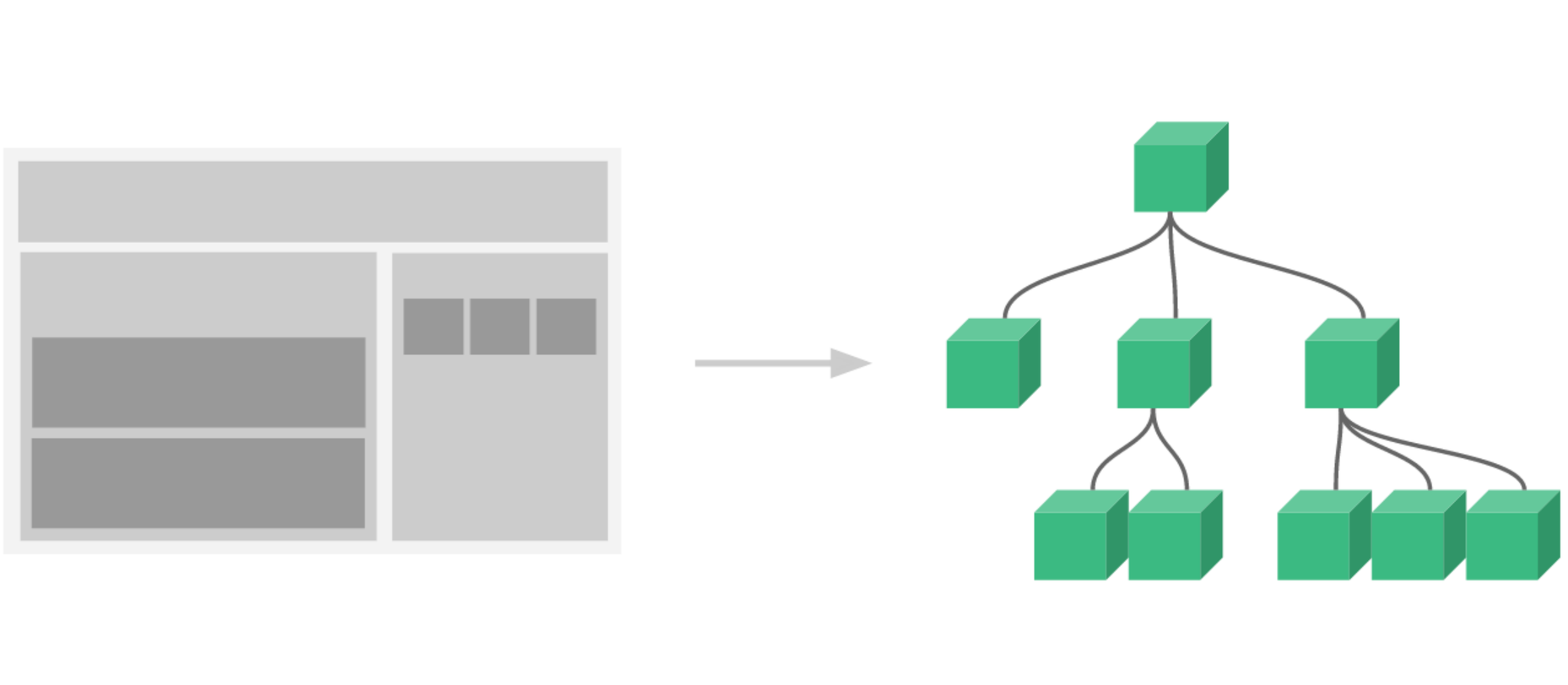
```
function App() {
  return (
    <div>
      <Greeting />
      <div>
        <Greeting />
      </div>
    </div>
  );
}
```

Use Greeting component in another component

Components without a closing tag REQUIRE a closing slash

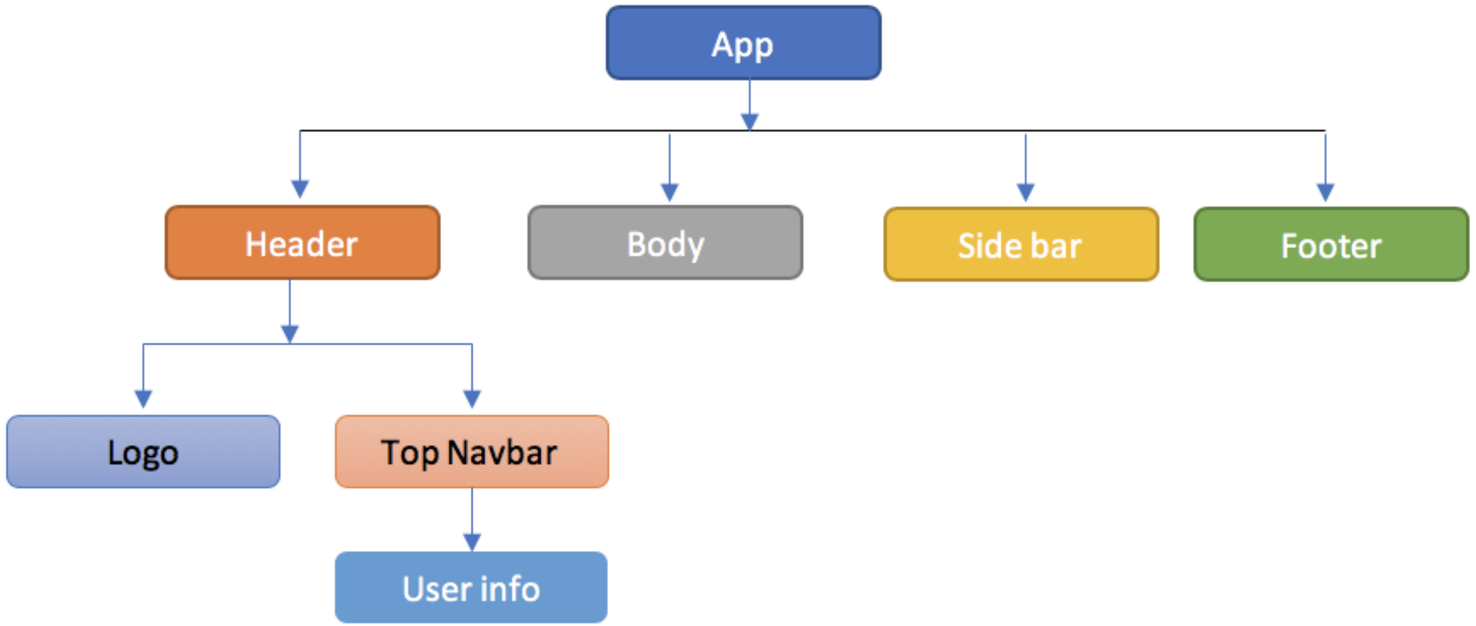
https://twitter.com/chrisachard/status/1175022111758442497

Components, Components, Components



Build Reusable Components

UI can be broken down to small block or modules called components

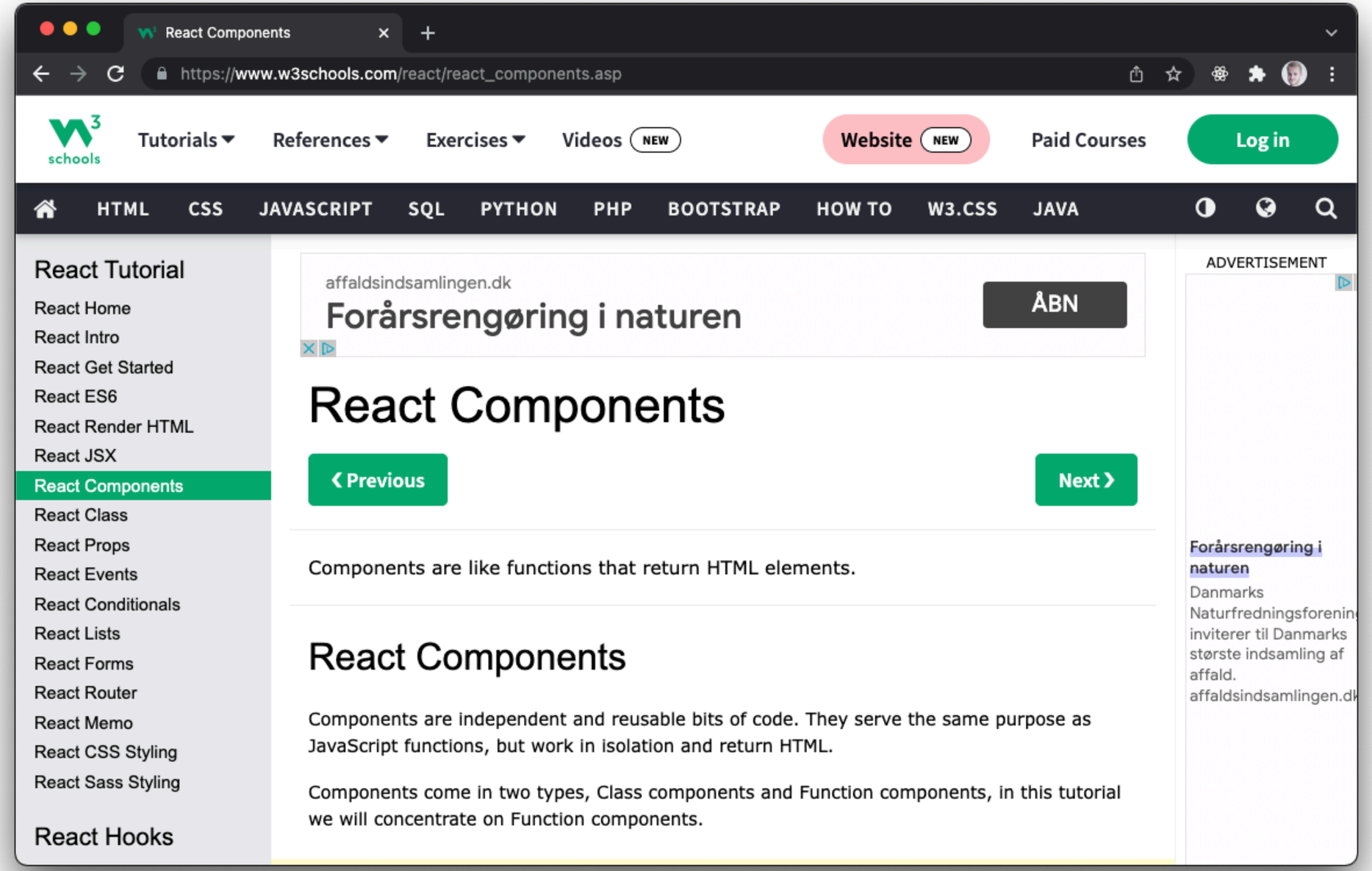


It’s all Components!

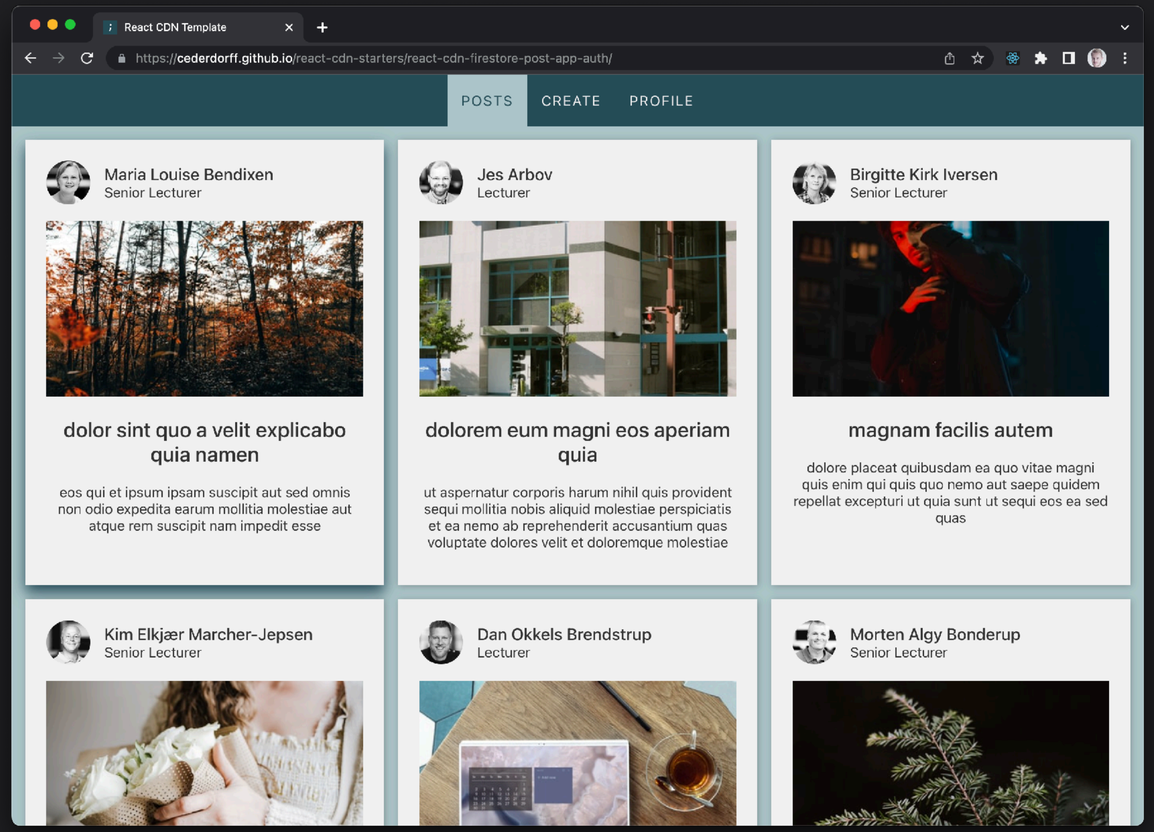
```
function Navigation() {
    return (
        <nav>
            <a href="#">Home</a>
        </nav>
    );
}

function Header() {
    return (
        <header>
            <h1>React Page Template</h1>
        </header>
    );
}

function PageContent() {
    return (
        <section>
            <h2>Here goes some page content</h2>
        </section>
    );
}

function Footer() {
    return (
        <footer>
            <p>This is my Footer</p>
        </footer>
    );
}
```

```
function App() {
    return (
        <>
            <Navigation />
            <Header />
            <PageContent />
            <Footer />
        </>
    );
}

ReactDOM.render(<App />, document.getElementById("root"));
```

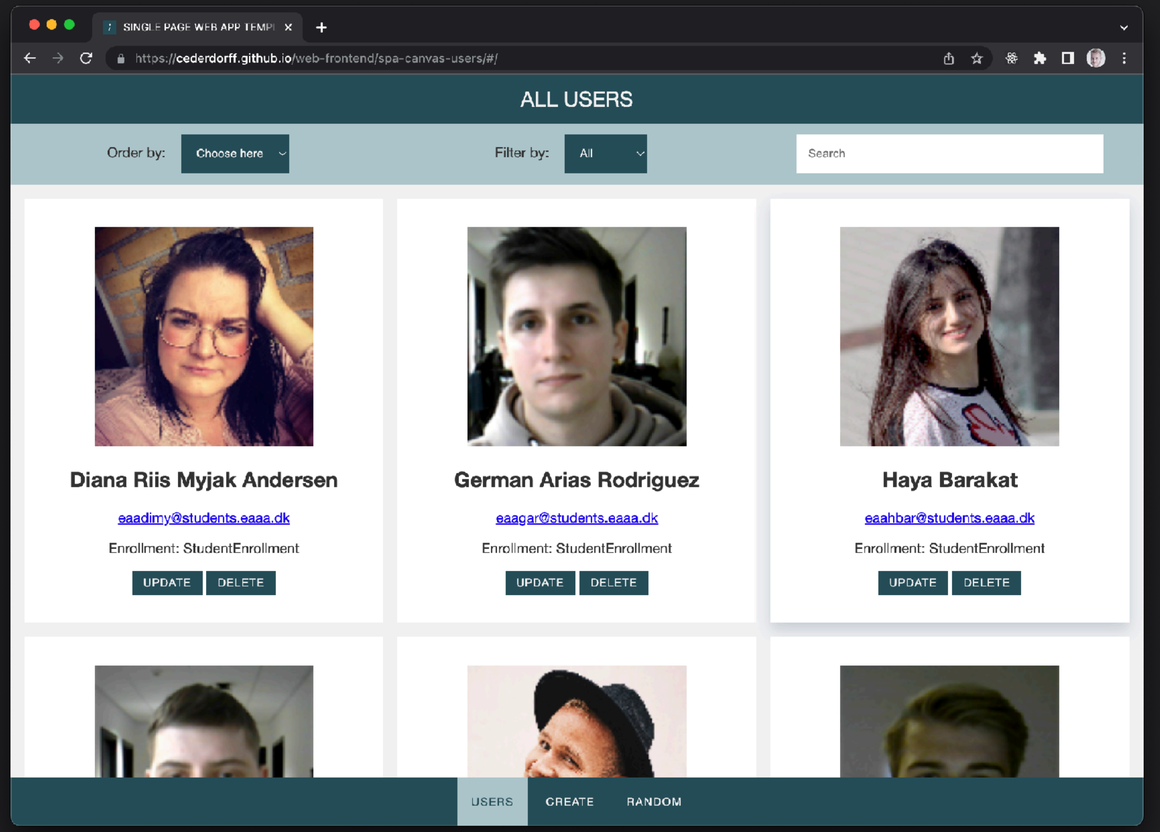
react-cdn-page-template


React Components
https://www.w3schools.com/react/react_components.asp
Tutorials
References
Exercises
Videos NEW
Website NEW
Paid Courses
Log in
HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP HOW TO W3.CSS JAVA
React Tutorial
React Home
React Intro
React Get Started
React ES6
React Render HTML
React JSX
React Components
React Class
React Props
React Events
React Conditionals
React Lists
React Forms
React Router
React Memo
React CSS Styling
React Sass Styling
React Hooks
affaldsindsamlingen.dk
Forårsrengøring i naturen
ÅBN
ADVERTISEMENT
React Components
Previous
Next
Components are like functions that return HTML elements.
React Components
Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML.
Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.
Forårsrengøring i naturen
Danmarks Naturfredningsforening inviterer til Danmarks største indsamling af affald.
affaldsindsamlingen.dk

Components?



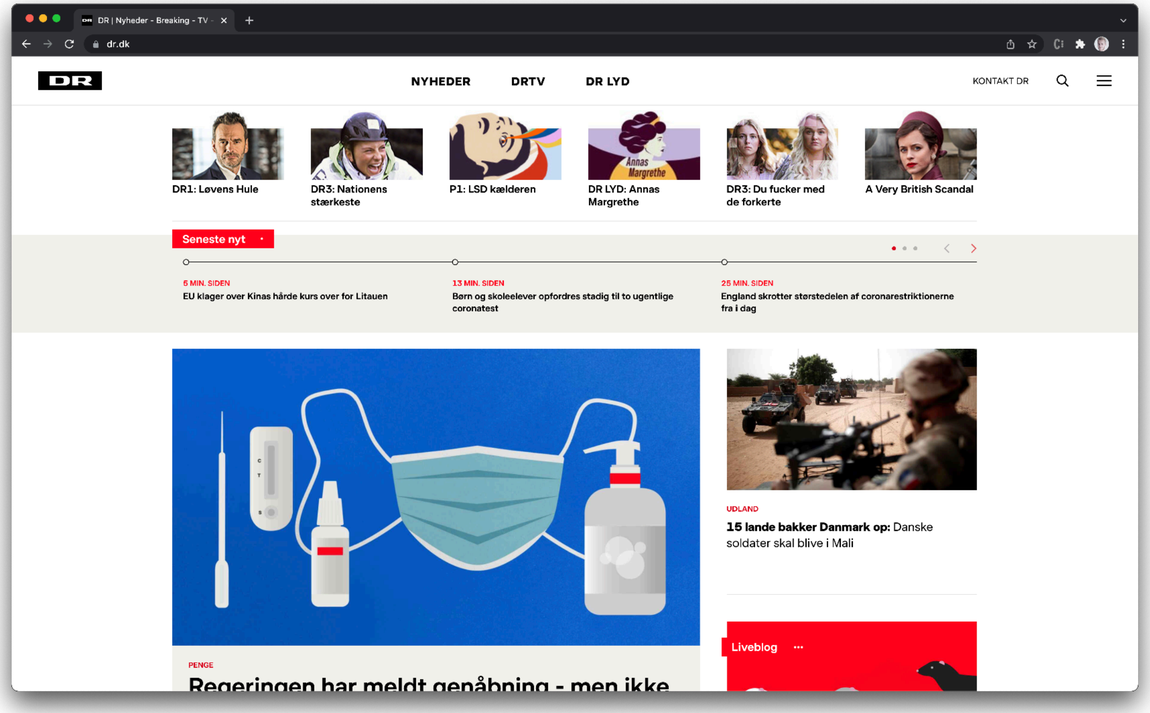
react-cdn-firestore-post-app-auth

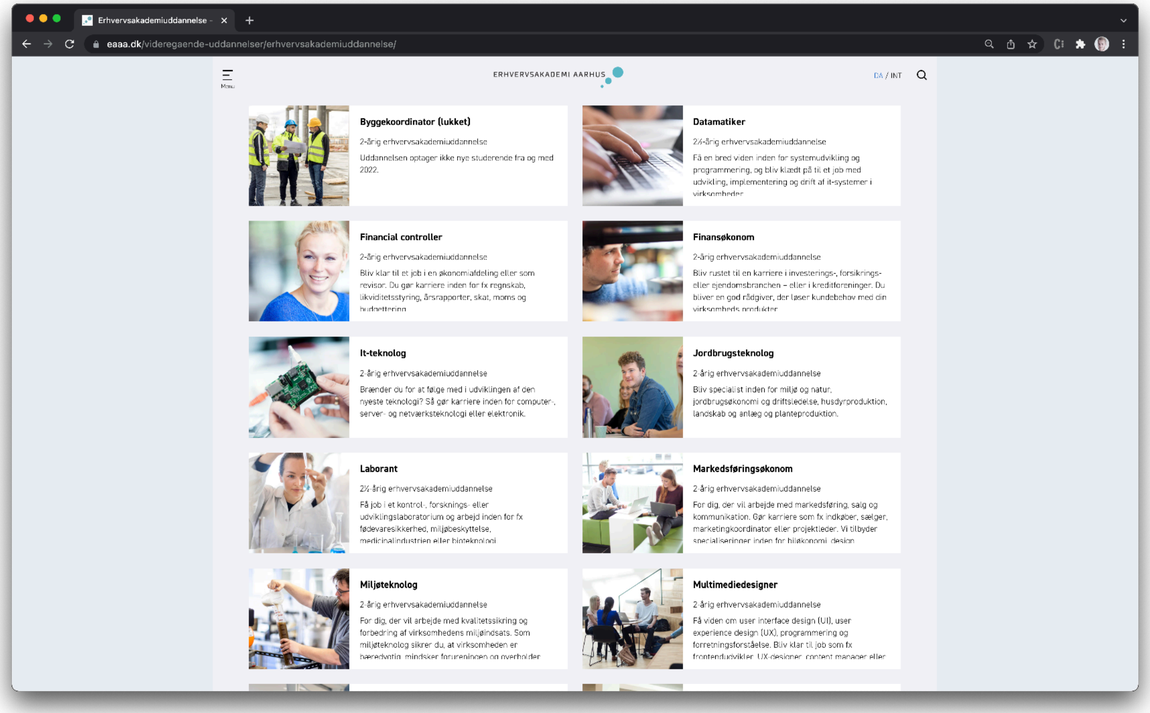


Template: `spa-canvas-users-template`

Solution: `spa-canvas-users`

And?

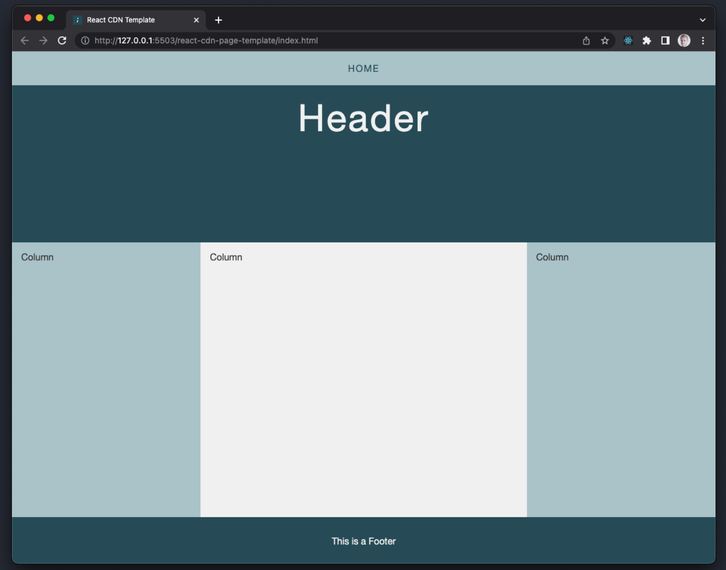




... but we thought it was all objects and array?

Component Page Layout

1. Use react-cdn-template
2. Implement a page layout and create the following components: `Navigation`, `Header`, `MainContent` & `Footer`
3. Add some placeholder content in every component like headlines and/or text. `MainContent` should consist of 3 columns (use the CSS classes `left`, `middle` & `right`).
4. Make sure to render all components in the `App` component.
5. Style your page layout using a grid, flexbox or use the existing CSS styling from app.css (`.page-layout`, `nav`, `header`, `footer`, `left`, `middle` & `right`)




```
function App() {
    return (
        <main className="page-layout">
            <Navigation />
            <Header />
            <MainContent />
            <Footer />
        </main>
    );
}

ReactDOM.render(<App />, document.getElementById("root"));
```

Props

... arguments passed into a React Component. Props are passed to components via HTML attributes.

```
function Greeting({ name }) {
    return <h1>Hello, {name}</h1>;
}

function App() {
    return <Greeting name="world" />;
}
```

Function Component with Props

```jsx
function Greeting(props) {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
    </div>
  );
}

function App() {
  return (
    <div>
      <Greeting name="React" />
      <div>
        <Greeting name="Chris" />
      </div>
    </div>
  );
}
```

props is first argument to function components

Access props inside of curly braces to show value

Set a prop on a child component by passing an attribute

Different instances of component can have different prop values

https://twitter.com/chrisachard/status/1175022111758442497

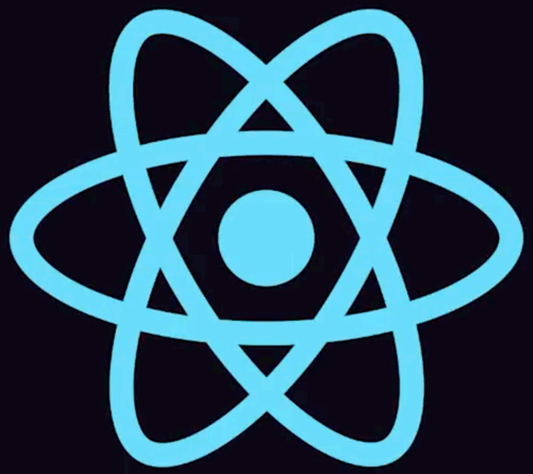
Add Props

1. Use your implemented Page Layout or `react-cdn-page-template`.
2. Add a prop to your `Header` component called `title`. Make sure the component (function) takes in an argument. The `title` prop must be rendered in the JSX inside the `h1`.
3. Locate the `Header` component in `App` and add the attribute `title` and pass in a `title` (“My Title”, “Your Name” etc.).
4. Test your solution and try to change the `title` attribute value.
5. Implement and try out props for some of the other components in your Page Layout.

```
function Greeting({ name }) {
    return <h1>Hello, {name}</h1>;
}

function App() {
    return <Greeting name="world" />;
}
```

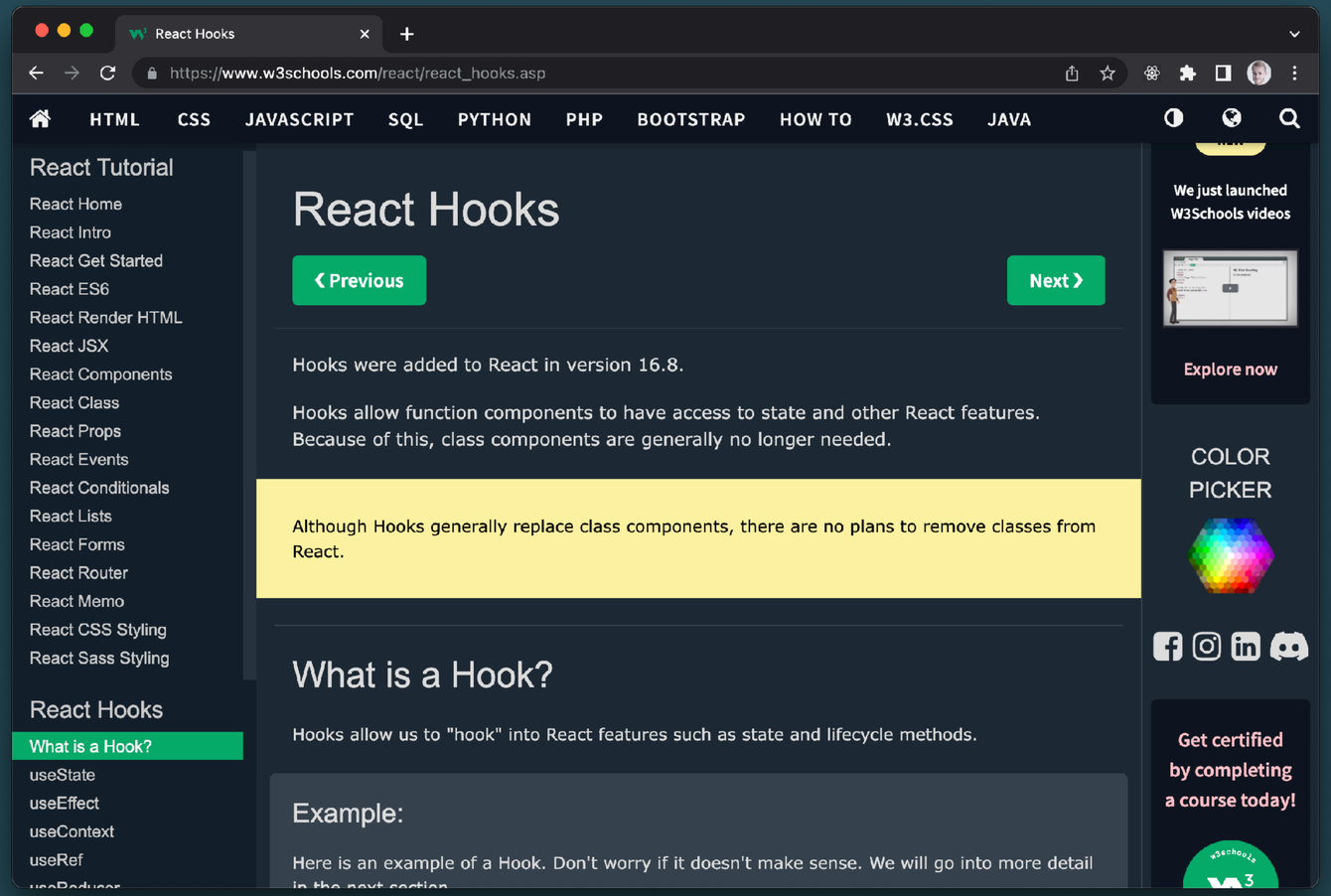
React Hooks

useState & useEffect



https://www.youtube.com/watch?v=TNhaISOUy6Q

React Hooks



w3schools.com/react/react_hooks.asp

https://www.youtube.com/watch?v=TNhaISOUy6Q

useState

... a hook that allows us to track states in a function component. The purpose is to handle reactive data. When data in a state change, React reacts and re-renders the UI.

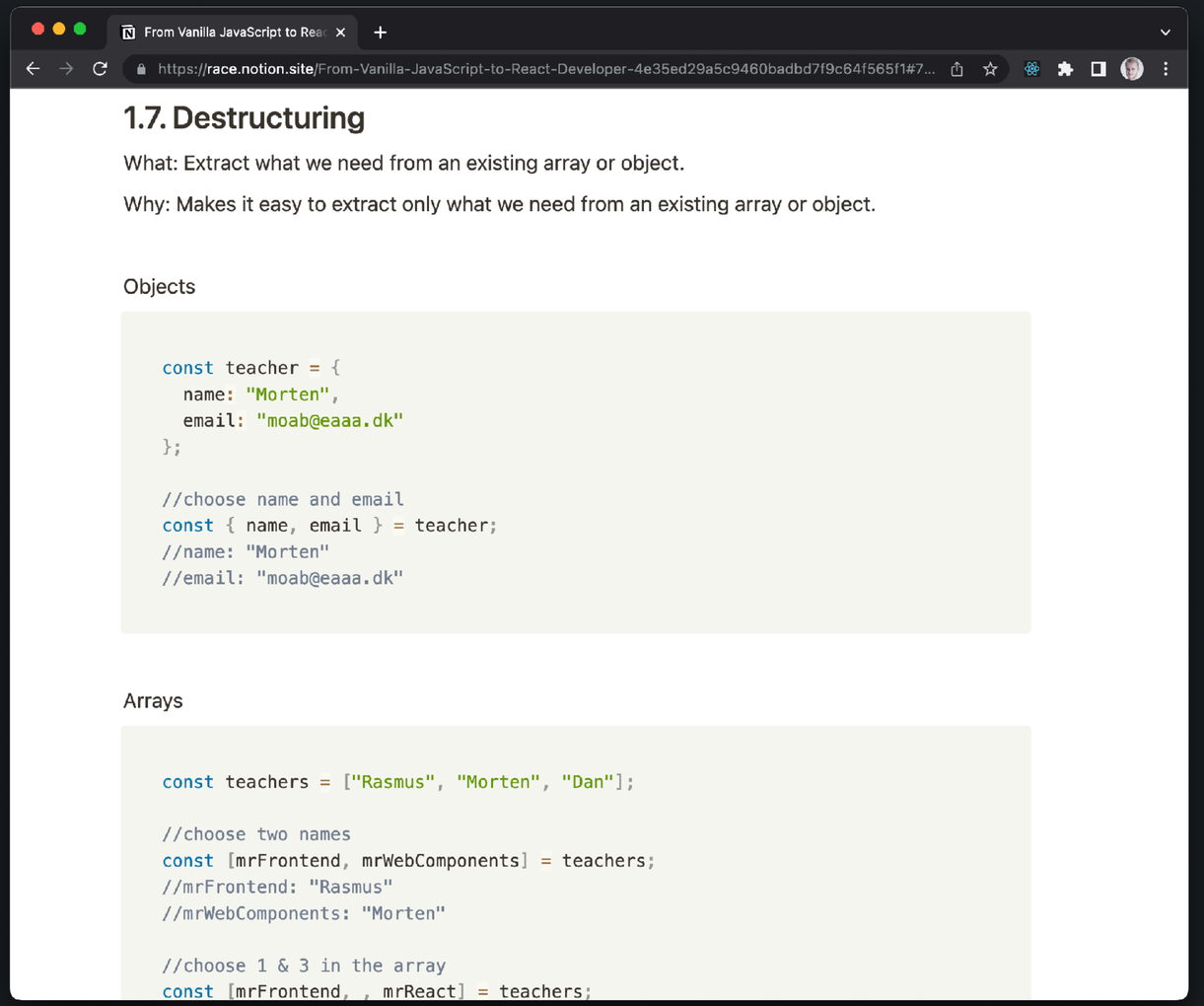
useState(...)

... a hook that allows us to track states in a function component.

The purpose is to handle reactive data. When data in a state change, React reacts and re-renders the UI.

https://twitter.com/chrisachard/status/1175022111758442497

Destructuring



Destructuring

When States & Props change, your components automatically is updated!

React *reacts* to the changes.

```jsx
import React, { useState } from "react";

function Greeting(props) {
  const [count, setCount] = useState(0)

  const updateCount = () => {
    setCount(count + 1)
  }

  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>You've clicked {count} times</p>
      <button onClick={updateCount}>
        Click me
      </button>
    </div>
  );
}
```

Call the update function with a new value to set the state. NEVER set the value directly

count will update AUTOMATICALLY!

Use curly braces to set attribute to JS value

Set onClick attribute of a button to a custom function

https://twitter.com/chrisachard/status/1175022111758442497

Tryout useState

1. Use a previous React project or `react-cdn-template`,
2. Create a component called `Counter`.
3. The Counter component should have a similar structure as the component to the right, with a `count` state, `updateCount` function, displaying the count state and a button calling `updateCount`.
4. Render the `Counter` component in your `App` component or another component.
5. Test and think of the pros of using React and states compared to Vanilla JS.
6. Try to render the `Counter` component in another component.


```jsx
import React, { useState } from "react";

function Greeting(props) {
  const [count, setCount] = useState(0)

  const updateCount = () => {
    setCount(count + 1)
  }

  return (
    <div>
      <h1>Hello, {props.name}!</h1>
      <p>You've clicked {count} times</p>
      <button onClick={updateCount}>
        Click me
      </button>
    </div>
  );
}
```

List, loop and map

```
const DATA = [
  { id: 4, title: 'A New Hope' },
  { id: 5, title: 'The Empire Strikes Back' },
  { id: 6, title: 'Return of the Jedi' }
]

const App = () => <MyList items={DATA} />

function MyList(props) {
  return (
    <div>
      {
        props.items.map(item => {
          return <p key={item.id}>{item.title}</p>
        })
      }
    </div>
  )
}
```

Wrap entire JS in curly braces

Map over data that was passed in as props

Return JSX from each iteration

Pass unique key attribute to each element

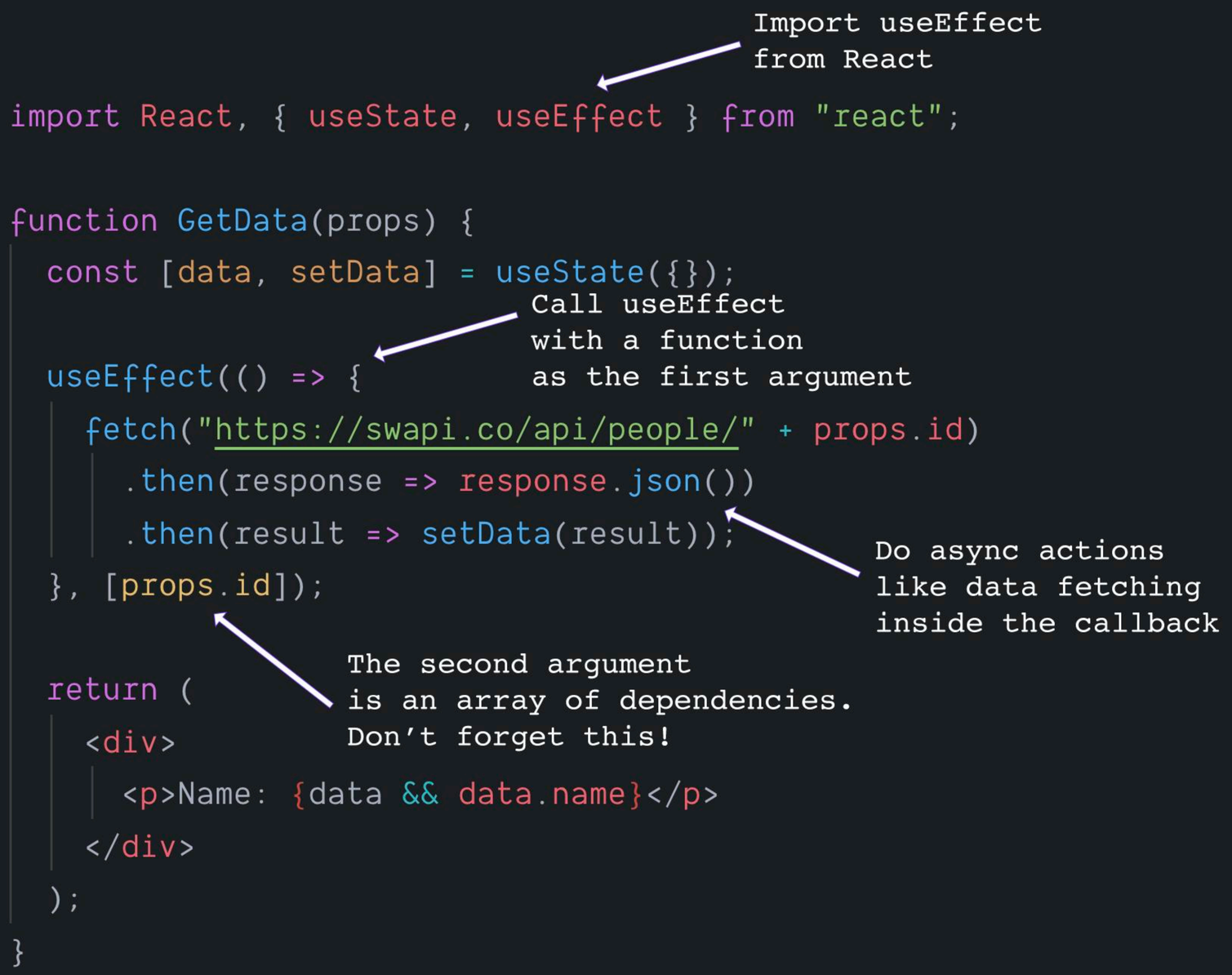
Use curly braces like normal to display data

https://twitter.com/chrisachard/status/1175022111758442497

https://www.youtube.com/watch?v=TNhaISOUy6Q

useEffect(...)

... a hook that allows us to do side effects in a component, like fetching data, directly updating the DOM, and timers.



https://twitter.com/chrisachard/status/1175022111758442497

useEffect(...)

useEffect accepts two arguments.

We should always include the second parameter which accepts an array of dependencies.

useEffect runs on every render and when the dependencies change.

```
useEffect(() => {
    //Runs on every render
});

useEffect(() => {
    //Runs only on the first render
}, []);

useEffect(() => {
    //Runs on the first render
    //And any time any dependency value changes
}, [prop, state]);
```

https://www.w3schools.com/react/react_useeffect.asp

Fetch

... get & post data from and to a data source
... can perform network requests to a server

```
1  let promise = fetch(url, [options])
```

- `url` – the URL to access.
- `options` – optional parameters: method, headers etc.

Without `options`, this is a simple GET request, downloading the contents of the `url`.

https://javascript.info/fetch

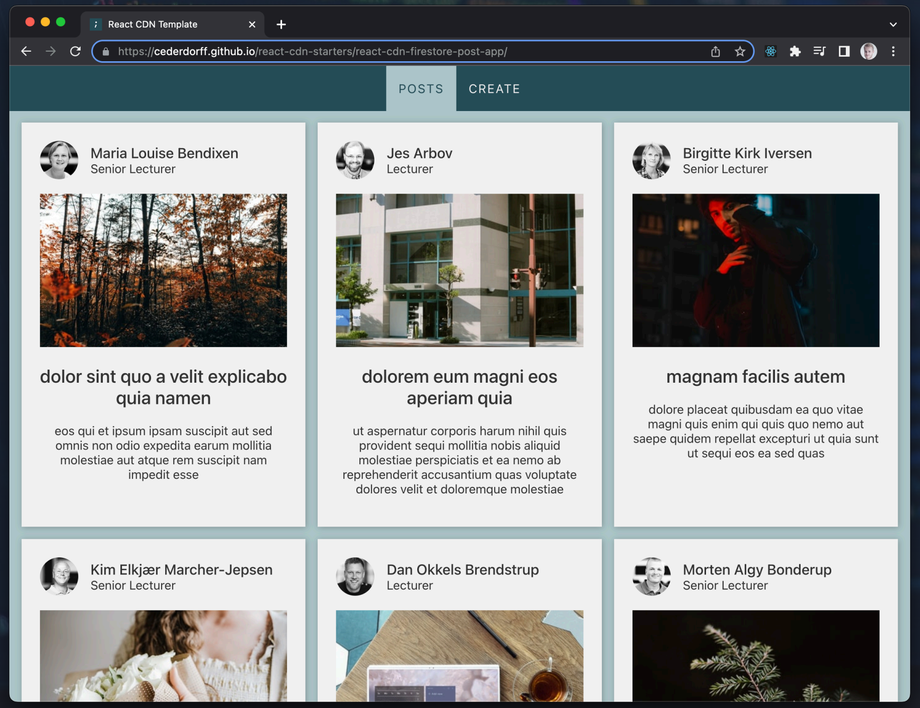
Fetch & useEffect in React

In `useEffect`, perform the sideeffect (fetch) to get data from a data source.
And save the response data in a state using `useState`.

```
function PostsPage() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
        async function getPosts() {
            const url = "https://raw.githubusercontent.com/cederdorff/web-frontend/main/data/posts.json";
            const response = await fetch(url);
            const data = await response.json();
            setPosts(data);
        }
        getPosts();
    }, []);

    return (
        <section className="page">
```

React Fetch Posts

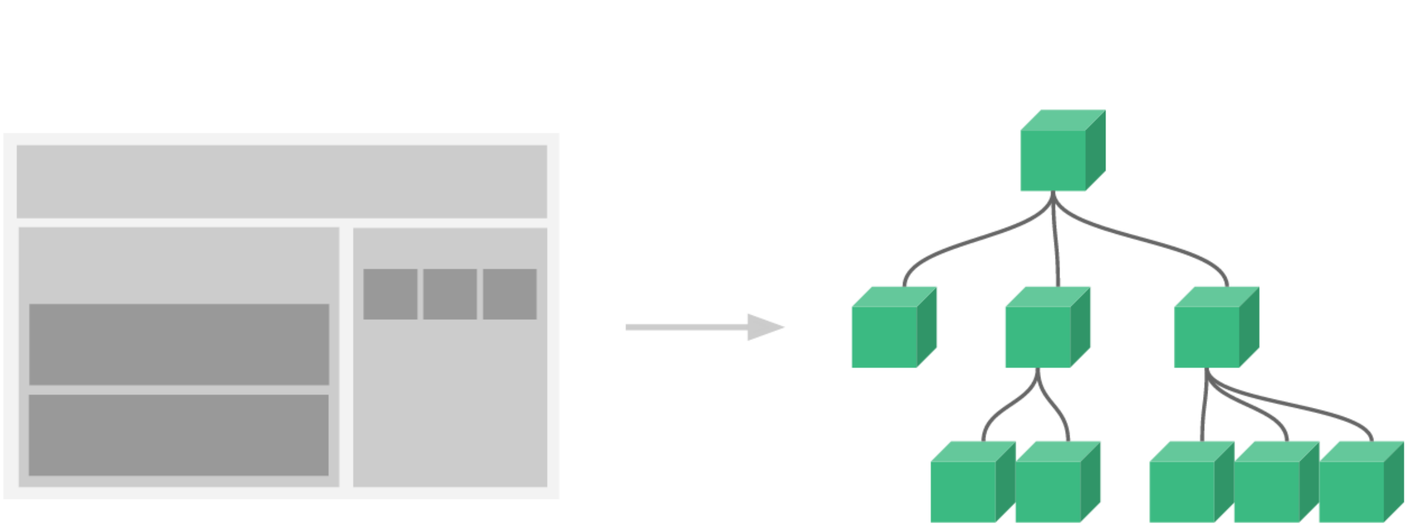
https://raw.githubusercontent.com/cederdorff/web-frontend/main/data/posts.json



1. Use `react-cdn-template` or `react-cdn-page-template`. The goal is to fetch and display a grid of posts using `useState` and `useEffect`. Similar to the JS Fetch Posts (previous lecture). You can use the predefined grid styling in app.css
2. Create a component called Posts returning a `section` with an `h1` (with a title) and another section as your grid container. Render the Post component in the App and test it in the browser.
3. In `Posts` component, implement a state called `posts`. The state must be initialised with an empty array in `useState`.
4. Use useEffect to fetch post data from the URL to the right. Save the fetched posts in the state. Use console.log(...) to test your fetch call.
5. In the grid container, map through the `posts` state and create an `article` for every post (similar to the JS Fetch Posts exercise). Display the properties image, title and body in matching HTML tags inside of the article tag. Also add a key attribute on the `article` tag using the `post.id` value.

Build Reusable Components

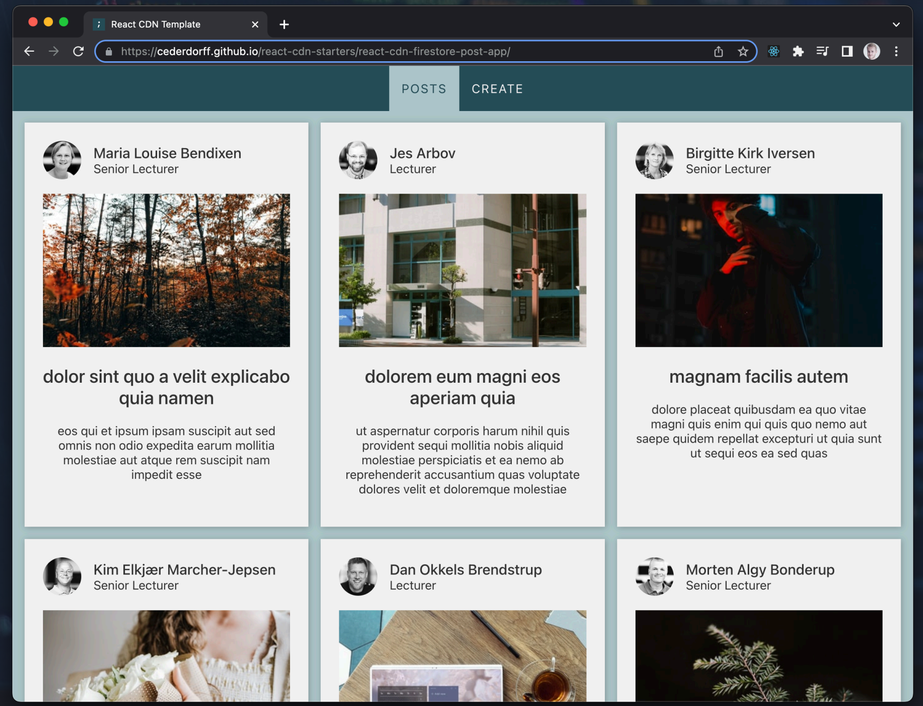
UI can be broken down to small block or modules called components. We strive for independent (and small) components.



6. Split your `Posts` component into more components. Start by creating a new component called `PostCard`.
7. Extract the post article logic from the `Posts` component to the `PostCard` component (only the article not the `.map` part).
8. Make sure `PostCard` takes in a prop called `post`. Use the prop to render the post in `PostCard`.
9. Inside of your `.map` in your Posts component render a `PostCard` and pass in `post` as a prop. A `PostCard` component will be rendered for every `post`. Remember to add a `key` attribute on the `PostCard`. (and remove it from the `article`).

React Fetch Posts

https://raw.githubusercontent.com/cederdorff/web-frontend/main/data/posts.json



Style your component

Regular Stylesheet

```
import "./styles.css";

function NormalCSS() {
  return <p className="big-text">
    BIG!
  </p>;
}
```

Use className instead of class, because class is a restricted word in javascript

Inline Styles

```
function InlineStyle() {
  return (
    <p
      style={{
        fontSize: 20,
        color: "#0000ff"
      }}
    >
      Blue Text
    </p>
  );
}
```

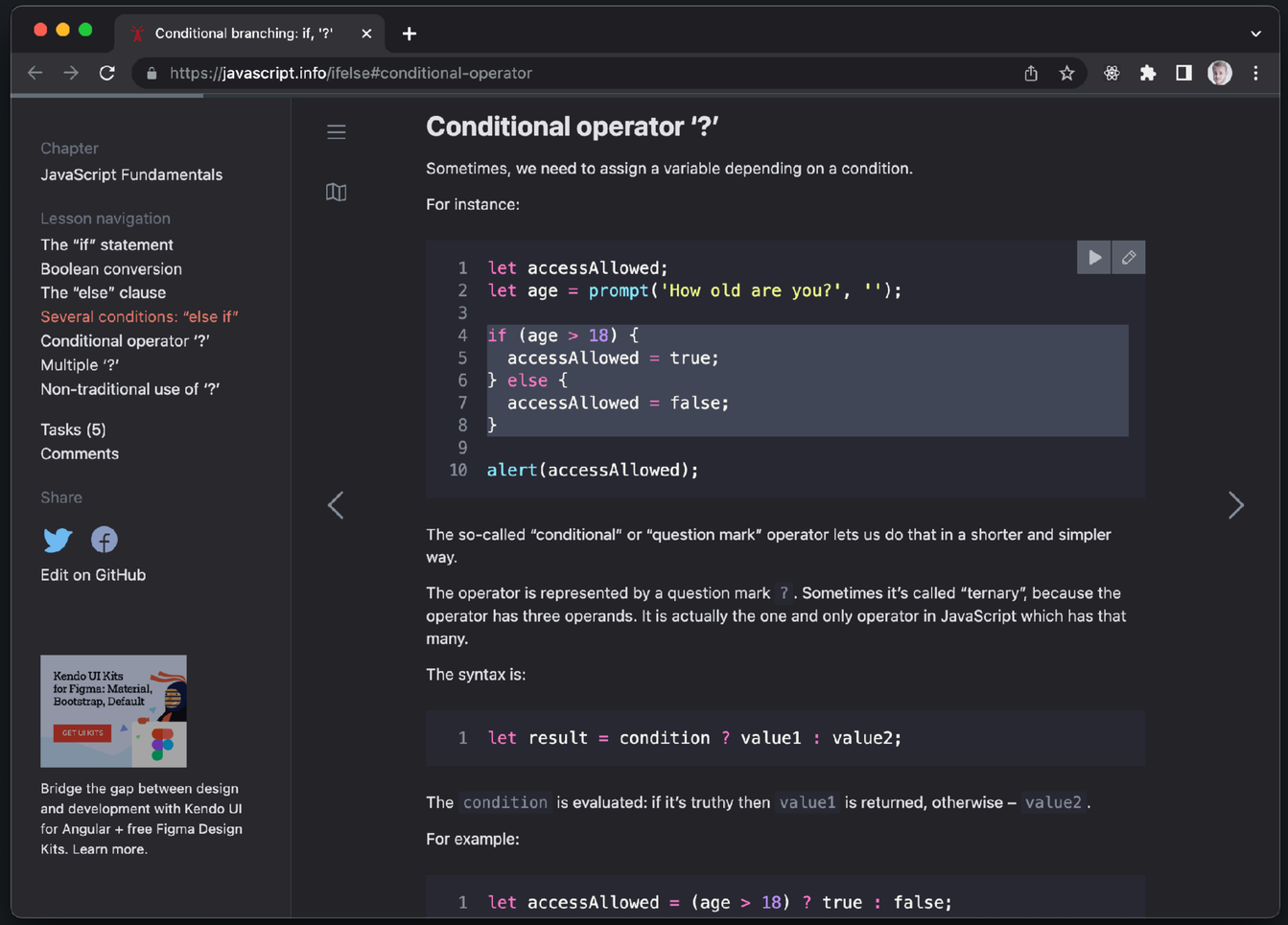
Double curly braces because you're defining a JS object inside of JSX

Camel case attributes

Values like colors must be in strings

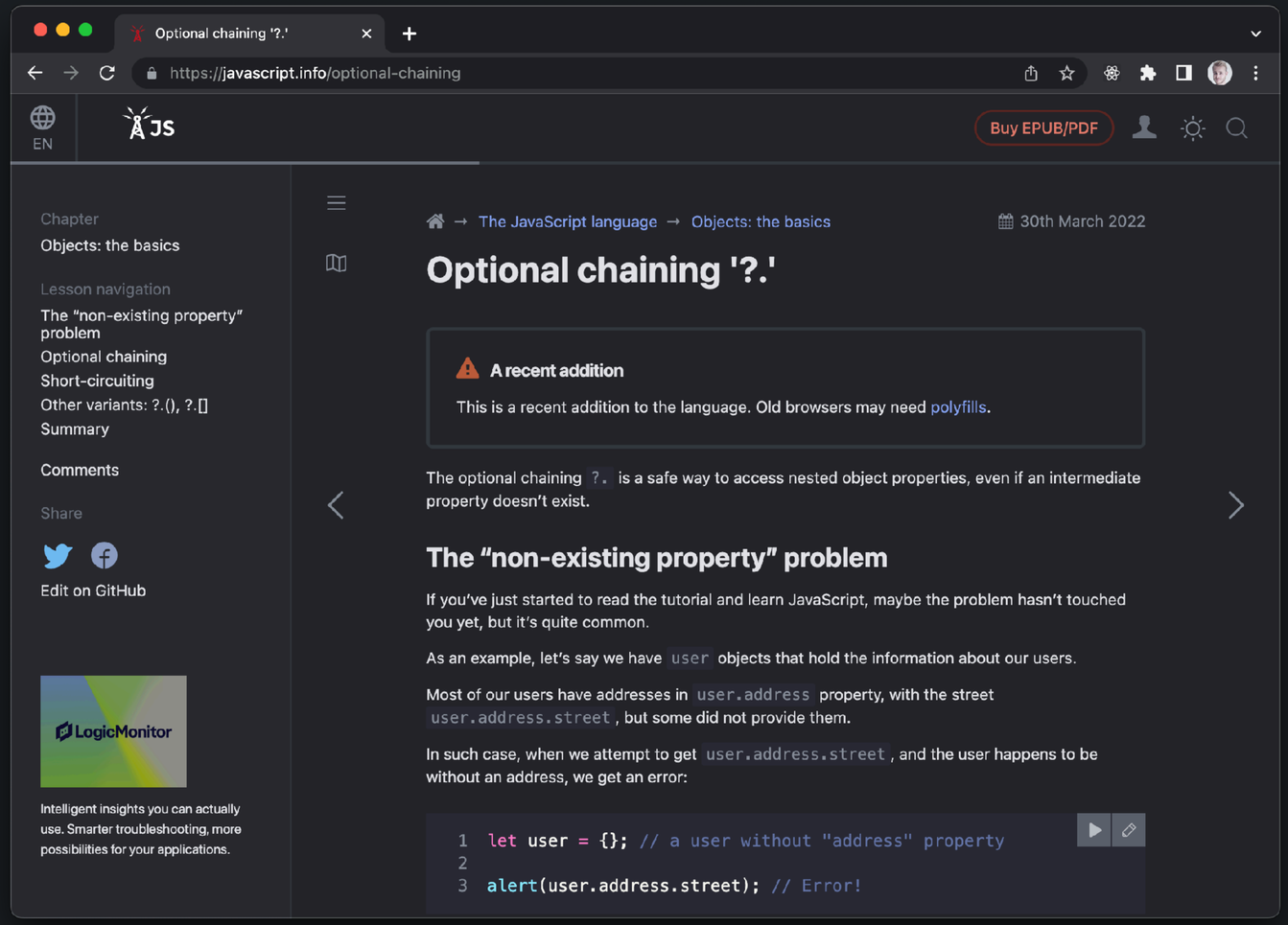
https://twitter.com/chrisachard/status/1175022111758442497

Shorter and simpler way to do `if/else`



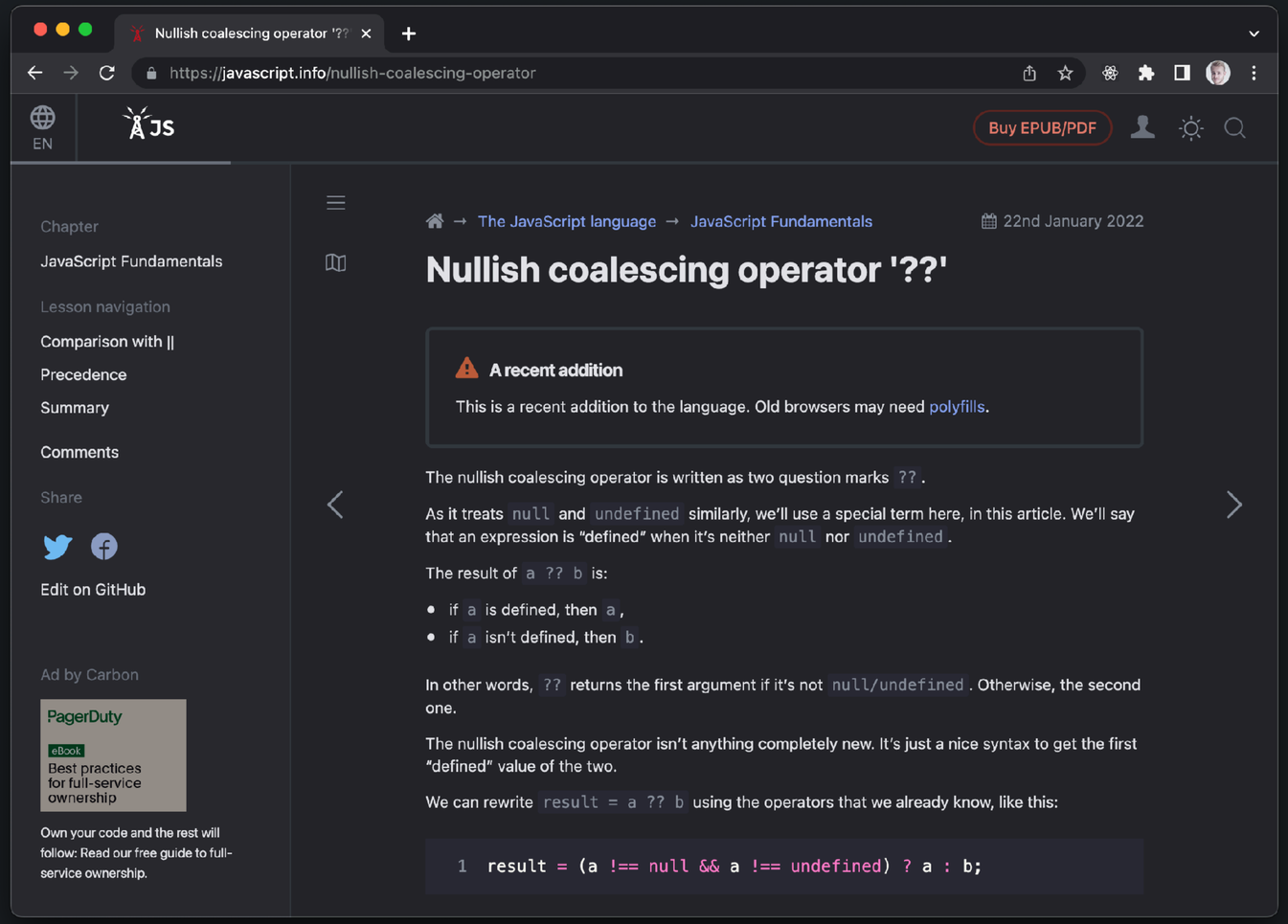
https://javascript.info/ifelse#conditional-operator
Ternary Operator

?. is a safe way to access nested object properties



https://javascript.info/optional-chaining

?? returns the first argument if it's not null/undefined



https://javascript.info/nullish-coalescing-operator

Async JS In React

`fetch(…)`, `await` & `async`

Async JS

JavaScript reads and runs the script from top to bottom.

JavaScript functions are executed in the sequence they are called. Not in the sequence they are defined.

... by default JavaScript is synchronous.

JS is Synchronous & Single-Threaded

```
function myFirst() {
    console.log("Hello");
}

function mySecond() {
    console.log("Goodbye");
}

mySecond();
myFirst();
```

```
/* ---------- Global Variables ---------- */
let _users = [];
let _selectedUserId;

/* ------------------------------------- */

async function fetchUsers() {…
}
function appendUsers(usersArray) {…
}

// ========= INIT APP =========

async function initApp() {
    await fetchUsers();
    appendUsers(_users);
}

initApp();
```

With callbacks, we can make JS Asynchronous

```
setTimeout(() => {
    console.log("Hey, I'm async!");
}, 3000);

btn.addEventListener('click', () => {
    alert("Hey, you clicked me!");
});
```

```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canva
    .then(function (response) {
        return response.json();
    })
    .then(function (data) {
        console.log(data);
    });
```

Fetch is Asynchronous

And it’s about making HTTP requests in JavaScript.
... and a way to get & post date from and to a data source.

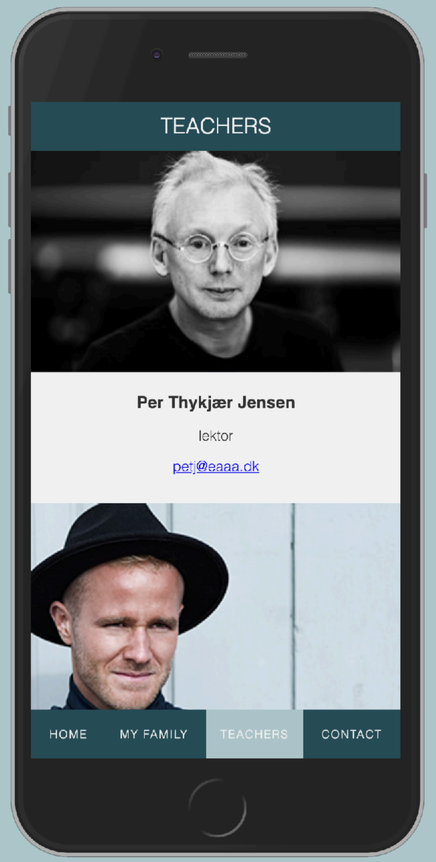
```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json")
    .then(function (response) {
        return response.json();
    })
    .then(function (data) {
        console.log(data);
    });
//
//
// or with async/await
const response = await fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json");
const data = await response.json();
console.log(data);
```

Web Development

WEB APP

API

DATA SOURCE



FETCH(...)

REQUEST

JSON DATA

RESPONSE

JSON

JSON

{ api }

"`async` and `await` make promises easier to write"

`async` makes a function return a Promise

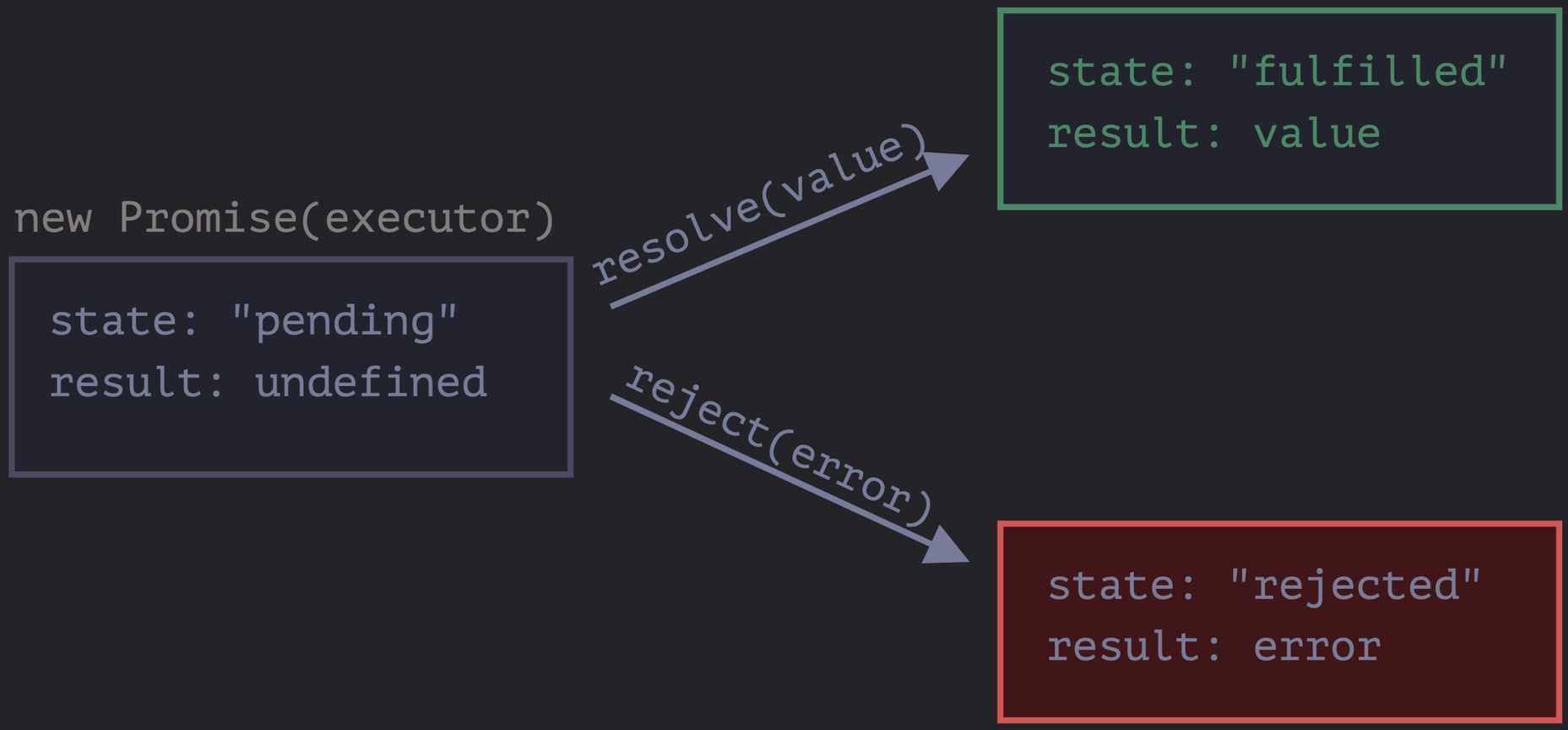
`await` makes a function wait for a Promise

https://www.w3schools.com/js/js_async.asp

Fetch returns a promise

We can use `await` to wait for fetch to finish 🤷

https://www.w3schools.com/js/js_async.asp



https://javascript.info/promise-basics

Fetch: callback (`then`) vs `async/await`

```
// fetch with callbacks
fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json")
    .then(function (response) {
        return response.json();
    })
    .then(function (data) {
        console.log(data);
    });
//
//
// or with async/await
const response = await fetch("https://cederdorff.github.io/web-frontend/canvas-users/data.json");
const data = await response.json();
console.log(data);
```

All you need to know

- Use `await` to tell JS to wait for a fetch call to finish.
- When using `await` you must tell JS that here goes some asynchronous code by wrapping it in an `async` function.

```
function PostsPage() {
    const [posts, setPosts] = useState([]);

    useEffect(() => {
        async function getPosts() {
            const url = "https://raw.githubusercontent.co
            const response = await fetch(url);
            const data = await response.json();
            setPosts(data);
        }
        getPosts();
    }, []);

    return (
        <section className="page">
            <h1>Posts</h1>
            <section className="grid-container">
                {posts.map(post => (
                    <PostItem post={post} key={post.id} /
```

Builds tools,
Package Managers,
Compilers, Transpilers,
Module Bundlers &
Command-line Interface (CLI)

It is simply just tools!

JavaScript runtime environment

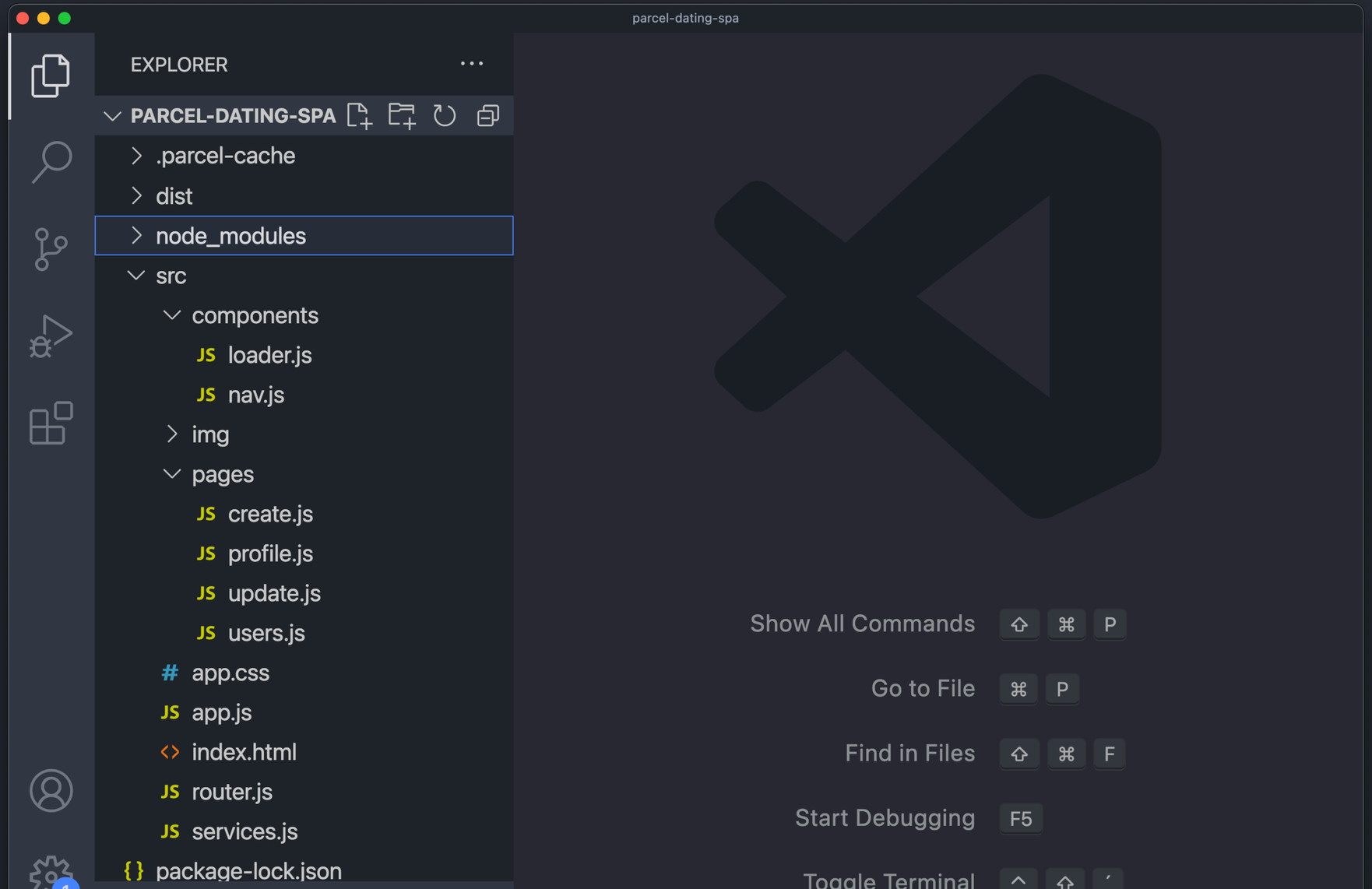
Executes JavaScript outside of the browser

Npm is a package manager for node.js packages - or modules 🤷

Node.js packages contains all the files you need for a module.

Modules are JavaScript libraries you can include in your project.

node_modules



npm consists of

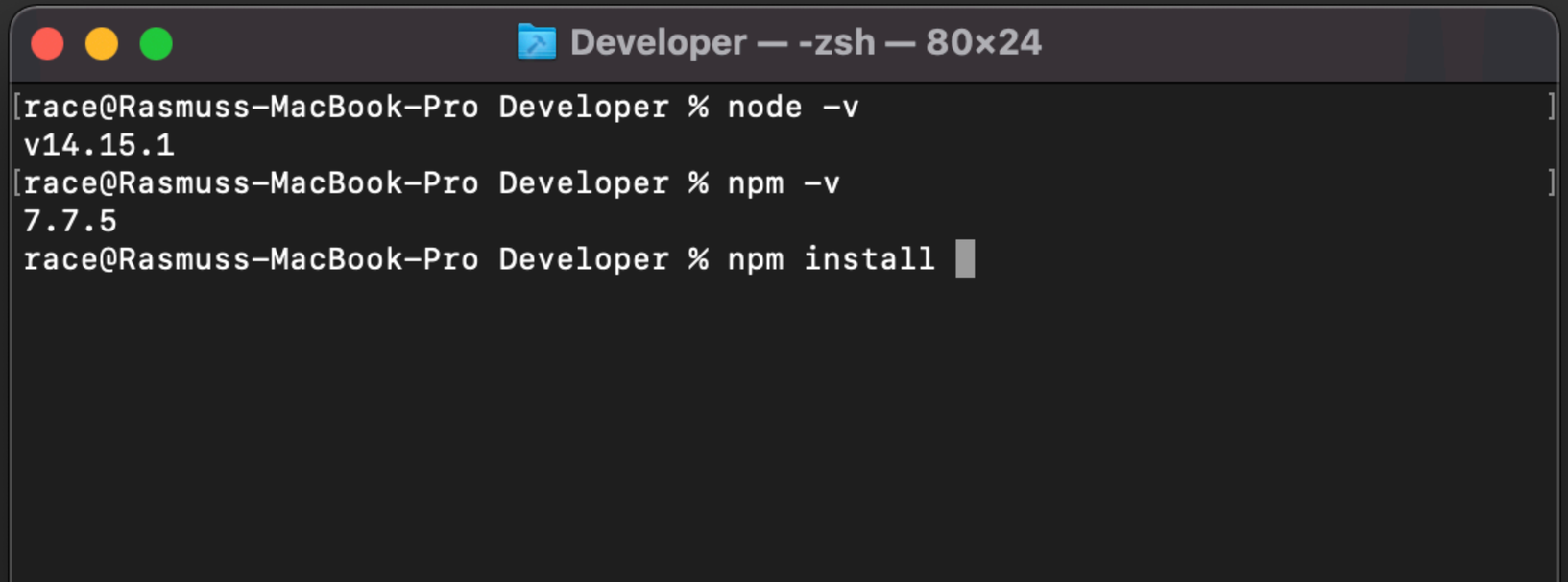
1. Website to discover packages
2. Command line interface (cli)
3. A registry - database with JS software / node modules

https://docs.npmjs.com/about-npm/

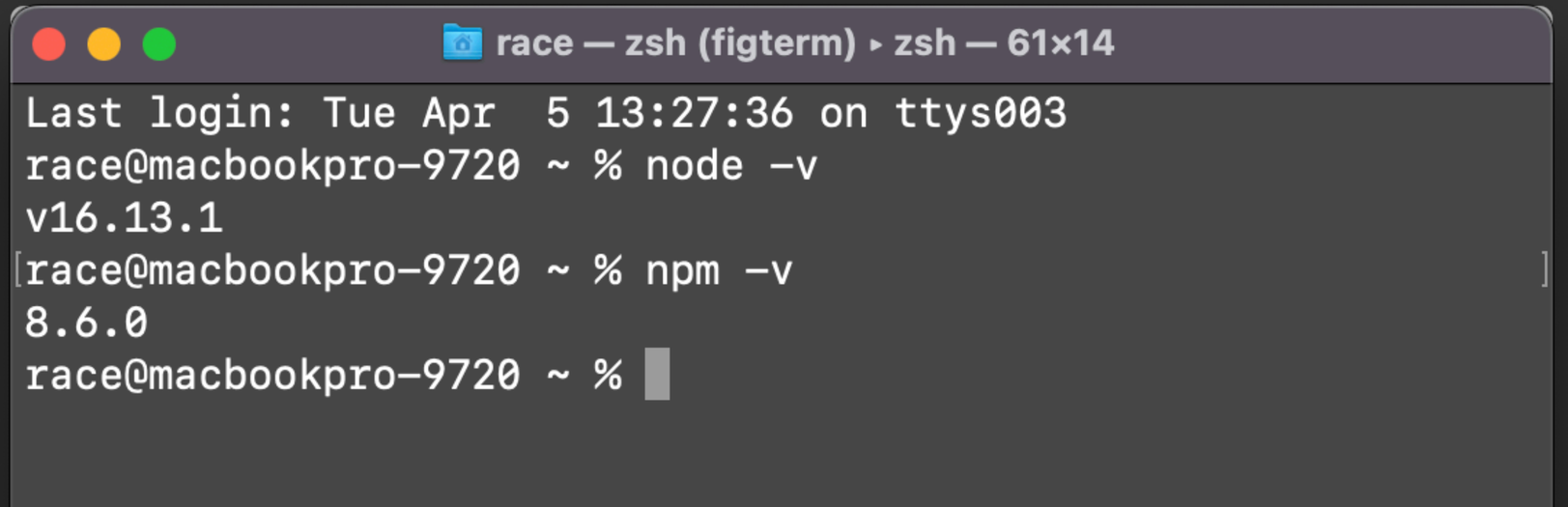
Command Line Interface

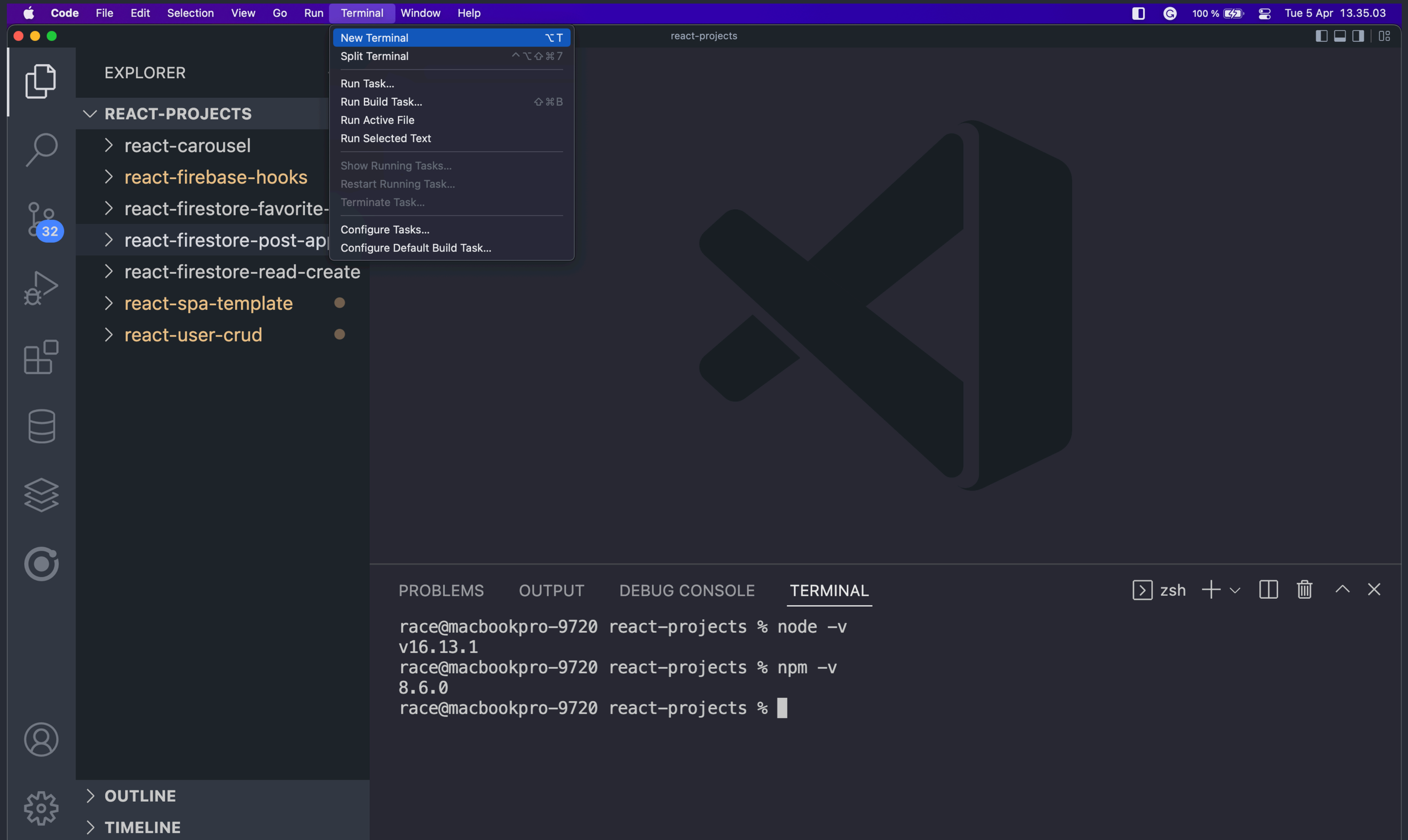
... a program that allows users to type text commands to execute functions.

npm uses CLI to install software & packages.



1. Download & install Node.js & npm: nodejs.org/en/
2. Check your version of npm & Node.js




Code File Edit Selection View Go Run Terminal Window Help
100 % Tue 5 Apr 13.35.03
New Terminal ⌥T
Split Terminal ^⌥⇧⌘7
Run Task...
Run Build Task... ⇧⌘B
Run Active File
Run Selected Text
Show Running Tasks...
Restart Running Task...
Terminate Task...
Configure Tasks...
Configure Default Build Task...
react-projects
EXPLORER
REACT-PROJECTS
react-carousel
react-firebase-hooks
react-firestore-favorite-
react-firestore-post-ap
react-firestore-read-create
react-spa-template
react-user-crud
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
zsh
race@macbookpro-9720 react-projects % node -v
v16.13.1
race@macbookpro-9720 react-projects % npm -v
8.6.0
race@macbookpro-9720 react-projects %

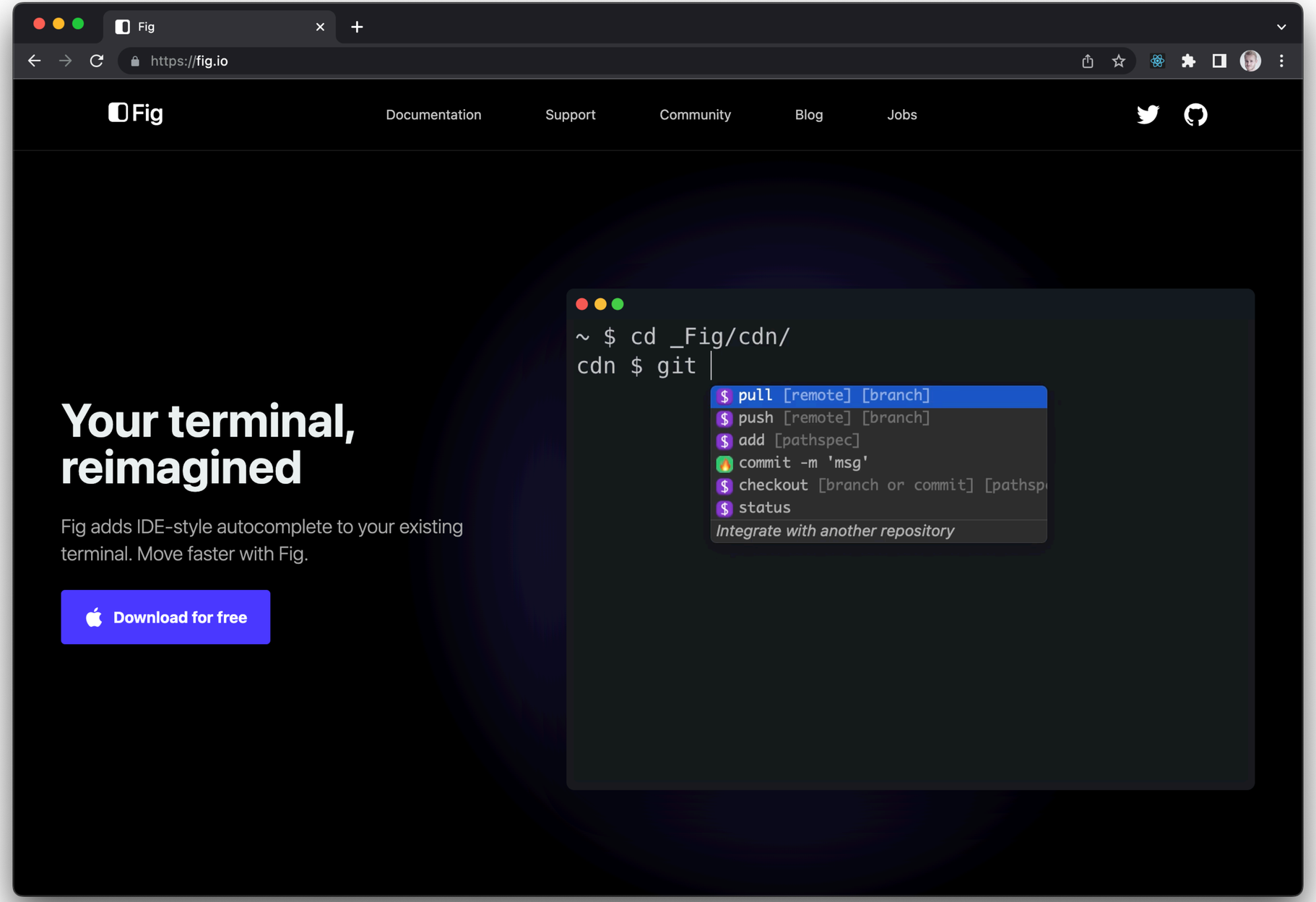
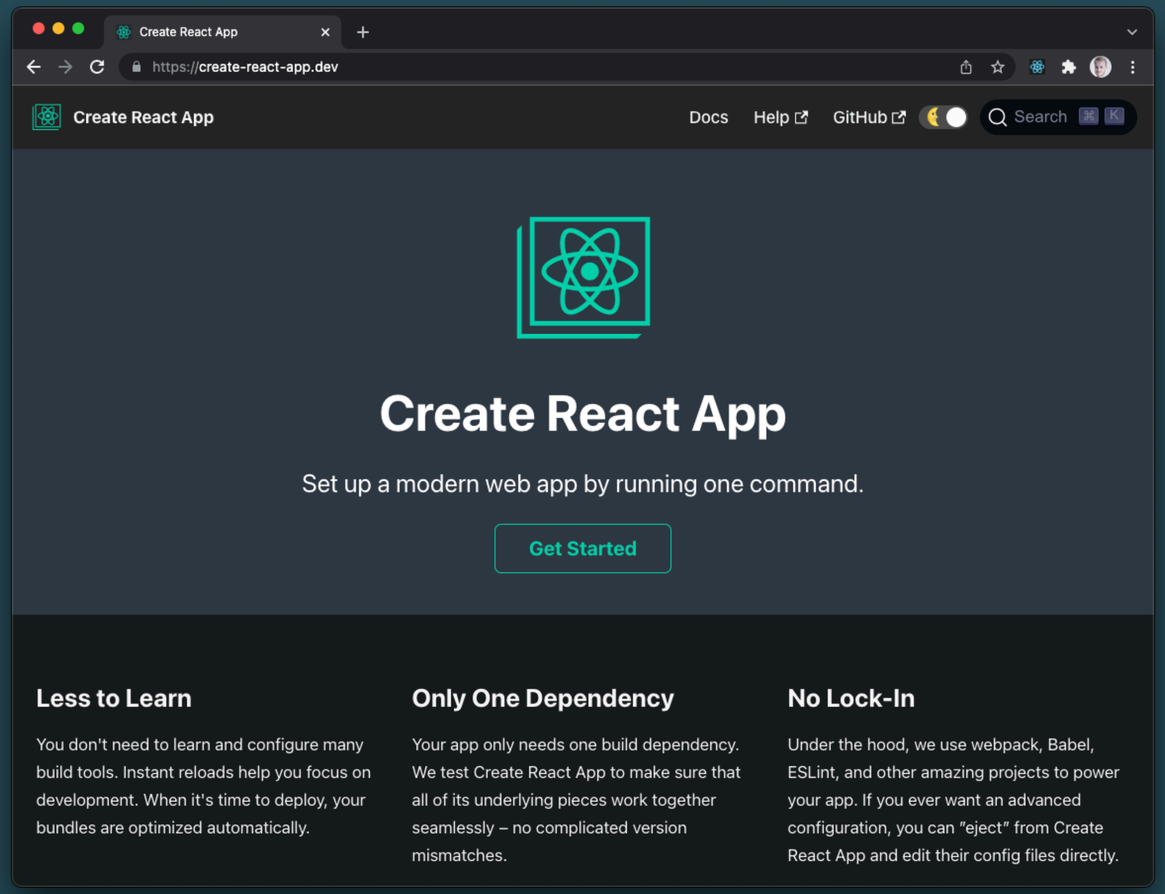


Fig
https://fig.io
Fig
Documentation
Support
Community
Blog
Jobs
Your terminal, reimagined
Fig adds IDE-style autocomplete to your existing terminal. Move faster with Fig.
Download for free
~ $ cd _Fig/cdn/
cdn $ git
pull [remote] [branch]
push [remote] [branch]
add [pathspec]
commit -m 'msg'
checkout [branch or commit] [pathsp
status
Integrate with another repository

https://fig.io/

create-react-app

Starter template for your react app



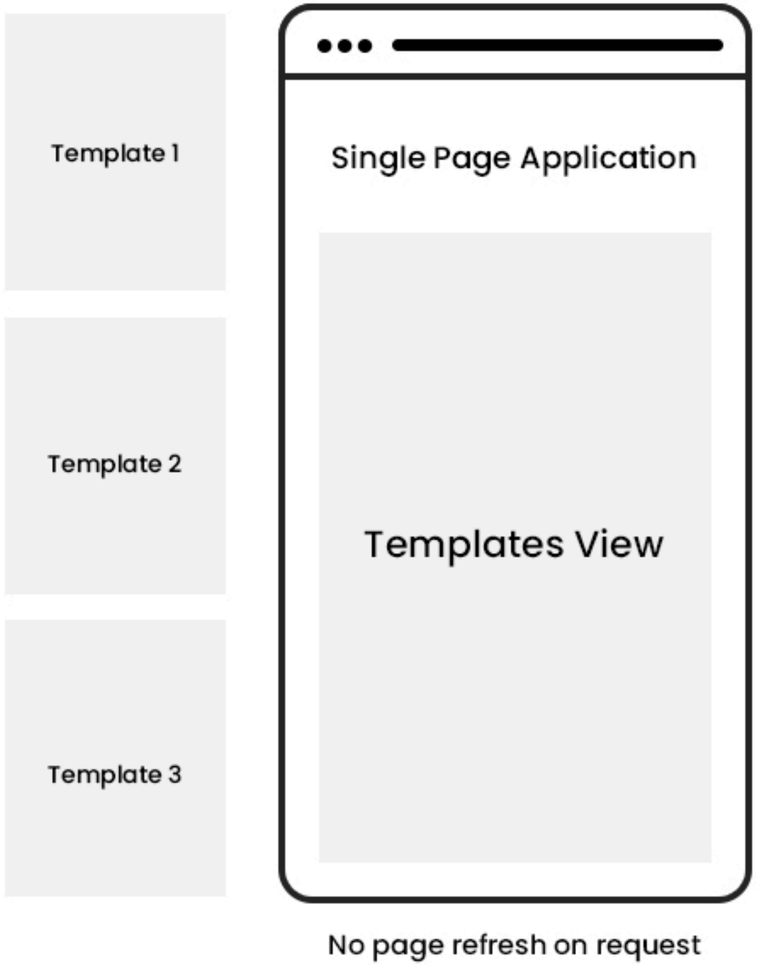
Generate create-react-app

Single Page Apps

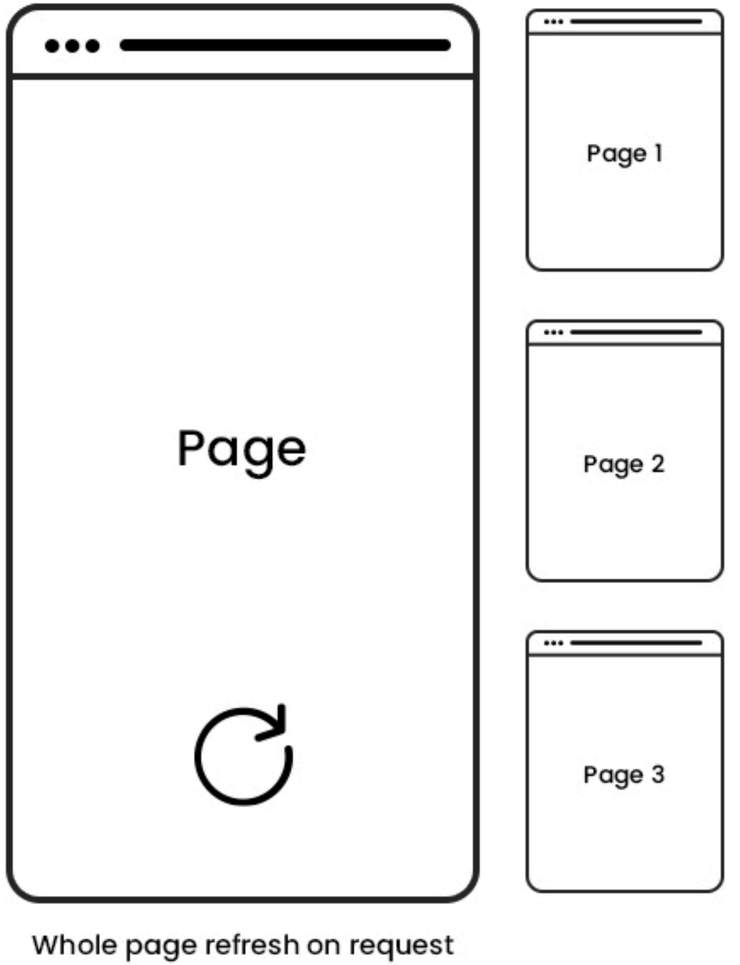
“A single-page application is an application that loads a single HTML page and all the necessary assets (such as JavaScript and CSS) required for the application to run. Any interactions with the page or subsequent pages do not require a round trip to the server which means the page is not reloaded.”

https://reactjs.org/docs/glossary.html#single-page-application

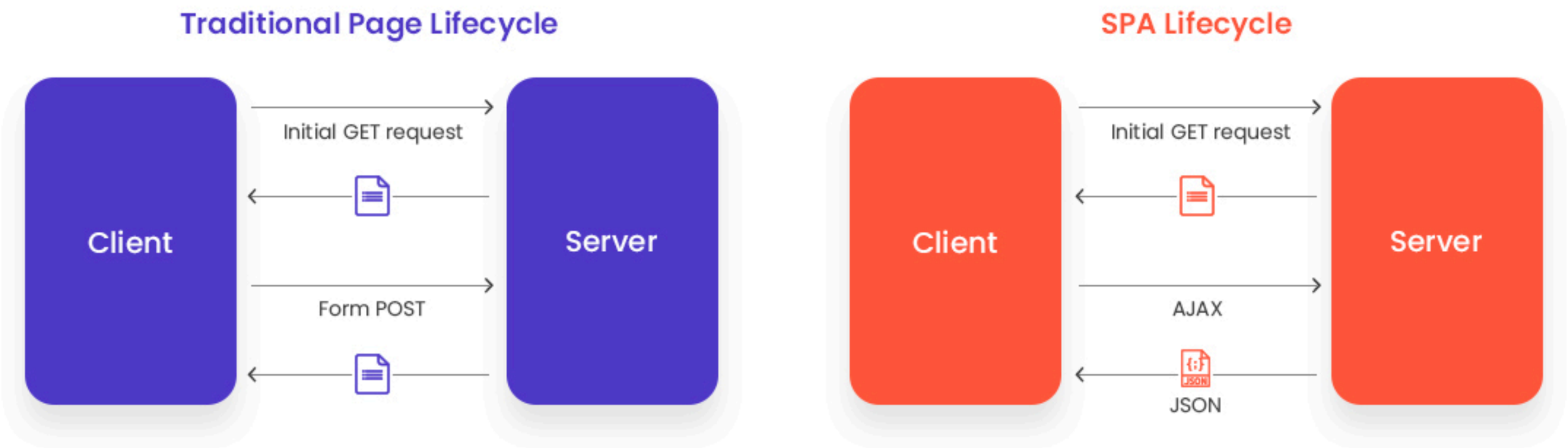
SPA



MPA



excellentwebworld.com/what-is-a-single-page-application



excellentwebworld.com/what-is-a-single-page-application

React Router

... a client & server-side routing library for React

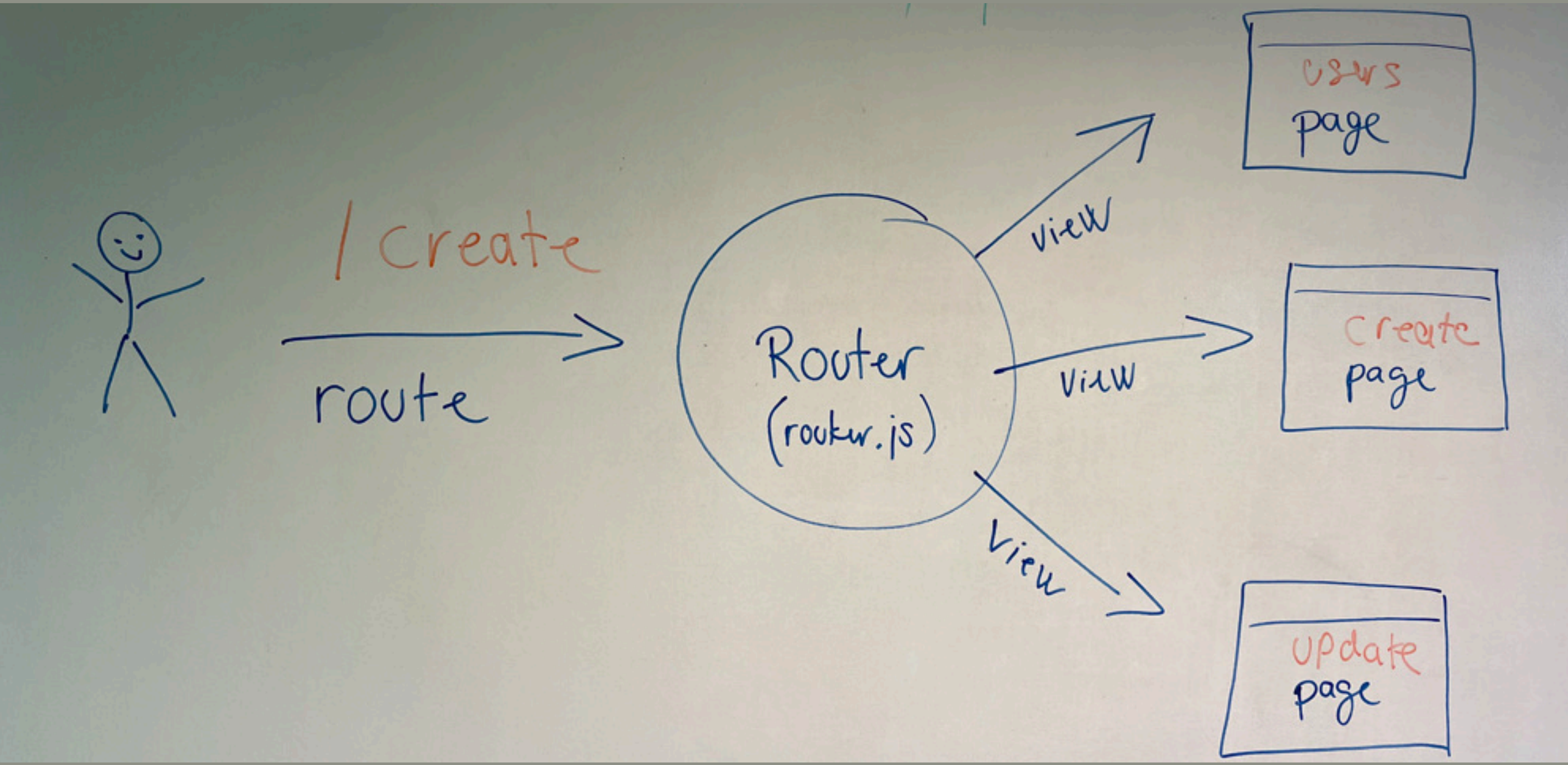
```
<Routes>
    <Route path="/" element={<HomePage />} />
    <Route path="about" element={<AboutPage />} />
    <Route path="*" element={<Navigate to="/" />} />
</Routes>
```

What’s a Router?

“*It is the piece of software in charge to organize the states of the application, switching between different views.*”

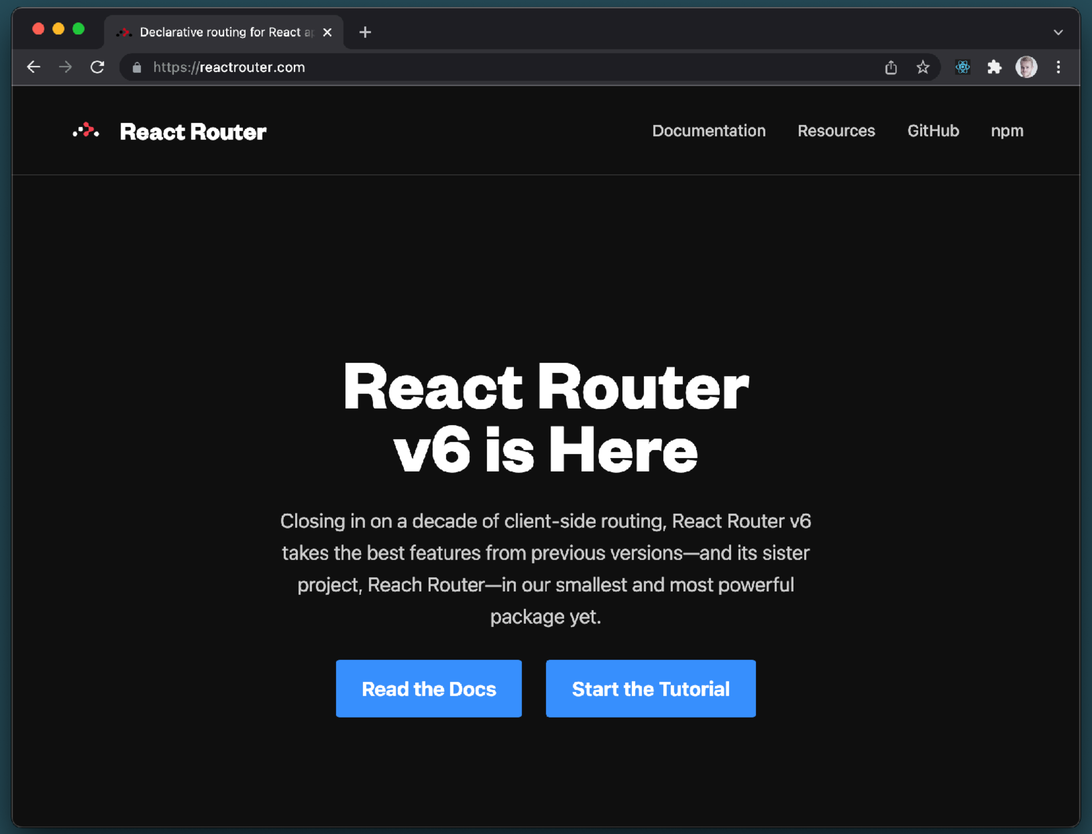
It’s a key component in Single Page Applications.

https://medium.com/@fro_g/routing-in-javascript-d552ff4d2921


/create
route
Router
(router.js)
view
view
view
users
page
create
page
update
page

Create React SPA

With React Router

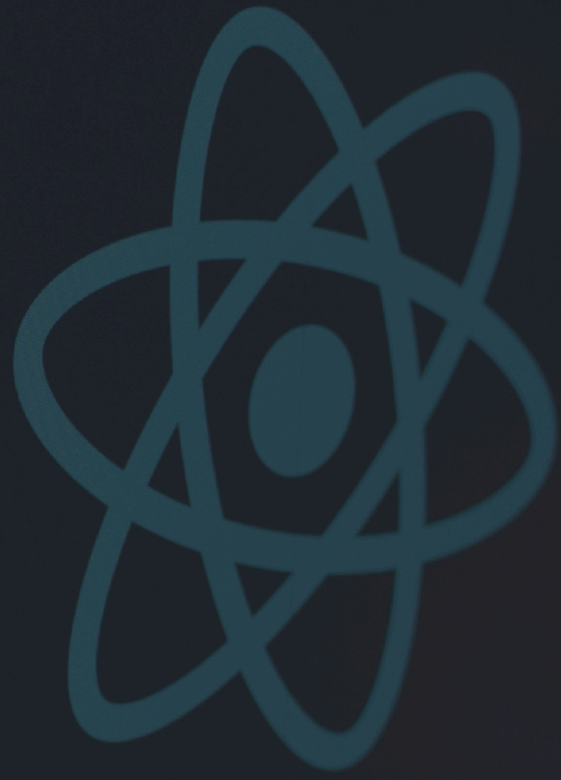


Create React SPA

React Practice

Practice your React knowledge with create-react-app and react-router:

1. Build a Portfolio site template with pages (components) like Home, Projects, About and Contact.
2. Do a remake of Canvas Users SPA with React. You can find inspiration on GitHub: `react-starters` and `react-cdn-starters`
3. Checkout React Self-study and Guides & Tutorials



Code Every Day