

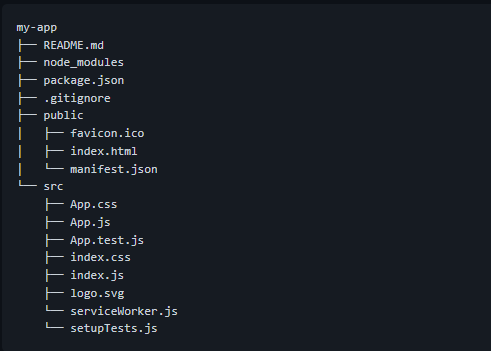
To create a project : npx create-react-app my-app

When new versions of Create React App are released, you can upgrade using a single command:

npm install react-scripts@latest

npm is all about code sharing and reuse. You can **use other people's code** in your own projects, and you can also **publish your own Node.js modules** so that other people can use them.

Structure:



package-loc.json file has

all the information for npm to be

able to rebuild those files reliably.

This file is there to ensure the npm

tracks all the modules installations properly.

Robot.txt file is used for search engine optimization.

manifest.json file,

which is used to provide some metadata to a device

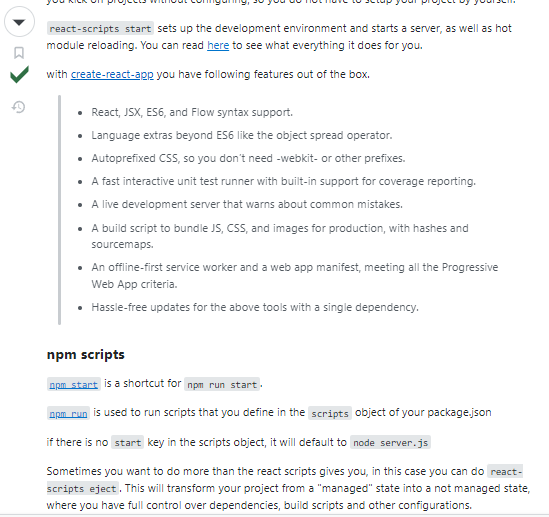
when you're React powered web app is installed on it

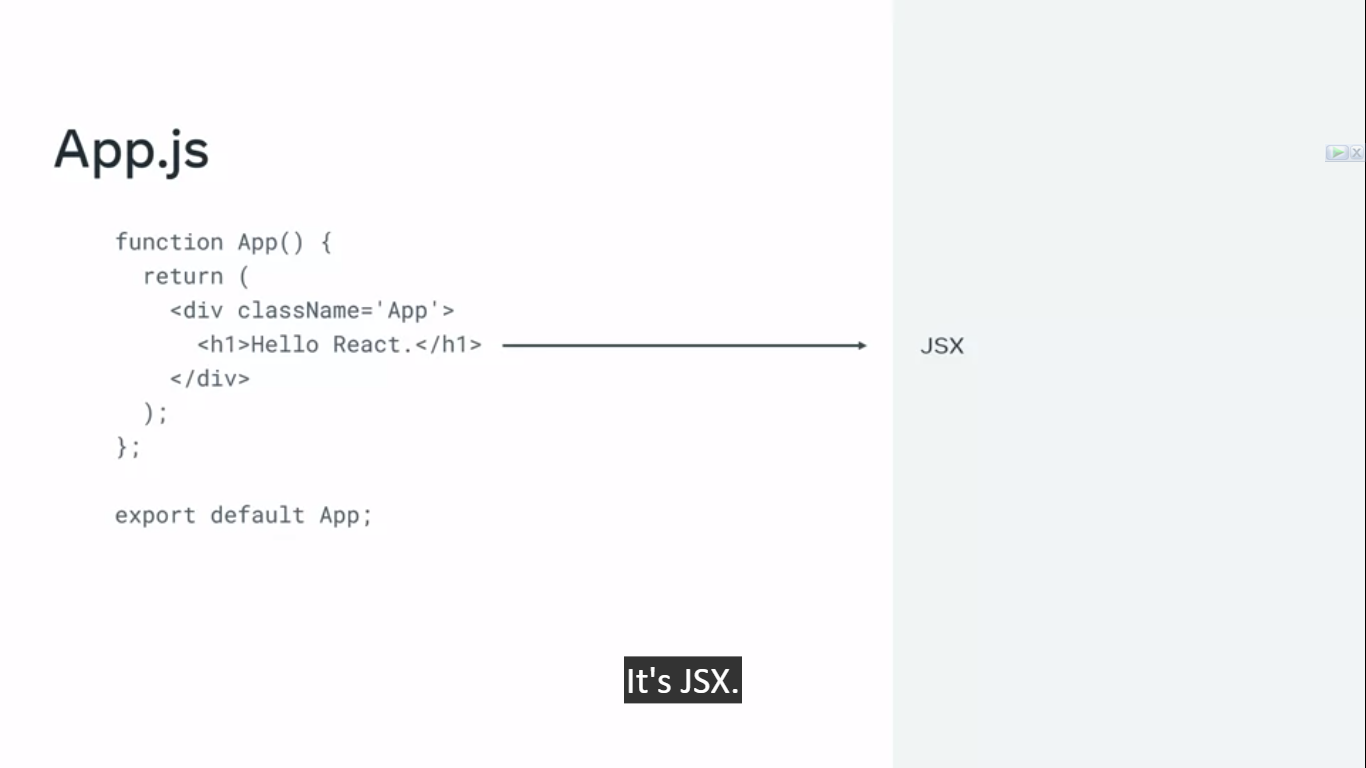
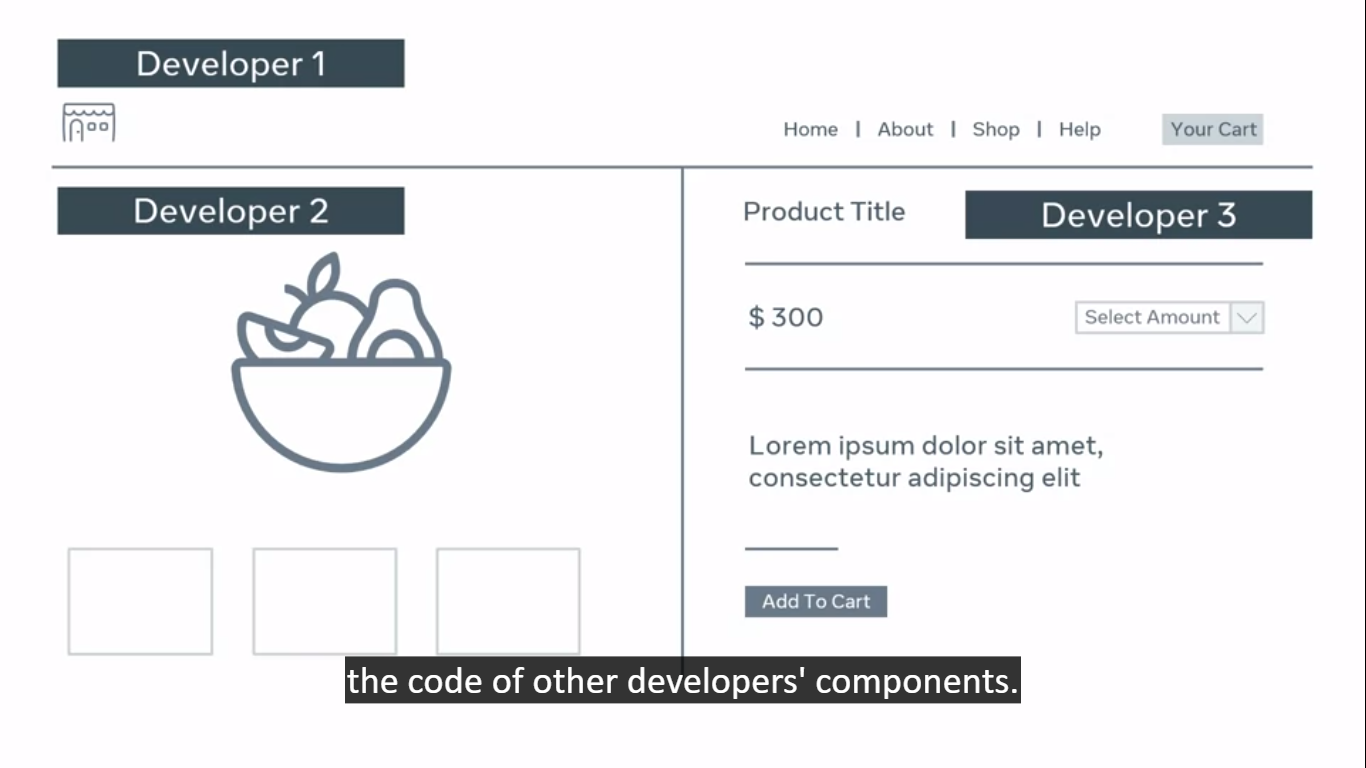
and rest of the files in src folder are related to test and performance.

**npx**: **npx** is a command-line tool that comes bundled with npm (Node Package Manager). It allows you to run packages without having to install them globally.

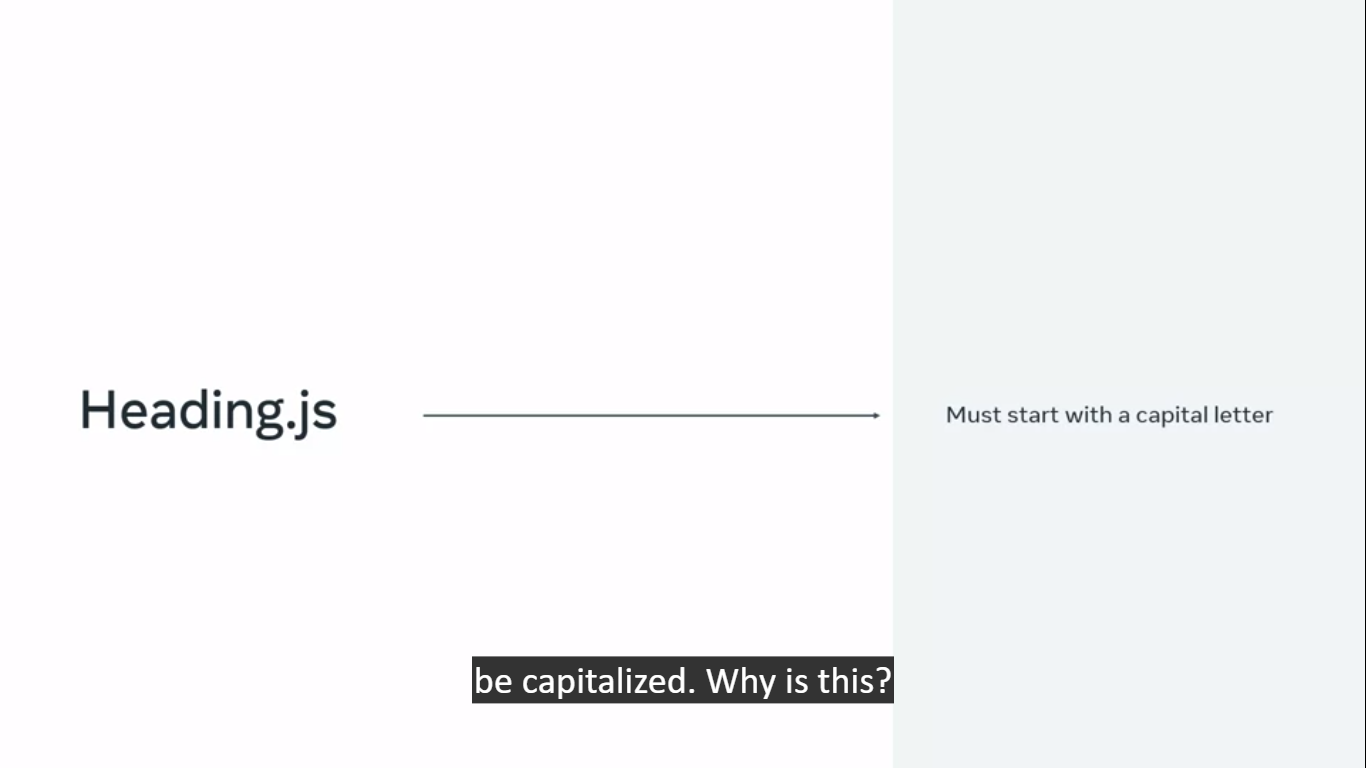
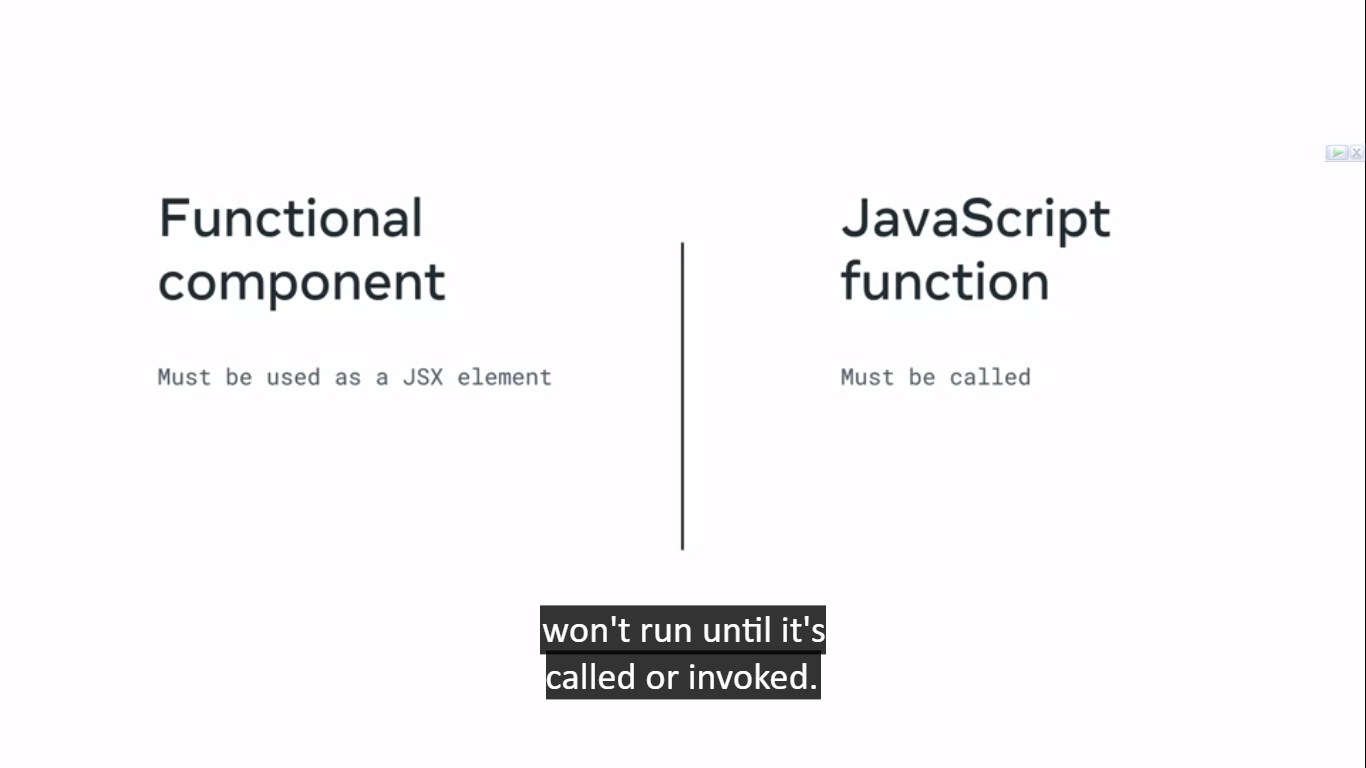
**create-react-app**: **create-react-app** is an officially supported command-line tool that sets up a new React application with a basic project structure and configuration. It sets up all the necessary files and dependencies to get started with React development.

react-scripts is a set of scripts from the [create-react-app](https://github.com/facebook/create-react-app) starter pack. create-react-app helps you kick off projects without configuring, so you do not have to setup your project by yourself.

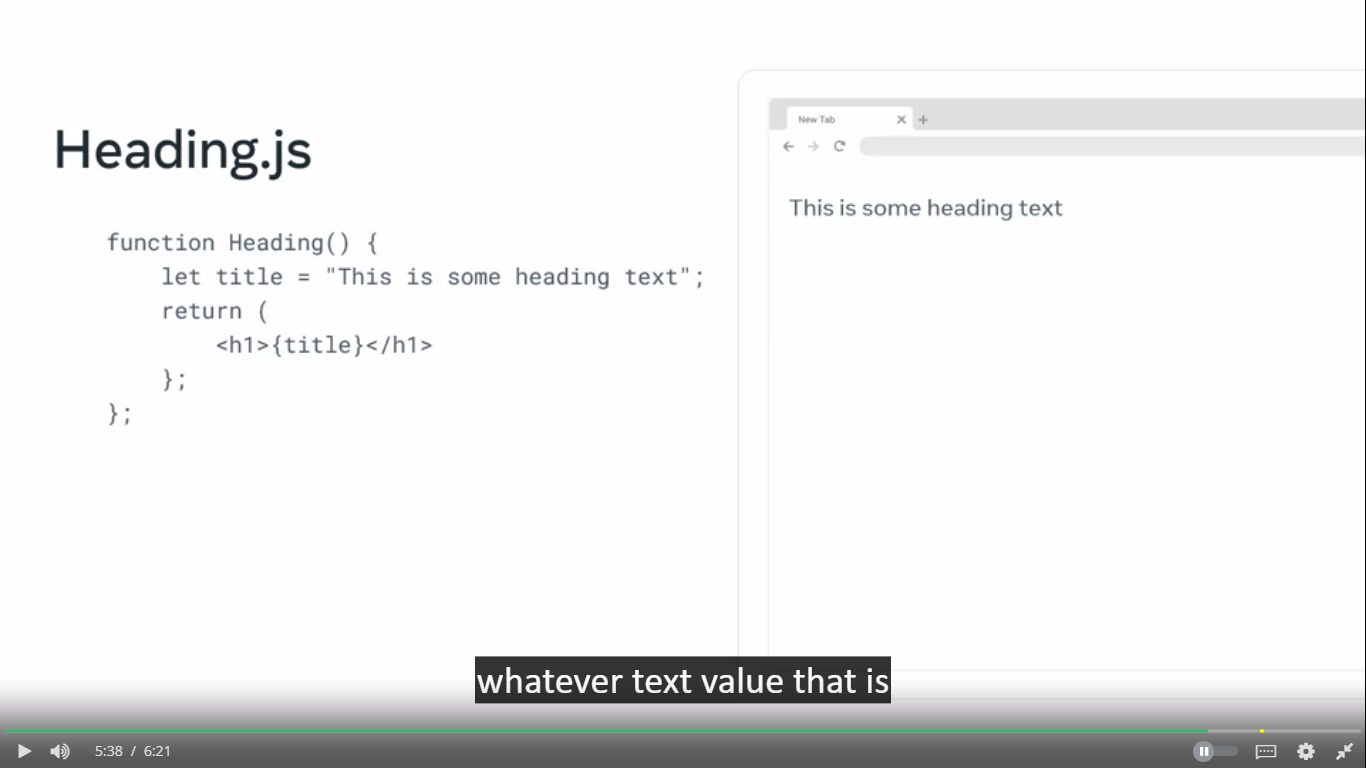




In react it is actually jsx code. Which, looks like html



React component names should be capitalized because non capitalize treated as html element. So it is necessary to tell react it is react component



A **transpiler** takes a piece of code and transforms it into some other code. Within same level (not like compiler which changes high level language into lower level)

Actually it is used to run modern js code in old browers like which an browser which support es5 engine and you want to run es6 code on it then **transpiler** is an important concept.

So **babel** is works on transpiler concept so every browser can run modern js code and removes compatibility issue

(When JSX code is transpiled by Babel, it is transformed into regular JavaScript code, not HTML. The transpiled code typically consists of function calls and object creations that represent the structure and behavior of React components.

To render JSX components in a web browser, you need to use a build tool like webpack or a bundler like Parcel, along with a JSX compiler like Babel, to transform the JSX code into browser-compatible JavaScript code.

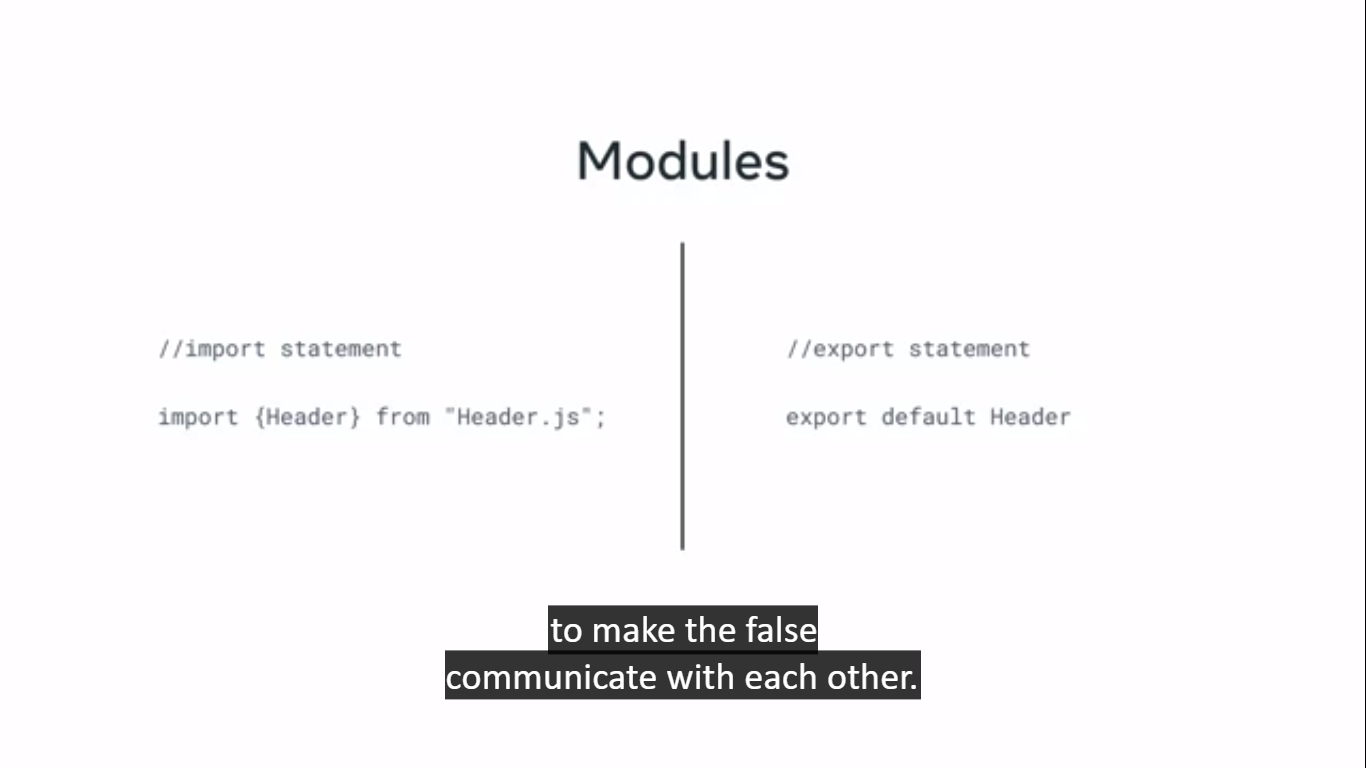
React includes a component called the "ReactDOM" library, which is responsible for rendering React components into the browser's DOM (Document Object Model). It takes the transpiled JavaScript code and converts it into the corresponding HTML elements, which are then rendered on the web page.

In summary, Babel's primary role is to transpile JSX code into regular JavaScript code, while React's ReactDOM library is responsible for rendering the resulting JavaScript code as HTML elements in the browser's DOM.

) cut is

Note:  
1.in react link tag is used for internal links and anchor tag for external links.

2. jsx is a king of js code so class word is reserved so for unlike html we have to use className rather using class like html.



There are two types of export 1. Default and 2.named

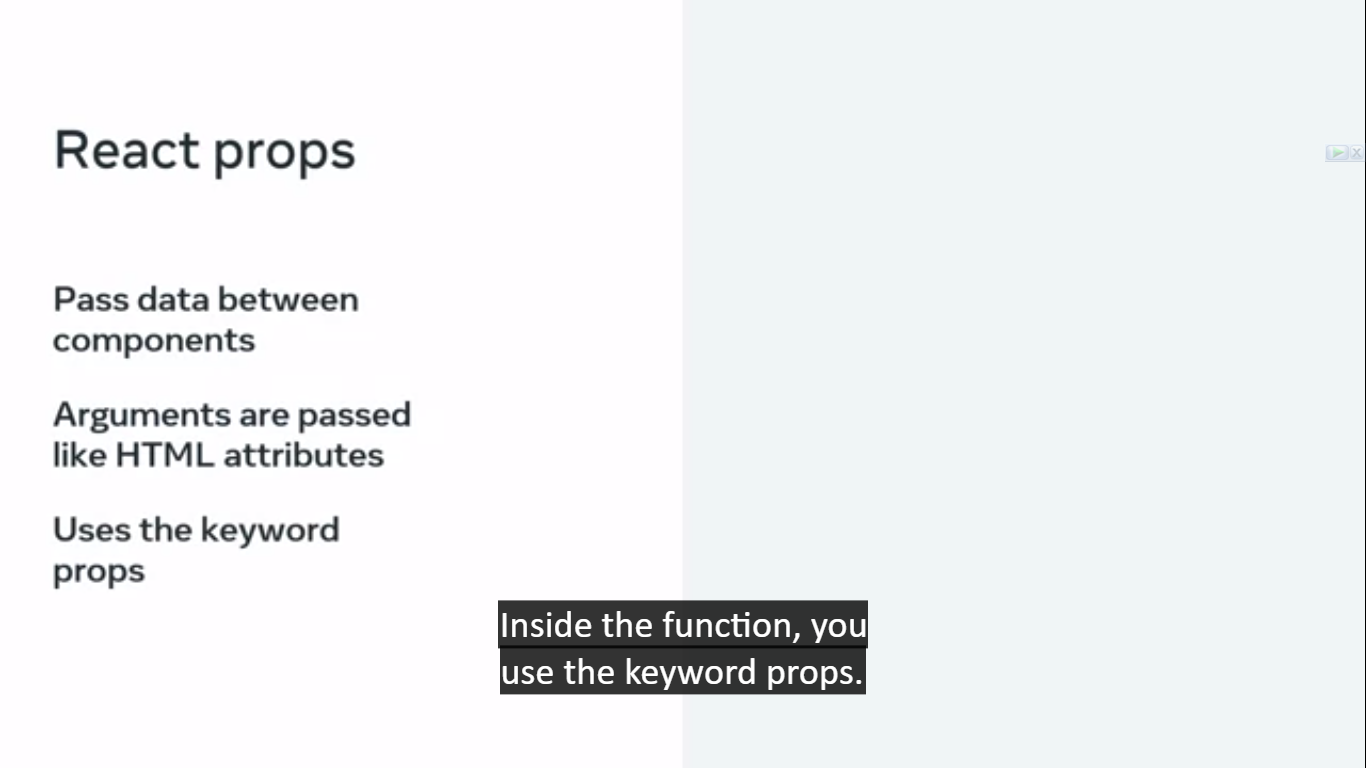


**Additional resources for React components and where they live**

Below you will find links to helpful additional resources.

* [Basic Concepts of Flexbox](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox)
* [Importing a Component](https://create-react-app.dev/docs/importing-a-component/)
* [Babeljs.io](https://babeljs.io/)
* [NPM docs: package.json](https://docs.npmjs.com/cli/v7/configuring-npm/package-json)
* [git docs: gitignore](https://git-scm.com/docs/gitignore)
* [NPM docs: node modules folder](https://docs.npmjs.com/cli/v8/configuring-npm/folders#node-modules)
* [webpack docs: DevServer](https://webpack.js.org/configuration/dev-server/)
* [webpack/webpack-dev-server on GitHub](https://github.com/webpack/webpack-dev-server)
* [Visual Studio Code keyboard shortcuts (Windows)](https://code.visualstudio.com/shortcuts/keyboard-shortcuts-windows.pdf)
* [Visual Studio Code keyboard shortcuts (macOS)](https://code.visualstudio.com/shortcuts/keyboard-shortcuts-macos.pdf)

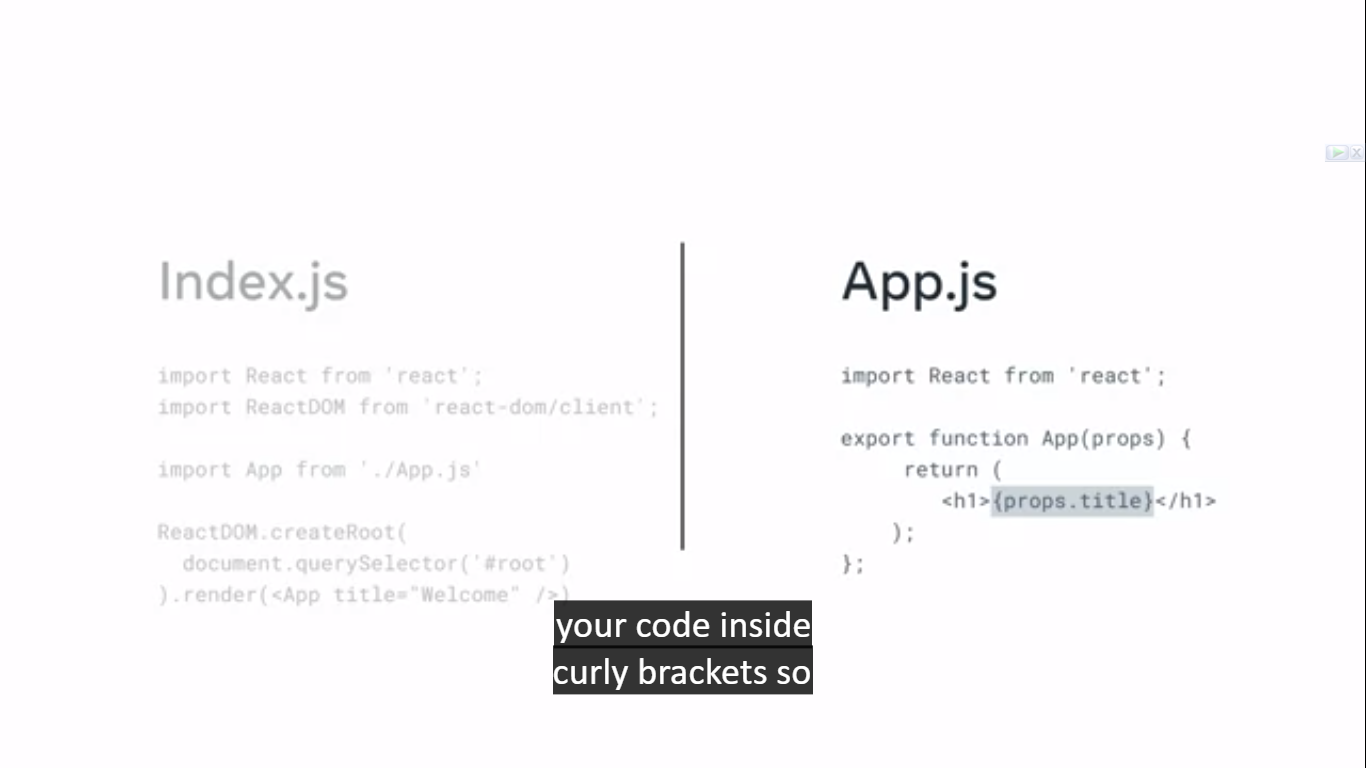
## Component Use and Styling:



Props are like js object which value can be accessed like object like props.value

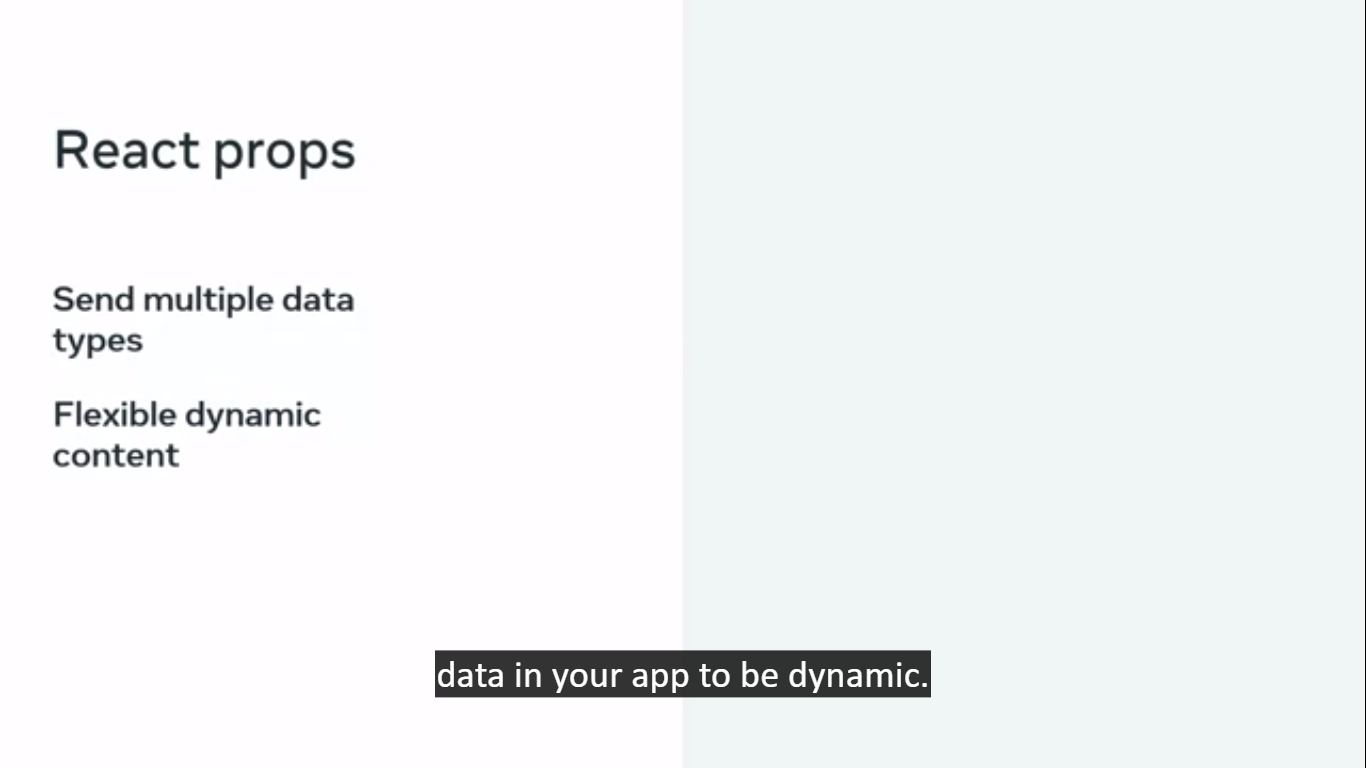
Props are passed using JSX syntax.

Actually it is an argument received by and component.



React props allows all data types to send like string, number, array, function, and also object.

Makes content dynamic.

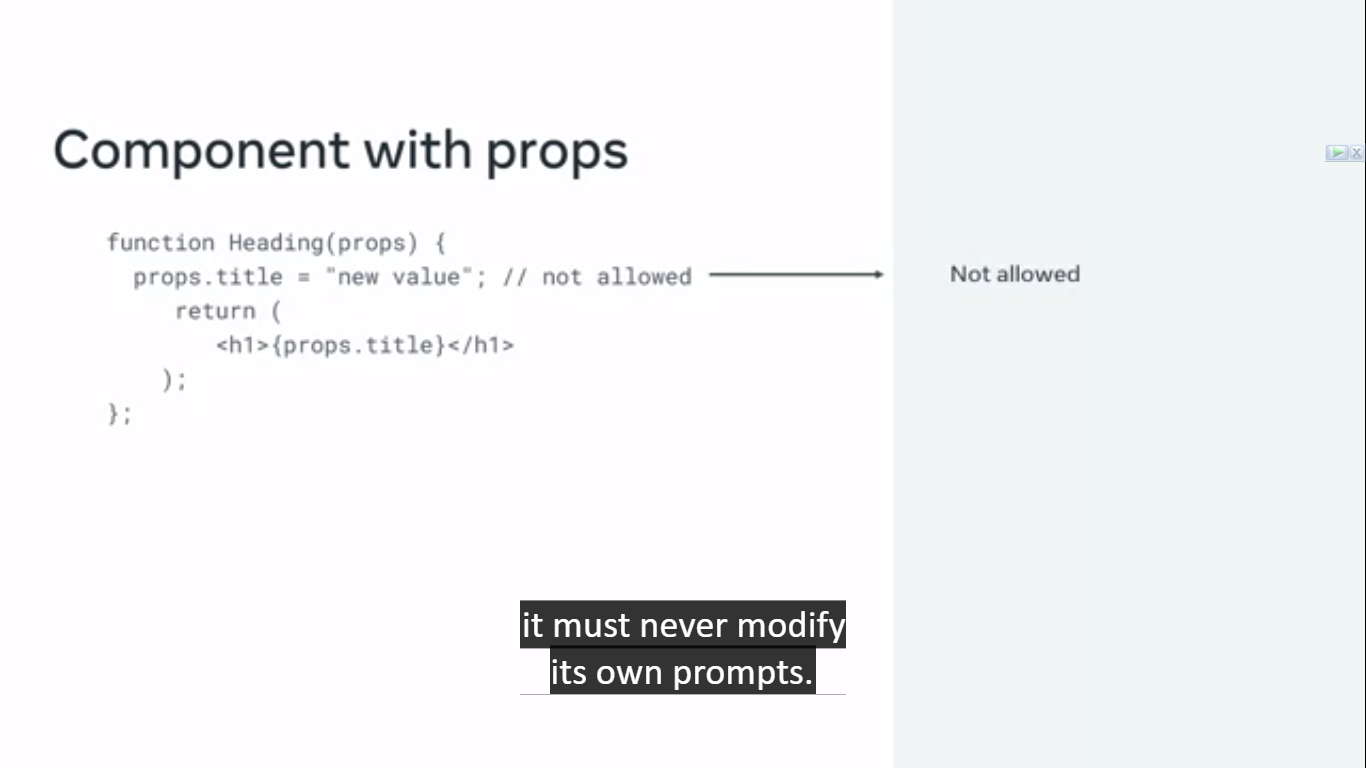


Limitations of props

1.

The component send props data is called parent component and Which receives is called child component. But it is one directional communication which only happened only(from parent to child)

2.



like parameters in a JavaScript function which allow you to pass in values as arguments, React uses properties, or **props**, to pass data between components.

transpiler to break JSX code to plain JavaScript, bable do that thing

|  |  |
| --- | --- |
| function App() {    return <h1>Hello there</h1>  } | "use strict";  function App() {      return /\*#\_\_PURE\_\_\*/React.createElement("h1", null, "Hello there");  } |

**createElement** function receives three arguments:

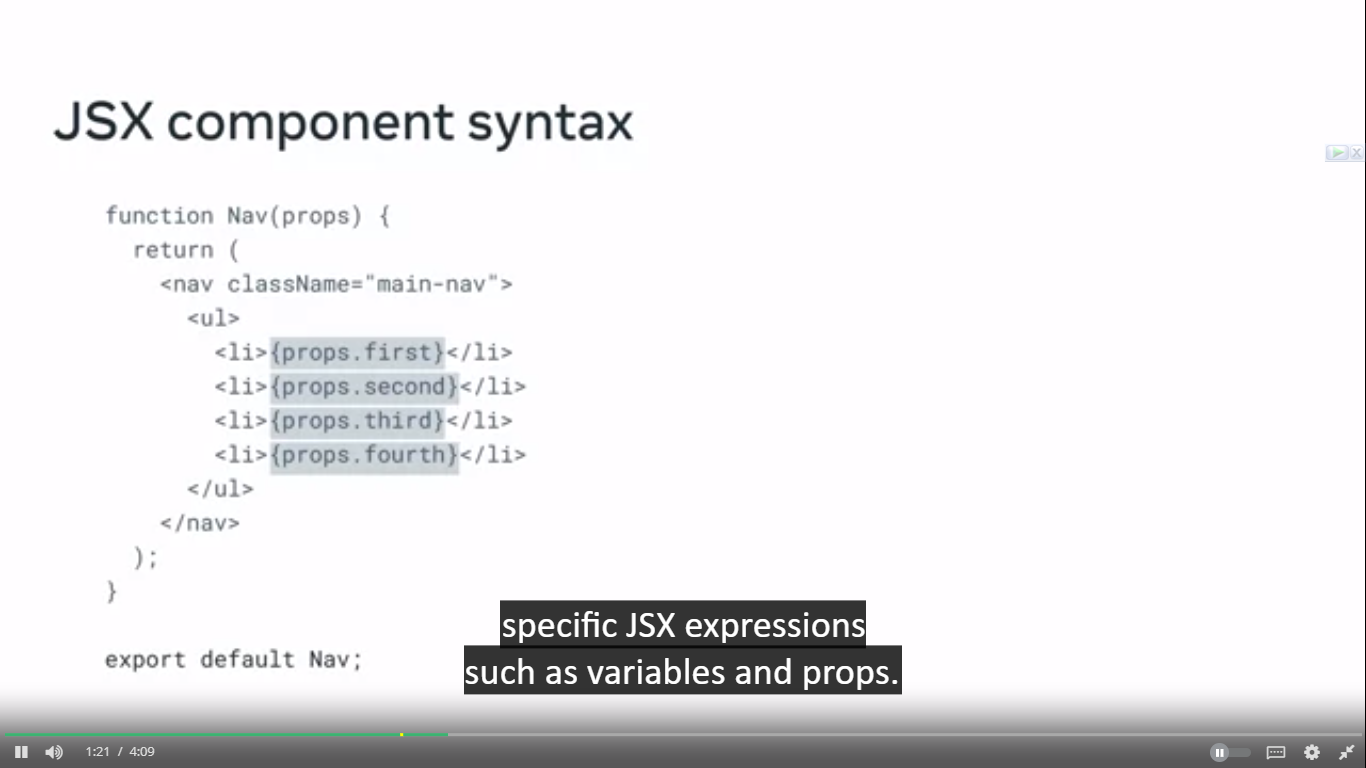
1. The wrapping element to render.
2. A null value (which is there to show an absence of an expected JavaScript object value).
3. The inner content that will go inside the wrapping element.

|  |  |
| --- | --- |
| "use strict";  function App() {    return /\*#\_\_PURE\_\_\*/React.createElement("div", null, /\*#\_\_PURE\_\_\*/React.createElement("h1", null, "Hello there"));  } | function App() {    return React.createElement(      "div",      null,      React.createElement("h1", null, "Hello there")    );  } |

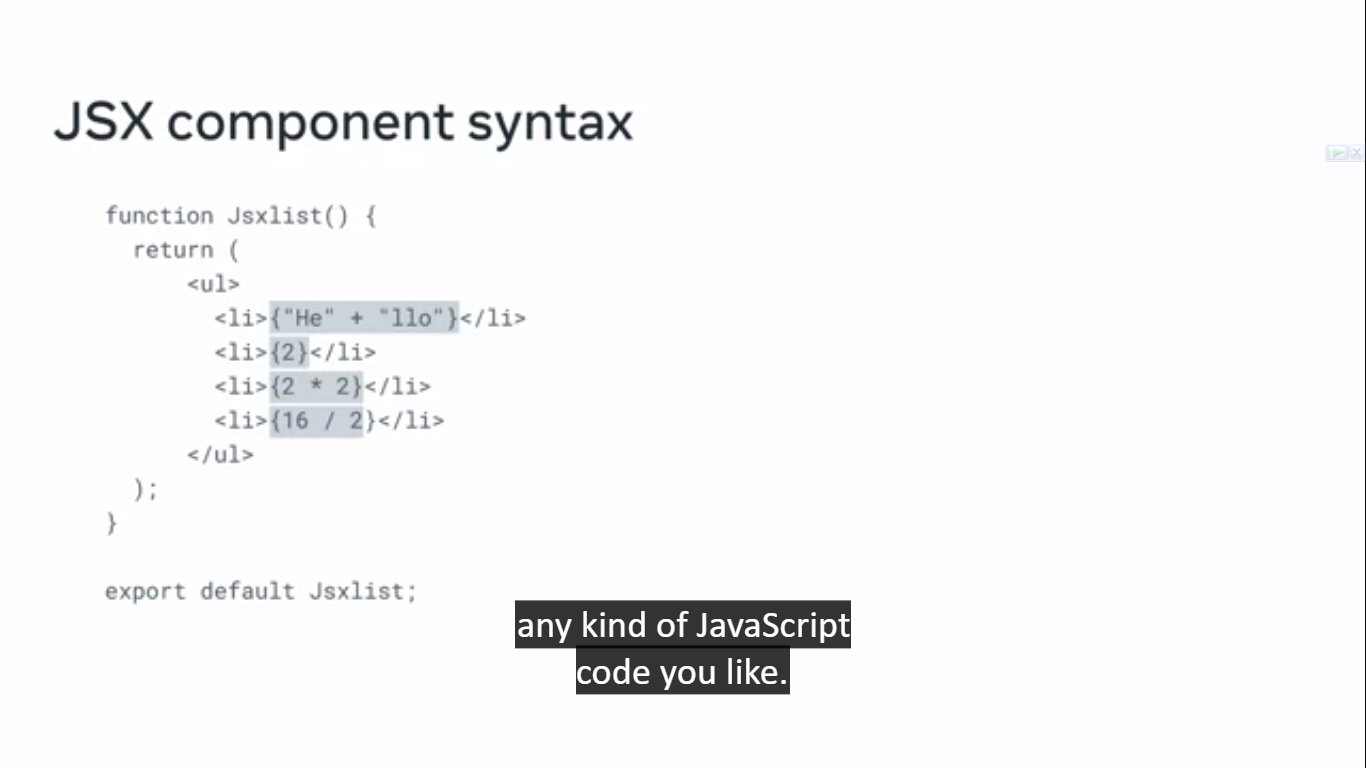
**Jsx:**



almost identical to HTML or XML.



Inside a the curly braces you can write regular js code where it is any kind of variable or expression.



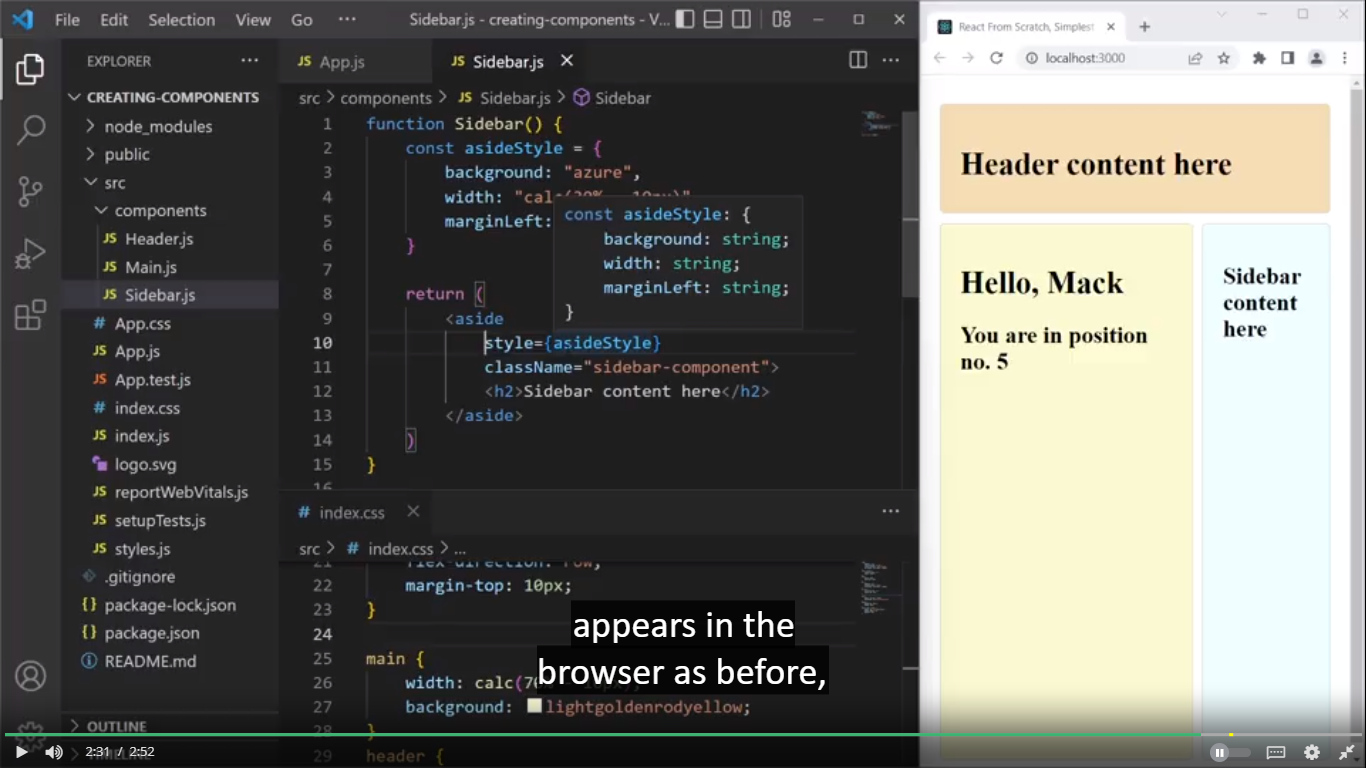
**Some rules of jsx:**

* 1. **To return multiple lines use ()**
  2. **Put them inside a parent element like div or fragment <></>**
  3. **You can’t use class keyword for css use className.**
  4. **props.children** property is a special prop that allows you to pass components or elements as children to another component. It is used when you want to create a component that acts as a wrapper or container for other components or elements.

|  |
| --- |
| **// WrapperComponent.js**  **import React from 'react';**  **const WrapperComponent = (props) => {**  **return (**  **<div className="wrapper">**  **{/\* Render the children \*/}**  **{props.children}**  **</div>**  **);**  **};**  **export default WrapperComponent;** |
|  |
| **// App.js**  **import React from 'react';**  **import WrapperComponent from './WrapperComponent';**  **const App = () => {**  **return (**  **<WrapperComponent>**  **<h1>Hello, World!</h1>**  **<p>This is a paragraph inside the wrapper.</p>**  **<button>Click me</button>**  **</WrapperComponent>**  **);**  **};**  **export default App;** |

**Styling with jsx**

|  |
| --- |
| function Promo(props) {  const styles = {      color: "tomato",      fontSize: "40px"  }  return (          <div className="promo-section">              <div>                  <h1 style={styles}>                      {props.heading}                  </h1>              </div>              <div>                  <h2>{props.promoSubHeading}</h2>              </div>          </div>      );  } |



Using React, you can easily convert a CSS rule to a JavaScript object, where each key-value pair describes a CSS declaration.

Ternary operators in jsx:

name == Bob ? "Yes, it is Bob" : "I don't know this person";

|  |
| --- |
| function Example() {      return (          <div className="heading">              <h1>{Math.random() >= 0.5 ? "Over 0.5" : "Under 0.5"}</h1>          </div>      );  }; |

**Expression as props:**

|  |
| --- |
| const bool = false;  const str1 = "just";  function Example(props) {      return (          <div>              <h2>                  The value of the toggleBoolean prop is:{props.toggleBoolean.toString()}              </h2>              <p>The value of the math prop is: <em>{props.math}</em></p>              <p>The value of the str prop is: <em>{props.str}</em></p>          </div>      );  };  export default function App() {      return (          <div className="App">              <Example                  toggleBoolean={!bool}                  math={(10 + 20) / 3}                  str={str1 + ' another ' + 'string'}              />          </div>      );  }; |