**Avoiding npm**

In order to avoid setting up npm for the sake of a *single file*, we can reference a CDN for the two main React libraries, react and react-dom:

Alternatively, you can manually save those two JS files into your workspace and reference them locally. (it’s no big concern for a sample file, but I did on one occasion find the CDN had been updated with a broken file, which broke my sample project)

Like\_button.js

'use strict';

const e = React.createElement;

class LikeButton extends React.Component {

constructor(props) {

super(props);

this.state = { liked: false };

}

render() {

if (this.state.liked) {

return 'You liked this.';

}

return e(

'button',

{ onClick: () => this.setState({ liked: true }) },

'Like'

);

}

}

const domContainer = document.querySelector('#like\_button\_container');

ReactDOM.render(e(LikeButton), domContainer);

Index.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<title>Add React in One Minute</title>

</head>

<body>

<h2>Add React in One Minute</h2>

<p>This page demonstrates using React with no build tooling.</p>

<p>React is loaded as a script tag.</p>

<!-- We will put our React component inside this div. -->

<div id="like\_button\_container"></div>

<!-- Load React. -->

<!-- Note: when deploying, replace "development.js" with "production.min.js". -->

<script src="https://unpkg.com/react@16/umd/react.development.js" crossorigin></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js" crossorigin></script>

<!-- Load our React component. -->

<script src="like\_button.js"></script>

</body>

</html>

HTML file with JSX

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8" />

<title>Hello World</title>

<script src="https://unpkg.com/react@16/umd/react.development.js"></script>

<script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"></script>

<!-- Don't use this in production: -->

<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

</head>

<body>

<div id="root"></div>

<script type="text/babel">

ReactDOM.render(

<h1>Hello, world!</h1>,

document.getElementById('root')

);

</script>

<!--

Note: this page is a great way to try React but it's not suitable for production.

It slowly compiles JSX with Babel in the browser and uses a large development build of React.

Read this section for a production-ready setup with JSX:

https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project

In a larger project, you can use an integrated toolchain that includes JSX instead:

https://reactjs.org/docs/create-a-new-react-app.html

You can also use React without JSX, in which case you can remove Babel:

https://reactjs.org/docs/react-without-jsx.html

-->

</body>

</html>

This approach is fine for learning and creating simple demos. However, it makes your website slow and **isn’t suitable for production**. When you’re ready to move forward, remove this new <script> tag and the type="text/babel" attributes you’ve added. Instead, in the next section you will set up a JSX preprocessor to convert all your <script> tags automatically.

JSX Started code

'use strict';

class LikeButton extends React.Component {

constructor(props) {

super(props);

this.state = { liked: false };

}

render() {

if (this.state.liked) {

return 'You liked this.';

}

return (

<button onClick={() => this.setState({ liked: true }) }>

Like

</button>

);

}

}

let domContainer = document.querySelector('#like\_button\_container');

ReactDOM.render(<LikeButton />, domContainer);

### **Add JSX to a Project**

Adding JSX to a project doesn’t require complicated tools like a bundler or a development server. Essentially, adding JSX **is a lot like adding a CSS preprocessor.** The only requirement is to have [Node.js](https://nodejs.org/) installed on your computer.

Go to your project folder in the terminal, and paste these two commands:

1. **Step 1:** Run npm init -y (if it fails, [here’s a fix](https://gist.github.com/gaearon/246f6380610e262f8a648e3e51cad40d))
2. **Step 2:** Run npm install babel-cli@6 babel-preset-react-app@3

**Tip**

We’re **using npm here only to install the JSX preprocessor;** you won’t need it for anything else. Both React and the application code can stay as <script> tags with no changes.

Congratulations! You just added a **production-ready JSX setup** to your project.

### Run JSX Preprocessor

Create a folder called src and run this terminal command:

npx babel --watch src --out-dir . --presets react-app/prod

**Note**

npx is not a typo — it’s a [package runner tool that comes with npm 5.2+](https://medium.com/@maybekatz/introducing-npx-an-npm-package-runner-55f7d4bd282b).

If you see an error message saying “You have mistakenly installed the babel package”, you might have missed [the previous step](https://reactjs.org/docs/add-react-to-a-website.html#add-jsx-to-a-project). Perform it in the same folder, and then try again.