qwik | qwik-ts

Scaffolding Your First Vite Project

Compatibility Note

Vite requires Node.js version 18+. 20+. However, some templates require a higher Node.js version to work, please upgrade if your package manager warns about it.

NPM | Yarn | PNPM | Bun

```
$ npm create vite@latest
```

Then follow the prompts!

You can also directly specify the project name and the template you want to use via additional command line options. For example, to scaffold a Vite + Vue project, run:

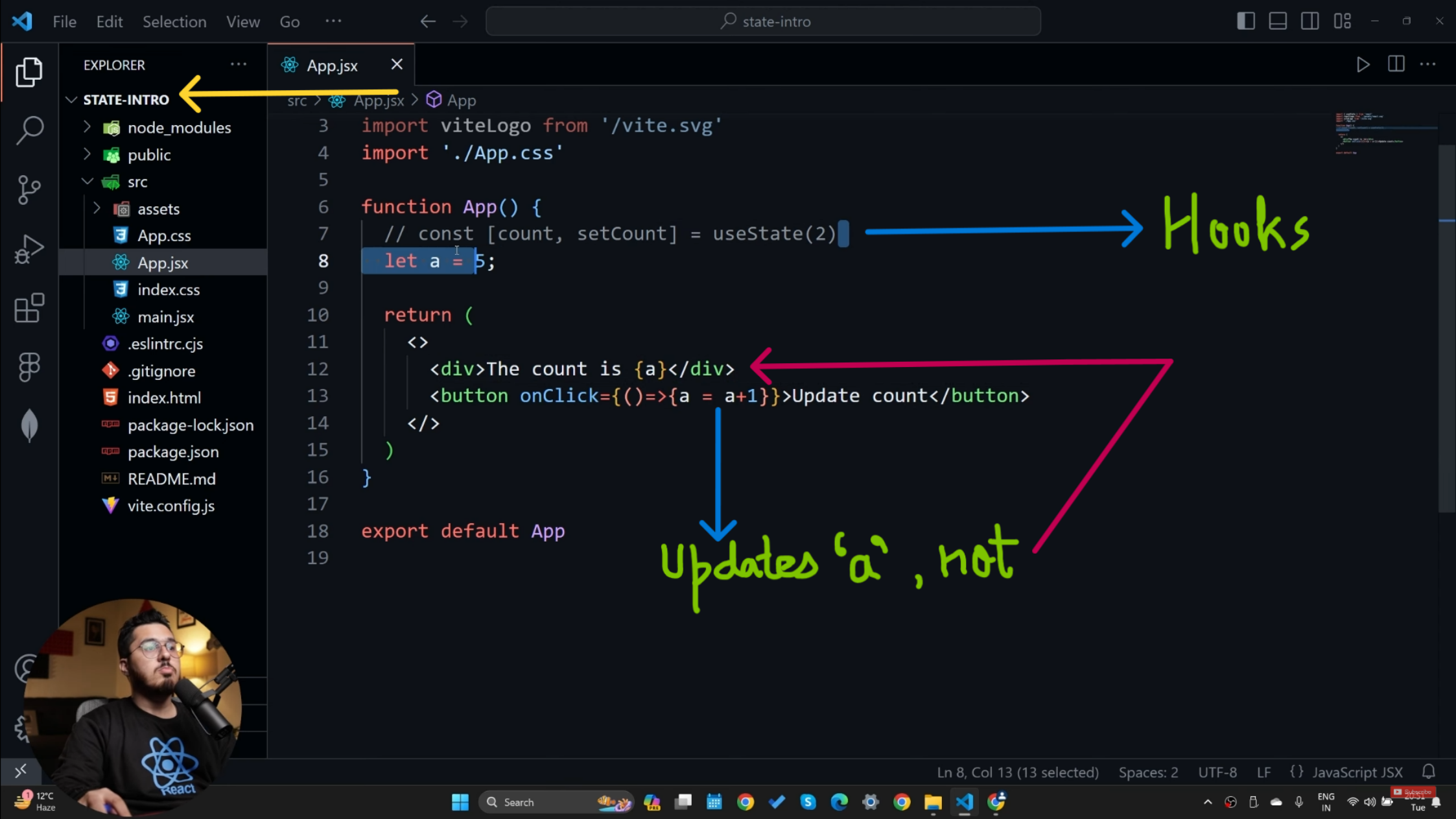
```bash
# npm 7+, extra double-dash is needed:
npm create vite@latest my-vue-app -- --template vue

# yarn
yarn create vite my-vue-app --template vue

# pnpm
pnpm create vite my-vue-app --template vue

# bun
bunx create-vite my-vue-app --template vue
```

See create-vite for more details on each supported template: `vanilla`, `vanilla-ts`, `vue`, `vue-ts`, `react`, `react-ts`, `react-swc`, `react-swc-ts`, `preact`, `preact-ts`, `lit`, `lit-ts`,


File Edit Selection View Go
state-intro
EXPLORER
App.jsx
STATE-INTRO
node_modules
public
src
assets
App.css
App.jsx
index.css
main.jsx
.eslintrc.cjs
.gitignore
index.html
package-lock.json
package.json
README.md
vite.config.js
src > App.jsx > App
3 import viteLogo from '/vite.svg'
4 import './App.css'
5
6 function App() {
7 // const [count, setCount] = useState(2)
8 let a = 5;
9
10 return (
11 <>
12 The count is {a}
13 <button onClick={()=>{a = a+1}}>Update count</button>
14 </>
15)
16 }
17
18 export default App
19
Hooks
Updates 'a', not
Ln 8, Col 13 (13 selected) Spaces: 2 UTF-8 LF { } JavaScript JSX
12°C Haze
Search
ENG IN
Tue

```
import viteLogo from '/vite.svg'
import './App.css'

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

  return (
    <>
      <div>The count is {count}</div>
      <button onClick={()=>{setCount(count + 1)}}>Update count</button>
    </>
  )
}

export default App
```

Built-in React Hooks

Hooks let you use different Rèact features from your components. You can either use the built-in Hooks or combine them to build your own. This page lists all built-in Hooks in React.

State Hooks

State lets a component "remember" information like user input. For example, a form component can use state to store the input value, while an image gallery component can use state to store the selected image index.

To add state to a component, use one of these Hooks:

- `useState` declares a state variable that you can update directly.
- `useReducer` declares a state variable with the update logic inside a reducer function.

```
function ImageGallery() {
  const [index, setIndex] = useState(0);
  // ...
```

Context Hooks

Context lets a component receive information from distant parents without passing it as props. For example, your app's top-level component can pass the current UI theme to all components below, no matter how deep.

- `useContext` reads and subscribes to a context.