Creating a Vite project

Now that we understand more about the powerful combination of TypeScript and Vite, let's dive into the demo portion of this tutorial.

First, ensure that you have Node.js ≥v18 installed on your machine, then create a Vite project by running the following command in the terminal:

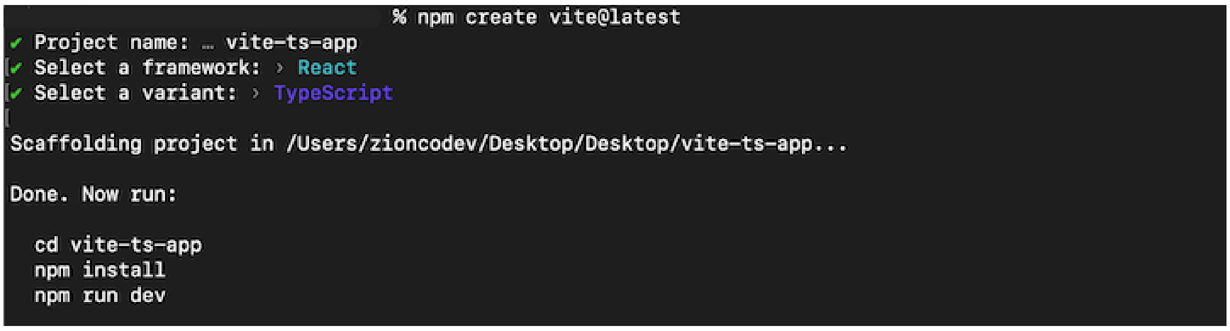
```
npm create vite@latest
```

This command will prompt you to choose a name for your project. Feel free to choose any name; then press **Enter** to continue. For this demonstration, we'll use the project name `vite-ts-app`.

Next, you'll be asked to select a framework for your Vite project. Vite provides a variety of frameworks that may be used for an application: React, Vue.js, Lit, Preact, vanilla JavaScript, and Svelte. For this demo, we'll select **React**.

Lastly, you'll be prompted to choose a variant for your application. For this demo, we're building a TypeScript app with Vite, so we'll select **TypeScript**.

Here are our selections for the Vite project prompts:



```
                                      % npm create vite@latest
✔ Project name: … vite-ts-app
✔ Select a framework: › React
✔ Select a variant: › TypeScript

Scaffolding project in /Users/zioncodev/Desktop/Desktop/vite-ts-app...

Done. Now run:

  cd vite-ts-app
  npm install
  npm run dev
```

Project structure

After processing the project information we just submitted, Vite will generate the project's folder structure:

```
📦vite-ts-app
 ┣ 📂public
 ┃ ┗ 📜vite.svg
 ┣ 📂src
 ┃ ┣ 📂assets
 ┃ ┃ ┗ 📜react.svg
 ┃ ┣ 📜App.css
 ┃ ┣ 📜App.tsx
 ┃ ┣ 📜index.css
 ┃ ┣ 📜main.tsx
 ┃ ┗ 📜vite-env.d.ts
 ┣ 📜.gitignore
```

```
├ 📜index.html
├ 📜package-lock.json
├ 📜package.json
├ 📜tsconfig.json
├ 📜tsconfig.node.json
└ 📜vite.config.ts
```

Below are the key files from the `vite-ts-app` project folder:

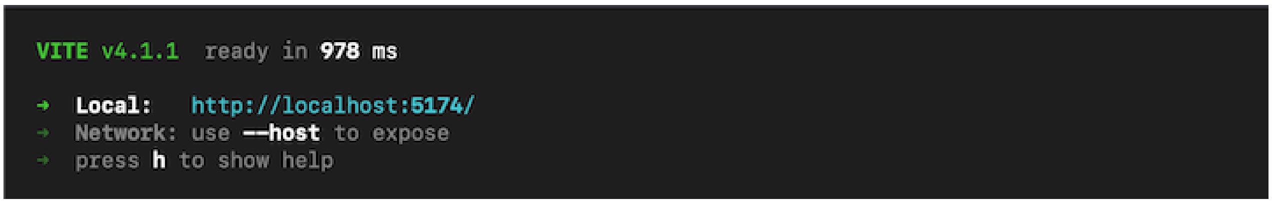
- `index.html`: The main file, typically found in a public directory in a Vite project
- `main.tsx`: Where the code for producing the browser output is executed. This file is common for Vite projects
- `vite.config.json`: The configuration file for any Vite project

Running the application

We've completed the prompts to create a Vite project. Now, let's cd into the project folder and use the below commands to run the application:

```
cd vite-ts-app
npm install
npm run dev
```

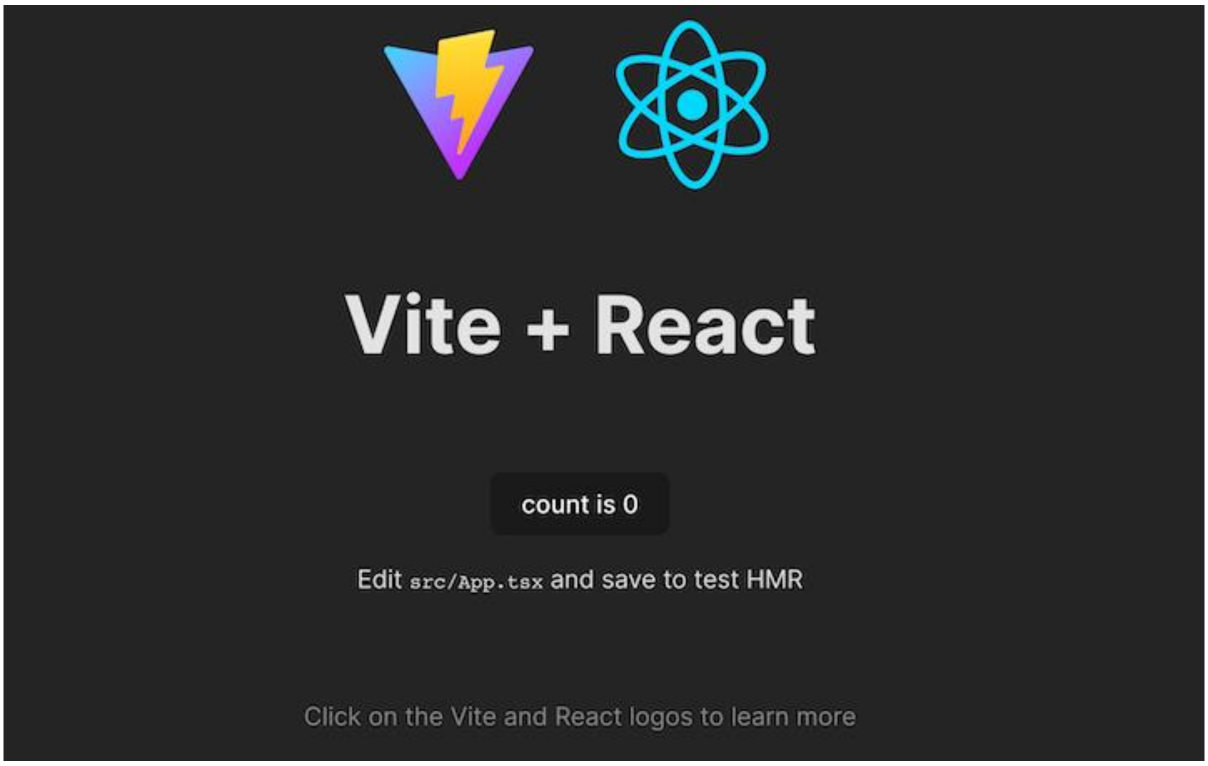
To confirm that the application is running, check the terminal — and you should see the following:



```
VITE v4.1.1  ready in 978 ms

➜  Local:   http://localhost:5174/
➜  Network: use --host to expose
➜  press h to show help
```

Press the `o` key to open the application in your web browser:


Vite + React
count is 0
Edit src/App.tsx and save to test HMR
Click on the Vite and React logos to learn more

Building a blog application

With the Vite app up and running in our browser, let’s create a blog application using Vite and React that renders some static blog data from a JSON file.

To get started, let’s update the code in the `App.tsx` file to add a navbar to the application’s UI:

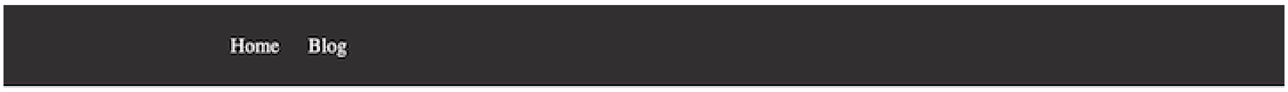
```
import './App.css'
function App() {

 return (
  <div className="App">
   <div className="navbar">
    <ul>
     <li>Home</li>
     <li>Blog</li>
    </ul>
   </div>
  </div>
 )
}
export default App
```

Next, let’s update the `App.css` file to add some new styles to the application:

```
* {
 padding: 0px;
 margin: 0px;
 box-sizing: border-box;
}
.navbar {
 background-color: rgb(50, 47, 47);
 color: white;
 padding: 10px;
}
.navbar ul {
 display: flex;
 width: 600px;
 margin: 0px auto;
 font-size: 14px;
 list-style: none;
}
.navbar ul li {
 margin: 10px;
}
```

The resulting UI will look like the following:



Creating the blog data

Next, we'll need to add data to our blog application. Let's create a `blog.json` file in the project's root directory and add the following data:

```
[
 {
  "id": 1,
  "title": "Building a Todo App with Vue",
  "cover": "https://nextjs.org/static/images/learn/foundations/next-
app.png",
  "author":"John Doe"
 },
 {
  "id": 2,
  "title": "Getting started with TypeScript",
  "cover":
"https://nextjs.org/static/images/learn/foundations/components.png",
  "author":"Claman Joe"
 }
]
```

Here we defined some arrays of blog objects, which we'll render in our Vite app's UI.

Creating a blog component

Now, let's create a `components` folder in the `src` directory. Then, we'll create a `Blog.tsx` file and add the below snippet:

```
import blogData from '../../blog.json'
type Blog = {
  id: number,
  title: string,
  cover: string,
  author: string
}
```

```
export function Blog() {
  return (
    <div className="container">
      <div className="blog">
        {blogData.map((blog: Blog) =>
          <div className="card" key={blog.id}>
            <img src={blog.cover} alt="" />
            <div className="details">
              <h2>{blog.title}</h2>
              <h4>{blog.author}</h4>
            </div>
          </div>
        )}
      </div>
    </div>
  )
}
```

This code defines a function that returns a container for blog posts and includes a list of blog cards. Each card displays the title, cover image, and blog post author. The code uses a `map` function to loop through a `blogData` array and create a `card` for each item.

Next, let's update the `App.css` file to style the `Blog` component:

```
.App {
 background: rgb(44, 183, 134);
 height: 100vh;
}
.container {
 width: 600px;
 margin: 0px auto;
}
.container .blog {
 display: flex;
 padding: 10px;
}
.container .card {
 background-color: white;
 margin: 10px;
 padding: 10px;
 border-radius: 4px;
 width: 50%;
 font-size: 10px;
 color: rgb(50, 47, 47);
}
.container .card img {
 width: 100%;
}
```

Lastly, let's update the `App.tsx` component to import and render the `Blog` component:

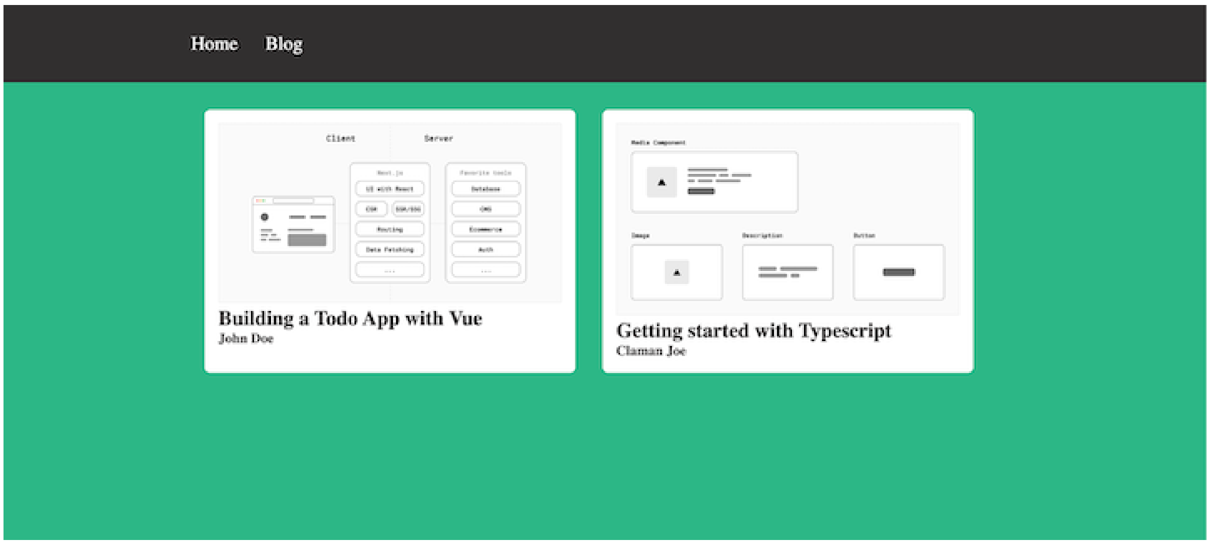
```
import './App.css'
import { Blog} from './components/Blog'

function App() {

 return (
  <div className="App">
   <div className="navbar">
```

```
    <ul>
     <li>Home</li>
     <li>Blog</li>
    </ul>
   </div>
    <Blog />
  </div>
 )
}
export default App
```

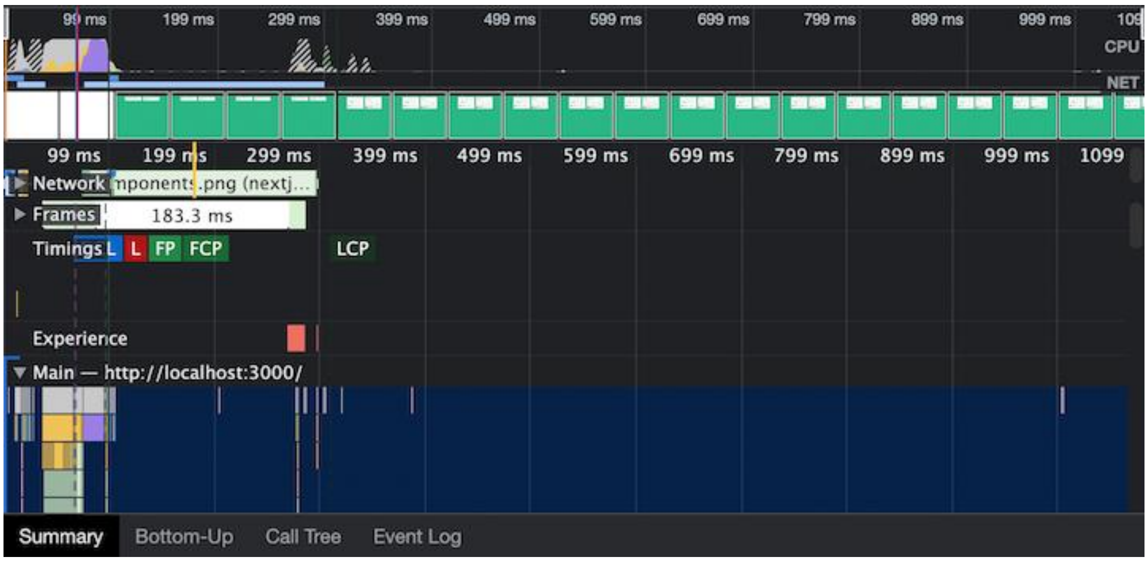
And with that, we’ve successfully created a blog application using TypeScript and Vite! If all went well, it should look like the image below:



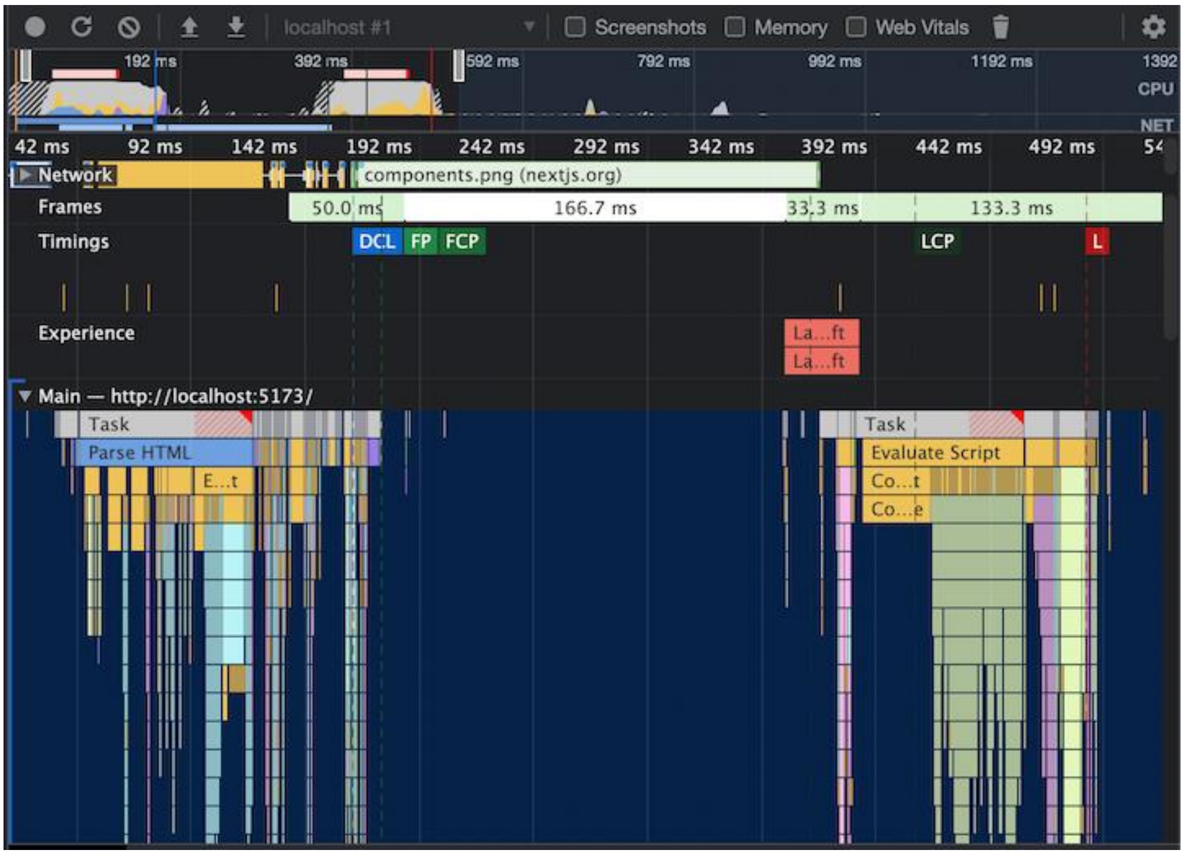
Performance comparison: CRA vs. Vite

To compare the startup time of a Vite app to an app built with an alternative like Create React App (CRA), we’d need to build and test both apps under similar conditions. To demonstrate this, I built the same demo application that we just created in this tutorial, except I used CRA. Then, I used the performance inspection feature in Chrome DevTools to test the start time for each version of the app.

Here’s the performance result for the TypeScript app built with CRA; the startup time was 99ms:



And here's the performance of the TypeScript app built with Vite; the startup time was 42ms:



In our test, the TypeScript application built with Vite started 58 percent faster than the TypeScript application built with Create React App.

Conclusion

In this article, we discussed the many benefits of combining React, TypeScript, and Vite, demonstrated how to build a simple React-based blog application using TypeScript and Vite, and then compared the performance of our app with that of a TypeScript app built with Create React App.