تحقیق درس برنامه سازی وب

دانشگاه صنعتی شریف

سپهر ذوالفقاری ۴۰۱۱۰۵۹۲۳

باربد شهرآبادی ۴۰۱۱۰۶۱۲۵

علی امینی ۴۰۱۱۷۰۵۲۹

پاییز ۱۴۰۳

جدول

مقدمه

این گزارش، نحوه ی استفاده از ابزار Vite برای ساخت کد React و کارکرد آن را بررسی می‌کند و اصلا می‌بینیم که این ابزار چگونه به کمک می‌کند. اما اصلا خود Vite چیست؟ ویت (Vite) یک ابزار ساخت مدرن و سریع و سرور توسعه برای پروژه‌های وب است که با هدف بهبود تجربه توسعه‌دهنده طراحی شده است. این ابزار بر ارائه جایگزینی ماژول داغ (HMR) سریع و بهینه‌سازی عملکرد ساخت با استفاده از ماژول‌های ES محلی و ویژگی‌های مدرن جاوا اسکریپت تمرکز دارد. با استفاده از یک سرور توسعه سبک و سریع و بهینه‌سازی دارایی‌ها (Assests) در زمان ساخت، ویت به طور چشمگیری سرعت و کارایی فرآیندهای توسعه فرانت‌اند را بهبود می‌بخشد.

پیش نیاز ها

قبل از ادامه دادن این تحقیق و بررسی کامل جوانب ویت، نیاز به بررسی و فهمیدن چند مفهوم مهم داریم که در ادامه به فهم این مطالب به ما کمک خواهند کرد. در نتیجه با بخش پیش نیاز شرع میکنیم.

۱. ES Modules چیست؟

ماژول‌های ES (ESM) به استاندارد رسمی برای کار با ماژول‌ها در جاوا اسکریپت اشاره دارند که در نسخه ES6 (اکمااسکریپت ۲۰۱۵) معرفی شدند. به زبان ساده‌تر، این یک روش برای تقسیم کد جاوا اسکریپت به قطعات کوچک‌تر و قابل استفاده مجدد، یا ماژول‌ها، است که سپس می‌توان آن‌ها را وارد یا صادر کرد.

وارد کردن (Import) : به شما این امکان را می‌دهد که عملکرد خاصی را از یک ماژول دیگر وارد کنید.
خروجی گرفتن (Export): به شما این امکان را می‌دهد که بخش‌هایی از ماژول خود را برای دسترسی سایر ماژول‌ها قابل دسترسی کنید.

برای مثال در این شکل یک نوع از این کار را می‌بینید:

```
// در myModule.js
export function sayHello() {
    console.log('Hello, world!');
}

// در main.js
import { sayHello } from './myModule.js';
sayHello();  // خروجی: Hello, world!
```

شکل ۱

۲. مفهوم جایگزینی ماژول داغ (HNR) :

جایگزینی ماژول داغ (HMR) ویژگی‌ای در ابزارهای توسعه مدرن است که به شما این امکان را می‌دهد که تنها کد تغییر یافته را در مرورگر جایگزین کنید بدون اینکه نیازی به تازه‌سازی کامل صفحه باشد.

هنگامی که یک فایل را در پروژه خود تغییر می‌دهید مثلاً به‌روزرسانی یک کامپوننت React ، HMR بلافاصله تنها بخش تغییر یافته را در مرورگر به‌روزرسانی می‌کند .این ویژگی تجربه توسعه سریع‌تری را فراهم می‌آورد زیرا نیازی به صبر کردن برای بارگذاری دوباره کل صفحه ندارید. این ویژگی به ویژه زمانی مفید است که با اجزای رابط کاربری (UI) کار می‌کنید.

مثال :هنگامی که با React کار می‌کنید، یک کامپوننت را تغییر می‌دهید و HMR تغییر را بلافاصله در مرورگر نشان می‌دهد بدون اینکه وضعیت برنامه از دست برود.

۳. مفهوم بسته‌بندی (Bundling) :

بسته‌بندی فرآیند ترکیب تمام فایل‌های جاوا اسکریپت و سایر دارایی‌ها مانند CSS یا تصاویر به یک فایل یا چند فایل است تا بتوان آن‌ها را به‌طور مؤثر به مرورگر ارائه داد. هدف اصلی از بسته‌بندی کاهش تعداد درخواست‌های شبکه‌ای است که مرورگر باید برای بارگذاری اپلیکیشن شما انجام دهد.

چرا؟ فایل‌های جاوا اسکریپت معمولاً به قطعات کوچک‌تر تقسیم می‌شوند، بنابراین بسته‌بندی آن‌ها را به یک فایل ترکیب می‌کند که زمان بارگذاری را کاهش می‌دهد.

مثال :شما ممکن است چندین فایل جاوا اسکریپت کوچک (یکی برای هر ماژول) داشته باشید، اما زمانی که مرورگر وب‌سایت شما را بارگذاری می‌کند، نمی‌خواهید برای هر ماژول فایل جدیدی بارگذاری شود. بسته‌بندی آن‌ها را ترکیب می‌کند.

در ابزارهای مدرن مانندVite ، بسته‌بندی در هنگام ساخت نسخه تولید (production build) انجام می‌شود تا سرعت بارگذاری بهینه شود.

esbuild یک ابزار بسته‌بندی و فشرده‌سازی جاوا اسکریپت است. این یک ابزار سریع نوشته شده در زبان Go است که می‌تواند جاوا اسکریپت مدرن مانندJSX ، TypeScriptرا ترنسپایل کرده و برای مرورگر بسته‌بندی کند. این ابزار نسبت به ابزارهای قدیمی‌تر مانند Webpack بسیار سریع‌تر است، زیرا در زبان Go نوشته شده و به‌صورت موازی عمل می‌کند.

۴. مفهوم Static Site Generation (SSG) :

تولید سایت ایستا (SSG) روشی است که در آن صفحات HTML در زمان ساخت پیش‌ساخته می‌شوند به‌جای اینکه به‌صورت پویا در زمان اجرا تولید شوند.

سایت‌های ایستا :این‌ها وب‌سایت‌هایی هستند که شامل فایل‌های ثابت و پیش‌ساختهHTML ، CSS، جاوا اسکریپت هستند که مرورگر می‌تواند آن‌ها را به‌طور مستقیم به کاربر ارائه دهد بدون اینکه نیاز باشد هر بار با سرور ارتباط برقرار کند.

چطور؟ باSSG ، HTMLبرای هر صفحه زمانی که اپلیکیشن خود را می‌سازید مثلاً با استفاده از ابزاری مانند Vite یا Next.js تولید می‌شود و سپس آماده است تا توسط سرور یا CDN ارائه شود.

مثال :اگر یک وبلاگ با استفاده از SSG بسازید، هر پست وبلاگ به‌عنوان یک صفحه HTML ایستا در زمان ساخت تولید می‌شود، بنابراین زمانی که کسی به صفحه مراجعه می‌کند، این صفحه به‌طور فوری از سرور ارائه می‌شود.

۵. مفهوم ماژول های بومی ES

ماژول‌های ES بومی همان ماژول‌های ES هستند که مستقیماً توسط مرورگرهای وب مدرن پشتیبانی می‌شوند و نیازی به بسته‌بندی‌کننده(تعریف شده در بخش قبل) یا ترنسپایلر ندارند.

این بدین معنی است که می‌توانید از دستورات وارد کردن و صادرات import و export در جاوا اسکریپت خود به‌طور مستقیم در مرورگر استفاده کنید و مرورگر به‌طور داینامیک ماژول‌ها را بارگذاری و اجرا می‌کند. برای مثال می‌توانید در این عکس ببینید:

```
<script type="module">
  import { sayHello } from './myModule.js';
  sayHello();
</script>
```

شکل ۲

مرورگر myModule.js را به‌عنوان یک ماژول بارگذاری خواهد کرد، بدون اینکه نیازی به بسته‌بندی آن با دیگر فایل‌های جاوا اسکریپت باشد.

۶. ریلود داغ چیست و چگونه Vite این مشکل را حل می‌کند؟

ریلود داغ به معنی تازه‌سازی خودکار بخشی از اپلیکیشن شما هنگام تغییر کد است، بدون اینکه نیاز به بارگذاری کامل صفحه باشد. این ویژگی فرآیند توسعه را سریع‌تر کرده و وضعیت اپلیکیشن را حفظ می‌کند.

Vite این کار را با استفاده از جایگزینی ماژول داغ (HMR) انجام می‌دهد:

مکانیزم :زمانی که یک فایل مانند یک کامپوننت React تغییر می‌کند، Vite تنها آن فایل را در مرورگر به‌روزرسانی می‌کند، اپلیکیشن همچنان در حال اجراست و وضعیت آن حفظ می‌شود .این فرآیند بسیار سریع‌تر از بارگذاری کامل صفحه است، زیرا اپلیکیشن نیازی به راه‌اندازی مجدد کل صفحه ندارد.

حالا که با پیش نیاز ها آشنا شدیم، به سراغ معرفی خود Vite می رویم.

معرفی ابزار و کاربرد های آن

Vite چه ابزاری است؟

ویت یک ابزار ساخت نسل بعدی و سریع برای توسعه وب است. این ابزار تمرکز خود را بر ارائه یک تجربه توسعه فوق‌العاده سریع برای پروژه‌های وب مدرن، به‌ویژه فریم‌ورک‌های فرانت‌اند مانندReact ، Vueیا Svelte گذاشته است. ویت از ویژگی‌های مدرن

جاوا اسکریپت مانند ماژول‌های ES و پشتیبانی مرورگرها از Import ماژول‌های بومی استفاده می‌کند. این ابزار برای هر دو بخش توسعه (با ویژگی‌هایی مانند جایگزینی ماژول داغ) و تولید (با فرآیند بسته‌بندی بهینه‌شده) بهینه شده است.

کاربردها و موارد استفاده ویت

- **سرور توسعه** : ویت یک سرور توسعه فوق‌العاده سریع فراهم می‌کند که از جایگزینی ماژول داغ (HMR) پشتیبانی می‌کند. این به این معنی است که می‌توانید تغییرات اپلیکیشن خود را تقریباً به‌طور فوری مشاهده کنید بدون نیاز به تازه‌سازی صفحه.
- **بسته‌بندی** : ویت از **esbuild** برای بسته‌بندی در تولید استفاده می‌کند. این ابزار کد را سریع‌تر از بسته‌بندهای سنتی مانند Webpack کامپایل می‌کند که باعث کاهش زمان ساخت و بهبود عملکرد می‌شود.
- **بهینه‌سازی برای فریم‌ورک‌های مدرن** : ویت به‌طور پیش‌فرض برای کار با فریم‌ورک‌هایی مانندReact ، Vueو Svelteطراحی شده است که کار ساخت پروژه‌ها با این تکنولوژی‌ها را بسیار آسان می‌کند.
- **پشتیبانی از ماژول‌های ES :** به‌جای بسته‌بندی سنتی برای توسعه، ویت از ماژول‌های بومی ES برای ارائه فایل‌های منبع در توسعه استفاده می‌کند که باعث تسریع فرآیند ساخت می‌شود.
- **تولید سایت ایستا (SSG)** : ویت از تولید سایت ایستا برای پیش‌ساخت صفحات و ساخت وب‌سایت‌های ایستا پشتیبانی می‌کند، که این ویژگی آن را برای اپلیکیشن‌های سازگار با SEO مناسب می‌سازد.

مزیت های ویت

ویت عمدتاً مشکلاتی را که از استفاده از ابزارهای ساخت سنتی مانند Webpack در توسعه وب مدرن به وجود می‌آید، حل می‌کند. در اینجا بررسی می‌کنیم که ویت چگونه این مشکلات را حل می‌کند:

زمان ساخت کند در هنگام توسعه (بارگذاری داغ):

مشکل :بسته‌بندهای سنتی مانند Webpack می‌توانند زمان زیادی را برای بازسازی و تازه‌سازی اپلیکیشن در حین توسعه صرف کنند، به‌ویژه وقتی که اندازه پروژه افزایش می‌یابد. این موضوع می‌تواند چرخه بازخورد را کند کند و باعث ناامیدی توسعه‌دهندگان شود.

راه‌حل :ویت فایل‌ها را در حین توسعه از طریق ماژول‌های ES بومی (ESM) ارائه می‌دهد، به این معنی که تنها ماژول‌های تغییر یافته بارگذاری می‌شوند، که منجر به جایگزینی سریع‌تر ماژول داغ (HMR) می‌شود. این فرآیند به‌شدت زمان لازم برای انتظار در هنگام تغییر کد را کاهش می‌دهد.

زمان ساخت طولانی برای تولید:

مشکل :بسته‌بندهایی مانند Webpack معمولاً زمان ساخت کندی برای تولید دارند زیرا به ترنسپایلرهای جاوا اسکریپت مانند Babelو بسته‌بندها مانند Webpack متکی هستند که به‌طور کامل از ویژگی‌های جدید جاوا اسکریپت استفاده نمی‌کنند، که منجر به ساخت‌های ناکارآمد می‌شود.

راه‌حل :ویت از **esbuild** نوشته شده در زبان Go برای ترنسپایل و بسته‌بندی کد استفاده می‌کند که به‌طور قابل توجهی سریع‌تر از ابزارهای مبتنی بر جاوا اسکریپت است. این ابزار کد را به‌طور موازی کامپایل می‌کند که منجر به زمان‌های ساخت سریع‌تر برای تولید می‌شود.

پیکربندی پیچیده:

مشکل :بسته‌بندهای قدیمی معمولاً نیاز به پیکربندی گسترده دارند، به‌ویژه زمانی که با فناوری‌های مختلف فرانت‌اند مانند React، Vue، TypeScriptو غیره کار می‌کنید. این می‌تواند زمان‌بر و مستعد خطا باشد.

راه‌حل :ویت پیکربندی صفر برای اکثر فریم‌ورک‌های فرانت‌اند فراهم می‌کند. این ابزار از ابزارهای مدرن مانندTypeScript ، JSXو Vue پشتیبانی بومی دارد و شروع به کار را بدون نیاز به دستکاری فایل پیکربندی پیچیده آسان می‌سازد.

شروع کند در هنگام توسعه (Cold Start):

مشکل :پروژه‌های مبتنی بر Webpack معمولاً زمان شروع اولیه کندی (cold start) دارند به دلیل نیاز به بسته‌بندی تمام ماژول‌ها به‌طور همزمان.

راه‌حل :ویت از Import ماژول‌های بومی ES در هنگام توسعه استفاده می‌کند. به‌جای بسته‌بندی همه چیز از ابتدا، کد منبع به‌طور جداگانه به مرورگر ارسال می‌شود که باعث زمان شروع تقریباً آنی می‌شود.

نحوه استفاده

باید دقت کنیم که هنگامی که شما یک اپلیکیشن React (یا سایر فریم‌ورک‌های فرانت‌اند) را با ویت می‌سازید، همچنان از جاوا اسکریپت استفاده می‌کنید. در واقع، ویت یک ابزار ساخت برای جاوا اسکریپت و سایر دارایی‌های وب مانند HTML و CSS است.

وظیفه اصلی ویت این است که کد جاوا اسکریپت را بسته‌بندی، بهینه‌سازی و ارائه دهد، همراه با فایل‌های CSS و .HTML حتی اگر شما با یک اپلیکیشن React کار می‌کنید که در آن ازJSX ، یک سینتکس اکستنشن برای جاوا اسکریپت استفاده می‌شود، در نهایت همه چیز در هنگام ساخت اپلیکیشن برای تولید به جاوا اسکریپت ترجمه می‌شود.

کد React : کامپوننت‌های React شما با استفاده از JSX نوشته می‌شوند، که سپس توسط ویت در فرآیند ساخت به جاوا اسکریپت ترنسپایل می‌شود.

ماژول‌های جاوا اسکریپت : کامپوننت‌ها، کتابخانه‌ها و وابستگی‌های React شما با استفاده از ماژول‌های(ES (ESM ، استاندارد مدرن جاوا اسکریپت وارد می‌شوند. بنابراین، ویت مسئولیت ارائه و بسته‌بندی این ماژول‌های جاوا اسکریپت را بر عهده دارد.

حال که اپلیکیشن را معرفی کردیم، به سراغ نصب آن، استفاده از آن و زدن مثال از آن می‌رویم.

در ابتدا باید ویت را نصب و آن را استفاده کنیم. برای نصب آن این مراحل را طی میکنیم:

گام ۱: نصب Node.js

Vite نیاز به نصب Node.js بر روی سیستم شما دارد، زیرا این ابزار بر روی Node ساخته شده است و از npm یا Yarn برای مدیریت وابستگی‌ها استفاده می‌کند.

بررسی نصب Node.js:

ترمینال/پرامپت دستورات خود را باز کنید و دستور زیر را وارد کنید:

node -v

اگر شماره نسخه‌ای (مثلاً v16.x.x) را مشاهده کردید، همه چیز تا اینجا اوکیه!

```
C:\Users\No1\Desktop\New Vite\my-react-app> node -v
v22.12.0
```

شکل ۳

گام ۲: ایجاد پروژه React با استفاده از Vite

حالا که Node.js نصب شده است، بیایید Vite را نصب کرده و یک پروژه React جدید ایجاد کنیم.

ترمینال خود را باز کنید و به پوشه‌ای که می‌خواهید پروژه را ایجاد کنید بروید.

برای ایجاد یک پروژه React جدید با استفاده از Vite دستور زیر را وارد کنید:

npm create vite@latest my-react-app --template react

my-react-app نام پروژه شما است. می‌توانید آن را با هر نامی که می‌خواهید جایگزین کنید. سپس در ادامه ری اکت و جاوا اسکریپت را انتخاب کنید.

template react-- به Vite می‌گوید که می‌خواهید یک اپلیکیشن React ایجاد کنید.

```
C:\Users\No1\Desktop\New Vite>npm create vite@latest my-react-app --template react

> npx
> create-vite my-react-app react

√ Select a framework: » React
√ Select a variant: » JavaScript

Scaffolding project in C:\Users\No1\Desktop\New Vite\my-react-app...

Done. Now run:

  cd my-react-app
  npm install
  npm run dev
```

شکل ٤

حالا به پوشه پروژه جدید بروید:

cd my-react-app

نصب وابستگی‌ها: پس از وارد شدن به پوشه پروژه، باید تمام وابستگی‌های لازم را نصب کنید:

npm install

```
C:\Users\No1\Desktop\New Vite\my-react-app>npm install

added 259 packages, and audited 260 packages in 52s

108 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

شکل ۵

گام ۳: راه‌اندازی سرور توسعه

حالا که پروژه شما تنظیم شده است، بیایید سرور توسعه را راه‌اندازی کنیم تا بتوانید شروع به کدنویسی کنید.

برای راه‌اندازی سرور توسعه دستور زیر را وارد کنید:

npm run dev

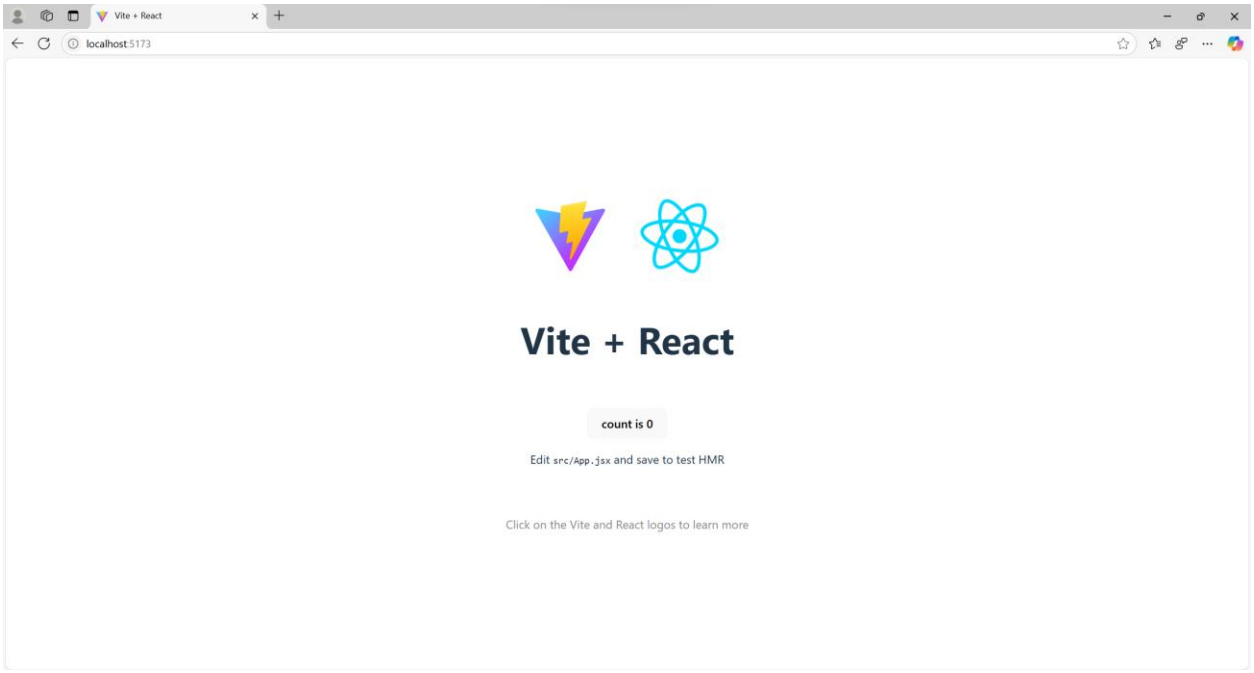
بررسی در مرورگر: بعد از اجرای دستور بالا، باید خروجی مشابه زیر را ببینید:

```
C:\Users\No1\Desktop\New Vite\my-react-app>npm run dev

> my-react-app@0.0.0 dev
> vite


  VITE v6.0.11  ready in 290 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

شکل ۶ : خروجی دستور بالا

دقت کنید الان در محیط توسعه دهی هستید و اگر برای سرور خود پروژه را می‌خواهید باید پس از آن، آن را بیلد کنید.

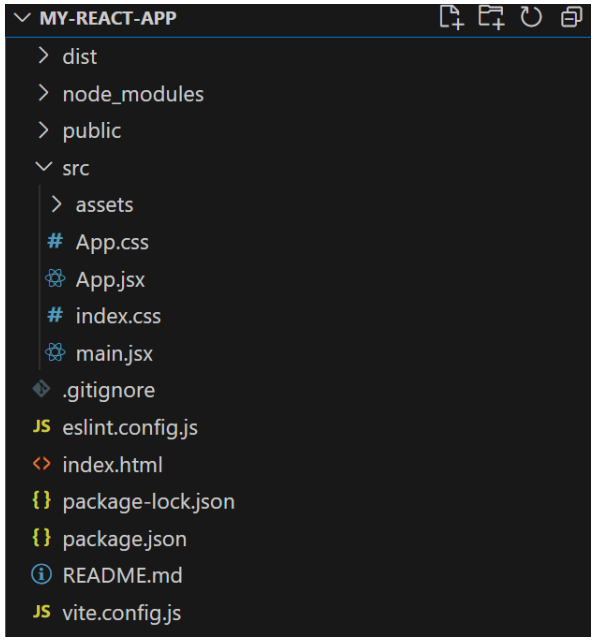
مرورگر خود را باز کنید و به آدرس http://localhost:5173/ بروید. شما باید اپلیکیشن React پیش‌فرض Vite را مشاهده کنید! به صورت دیفالت به این شکل است:



شکل ۷: دیفالت ویت

گام ۴: ساختار پروژه

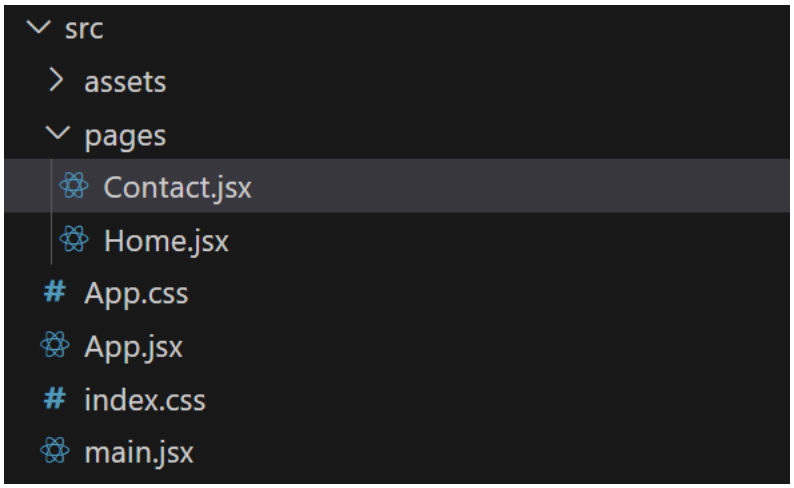
در شکل زیر یک نمای کلی از ساختار پروژه جدید شما آورده شده است.



شکل ۸

گام ۵: ویرایش کد

حالا آماده هستید که کدنویسی را شروع کنید! می‌توانید فایل src/App.jsx را ویرایش کنید تا رابط کاربری را به‌روزرسانی کرده یا ویژگی‌هایی به اپلیکیشن خود اضافه کنید. قابلیت جایگزینی ماژول داغ (HMR) به شما این امکان را می‌دهد که تغییرات را به‌طور زنده در مرورگر مشاهده کنید بدون اینکه نیاز به رفرش صفحه داشته باشید. برای مثال ما یک مثال ساده ری اکت میزنیم.ما یک فولدر به اسم pages درست میکنیم و در آن ۲ صفحه و مسیر Home.jsx و Contact.jsx می‌سازیم. سپس App و Main را به طور مناسب تغییر می‌دهیم:



شکل ۹

شکل ۱۰

```jsx
import React, { useState } from 'react';
const Contact = () => {
  const [message, setMessage] = useState('');

  const handleChange = (e) => {
    setMessage(e.target.value);
  };
  const handleSubmit = (e) => {
    e.preventDefault();
    alert(`Message Submitted: ${message}`);
    setMessage('');
  };
  return (
    <div>
      <h1>Contact Us</h1>
      <form onSubmit={handleSubmit}>
        <label>
          Your Message:
          <textarea
            value={message}
            onChange={handleChange}
            placeholder="Write your message here"
            rows="4"
            cols="50"
          />
        </label>
        <br />
        <button type="submit">Submit</button>
      </form>
    </div>
  );
};
export default Contact;
```

Contact.jsx | main.jsx | App.jsx | Home.jsx

my-vite-app > src > App.jsx > ...

```jsx
import React from 'react';
import { Routes, Route, Link } from 'react-router-dom';

import Home from './pages/Home';
import Contact from './pages/Contact';
import './App.css';
const App = () => {
  return (
    <div>
      <nav>
        <ul>
          <li><Link to="/">Home</Link></li>
          <li><Link to="/contact">Contact</Link></li>
        </ul>
      </nav>

      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/contact" element={<Contact />} />
      </Routes>
    </div>
  );
};

export default App;
```

شکل ۱۱

Contact.jsx | Home.jsx

my-vite-app > src > pages > Home.jsx > ...

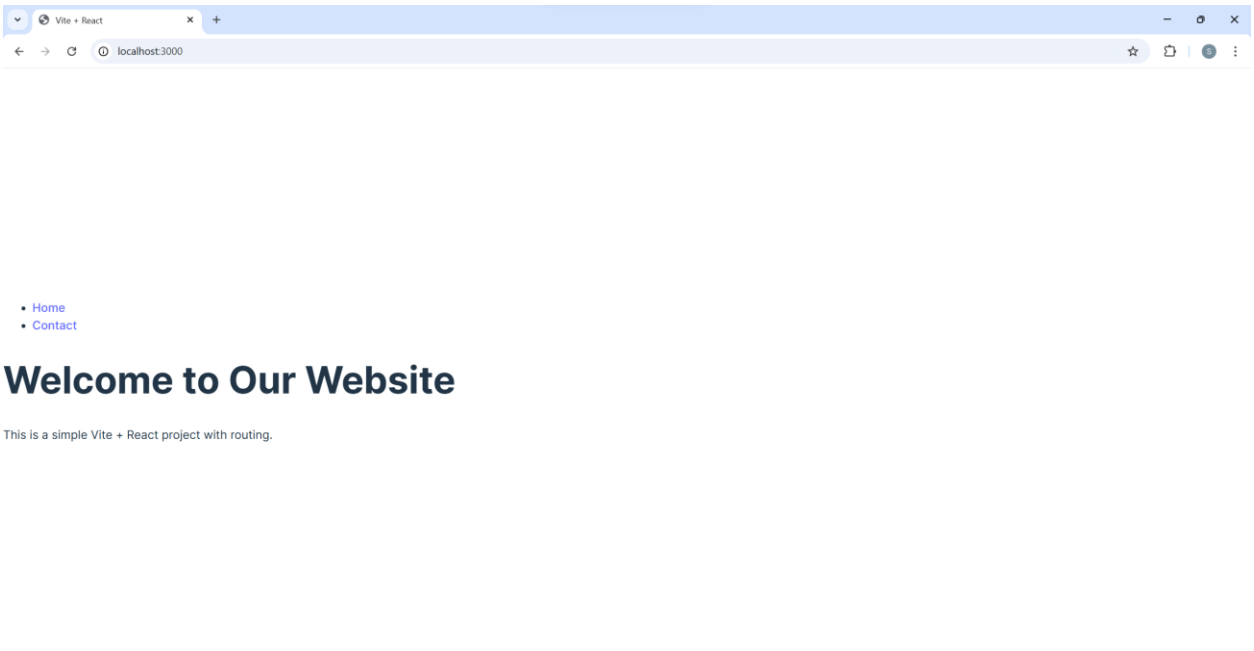
```jsx
import React from 'react';

const Home = () => {
  return (
    <div>
      <h1>Welcome to Our Website</h1>
      <p>This is a simple Vite + React project with routing.</p>
    </div>
  );
};

export default Home;
```

Contact.jsx | main.jsx | App.jsx | Home.jsx

my-vite-app > src > main.jsx > ...

```jsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom';
import App from './App';
import './index.css';

const root = ReactDOM.createRoot(document.getElementById('app'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

خروجی به این صورت است.



شکل ۱۲: بخش خانه

شکل ۱۳: بخش کانتکت

حال صرفا کافی است یک چیز را از بخش کانتکت تغییر بدهیم و سیو کنیم. آنگاه، درجا و به صورت آنی صفحه عوض می شود.
برای مثال من کد درون کانکت را عوض کردم و نتیجه به صورت درجا تغییر کرد به:

```
<button type="submit">Submitttttttttt</button>
```

شکل ۱۴

- Home
- Contact

Contact Us

Your Message: Write your message here

Submitttttttttt

شکل ۱۵

که یک مثال کاربردی از این ابزار است.

اگر سرور راه‌اندازی نشد، اطمینان حاصل کنید که نسخه Node.js شما به‌روز است.

اگر با مشکلاتی مانند وابستگی‌های گم‌شده یا خطاهایی در حین نصب مواجه شدید، می‌توانید دستور زیر را امتحان کنید:

npm install --legacy-peer-deps

در نهایت اگر میخواهید پورت دیفالت را عوض کنید صرفا کافی است فایل vite.config.js را تغییر دهید. برای مثال:

```
1  import { defineConfig } from 'vite'
2  import react from '@vitejs/plugin-react'
3
4  export default {
5    server: {
6      host: '127.0.0.1',
7      port: 3000,
8    },
9  };
```

شکل ۱۶

گام ۶: ساخت برای تولید

وقتی که اپلیکیشن خود را ساخته‌اید و آماده برای انتشار است، می‌توانید پروژه خود را برای تولید بسازید.

برای ساخت پروژه دستور زیر را وارد کنید:

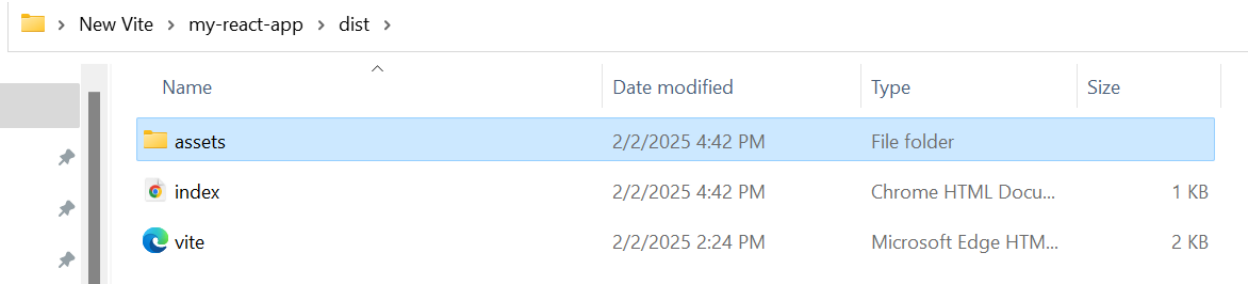
npm run build

و باید خروجیی مانند شکل زیر بگیرید.

```
C:\Users\No1\Desktop\New Vite\my-react-app>npm run build

> my-react-app@0.0.0 build
> vite build

vite v6.0.11 building for production...
✓ 30 modules transformed.
dist/index.html                   0.46 kB │ gzip:  0.29 kB
dist/assets/react-CHdo91hT.svg    4.13 kB │ gzip:  2.05 kB
dist/assets/index-n_ryQ3BS.css    1.39 kB │ gzip:  0.71 kB
dist/assets/index-DO18jdxB.js   143.91 kB │ gzip: 46.35 kB
✓ built in 818ms
```

شکل ۱۷ : خروجی بیلد کردن

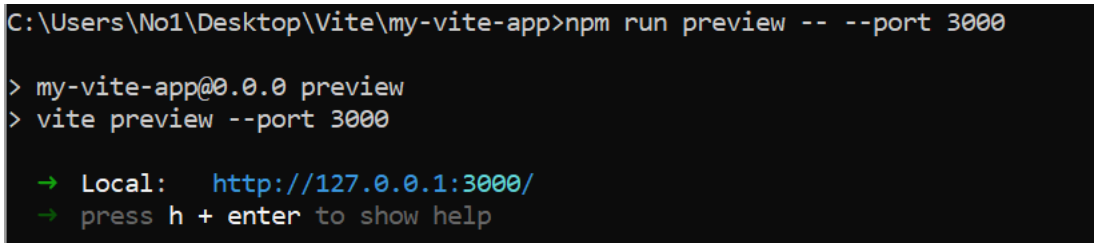
بررسی خروجی: فایل‌های ساخته شده در پوشه dist/ قرار خواهند گرفت. شما می‌توانید این فایل‌ها را در هر سرور فایل استاتیک (مثل Netlify، Vercel یا سرور خودتان) مستقر کنید. خروجی به صورت زیر می باشد:



شکل ۱۸

حالا شما آماده‌اید که با Vite، React شروع به کدنویسی کنید و در صورت نیاز از فایل های بیلد شده در سمت سرور استفاده کنید! و اگر میخواهید پروژه بیلد شده خود را ببینید، صرفا کافی است دستور زیر را وارد کنید تا بیلد شده آن را در لوکال هاست ۳۰۰۰ ببینید:

npm run preview -- --port 3000



شکل ۱۹

و صرفا کافی است لوکال هاست ۳۰۰۰ را باز کنید و نتیجه ی پروژه ی خود را ببینید. امیدوارم این تحقیق براتون مفید بوده باشه خسته نباشید. 😊

مراجع

1. https://vitejs.dev/guide/
2. https://vitejs.dev/config/
3. https://vitejs.dev/guide/using-plugins.html
4. https://vitejs.dev/guide/production-deployment.html
5. https://developer.mozilla.org/en-US/docs/web/javascript/reference/statements/import
6. https://esbuild.github.io/getting-started/
7. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Hot-Module-Replacement
8. https://reactjs.org/docs/getting-started.html
9. https://vite.dev/guide/static-deploy.html