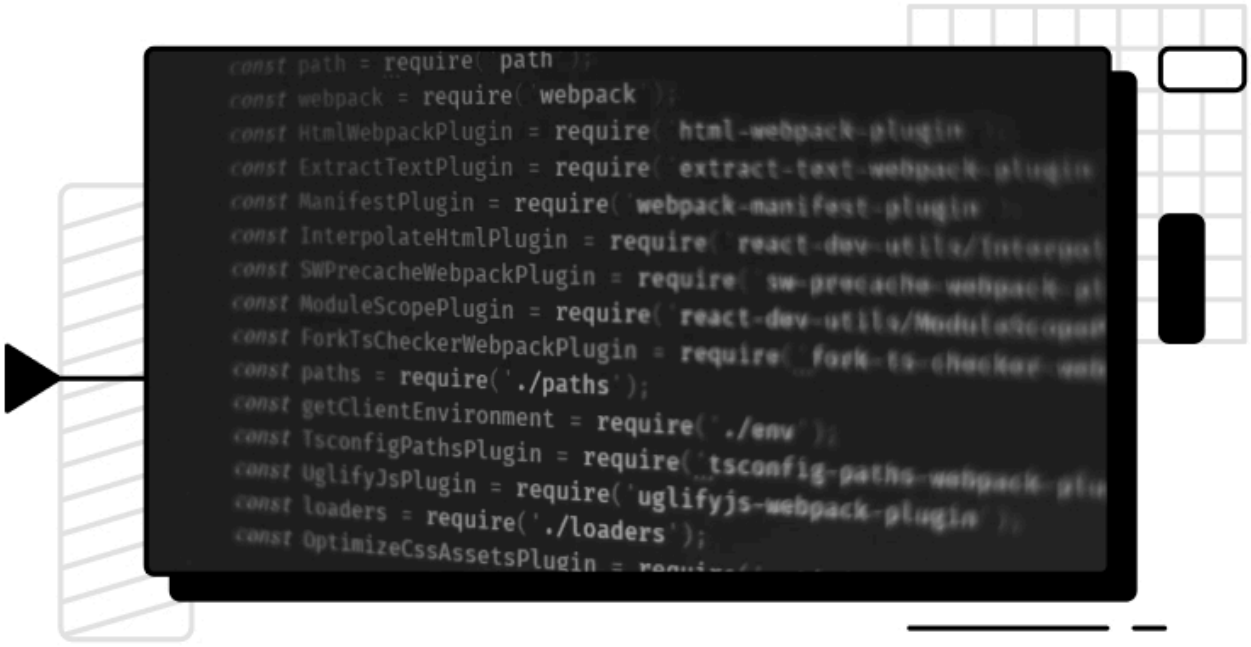
Pieces 🌟
Posted on 26 Sept 2024 • Originally published at code.pieces.app

Vite vs Webpack: Which Build Tool is Right for Your Project?



Do you ever wonder how development code is transformed into production code? Yes, libraries like React, Vue, and Svelte are great to code in, but the browser cannot

understand the code written in these front-end frameworks, and that's where JavaScript build tools come in. Build tools help automate tasks by optimizing code, bundling assets, and improving the performance of the application. Vite and Webpack, two popular JavaScript build tools, help to transpile code into a format the browser understands.

In this tutorial, we will take a look at Vite vs Webpack, their differences, and which build tool best fits your project.

What is Vite?

If you're into front-end development, then you've probably heard of Vite or seen it in action. Vite is a fast, modern front-end build tool that speeds up web performance by addressing the shortcomings of the traditional bundling process. Vite is known for its speed and serves as a build tool for various front-end frameworks such as React, Vue, and Svelte. Fun fact: "Vite" is a French word that translates to "fast" or "quick", which gives a clue to the major focus of Vite. You can install Vite by running the command below:

```
npm install -D vite
```

Features of Vite

In this section, we'll dive into the key features of Vite. They include:

- **Fast Server Start**: Compared to other build tools, applications built with Vite have a relatively fast server time, this is because Vite divides the modules into two (dependencies and source code), thereby improving the code transformation process.
- **Native ES Modules Support**: Vite uses native ES Modules (ESM) which have full support for modern browsers. Rather than bundling modules in development (which is the case for other build tools), Vite loads the ES modules as separate files, which significantly improves the performance.
- **Fast Hot Module Replacement**: Hot Module Replacement (HMR) allows only a particular module to be updated rather than bundling and reloading the entire application on a slight update in the codebase. Vite has a super fast HMR, providing instant feedback to developers when the code changes.
- **Fast Production Builds**: Vite uses `esbuild`, a fast Go-based bundler, to compile code during the production build process. This is faster and provides more optimized builds than traditional JavaScript bundlers.

- **Zero Configuration**: Even small amounts of time saved add up. Developers do not need to manually configure the Vite build tool as it does all the work under the hood. Although it works fine without configuration, developers can customize configurations to fit their needs.

What is Webpack?

Webpack is a module bundler used to build modern JavaScript applications. It allows developers to compile modules and optimize assets in an application. A key feature of Webpack is that it is flexible in its customization, which enables it to be suitable for both small- and large-scale applications. You can install Webpack by running the command below:

```
npm install --save-dev webpack
```

Features of Webpack

The key features of Webpack include:

- **Bundling**: Being a module bundler, Webpack gathers all the JavaScript, CSS, and images into one or more files, making it easier and faster to load in the browser. Developers can work on these files separately while Webpack handles the bundling process.
- **Loaders**: Loaders in Webpack are responsible for processing and transforming non-JavaScript files such as CSS, fonts, images, and TypeScript. Examples of Loaders in Webpack are `babel-loader`, `css-loader`, `style-loader`, etc. Loaders convert these files into modules that are compatible to be used in the JavaScript bundle.
- **Code Splitting**: Code splitting significantly improves performance and reduces the load time of an application. It allows developers to split code into smaller chunks, which can be loaded only when it is needed.
- **Flexible Configuration**: The flexible nature of Webpack makes it highly suitable for both small and large-scale applications. Developers can configure the bundling process according to their project needs.
- **Tree Shaking and Optimization**: Bundling the codebase of a large-scale application can lead to a large bundle size. Webpack addresses this by adopting a dead code elimination technique that automatically removes unused code from the bundle.

Differences between Webpack and Vite

Let’s get into Webpack vs Vite, comparing both build tools using metrics such as bundling, server speed, HMR, configuration ease, and ecosystem. This will help you to choose the build tool that best fits your project. Let’s explore!

Bundling

The major difference between Vite vs Webpack lies in its functionality and its bundling process. Webpack is a bundler-first tool, which means it bundles all the files (including all dependencies) into one or more optimized files in development and also during the production build. Vite, on the other hand, serves files to the browser using ESM and does not bundle code during development. Vite bundles code during the production build process.

Speed

Is Vite faster than Webpack? Well, let’s see. Webpack works by bundling all the files before starting the development server. Server time is affected by this, especially when dealing with large-scale applications.

Vite starts the development server faster because it processes the source code and dependencies separately; the source code is loaded into the browser and ESM handles development. So, yes, Vite is faster than Webpack in terms of development server speed.

Hot Module Replacement

Webpack and Vite both have support for HMR, therefore you can choose either Vite or Webpack if this is a factor you consider highly. However, Vite has a faster implementation because it uses ESM and does not require bundling of the entire application when a section of the codebase is updated.

Bundle Size

Bundle size refers to the combined file size of all code, libraries, and dependencies that are loaded to the browser. A large bundle size will lead to slower page load time and a less responsive page because it takes up more memory in the browser.

Although Vite and Webpack are both great at optimizing bundle size, Vite’s Rollup bundling technique for production builds contributes to a smaller bundle size which leads to a more efficient browser performance. If bundle size is a major factor for you, then you should go for Vite instead of Webpack.

Configuration Flexibility

Flexibility in configuration is an aspect where Webpack significantly supersedes Vite. With Webpack, developers have the luxury of extensively toggling the configuration in a manner that suits their project. This makes applications built with Webpack highly scalable, as it caters to the needs of both small and large applications.

Vite requires zero configuration for developers to get started. It comes with a default setup that covers the build process of basic JavaScript applications. Although this might be particularly beneficial to developers working on projects that do not require complex configurations, it can be perceived as an inconvenience to developers working on large applications and looking to explore more configurations.

Community and Ecosystem

Webpack has been in use since 2012, which means it has a larger community of users, a vast ecosystem, and is used in most legacy applications. At the time of this writing, Webpack has about 26 million weekly downloads from NPM. Webpack also has extensive documentation, which makes it easier for developers to use. It has support for almost all JavaScript frameworks (Vue, React, Angular, Svelte, etc.).

Vite is relatively new and has been around since 2020. It was built by the creator of Vue.js. Although the community of users is growing rapidly, it is not as large as Webpack. At the time of this writing, Vite has about 15 million weekly downloads from NPM, which falls short of the downloads of Webpack. Vite is mostly suitable for modern frameworks like Vue and React, especially in projects that prioritize fast development.

How To Migrate from Webpack to Vite

Migrating from Webpack to Vite is a fairly straightforward process. Follow the steps below:

1. Install Vite in your Webpack's project directory:

```
npm install — save-dev vite
```

1. Install framework-specific plugins. You can check out the available Vite plugins here. For React:

```
npm install -- save-dev @vitejs/plugin-react
```

1. Update the scripts in your `package.json` file:

```
"scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
```

1. Create the `vite.config.js` configuration file at the root of your project (you can customize it according to your project's needs):

```
import { defineConfig } from 'vite';
import react from '@vitejs/plugin-react'; // since we're using React


export default defineConfig({
  plugins: [react()], //
  build: {
    outDir: 'dist', //
  },
});
```

1. Migrate the Webpack plugins in your project to their Vite equivalents. Here are some examples:

```
CopyWebpackPlugin -> vite-plugin-static-copy
BabelLoader -> vite-plugin-react
BundleAnalyzerPlugin -> rollup-plugin-visualizer
ESLintPlugin -> vite-plugin-eslint
CompressionWebpackPlugin -> vite-plugin-compression
```

1. Uninstall Webpack and its loaders/plugins in your project:

```
npm uninstall webpack
```

1. Finally, run the Vite development server (remember that we configured this in the `package.json` script):

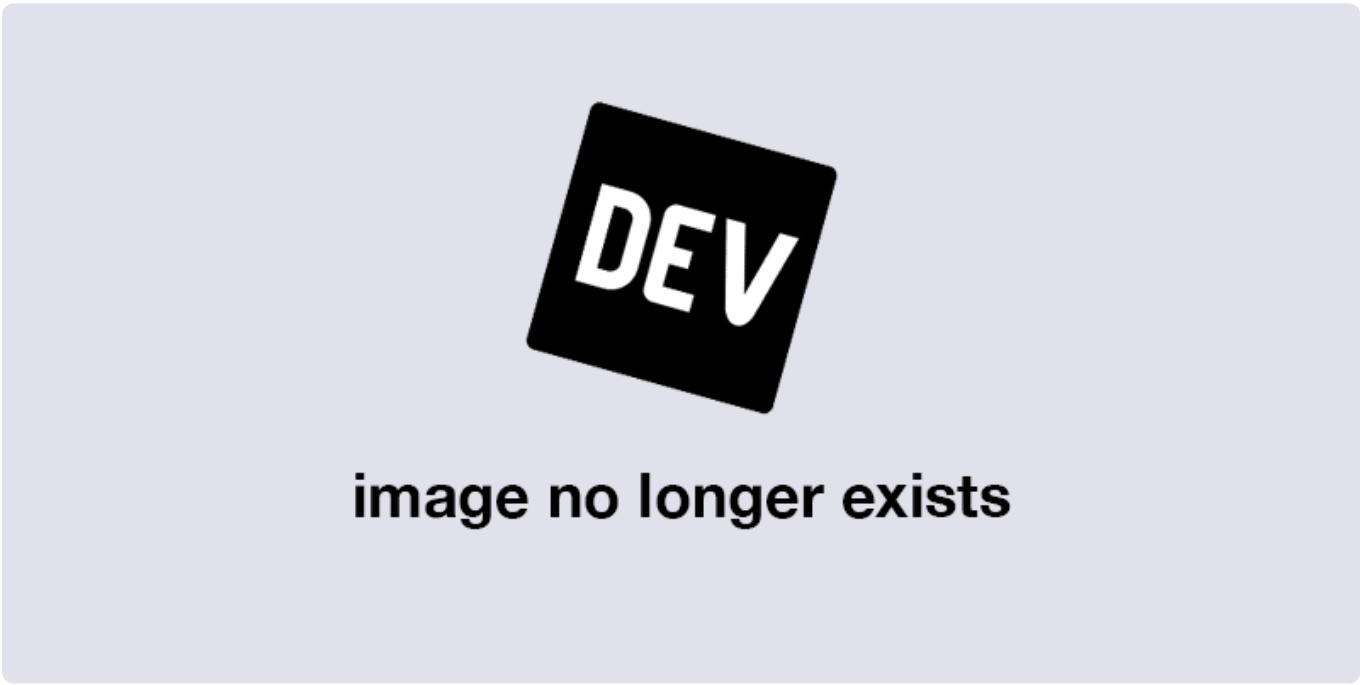
```
npm run dev
```

Build Tools and AI Tools: Leveraging Pieces in Development

Vite vs Webpack... have you made your choice? Well, whichever build tool you choose, you can always achieve greater productivity with Pieces. Pieces is an on-device AI coding assistant and snippet manager that boosts developers' productivity by offering a contextual understanding of their codebase. It also allows you to manage your code with actions such as save, search, share, generate, and more. Enough talk, let's see it in action.

How to Migrate Vite to Webpack with Pieces

We discussed how to migrate a Webpack project to Vite earlier. How about migrating Vite to Webpack? This time, we'll have Pieces do it for us!



In VS Code, I have a project built with Vite as a build tool. I right-clicked on the folder directory and then asked Pieces Copilot to convert this Vite project to Webpack. It gives a step-by-step process on how to do that. It also gives a **contextual reply** as it relates to my project and not a generic answer.

So, rather than manually looking for a Webpack plugin equivalent, you get a list of project-specific Webpack plugins, which saves a ton of time. Note that I have the Pieces for VS Code extension installed, but Pieces is also available on your favorite IDEs.

Pieces doesn't end here! Whichever language you're coding with, Pieces Copilot is a companion to guide you through every bit of it. Start exploring and dive into the numerous possibilities!

Conclusion

In this blog post, you have compared Vite vs Webpack, explored the backgrounds of Vite and Webpack, their features, and the differences between Vite and Webpack on various criteria such as the bundling, development server speed, HMR, flexibility in configuration, and the ecosystem. You also learned about how to leverage Pieces in your development. Happy coding, or better still, happy building!

Top comments (0)

Pieces

Bringing AI & machine learning right to where you are in your workflow. Supercharging tools like GitHub, VS Code, JetBrains, Browsers, and more! Try Pieces for Developers Suite for free today!

LOCATION
Cincinnati

JOINED
9 Sept 2022

More from Pieces

What's Ahead for Programmers: Tools Shaping the Future

AI Transformation: How AI is Reshaping Enterprise Development

TypeScript Interface vs Type: Differences and Best Use Cases