

Vite.js vs Webpack

Andrej Kocev (173148), Filip Tanevski (173165)

1. Introduction	2
2. Background	2
3. Objectives	5
Project Setup Speed:	5
Development Experience:	5
Build Speed & Bundle Size:	5
Development Server Performance:	5
Configuration and Customization:	6
Plugin Ecosystem:	6
Community and Documentation:	6
4. Comparisons	6
Project Setup Speed	6
Development Experience	8
Build Speed & Bundle Size	8
Development Server Performance	11
Configuration and Customization	13
Plugin Ecosystem	14
Community and Documentation	15
5. Use cases	18
6. Community & Support	18
7. Conclusion	18

1. Introduction

This report presents a side-by-side comparison of two widely-used front-end development tools: Vite and Vue CLI (with Webpack). To do this, we'll create two identical projects—one with Vite and another with Vue CLI.

Vite, recognized for its speed and modern approach, offers rapid development capabilities. It's particularly valued for its quick setup and instant code changes during development.

On the other hand, Vue CLI, built on the solid foundation of Webpack, provides extensive configuration options and compatibility for diverse project needs.

This report aims to present a clear picture of the strengths and weaknesses of these tools, making it easier for developers to choose the one that aligns best with their project requirements. We'll explore project setup, development experience, production build processes, and community support to help you decide between Vite and Vue CLI for your next project.

Let's begin our exploration of these two essential tools in the world of front-end development.

2. Background

Vue CLI, short for Vue Command Line Interface, is a command-line tool that simplifies the process of setting up, configuring, and managing Vue.js projects. It was created by the Vue.js team and is an official tool for building Vue.js applications. Vue CLI provides a standardized and efficient way to create, develop, and build Vue.js projects, making it easier for developers to get started and maintain their applications.

Some of the key aspects and features of Vue CLI are:

Project Generation: Vue CLI allows developers to quickly generate new Vue.js projects with a predefined project structure. It provides a set of project templates that include the necessary configuration files, development tools, and dependencies to kickstart development.

Interactive Project Setup: Vue CLI offers an interactive project setup wizard that guides developers through the initial project configuration. This includes choosing features, such as Vuex for state management, Vue Router for routing, and CSS pre-processors like SASS or Less.

Plugin System: Vue CLI is extensible through a plugin system. Developers can create and use plugins to add additional functionality and features to their projects. There are many official and community plugins available for tasks like internationalization, unit testing, and more.

Development Server: Vue CLI includes a development server with hot module replacement (HMR) support. This allows for a seamless development experience

where changes to code are immediately reflected in the running application without the need for a full page refresh.

Production Builds: Vue CLI provides optimized production builds that include features like code minification, tree shaking (removing unused code), and asset optimization to ensure that the final application is as small and performant as possible.

Configuration Management: Developers have the flexibility to customize project configuration using a `vue.config.js` file. This file allows you to modify settings related to the build process, CSS, dev server, and more.

Modern JavaScript and Babel Support: Vue CLI is built with modern JavaScript and provides support for Babel and TypeScript, allowing developers to write code using the latest language features and transpile it for compatibility with older browsers.

Vue UI: In addition to the command-line interface, Vue CLI offers a graphical user interface (Vue UI) that provides a visual way to create, configure, and manage Vue.js projects. It's especially useful for developers who prefer a GUI over the command line.

Easy Upgrades: Vue CLI simplifies the process of upgrading your project to newer versions of Vue.js and related dependencies. It can also help identify and resolve compatibility issues.

Active Community: Vue CLI benefits from the active Vue.js community and ecosystem. Many third-party plugins and extensions are available to enhance its capabilities.

On the other hand Vite.js is a build tool and development environment that was created by Evan You, the same individual who created Vue.js. Vite, which is pronounced like "veet" (the French word for "fast"), was first released in 2020. It's designed to be a fast, lightweight, and highly efficient build tool for modern web development, especially when working with JavaScript frameworks like Vue.js or React. Vite is known for its speed and simplicity, making it an excellent choice for building web applications.

Some of the key aspects and features of Vite.js are:

ESM (ECMAScript Modules) by Default: Vite leverages ECMAScript Modules, which are the native module system in modern browsers. This allows for faster development and builds because modules are loaded on-demand, only when they are needed.

Dev Server with Hot Module Replacement (HMR): Vite includes a development server that supports hot module replacement. This means that as you make changes to your code, Vite can update the application in the browser without requiring a full page refresh, greatly speeding up the development process.

Native Support for Vue.js and React: Vite was initially designed with Vue.js in mind, and it provides a Vue-specific plugin (Vite-plugin-Vue) for seamless integration. It also supports React through a similar plugin (Vite-plugin-React). This native support enhances the development experience for these popular frameworks.

Plugin System: Vite has a flexible and extensible plugin system, making it easy to add custom functionality or integrate with third-party tools and libraries. Developers can create their own plugins to extend Vite's capabilities.

Efficient Production Builds: When it comes to building for production, Vite generates highly optimized, tree-shaken, and minified code bundles. This ensures that your application's output is as small and efficient as possible for deployment.

Config-Free Development: For many projects, Vite doesn't require a complex configuration file. It uses sensible defaults, which means you can start a project without having to spend time configuring build settings.

Support for CSS Pre-processors: Vite supports popular CSS pre-processors like SASS, Less, and Stylus, allowing you to write styles in your preferred format.

Optimized for Speed: Vite's development server is known for its speed and responsiveness, which makes the development experience smooth and efficient, especially for large-scale applications.

Scoped CSS: Vite encourages scoped CSS by default, ensuring that styles defined in a component don't leak to other parts of the application.

Static Site Generation (SSG) and Server-Side Rendering (SSR): Vite can also be used for generating static websites and server-side rendering (SSR) with Vue.js, making it versatile for various project types.

3. Objectives

Project Setup Speed:

Evaluate the time it takes to set up a new project from scratch using Vite and Vue CLI, including the installation of dependencies and initial configuration.

Development Experience:

Compare the development experience and implementing features using both tools, as well as focusing on hot module replacement (HMR) or live-reloading capabilities.

Build Speed & Bundle Size:

Measure the time it takes to build a production-ready bundle with both Vite and Vue CLI, analyzing the speed of the build process.

Development Server Performance:

Examine the performance of the development servers provided by Vite and Vue CLI, considering factors such as server start-up time and response times.

Configuration and Customization:

Explore the level of configuration and customization options available in Vite and Vue CLI for tailoring the build process to specific project requirements.

Plugin Ecosystem:

Assess the availability and maturity of plugins and extensions for Vite and Vue CLI, examining the ecosystem's ability to extend functionality.

Community and Documentation:

Analyze the size and activity of the developer community for both tools, as well as the quality and comprehensiveness of documentation and resources.

4. Comparisons

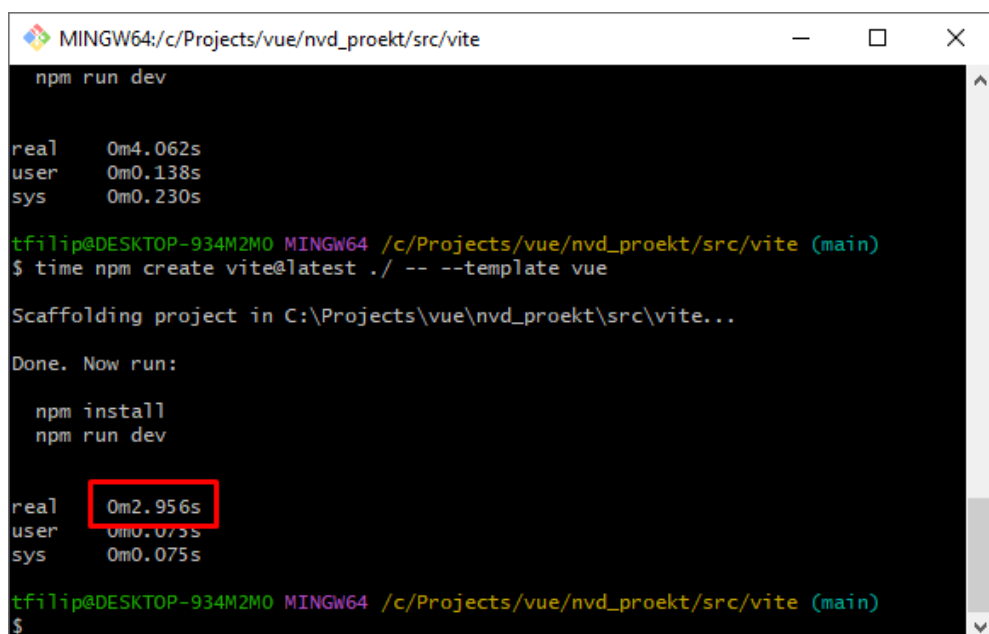
We would like to preface this section with the fact that for some of the comparisons where the commands didn't give out the time of execution, we used the **time** parameter before the actual command in terminals which gives us the time that the command spent running.

Project Setup Speed

- **Vite**

Vite has many templates to choose from when starting a project with it, and Vue is included as well. We created the project with the following command:

```
npm create vite@latest my-vue-app -- --template vue
```



The screenshot shows a terminal window titled 'MINGW64:/c/Projects/vue/nvd_proekt/src/vite'. The user enters 'npm run dev' and then '\$ time npm create vite@latest ./ -- --template vue'. The terminal output shows the command's execution time: 'real 0m2.956s', 'user 0m0.075s', and 'sys 0m0.075s'. The 'real' time is highlighted with a red box. The terminal also shows the scaffolding process and instructions to run 'npm install' and 'npm run dev'.

```
MINGW64:/c/Projects/vue/nvd_proekt/src/vite
npm run dev

real    0m4.062s
user    0m0.138s
sys     0m0.230s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$ time npm create vite@latest ./ -- --template vue

Scaffolding project in C:\Projects\vue\nvd_proekt\src\vite...

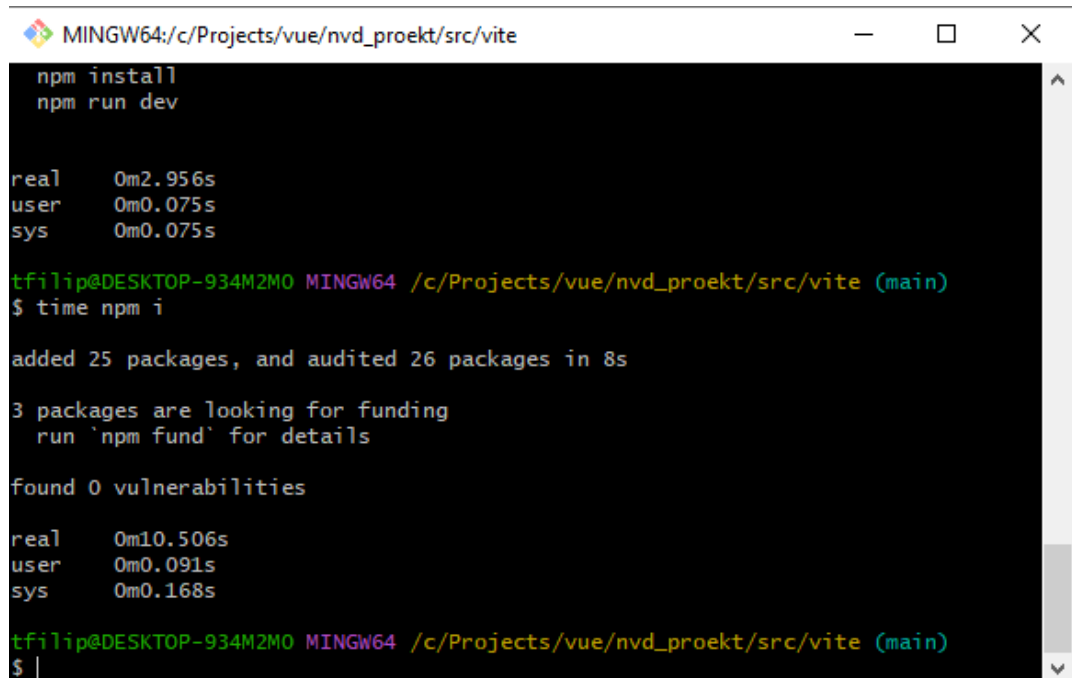
Done. Now run:

  npm install
  npm run dev

real    0m2.956s
user    0m0.075s
sys     0m0.075s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$
```

As we can see from the screenshot, the project creation only took **~3 seconds**, **however** this is without the npm install command. If we factor that in as well, we get around **13 seconds**:



```
MINGW64:/c/Projects/vue/nvd_proekt/src/vite
npm install
npm run dev

real    0m2.956s
user    0m0.075s
sys     0m0.075s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$ time npm i

added 25 packages, and audited 26 packages in 8s

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

found 0 vulnerabilities

real    0m10.506s
user    0m0.091s
sys     0m0.168s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$ |
```

- **Vue-CLI (Webpack)**
The Vue-CLI offers two ways of creating a project. One is through the command line, and the other is by using the GUI which you can start by running the `vue ui` command. We created our project through the command line by using:

```
vue create nvd_proekt
```

The Vue-CLI not only scaffolds the project, it also runs a `npm install` so you don't have to. Regardless of that, the whole process of creating the project took **1 minute and 25 seconds** as we can see from the screenshot below:

```
MINGW64:/c/Projects/vue/nvd_proekt/src/vue-cli

added 103 packages, and audited 966 packages in 10s

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

found 0 vulnerabilities
Running completion hooks...
Generating README.md...
Successfully created project vue-cli.
Get started with the following commands:

$ npm run serve

real    1m25.354s
user    0m0.015s
sys     0m0.106s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vue-cli (main)
$ |
```

Development Experience

In terms of development experience in our opinion Vite outshined Vue-CLI. Something that caught our eye was the HMR (Hot Module Replacement) which provided a really fast reaction to changes without even having to reload the page. Another thing that was really interesting was how fast Vite is in starting the development server. However, something that we really liked about the Vue-CLI was the `vue ui` which lets you create a project by using the graphical user interface.

Build Speed & Bundle Size

- **Vite**

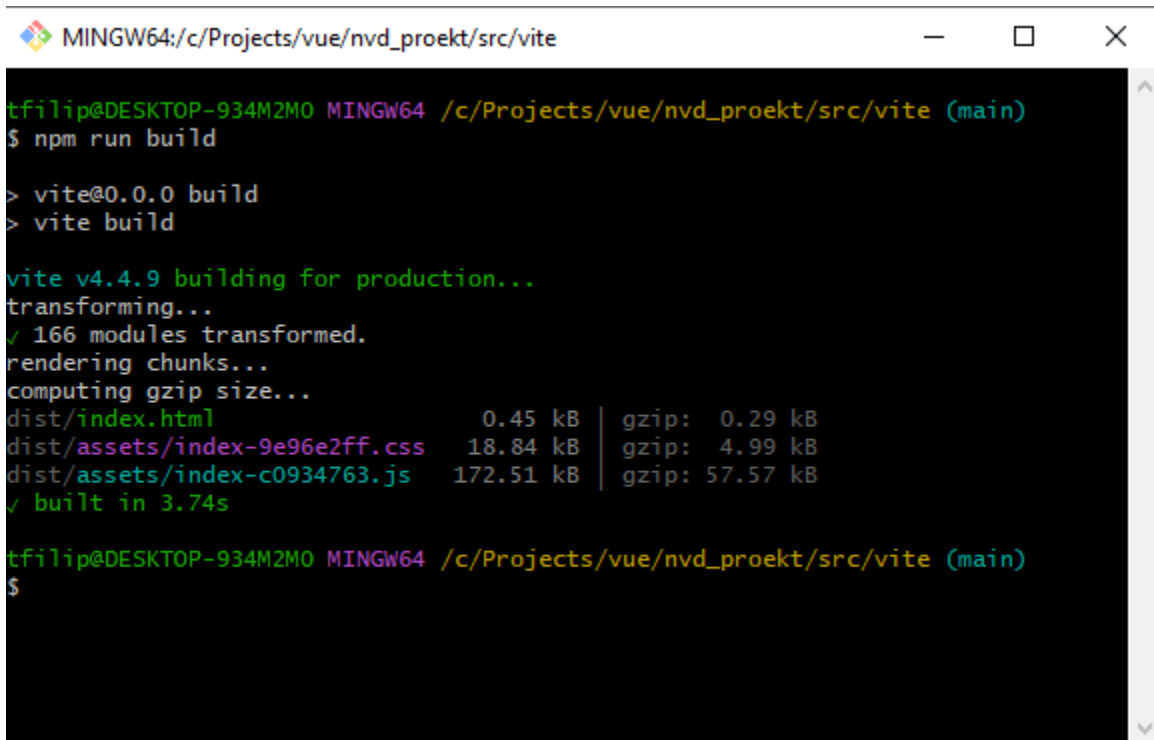
The time it took to create a production-ready build with Vite for the initial scaffolded project was just around **1 second** and the bundle size being around **53kb** unzipped:

```
Koce@DESKTOP-MG5D8IR MINGW64 ~/Desktop/nvd_proekt/src/vite (main)
$ npm run build

> vite@0.0.0 build
> vite build

vite v4.4.9 building for production...
transforming...
✓ 16 modules transformed.
rendering chunks...
computing gzip size...
dist/index.html          0.46 kB | gzip: 0.29 kB
dist/assets/vue-5532db34.svg 0.50 kB | gzip: 0.31 kB
dist/assets/index-14c7cc56.css 1.30 kB | gzip: 0.67 kB
dist/assets/index-16d64be8.js 51.54 kB | gzip: 20.90 kB
✓ built in 939ms
```


After some development, the production-ready bundle took about **3 seconds** with the size being about **190kb** unzipped.

A terminal window titled 'MINGW64:/c/Projects/vue/nvd_proekt/src/vite' showing the output of 'npm run build'. The output indicates that Vite v4.4.9 is building for production, transforming 166 modules, and rendering chunks. A table shows the file sizes for 'dist/index.html', 'dist/assets/index-9e96e2ff.css', and 'dist/assets/index-c0934763.js', along with their gzipped sizes. The build is completed in 3.74s.

```
tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$ npm run build

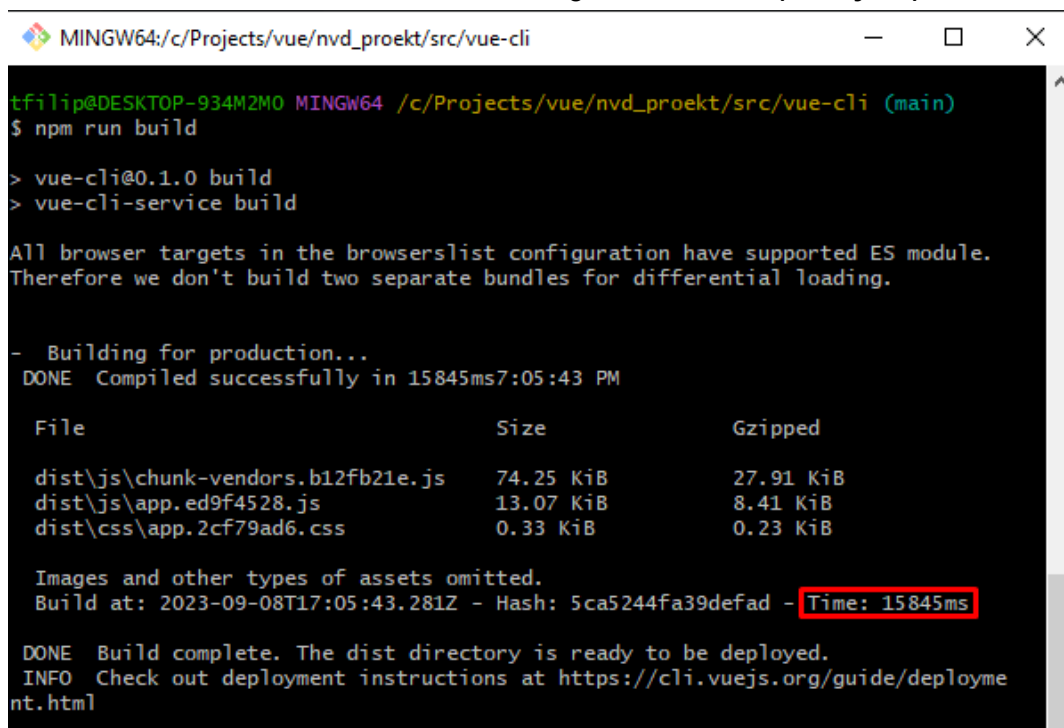
> vite@0.0.0 build
> vite build

vite v4.4.9 building for production...
transforming...
✓ 166 modules transformed.
rendering chunks...
computing gzip size...
dist/index.html          0.45 kB | gzip: 0.29 kB
dist/assets/index-9e96e2ff.css 18.84 kB | gzip: 4.99 kB
dist/assets/index-c0934763.js 172.51 kB | gzip: 57.57 kB
✓ built in 3.74s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$
```

- **Vue-CLI**

The time it took to create a production-ready build with Vue-CLI for the initial scaffolded project was around **16 seconds** with the bundle size being around **88KiB(kibibytes)** or **90kbs**

A terminal window titled 'MINGW64:/c/Projects/vue/nvd_proekt/src/vue-cli' showing the output of 'npm run build'. The output indicates that Vue-CLI@0.1.0 is building for production. It shows a table of file sizes for 'dist\js\chunk-vendors.b12fb21e.js', 'dist\js\app.ed9f4528.js', and 'dist\css\app.2cf79ad6.css', along with their gzipped sizes. The build is completed in 15845ms (15.845s).

```
tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vue-cli (main)
$ npm run build

> vue-cli@0.1.0 build
> vue-cli-service build

All browser targets in the browserslist configuration have supported ES module.
Therefore we don't build two separate bundles for differential loading.

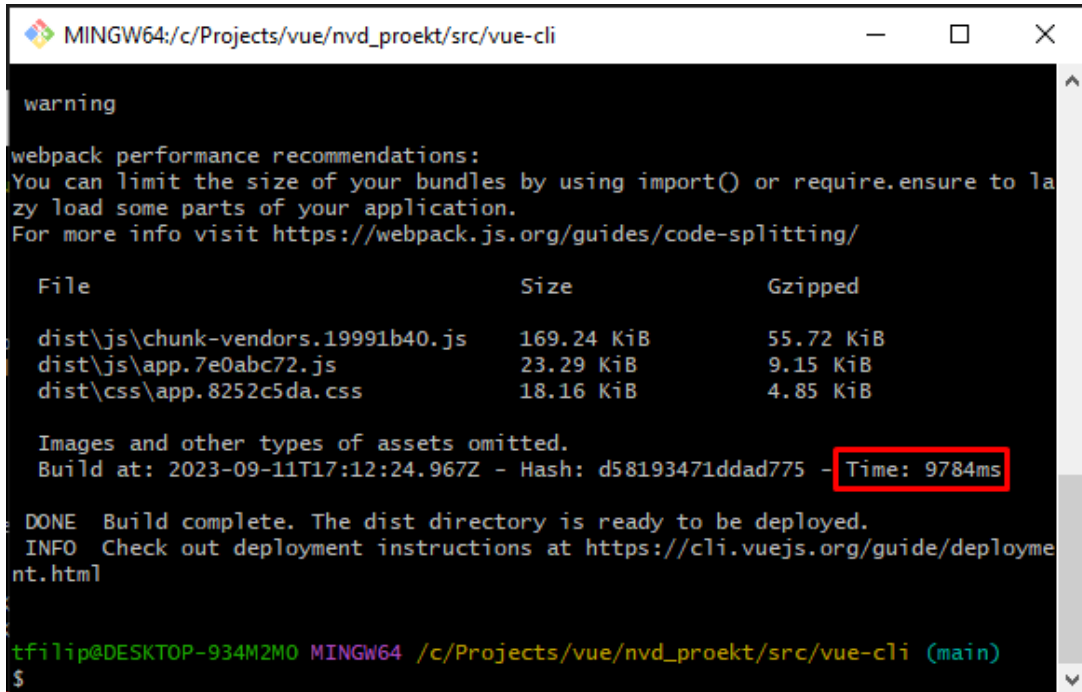
- Building for production...
DONE Compiled successfully in 15845ms7:05:43 PM

File                                Size                                Gzipped
dist\js\chunk-vendors.b12fb21e.js    74.25 KiB                          27.91 KiB
dist\js\app.ed9f4528.js              13.07 KiB                          8.41 KiB
dist\css\app.2cf79ad6.css             0.33 KiB                           0.23 KiB

Images and other types of assets omitted.
Build at: 2023-09-08T17:05:43.281Z - Hash: 5ca5244fa39defad - Time: 15845ms

DONE Build complete. The dist directory is ready to be deployed.
INFO Check out deployment instructions at https://cli.vuejs.org/guide/deployment.html
```

After some development the same process took around **10 seconds** with the bundle size being **210KiB** or **215kbs**



```
MINGW64:/c/Projects/vue/nvd_proekt/src/vue-cli
warning
webpack performance recommendations:
You can limit the size of your bundles by using import() or require.ensure to lazy
load some parts of your application.
For more info visit https://webpack.js.org/guides/code-splitting/

  File                                Size              Gzipped
  dist\js\chunk-vendors.19991b40.js    169.24 KiB        55.72 KiB
  dist\js\app.7e0abc72.js              23.29 KiB         9.15 KiB
  dist\css\app.8252c5da.css            18.16 KiB         4.85 KiB

Images and other types of assets omitted.
Build at: 2023-09-11T17:12:24.967Z - Hash: d58193471ddad775 - Time: 9784ms

DONE  Build complete. The dist directory is ready to be deployed.
INFO  Check out deployment instructions at https://cli.vuejs.org/guide/deployme
nt.html

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vue-cli (main)
$
```

The reason why the time was faster now instead of the initial build was because some of the dependencies that came with the project scaffolding were removed.

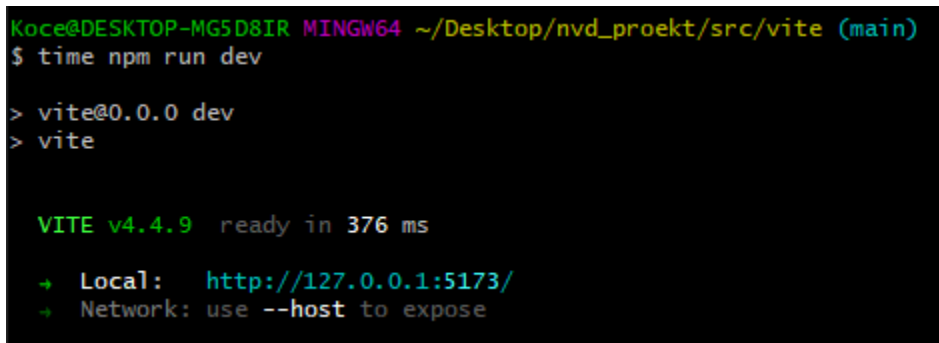
Here we noticed that Vite is more optimized in comparison to Vue-CLI, both in build speed and bundling the production-ready builds.

Development Server Performance

For the development server we measured the time that it takes to get a local server up and running for both Vite and Vue-CLI, both at the start of the project with the initial boilerplate code, as well as after some development.

- **Vite**

The initial time that it took to start a development server in Vite was an impressive **376 ms**:



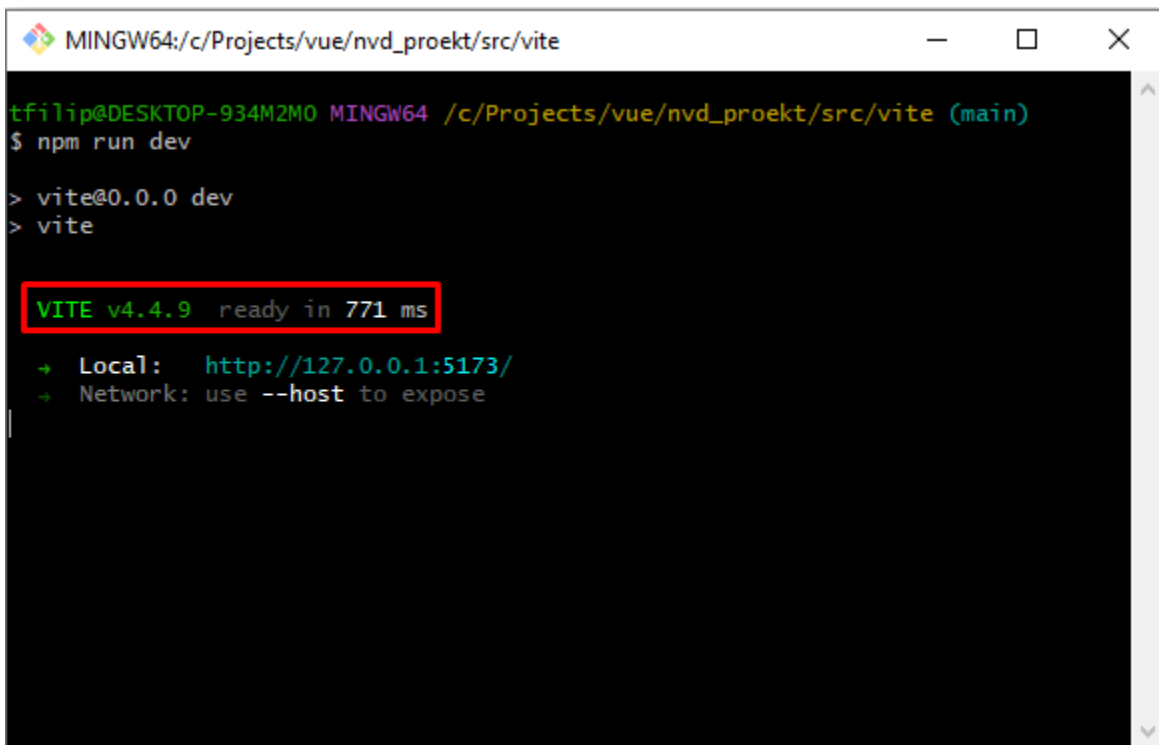
```
Koce@DESKTOP-MG5D8IR MINGW64 ~/Desktop/nvd_proekt/src/vite (main)
$ time npm run dev

> vite@0.0.0 dev
> vite

VITE v4.4.9 ready in 376 ms

➔ Local:   http://127.0.0.1:5173/
➔ Network: use --host to expose
```

After further development, we ran the local server again, and it was up and running in **771 ms**.

A screenshot of a Windows terminal window titled 'MINGW64:/c/Projects/vue/nvd_proekt/src/vite'. The terminal shows the command 'npm run dev' being executed, which runs 'vite@0.0.0 dev'. The output indicates that Vite v4.4.9 is ready in 771 ms. The URL 'http://127.0.0.1:5173/' is shown for local access, and a note suggests using '--host' for network exposure. The text 'VITE v4.4.9 ready in 771 ms' is highlighted with a red rectangle.

```
MINGW64:/c/Projects/vue/nvd_proekt/src/vite
tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vite (main)
$ npm run dev

> vite@0.0.0 dev
> vite

VITE v4.4.9 ready in 771 ms

➔ Local:   http://127.0.0.1:5173/
➔ Network: use --host to expose
```

Once again, it's worth mentioning that the hot module replacement from Vite provides a great development experience.

- **Vue-CLI**

The initial time that it took to start a development server with Vue-CLI was **~14 seconds**.

```
MINGW64:/c/Projects/vue/nvd_proekt/src/vue-cli
$ time npm run serve
> vue-cli@0.1.0 serve
> vue-cli-service serve

INFO Starting development server...
DONE Compiled successfully in 5290ms7:04:28 PM

App running at:
- Local: http://localhost:8081/
- Network: http://192.168.1.252:8081/

Note that the development build is not optimized.
To create a production build, run npm run build.

real    0m13.842s
user    0m0.075s
sys     0m0.168s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vue-cli (main)
$ |
```

After further development and removing some of the boilerplate that came with the project scaffolding we ran the local server again, and it was up and running in **~10 seconds**.

```
MINGW64:/c/Projects/vue/nvd_proekt/src/vue-cli
$ time npm run serve
> vue-cli@0.1.0 serve
> vue-cli-service serve

INFO Starting development server...
DONE Compiled successfully in 4181ms7:11:29 PM

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.1.252:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.

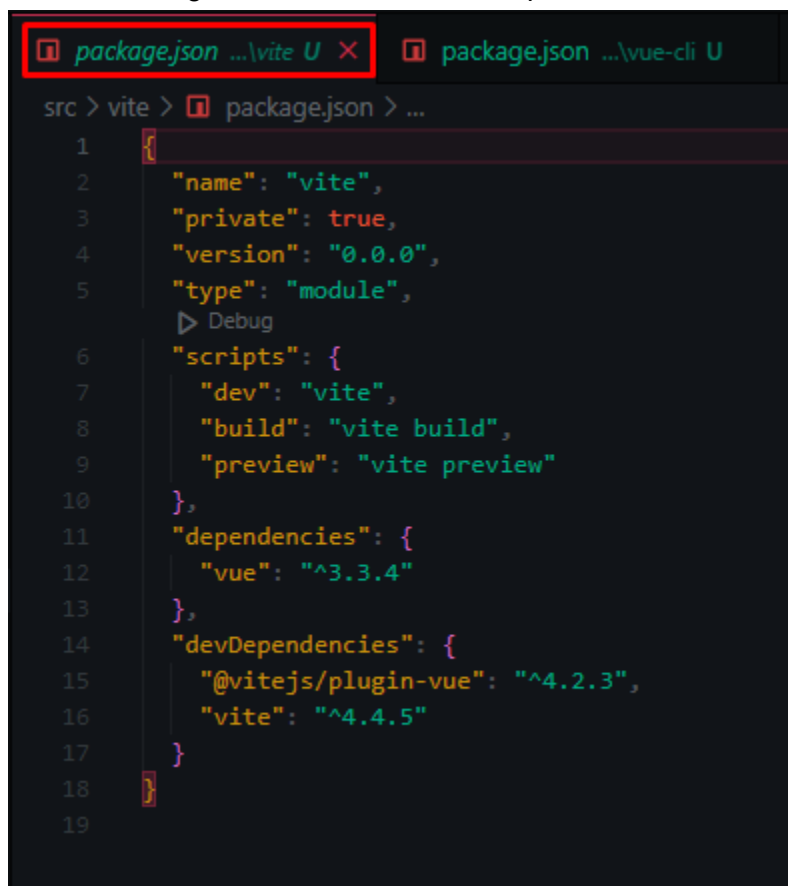
real    0m10.382s
user    0m0.061s
sys     0m0.106s

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/src/vue-cli (main)
$ |
```

Configuration and Customization

- **Vite**

There are multiple ways of creating a project with Vite. You can do so with **npm**, **yarn** and **pnpm**. The command for it is `npm create vite@latest`. Once we run this command in the terminal there are a couple of prompts where it asks us which template we want to choose (React, Vanilla, Vue, Svelte etc..), and if we want to use JavaScript or TypeScript. We can also directly create a project without having to select the template by passing the `--template` argument in the command like so: `vue create vite@latest --template vue[-ts]`. We add the `-ts` at the end if we want our project to use TypeScript. If not, we just omit that. Once we create the project, we are left with a very minimal and working version of said template. For Vue, this was the **package.json** file that was generated with all of its dependencies:



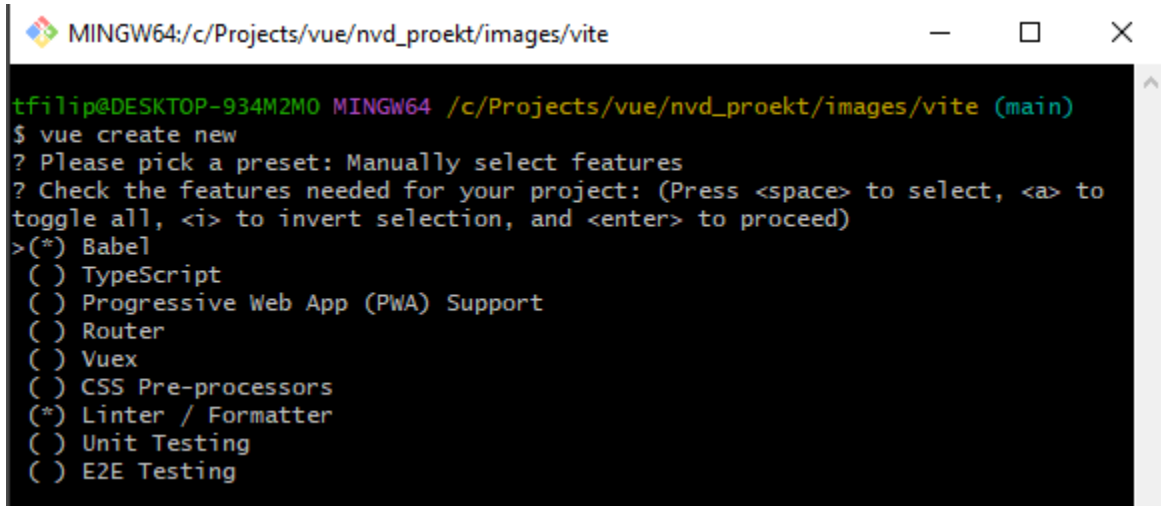
```
src > vite > package.json > ...
1  {
2    "name": "vite",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "vite build",
9      "preview": "vite preview"
10   },
11   "dependencies": {
12     "vue": "^3.3.4"
13   },
14   "devDependencies": {
15     "@vitejs/plugin-vue": "^4.2.3",
16     "vite": "^4.4.5"
17   }
18 }
```

As we can see this is a very minimal project requiring the bare essentials for it to work, having only vue as the dependency, and Vite related dependencies as devDependencies.

- **Vue-CLI**

In terms of configuration and customization, Vue-CLI outshines Vite, as it comes with many configuration settings. In order to create a project, we can use terminal commands, or use a GUI. With terminal commands we can run `vue create [project-name]`. This will then let us

choose whether we want to manually select features, or to use the Vue 2 or Vue 3 template that is provided, which comes with vue dependencies, and babel and eslint as extra. If we choose to manually select features we have many things to choose from that we want to include in the project as we can see from the following screenshot:



```
MINGW64:/c/Projects/vue/nvd_proekt/images/vite

tfilip@DESKTOP-934M2M0 MINGW64 /c/Projects/vue/nvd_proekt/images/vite (main)
$ vue create new
? Please pick a preset: Manually select features
? Check the features needed for your project: (Press <space> to select, <a> to
toggle all, <i> to invert selection, and <enter> to proceed)
>(*) Babel
  ( ) TypeScript
  ( ) Progressive Web App (PWA) Support
  ( ) Router
  ( ) Vuex
  ( ) CSS Pre-processors
  (*) Linter / Formatter
  ( ) Unit Testing
  ( ) E2E Testing
```

The GUI offers a wide range of options for your project. To start the GUI, we firstly need to navigate to a folder where we want to create the project, open up a terminal and use the ``vue ui`` command. This will spin up a developer server at localhost:8000 where we can follow the steps in order to generate the project. It's divided under 4 tabs, them being: **Details**, **Presets**, **Features**, **Configuration**.

- **Details**
This tab contains options like the project directory, which package manager do we want to use (npm, yarn, pnpm), whether or not to initialize a git repository etc.
- **Presets**
Just like in the terminal above, we can select the presets for Vue 2, Vue 3, manually select a preset, or retrieve a preset from a git repository to which we need to give the link to the repository.
- **Features**
This tab will be enabled only if we choose to select features manually from the presets tab. It contains all of the options on the screenshot above.
- **Configuration**
This tab is also only enabled if we choose to select features manually. It provides various configuration settings for all of the features that we chose in the previous step.

Plugin Ecosystem

- **Vite**

For Vite there are 3 types of plugins:

1. Official Plugins

As the name suggests these are plugins made officially by Vite, and they all provide mostly support for Vue, React, React SWC, JSX etc..

2. Community Plugins

The community plugins are a curated list of plugins for Vite which are all gathered into a single repository on github and they are all divided into categories. For example there are plugins for templates, integrations, loaders, bundling, transformers, helpers and so on.

3. Rollup plugins

In short, Vite uses Rollup under the hood to create small and optimized builds for production. The rollup plugins can help with transpilation, minification, code-splitting, asset handling, css preprocessing etc.

- **Vue-CLI**

Vue CLI comes with official and community plugins as well. The official plugins of Vue CLI are the ones that we can select as features from the project creation window. For example Babel, Jest/Mocha, ESLint, Router etc.

The number of community plugins that exist for Vue CLI are greater than for Vite, and the main reason for that is that Vue CLI has been around for a lot longer than Vite so it's only normal. A simple search for `vitejs` on [npm](https://www.npmjs.com/) yields around 300 results, while for vue-cli there are around 5800 results.

So, if we value speed and simplicity, Vite's minimalistic plugin ecosystem might be a good fit. However if we need extensive customization, a broader range of plugin options, or have an existing project that relies on Webpack, Vue CLI's comprehensive plugin ecosystem may be more suitable.

Community and Documentation

- **Vite:**

Emerging Community: Vite, being a relatively newer build tool, has been rapidly gaining popularity within the Vue.js and front-end development communities. It has garnered attention for its speed and innovative development experience.

Active Development: Vite is actively developed by Evan You, the creator of Vue.js. This close relationship with Vue.js contributes to its popularity within the Vue ecosystem.

Official Documentation: Vite's official documentation is well-maintained, concise, and provides clear guidance on getting started and using the tool effectively. The documentation is continuously updated to reflect the latest changes and features.

Growing Ecosystem: Vite's ecosystem of official and community-contributed plugins and extensions is growing steadily, offering additional functionality for different use cases.

Community Support: While the community is still evolving, Vite users can find support and engage in discussions on forums, social media, and platforms like GitHub.

- Vue CLI:

Established Community: Vue CLI has an established and active community of Vue.js developers. It has been a preferred choice for Vue.js projects for several years.

Community-Driven: Vue CLI is a community-driven project, and it benefits from the collective knowledge and contributions of Vue.js developers worldwide.

Robust Documentation: Vue CLI's documentation is comprehensive and covers a wide range of topics. It is suitable for both beginners and experienced developers. The documentation is well-maintained and updated regularly.

Rich Ecosystem: Vue CLI has a rich ecosystem of official and community-created plugins and extensions. These plugins provide solutions for various development tasks, enhancing the development experience.

Active Support Channels: Vue CLI users can find assistance and discuss issues through community forums, social media, and GitHub. The active community ensures that developers can seek help when needed.

If you prefer a tool with a rapidly growing community and a focus on innovation, Vite might be a compelling choice.

If you value the stability of a mature ecosystem with extensive documentation and a well-established user base, Vue CLI offers a strong community and support system.

5. Use cases

Both Vite and Vue CLI are reliable build tools when we are constructing our Vue applications but they each excel at different areas. Some of the advantages of Vite.js over Vue CLI are:

Rapid Prototyping: Vite.js is excellent for quickly prototyping Vue.js applications. Its fast development server with HMR allows developers to iterate and see changes in real-time, making it ideal for experimenting with ideas.

Small to Medium-Sized Projects: Vite.js shines in projects that are relatively small or medium-sized, where development speed and efficiency are essential. It excels in scenarios where you want to optimize for quick development cycles.

Static Site Generation (SSG): Vite.js's support for generating static websites is beneficial for projects like blogs, documentation sites, and marketing websites that can benefit from pre-rendered pages for improved performance and SEO.

Server-Side Rendering (SSR): If you're building a Vue.js application with server-side rendering requirements, Vite.js can be used in combination with frameworks like Nuxt.js to facilitate SSR development.

Modern JavaScript: When you want to leverage modern JavaScript features and ECMAScript Modules (ESM) in your Vue.js application without extensive transpilation, Vite.js is a suitable choice.

Learning Vue.js: For developers who are learning Vue.js and want a simple and lightweight development setup, Vite.js provides a less complex environment compared to Vue CLI.

Although Vite has its advantages in speed and flexibility over Vue CLI, the latter still has uses in larger and more complex projects. Here are some of the advantages of Vue CLI:

Large and Complex Projects: Vue CLI is well-suited for large and complex applications where extensive configuration and customization are required. It offers more flexibility and can handle a wide range of project requirements.

Legacy Browser Support: When your project needs to support older browsers that don't fully support ESM, Vue CLI's ability to transpile code and provide compatibility is valuable.

Migration from Vue 2: If you have an existing Vue 2 project and plan to migrate it to Vue 3, Vue CLI offers tools and guides to help with the migration process.

Enterprise Applications: For enterprise-level projects with rigorous testing, deployment, and monitoring requirements, Vue CLI provides a robust framework for integration with CI/CD pipelines and other enterprise systems.

Full-Stack Development: If you're building a full-stack application and need to integrate the front end with back-end technologies like Express.js or Node.js, Vue CLI's ecosystem has plugins and tools to support such development.

Customization and Configuration: Vue CLI is a better choice when you need extensive control over project configuration. Its `vue.config.js` file allows for fine-tuning various aspects of your project setup.

Long-Term Support: When long-term stability and official support are critical, Vue CLI, as the official CLI tool for Vue.js, provides a high level of confidence in terms of maintenance and updates.

Projects with Established Workflows: If your team already has established workflows and processes using Vue CLI, it may be more convenient to stick with Vue CLI for consistency and familiarity.

6. Community & Support

Despite being the younger of the technologies Vite.js has a thriving community that is growing rapidly and becoming more active. It has active forums and communities who support the evolution of this technology and help other users understand it and advance their knowledge. Vite also supports third party plug-ins which are developed mostly by the community surrounding it. Here is how the Vite.js community is connected:

GitHub Repository: Vite.js is an open-source project hosted on GitHub, and it has a thriving repository with many contributors. The GitHub repository is where the core development of Vite.js takes place, and it serves as a central hub for discussions, bug reports, feature requests, and contributions.

Discord Community: Vite.js has an official Discord server where developers can join to discuss various aspects of the project, seek help, share their experiences, and engage with the community. The Discord server is often used for real-time discussions and Q&A sessions.

Documentation and Guides: The Vite.js project maintains comprehensive and well-organized documentation, including guides, API references, and examples. This documentation helps developers understand how to use Vite.js effectively and serves as a valuable resource for learning.

Plugin Ecosystem: Vite.js has an ecosystem of official and community-contributed plugins. Developers can create and share plugins to extend Vite's functionality. This ecosystem allows for customization and integration with various tools and technologies.

Community Contributions: The Vite.js community actively contributes to the project by reporting issues, suggesting enhancements, and submitting pull requests. The project's maintainers and contributors are responsive to community input.

Twitter and Social Media: Developers and enthusiasts often share their Vite.js-related experiences, updates, and projects on social media platforms like Twitter, Reddit, and others, using hashtags like `#vitejs`.

Meetups and Conferences: Vite.js may have had a presence at Vue.js-related meetups, conferences, and web development events. These gatherings provide opportunities for community members to network, share knowledge, and learn from one another.

Online Forums and Communities: Developers interested in Vite.js can find discussions and help on platforms like Stack Overflow and Vue.js forums, where questions related to Vite.js are addressed.

Tutorials and Blog Posts: Members of the community often create tutorials and blog posts to share their experiences and insights into using Vite.js effectively. These resources can be valuable for both beginners and experienced developers.

VitePress: VitePress is a documentation generator and static site generator built with Vite.js. It is often used for creating documentation websites, blogs, and knowledge bases related to Vite.js and other topics.

The Vite.js community was characterized by its enthusiasm, helpfulness, and a focus on making web development faster and more efficient. Developers were encouraged to contribute to the project, share their knowledge, and help one another.

Despite being the older of the two technologies, Vue CLI is still widely used for its stability and consistency when building larger Vue projects. The community is vibrant, thriving and ever expanding. Eager to help each other and help improve Vue CLI. Here is how the Vue CLI community is connected:

GitHub Repository: Vue CLI is an open-source project hosted on GitHub. The GitHub repository serves as the central hub for Vue CLI's development, discussions, issue tracking, and contributions. It's a place where users can report issues, suggest enhancements, and collaborate on improving the tool.

Official Forum: The Vue.js community maintains an official forum (forum.vuejs.org), which includes a dedicated category for Vue CLI. Developers often visit the forum to seek help, share their experiences, and discuss best practices related to Vue CLI and Vue.js development.

Discord Server: Vue.js has an official Discord server, which includes channels for discussions about Vue CLI and its related ecosystem. The Discord server is a space for real-time conversations, Q&A sessions, and community interactions.

Documentation and Guides: Vue CLI provides extensive and well-maintained documentation. The documentation includes guides, configuration references, and tutorials that help developers understand how to use Vue CLI effectively and efficiently.

Plugin Ecosystem: Vue CLI has a rich ecosystem of official and community-contributed plugins. Developers can create custom plugins to extend Vue CLI's capabilities, and many plugins are available to enhance project setups and workflows.

Twitter and Social Media: Developers, Vue.js enthusiasts, and the official Vue.js Twitter account often share updates, tips, and experiences related to Vue CLI on social media platforms like Twitter. The hashtag #VueCLI is commonly used for such discussions.

Meetups and Conferences: Vue.js and Vue CLI may have a presence at web development meetups, conferences, and events. These gatherings provide opportunities for community members to connect, share knowledge, and learn from experts.

Online Communities: Developers interested in Vue CLI can find discussions, articles, and help on platforms like Stack Overflow, Reddit (such as the r/vuejs subreddit), and other online developer communities.

Tutorials and Blog Posts: The Vue CLI community regularly produces tutorials, blog posts, and videos to share insights, best practices, and practical examples of Vue CLI usage. These resources are valuable for developers of all skill levels.

Contributions: Vue CLI welcomes contributions from the community, including bug fixes, new features, and improvements. Many developers actively contribute to Vue CLI's development and maintenance.

7. Conclusion

The comparison between Vite and Vue CLI highlights their distinct strengths and characteristics. Vite excels in rapid development, offering a quick setup and modern development experience. Vue CLI, with its extensive plugin ecosystem and stable community, is a versatile choice for customization and reliability.

The choice between these tools depends on your project's needs. Vite is ideal for modern web development, while Vue CLI suits projects requiring customization and stability. Both tools empower developers to create exceptional web experiences in an ever-evolving field.

As you consider the right tool for your project, remember that staying informed about emerging technologies is essential. Your understanding of Vite and Vue CLI will enhance your ability to make informed decisions and create outstanding web applications.