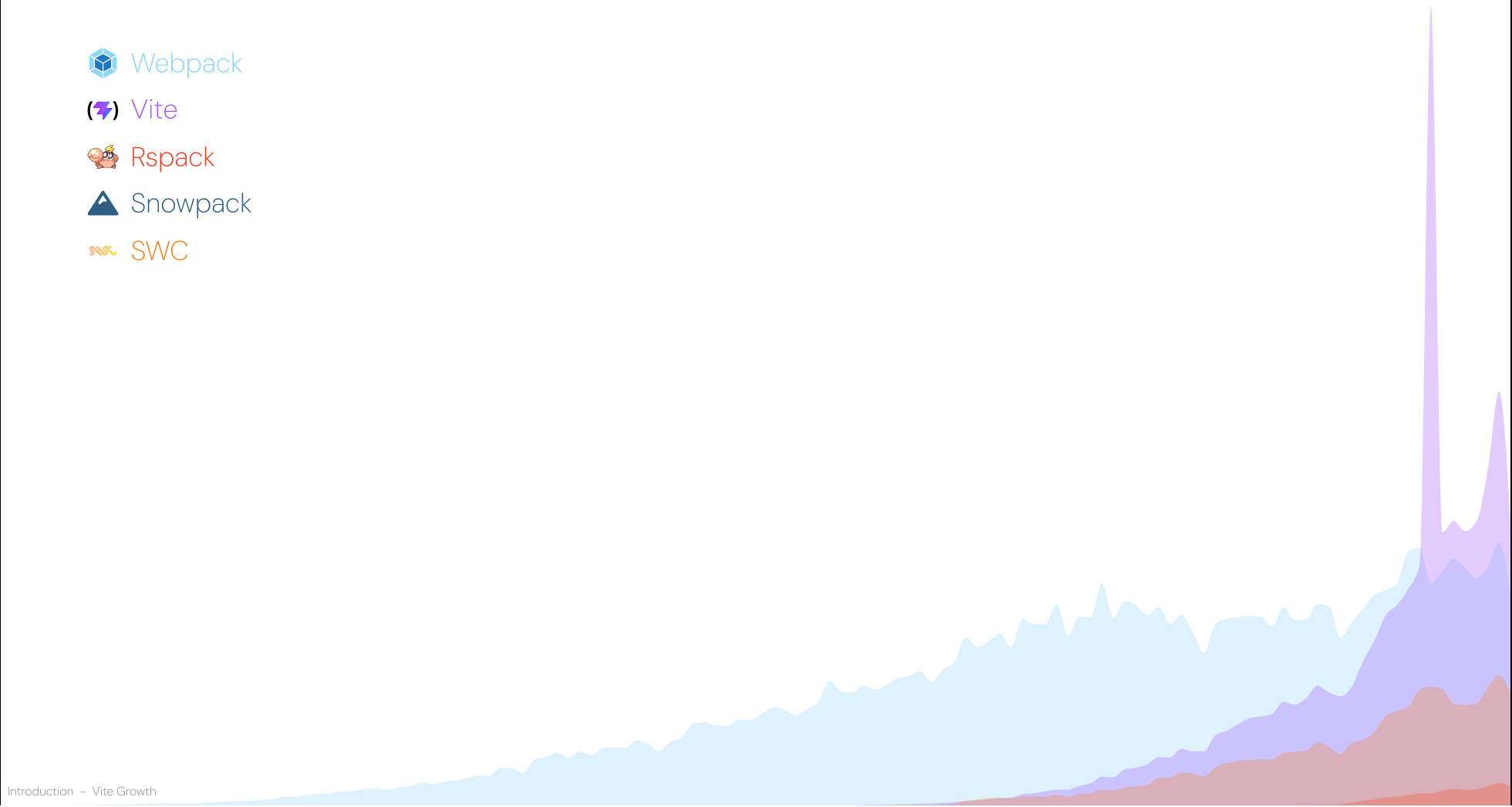
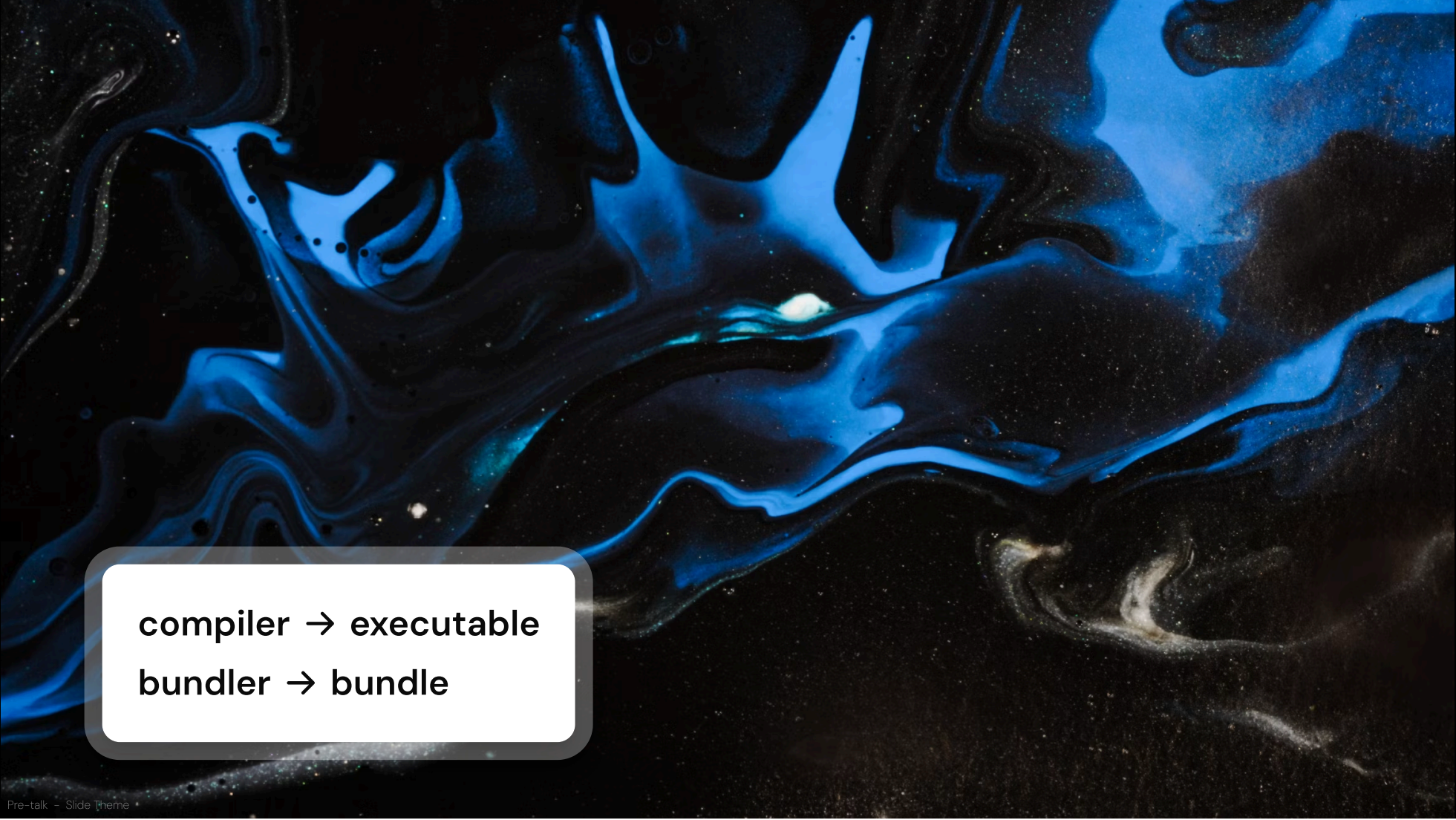
# Au cœur d’une pipeline: démystifions Vite et ses plugins

DevLille · 12 juin 2026


VITE

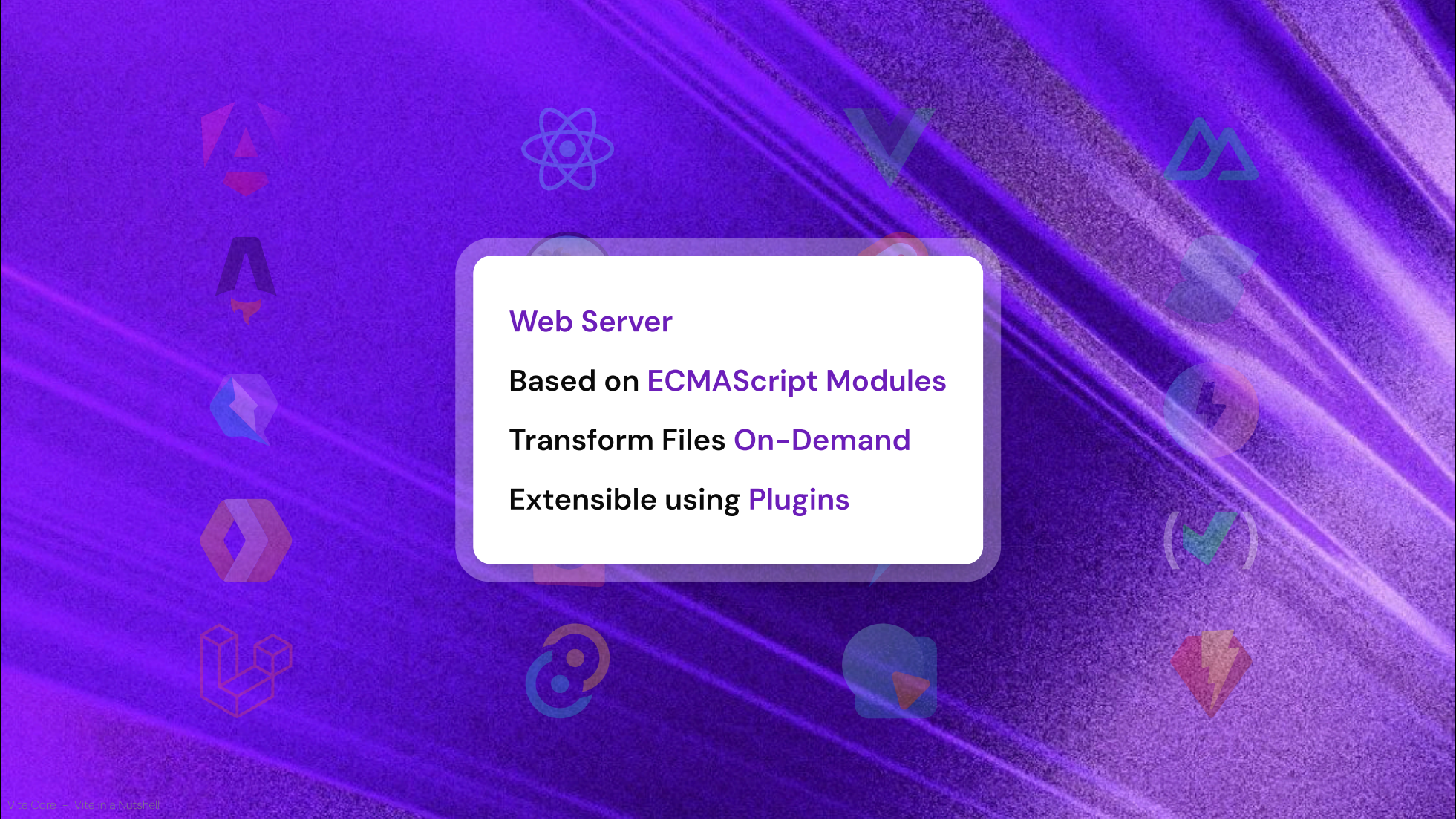
- Webpack
- Vite
- Rspack
- Snowpack
- SWC




compiler → executable
bundler → bundle



Type your message...


Web Server
Based on ECMAScript Modules
Transform Files On-Demand
Extensible using Plugins

HTTP Logs

```
GET http://localhost:5173/
GET http://localhost:5173/src/main.ts?t=1772882882631
GET http://localhost:5173/src/style.css?t=1772882882631
GET http://localhost:5173/node_modules/.vite/deps/vue.js?v=ed2c93c8
GET http://localhost:5173/src/App.vue
```

Native ESM instead of bundling-first

Dev server performance

On-demand transformation

Simpler configuration

Ecosystem alignment

## ECMAScript module N.

A JavaScript file that uses `import` and `export`. The standard module format in modern JavaScript.

*e.g.* `main.ts`, `App.vue` *once transformed for the browser.*

## module identifier N.

The string used in an import to request a module from Vite.

*e.g.* `'vue'`, `'./App.vue'`, `'virtual:my-module'`.

## hook N.

A function exposed by a plugin to intervene at a specific step in the pipeline.

*e.g.* `resolveId`, `load`, `transform`.

## virtual module N.

A module that does not exist on disk, but is generated on the fly by a plugin.

*e.g. an import like* `'virtual:my-module'`.

### HTML

Use HTML as an entry point with automatic script and asset handling.

### JSX

Use JSX with built-in transformation and framework support.

### CSS

Import CSS with support for HMR, modules, and preprocessors.

### Static Assets

Import static assets such as images, fonts, and media.

### JSON

Import JSON files directly as JavaScript modules.

### Glob Import

Import multiple modules using a file pattern.

### Dynamic Import

Load modules dynamically with native code splitting.

### WebAssembly

Import and run WebAssembly modules from JavaScript.

### Web Workers

Run code in background threads using Web Workers.

Feature Plugins = Everything is a Plugin

CSS Imported into a TypeScript File

27 / 87 · 31%

27
VOTES

An Image Loaded in a TypeScript File

27 / 87 · 31%

27
VOTES

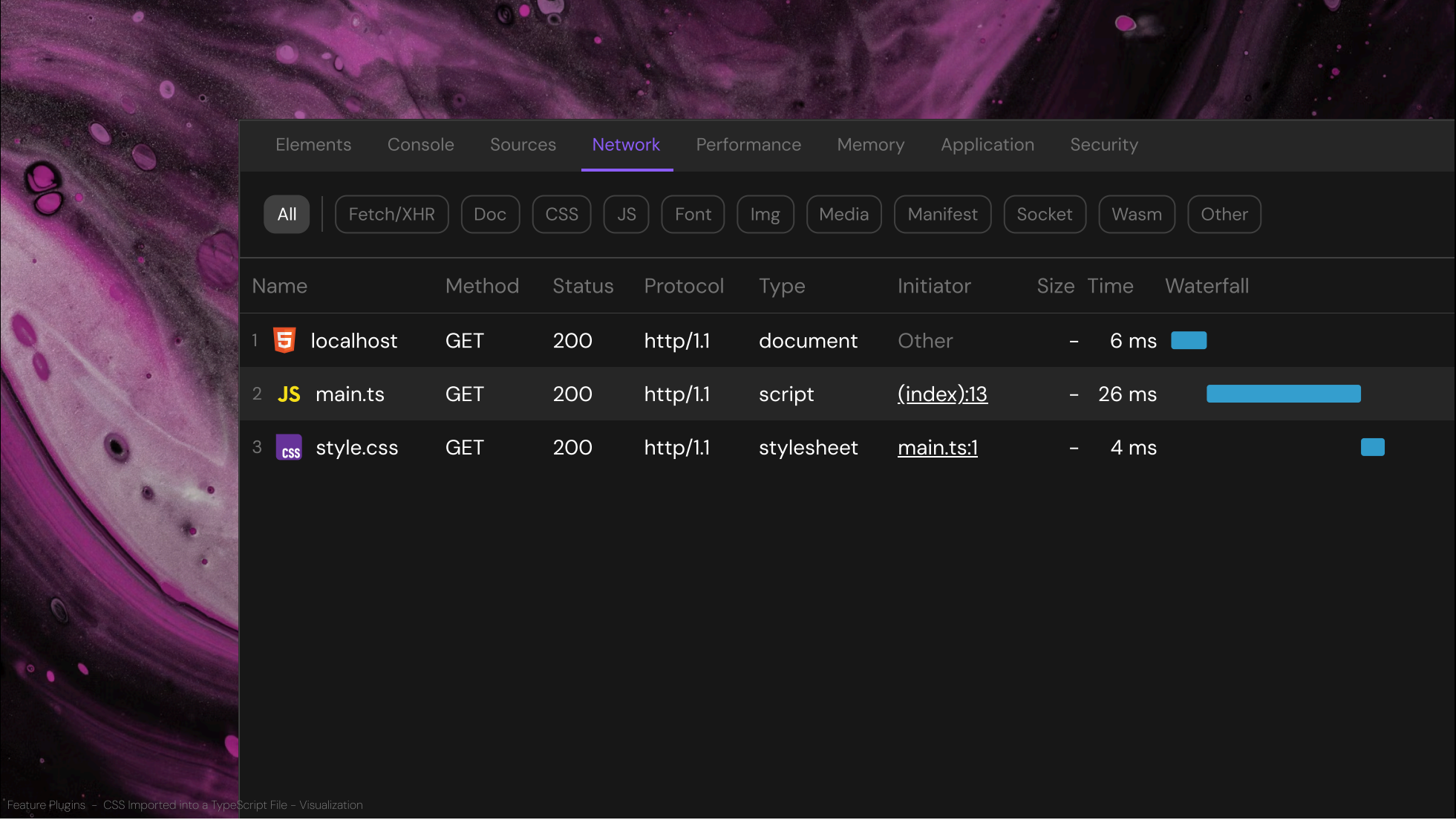
JSX Loaded Naturally

33 / 87 · 38%

33
VOTES

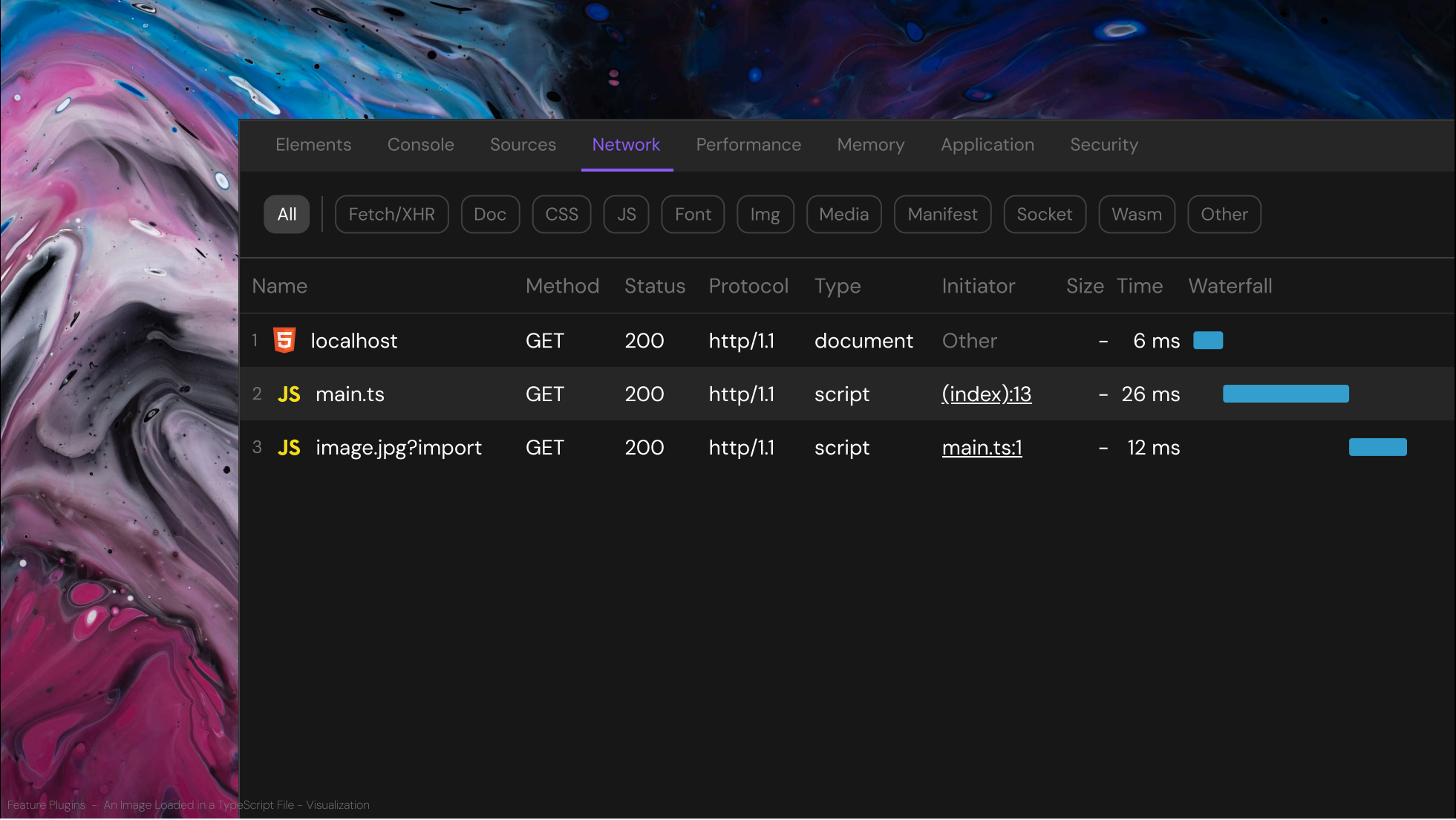
```
import './style.css'
```




Elements
Console
Sources
Network
Performance
Memory
Application
Security
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
Name
Method
Status
Protocol
Type
Initiator
Size
Time
Waterfall
1 localhost GET 200 http/1.1 document Other - 6 ms
2 JS main.ts GET 200 http/1.1 script (index):13 - 26 ms
3 CSS style.css GET 200 http/1.1 stylesheet main.ts:1 - 4 ms




```
import img from './image.png'
```

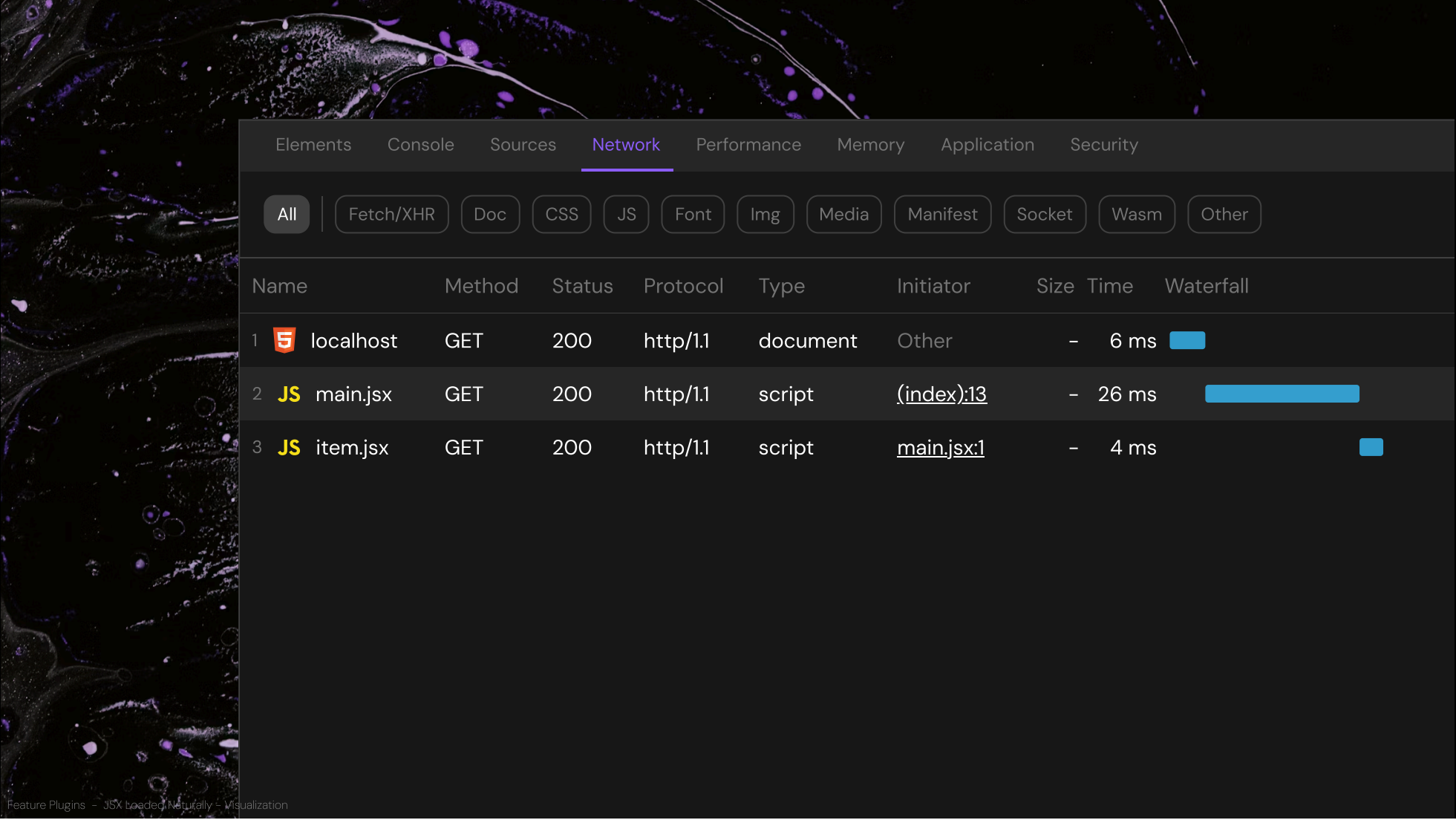



Feature Plugins – An Image Loaded in a TypeScript File – Visualization


Vite Build Output
dist
index.html
assets
image-BTTwLOJ2.jpg
JS
index-CglLWvq0.js

```
1   import { createRoot } fr
2   import { Item } from './I
3
4   function App() {
5     return (
6       <div>
7         <h1>My App</h1>
8         <Item name="Item 1"
9         <Item name="Item 2"
10      </div>
11    )
12  }
13
14  const root = createRoot(
15  root.render(<App />)
```






Feature Plugins – JSX Loaded Naturally – Visualization

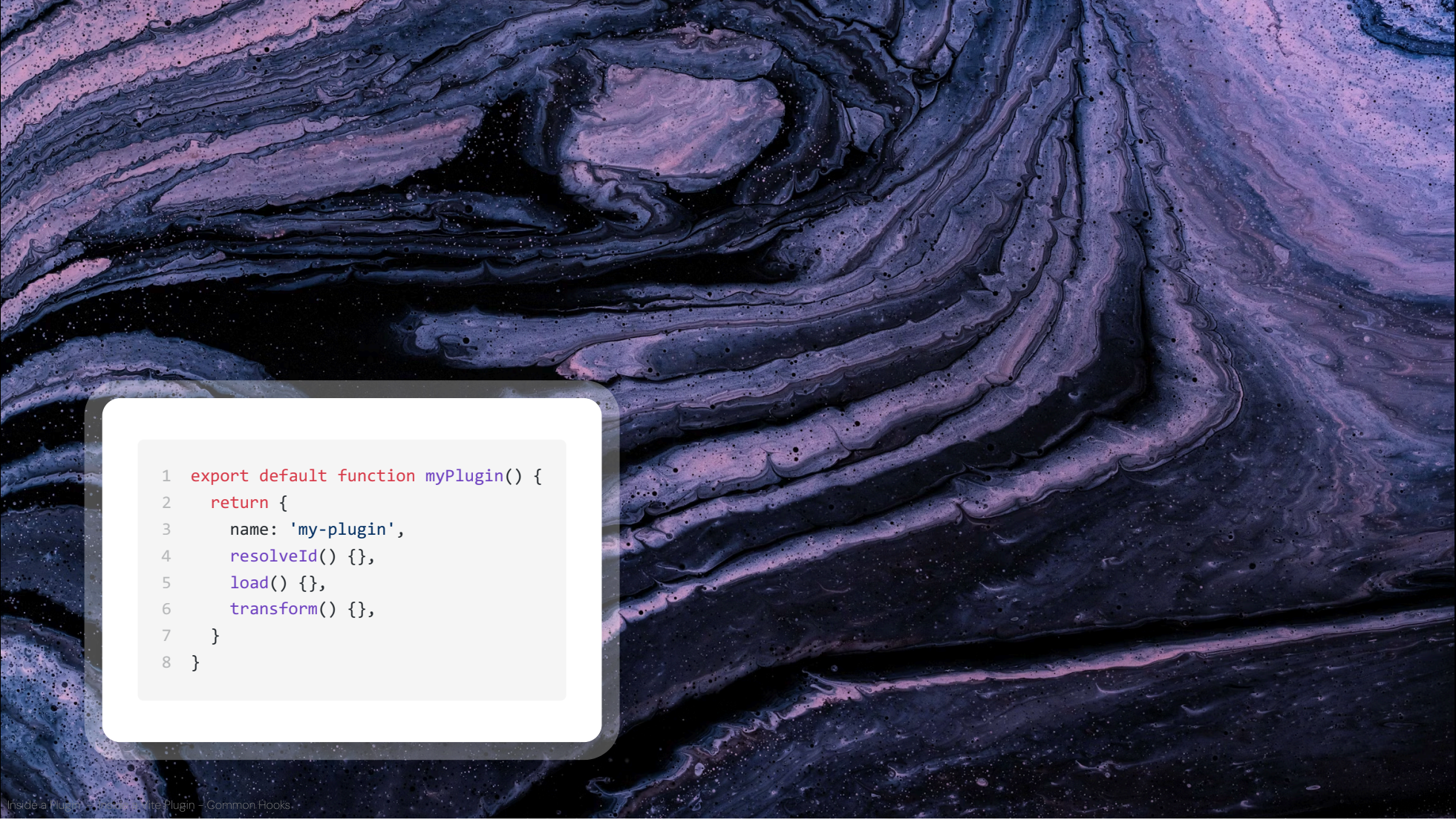
# Vite in 3 Points




1. **A web server for development**
   It handles requests that can be transformed on demand

2. **A bundler for production**
   It transforms our code for production

3. **A plugin system**
   To extend its features and make it do whatever we want


export default function myPlugin() {
return {
name: 'my-plugin',
resolveId() {},
load() {},
transform() {},
}
}


export function myPlugin() {
return {
name: 'my-plugin',
resolveId(id) { // /src/components/HelloWorld.vue
if (id.endsWith('.vue')) {
return id
}
},
}
}


export function myPlugin() {
return {
name: 'my-plugin',
async load() {
return "export default 'Hello World'"
}
},
}


export function myPlugin() {
return {
name: 'my-plugin',
transform(code, id) {
if (id.endsWith('.vue')) {
return transformSFC(code, filename)
}
}
},
}

# Multiple plugins?

request

**resolveId** `resolveId (id)`

VuePlugin

CustomPlugin

**load** `load (id)`

VuePlugin

CustomPlugin

**transform** `transform (code, id)`

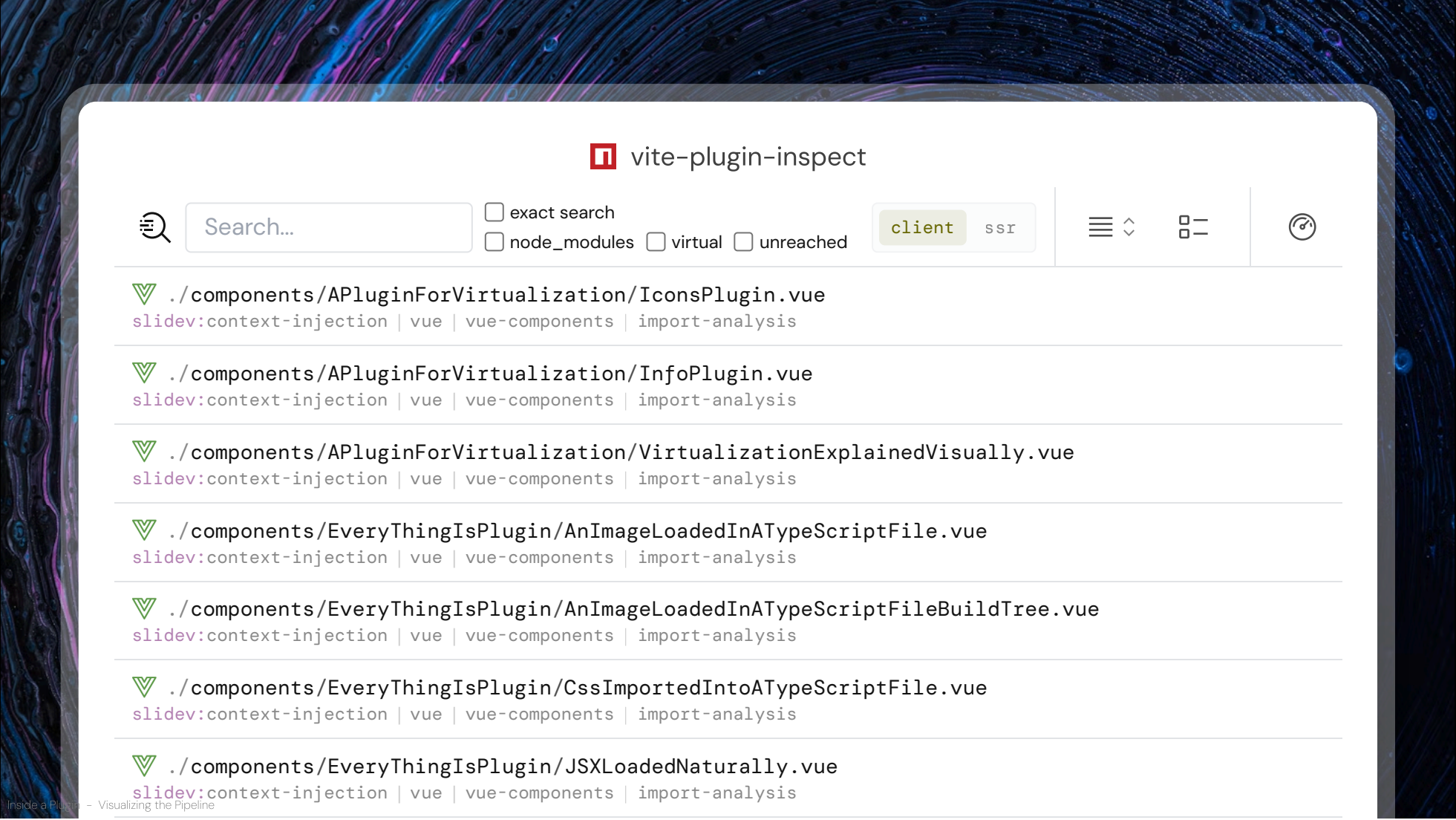
VuePlugin

CustomPlugin

response

input

output


vite-plugin-inspect
Search...
exact search
node_modules
virtual
unreached
client
ssr
./components/APluginForVirtualization/IconsPlugin.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/APluginForVirtualization/InfoPlugin.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/APluginForVirtualization/VirtualizationExplainedVisually.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/EveryThingIsPlugin/AnImageLoadedInATypeScriptFile.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/EveryThingIsPlugin/AnImageLoadedInATypeScriptFileBuildTree.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/EveryThingIsPlugin/CssImportedIntoATypeScriptFile.vue
slidev:context-injection | vue | vue-components | import-analysis
./components/EveryThingIsPlugin/JSXLoadedNaturally.vue
slidev:context-injection | vue | vue-components | import-analysis

```
export default function myPlugin() {
  let config = null
  return {
    name: 'my-plugin',
    config(config) {
      config.root = 'src'
    },
    configResolved(resolvedConfig) {
      config = resolvedConfig
    },
    async buildStart() {
      await scanDir(config.root)
    },
    async buildEnd() {
      await generateSitemap(config)
    },
  }
}
```


So what does
that look like?

# Simple Transform Plugin

HTTP Logs

```
GET http://localhost:5173/
GET http://localhost:5173/src/main.ts?t=1773357600000
```



```
export default function simpleTransform() {
  return {
    name: "simple-transform",
    transform(code) {
      return code.replace("__BUILD_TIME__", new Date().toString())
    }
  }
}
```

# External Markdown Loader

HTTP Logs

```
GET http://localhost:5173/
GET http://localhost:5173/src/main.ts?t=1773357600000
GET http://localhost:5173/src/daily-news.md?t=1773357600001
```

```ts
import markdownit from 'markdown-it'

export default function externalMarkdown() {
  const md = markdownit()

  return {
    name: "external-markdown",
    async load(id) { // /src/daily-news.md
      if (!id.endsWidth(".md"))
        return

      return fetch(`https://api.example.com/${id}`).then(res => res.text())
    },
    async transform(code, id) {
      if (!id.endsWidth(".md"))
        return

      return `const html = "${md.render(code)}"\n\nexport default html`
    }
  }
}
```

# The Vite Plugin in 3 Points




1. **A Vite plugin is a function that returns an object**
   That object must at least have a name property

2. **There are 3 main hooks to act on modules**
   resolveId, load, and transform

3. **There are also hooks for Vite's lifecycle**
   config, configResolved, buildStart, buildEnd, and closeBundle




```
export default function myPlugin() {
  return {
    name: 'my-plugin',
    resolveId(id) {
      // Used to intercept the module identifier and return a new one
    },
    load(id) {
      // Used to return the content of a module given its identifier
    },
    transform(code, id) {
      // Used to transform the content of a module given its identifier
    }
  }
}
```

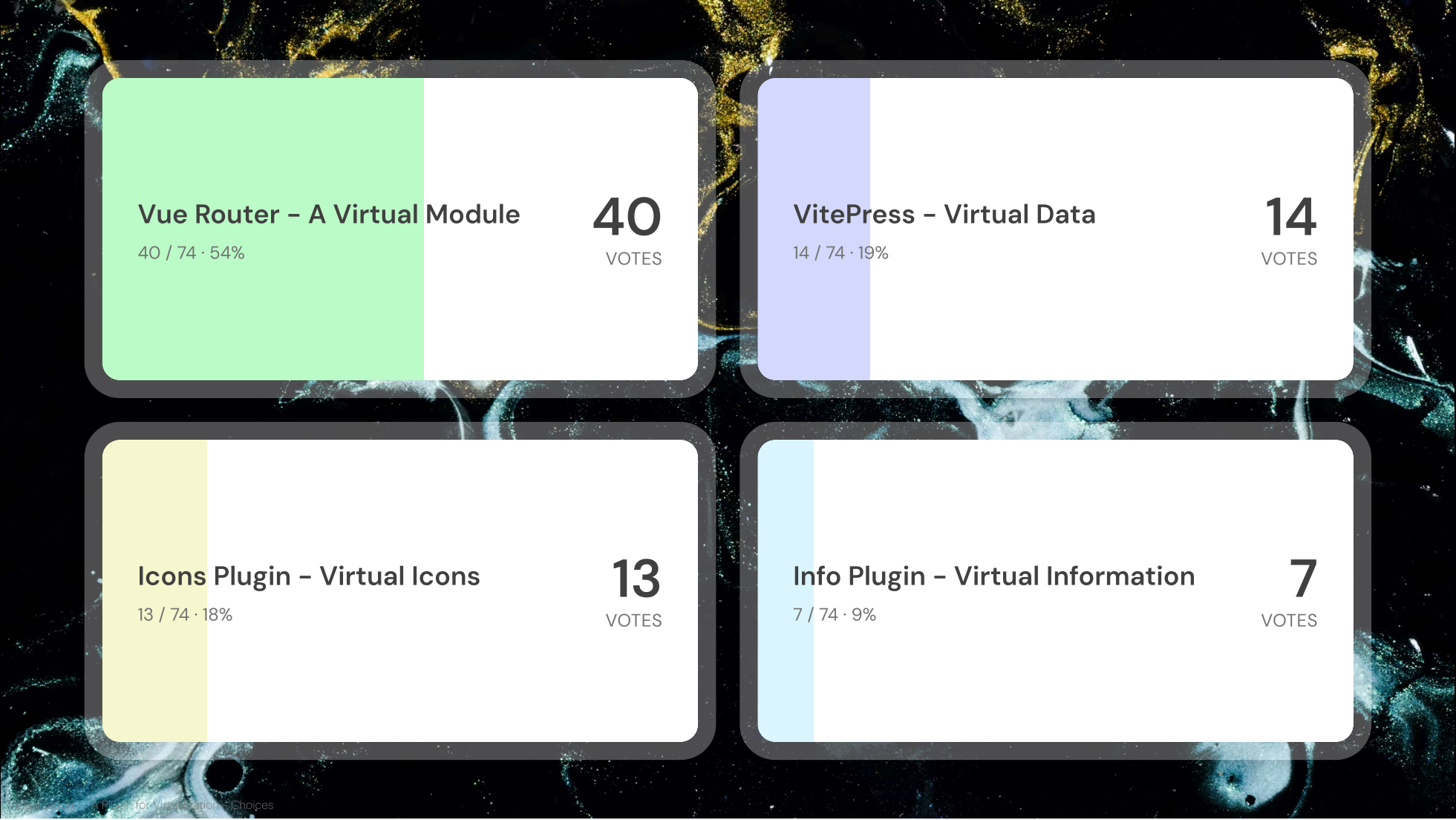
# A Plugin for Virtualization

HTTP Logs

```
GET http://localhost:5173/
GET http://localhost:5173/src/main.ts?t=1772882882631
GET http://localhost:5173/@id/__x00__virtual:my-module
```

```
export default function myVirtualModulePlugin() {
  const virtualModuleId = 'virtual:my-module'
  const resolvedVirtualModuleId = '\0' + virtualModuleId

  return {
    name: 'my-virtual-module-plugin',
    resolveId(id) {
      if (id === virtualModuleId) {
        return resolvedVirtualModuleId
      }
    },
    load(id) {
      if (id === resolvedVirtualModuleId) {
        return `export const msg = "from virtual module"`
      }
    },
  }
}
```

```ts
import { defineConfig } from 'vite'
import virtual from 'vite-plugin-virtual'

export default defineConfig({
  plugins: [
    virtual({
      'virtual:git:commit': () => {
        const commit = execSync('git rev-parse HEAD').toString().trim()
        return `export default "${commit}"`
      },
      'virtual:api:data': async () => {
        const res = await fetch('https://api.example.com/data')
        const data = await res.json()
        return `export default ${JSON.stringify(data)}`
      },
    })
  ]
})
```


Vue Router - A Virtual Module
40 / 74 · 54%
40
VOTES
VitePress - Virtual Data
14 / 74 · 19%
14
VOTES
Icons Plugin - Virtual Icons
13 / 74 · 18%
13
VOTES
Info Plugin - Virtual Information
7 / 74 · 9%
7
VOTES


import { createApp } from 'vue'
import { createRouter, createWebHistory } from 'vue-router'
import { routes } from 'vue-router/auto-routes'
import App from './App.vue'
const router = createRouter({
history: createWebHistory(),
routes,
})
createApp(App)
.use(router)
.mount('#app')

```ts
import fs from 'node:fs/promises'

export default function autoRoutes() {
  return {
    name: 'auto-routes',
    resolveId(id) {
      if (id === 'vue-router/auto-routes') {
        return '\0vue-router/auto-routes'
      }
    },
    load(id) {
      if (id === '\0vue-router/auto-routes') {
        const files = await fs.readdir('./src/pages')
        const routes = await generateRoutes(files)
        return `export const routes = ${JSON.stringify(routes)}`
      }
    },
  }
}
```

```
// example.data.js
export default {
  load() {
    return {
      hello: 'world'
    }
  }
}
```

```
<!-- example.vue -->
<script setup>
import { data } from './example.data.js'
</script>

<pre>
{{ data.hello }} <!-- "world" -->
</pre>
```

```ts
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [
    {
      name: 'data-plugin',
      resolveId(id) {
        if (id.endsWith('.data.js')) {
          return `\0${id}`
        }
      },
      async load(id) {
        if (id.startsWith('\0') && id.endsWith('.data.js')) {
          const dataLoader = await import(id.slice(1))
          const data = await dataLoader.default.load()
          return `export default ${JSON.stringify(data)}`
        }
      },
    },
  ],
})
```

```vue
<script setup>
import IconAccessibility from '~icons/carbon/accessibility'
import IconAccountBox from '~icons/mdi/account-box'
</script>

<template>
  <IconAccessibility />
  <IconAccountBox style="font-size: 2em; color: red" />
</template>
```





```ts
import { defineConfig } from 'vite'
import { loadIcon } from 'iconify'

export default defineConfig({
  plugins: [
    {
      name: 'icons',
      resolveId(id) {
        if (id.startsWith('~icons/')) {
          return '\0' + id
        }
      },
      async load(id) {
        if (id.startsWith('\0~icons/')) {
          const iconName = id.slice(7)
          const svg = await loadIcon(iconName)
          return `export default ${JSON.stringify(svg)}`
        }
      },
    },
  ],
})
```

```
import { sha } from '~build/info'
import now from '~build/time'

console.log(`Build ${sha} at ${now}`)
```

Build 724d7a5b05528e58a4330e04e414be9ebffeff8f at Sat Jun 20 2026 15:03:36 GMT+0200 (Central European Summer Time)

```
import { execSync } from 'child_process'
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [
    {
      name: 'build-info',
      resolveId(id) {
        if (id === '~build/time' || id === '~build/git') {
          return '\0' + id
        }
      },
      load(id) {
        if (id === '\0~build/time') {
          return `export default ${JSON.stringify(new Date().toISOString())}`
        }
        if (id === '\0~build/git') {
          const sha = execSync('git rev-parse HEAD').toString().trim()
          return `export default ${JSON.stringify(sha)}`
        }
      },
    },
  ],
})
```

# Virtualization in 3 Points




1. **Vite can respond to requests with virtual modules**
   Modules that do not exist on the disk but are generated on the fly by a plugin

2. **To create a virtual module, use the resolveId and load hooks**
   Take the ownership of an import with resolveId and return the code with load

3. **Used in many plugins and very powerful**
   It allows you to inject dynamic data, generate code on the fly, and much more




```
export default function myVirtualModulePlugin() {
  const virtualModuleId = 'virtual:my-module'
  const resolvedVirtualModuleId = `\0${virtualModuleId}`

  return {
    name: 'my-virtual-module-plugin',
    resolveId(id) {
      if (id === virtualModuleId) {
        return resolvedVirtualModuleId
      }
    },
    load(id) {
      if (id === resolvedVirtualModuleId) {
        // Do whatever you want
      }
    },
  }
}
```

### Hot Module Replacement (HMR)

Update modules in the browser without a full page reload, preserving application state.

### Middleware

Add additional routes to the Vite dev server to handle custom API requests or serve dynamic content.

### Command Runner

Execute custom commands alongside the Vite dev server, with access to the Vite configuration and lifecycle.

### Inter-process Communication

Allow different processes to communicate with each other to share data.

### Macros

Transform source code at build time to enable new syntax or features that are not natively supported by JavaScript.

### Server Framework

Extend Vite's dev server capabilities to build custom backends, APIs, or full-stack applications.

Hot Module Replacement (HMR)

23 / 52 · 44%

23 VOTES

Middleware - A Virtual File

7 / 52 · 13%

7 VOTES

Run Plugin - A Plugin for Running Commands

4 / 52 · 8%

4 VOTES

Laravel Vite - Inter-process Communication

5 / 52 · 10%

5 VOTES

unplugin-macro - A Plugin for Creating Macros

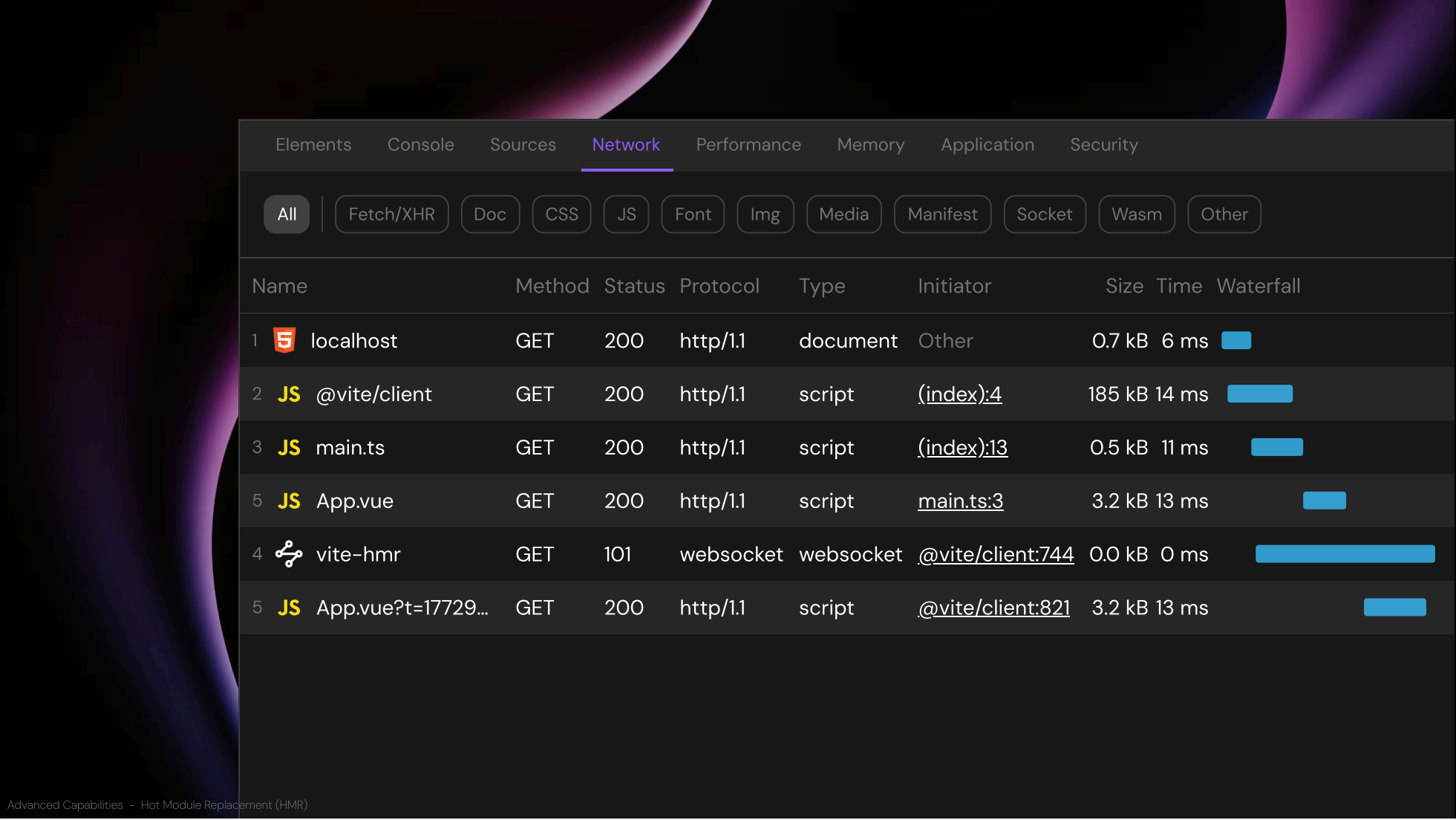
0 / 52 · 0%

0 VOTES

Nitro - A Backend

13 / 52 · 25%

13 VOTES

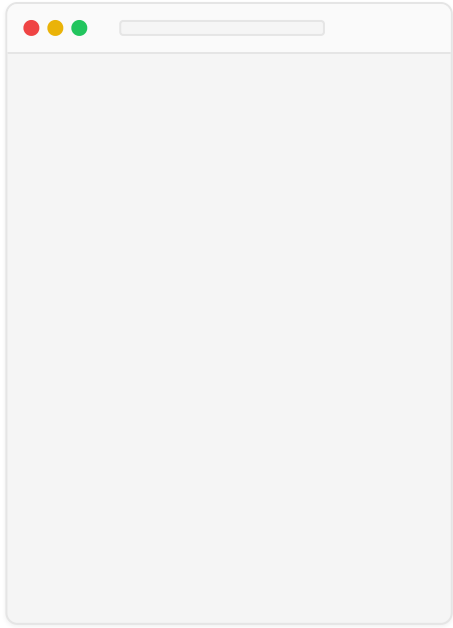

Elements
Console
Sources
Network
Performance
Memory
Application
Security
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
Socket
Wasm
Other
Name
Method
Status
Protocol
Type
Initiator
Size
Time
Waterfall
1 localhost GET 200 http/1.1 document Other 0.7 kB 6 ms
2 JS @vite/client GET 200 http/1.1 script (index):4 185 kB 14 ms
3 JS main.ts GET 200 http/1.1 script (index):13 0.5 kB 11 ms
5 JS App.vue GET 200 http/1.1 script main.ts:3 3.2 kB 13 ms
4 vite-hmr GET 101 websocket websocket @vite/client:744 0.0 kB 0 ms
5 JS App.vue?t=17729... GET 200 http/1.1 script @vite/client:821 3.2 kB 13 ms

```
export default function myHmrPlugin() {
  return {
    name: 'my-hmr-plugin',
    handleHotUpdate(ctx) {
      const { file, server, modules } = ctx
      if (file.endsWith('.myext')) {
        const mod = server.moduleGraph.getModuleById(file)
        if (mod) {
          server.moduleGraph.invalidateModule(mod)
        }
      }
    },
  }
}
```

```
import sirv from 'sirv'

{
  name: 'vite-plugin-inspect',
  configureServer(server) {
    server.middlewares.use(`${base}/__inspect`, sirv(DIR_CLIENT, {
      single: true,
      dev: true,
    }))
  }
}
```

```
import { defineConfig } from 'vite'
import { run } from 'vite-plugin-run'

export default defineConfig({
  plugins: [
    run([
      {
        name: 'generate ziggy types',
        run: ['php', 'artisan', 'ziggy:generate', '--types'],
        pattern: [/web.php$/, /api.php$/],
      }
    ]),
  ],
})
```

```
import { execSync } from 'node:child_process'
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [
    {
      name: 'vite-plugin-run',
      handleHotUpdate({ file }) {
        if (/web\.php$/.test(file)) {
          execSync('php artisan ziggy:generate --types')
        }
      }
    }
  ]
})
```

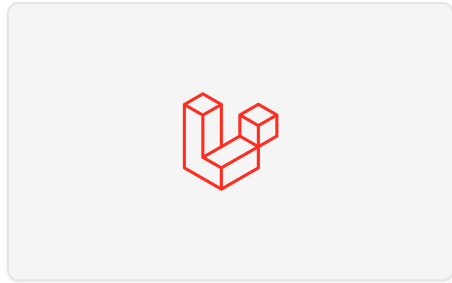


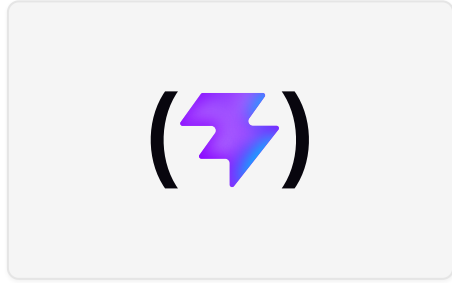
Ask Page →

← Send Page

Ask Assets →

← Send Assets





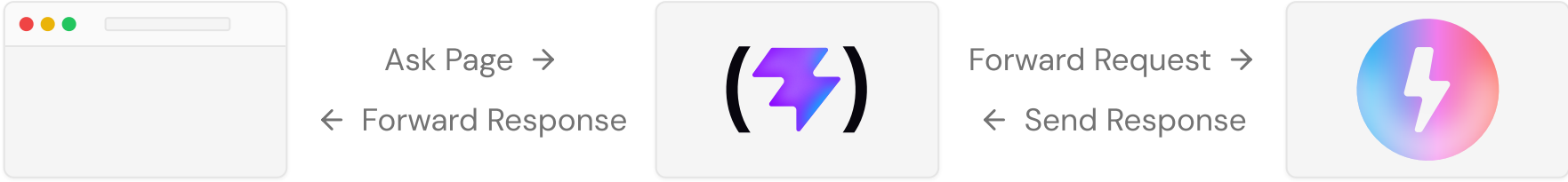
```
import fs from 'node:fs'

export default function laravel() {
  return {
    name: 'vite-plugin-laravel',
    configureServer(server) {
      server.httpServer?.once('listening', () => {
        const address = server.httpServer?.address()
        if (isAddressInfo(address)) {
          const viteDevServerUrl = resolveDevServerUrl(address, server.config)

          fs.writeFileSync('./public/hot', `${viteDevServerUrl}${server.config.base.replace(/\/$/, '')}`)
        }
      })
    }
  }
}
```

```ts
export function getRandom() {
  return Math.random()
}
export const buildTime = Date.now()
```

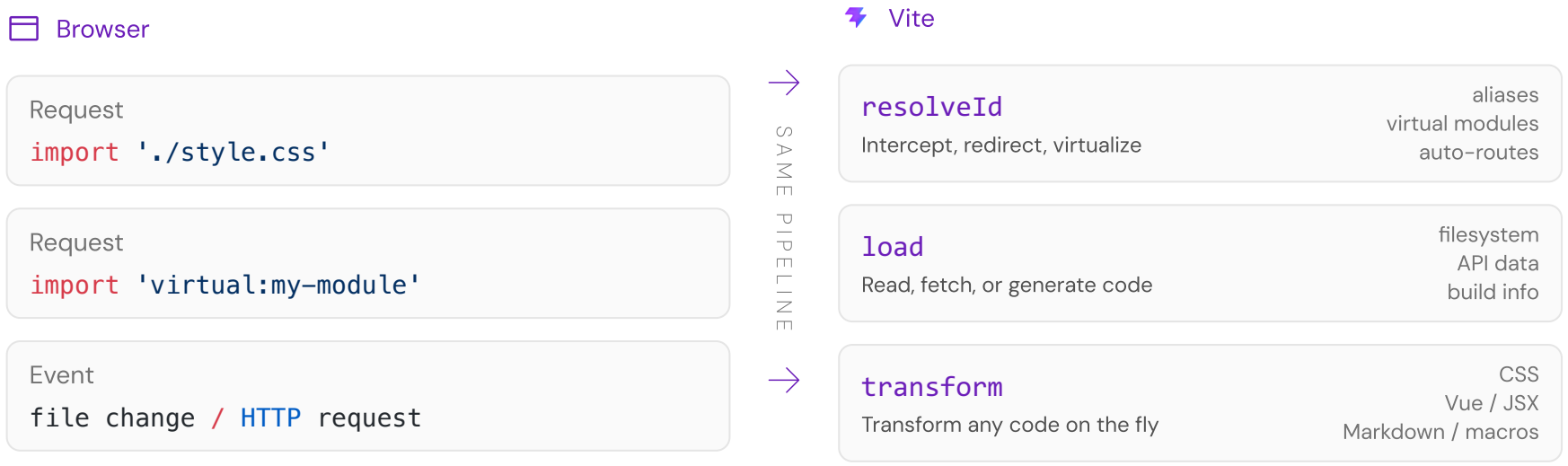
```ts
import { buildTime, getRandom } from './macros' with { type: 'macro' }

getRandom()
buildTime
```

```ts
import { defineConfig } from 'vite'

export default defineConfig({
  plugins: [
    {
      name: 'macro',
      async transform(code, id) {
        if (id.endsWith('.ts')) {
          const program = babelParse(code)
          const macros = collectMacro(program)

          for (macro of macros) {
            const result = await executeMacro(macro)
            code = replaceMacroWithResult(code, macro, result)
          }

          return code
        }
      }
    }
  ]
})
```

# Nitro as a Backend


Ask Page →
← Forward Response
Forward Request →
← Send Response

```
export default defineConfig({
  plugins: [(() => {
    return {
      name: 'nitro',
      configureServer(server) {
        server.middlewares.use((req, res, next) => {
          if (isNitroRequest(req.url)) {
            nitroDevMiddleware(req, res, next)
          }
          else {
            next()
          }
        })
      }
    }
  })()]
})
```




**Plugins**

CSS, JSX, Vue, images, Markdown...

**Generated modules**

Routes, icons, virtual data, build information...

**Around the pipeline**

HMR, middleware, commands, backend, lifecycle hooks...

# What to Keep in Mind




1. **A pipeline between your files and the browser**
   Each request goes through Vite, which can transform modules on demand

2. **Plugins can create modules that do not exist**
   Virtual modules, routes, icons, build info... everything can be generated on demand

3. **Everything can be extended by a plugin**
   Code transformation, HMR, middlewares, side effects, macros, backend...




```ts
export default defineConfig({
  plugins: [
    (() => {
      return {
        name: 'my-awesome-plugin',
        resolveId(id) {}, // if you want to create virtual modules
        load(id) {}, // if you want to create virtual modules
        transform(code, id) {}, // if you want to transform modules
        handleHotUpdate(ctx) {}, // if you want to handle HMR
        configureServer(server) {}, // if you want to do something to the dev server
        // ... and many more hooks for different capabilities
      }
    })()
  ]
})
```


import { defineConfig } from 'vite'
export default defineConfig({
plugins: [
{
name: 'vue-plugin',
transform: {
filter: /\.vue$/,
handler(code, id) {}
}
},
]
})

In 45 minutes, you discovered what you need to build your Vite plugin.






C'était

# Au cœur d'une pipeline : démystifions Vite et ses plugins

Pour **DevLille** (12 juin 2026)

Présenté par  Estéban Soubiran

⬇️ Votre feedback ⬇️

example.com

🎁 Contenu additionnel 🎁