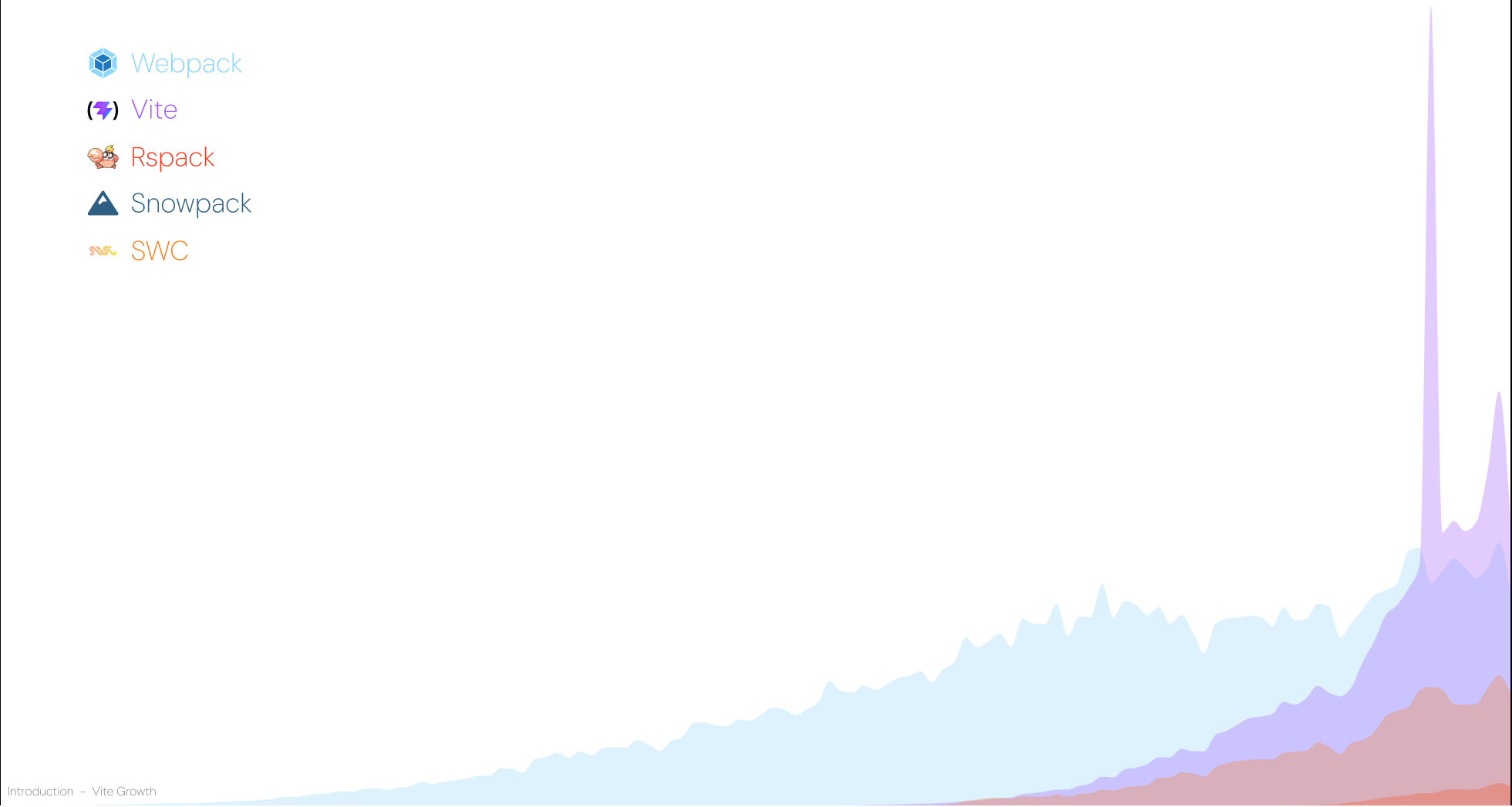
Au cœur d’une pipeline: démystifions Vite et ses plugins


VITE

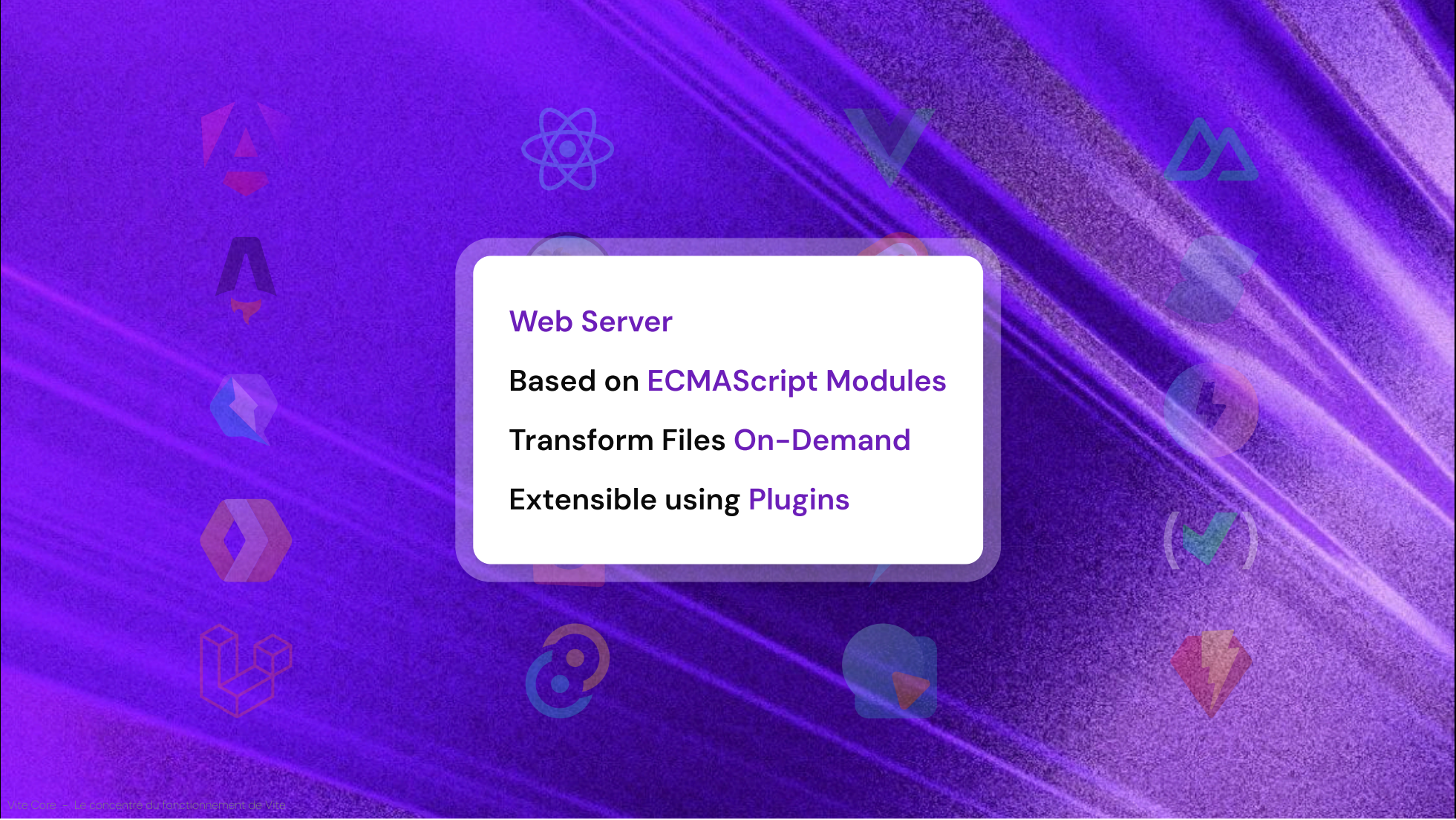
- Webpack
- Vite
- Rspack
- Snowpack
- SWC




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
```

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 - Toujnest que plugin

Du CSS importé dans un fichier TypeScript

1 / 3 · 33%

1

VOTES

Une image chargée dans un fichier TypeScript

1 / 3 · 33%

1

VOTES

JSX chargé naturellement

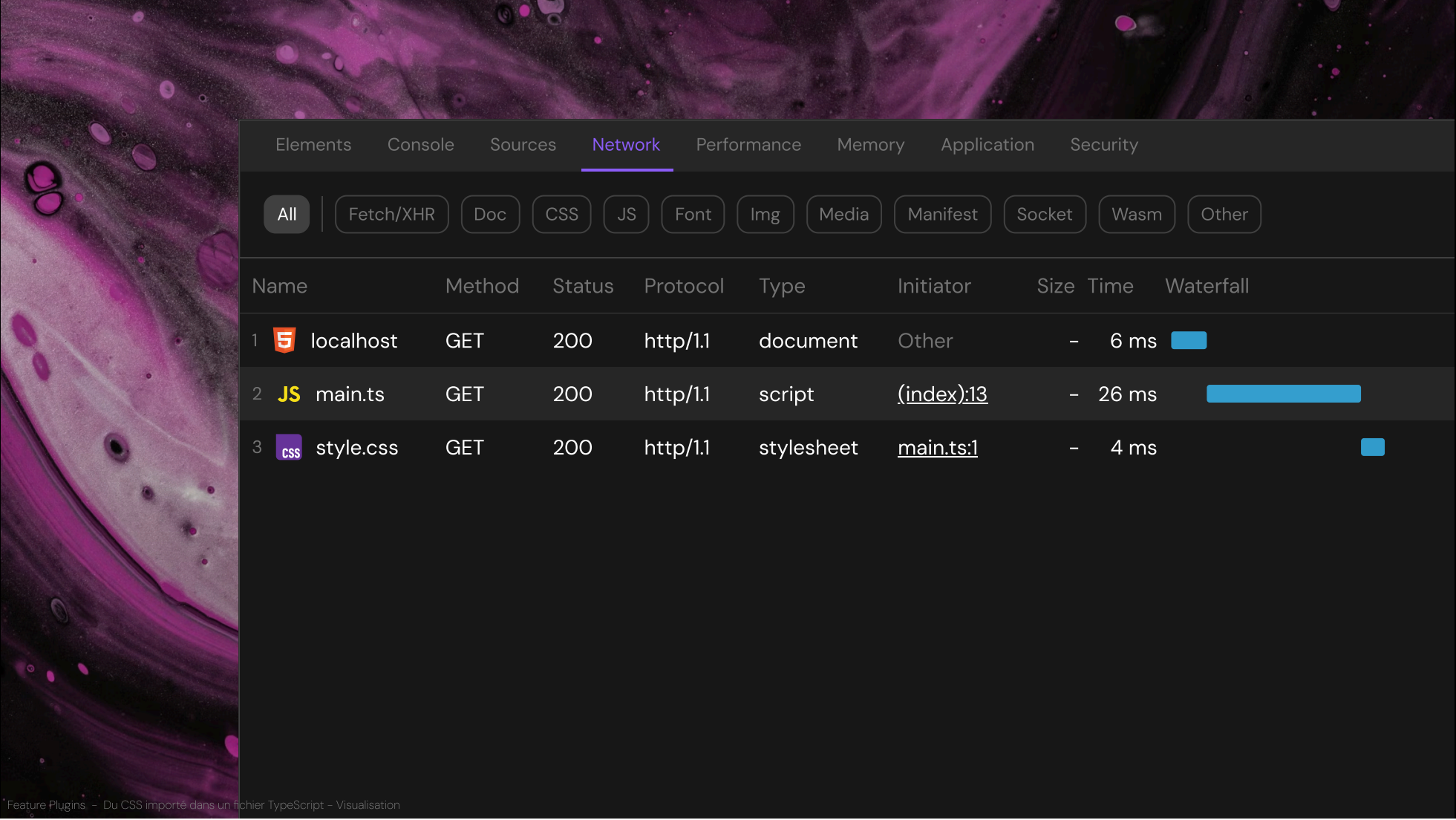
1 / 3 · 33%

1

VOTES

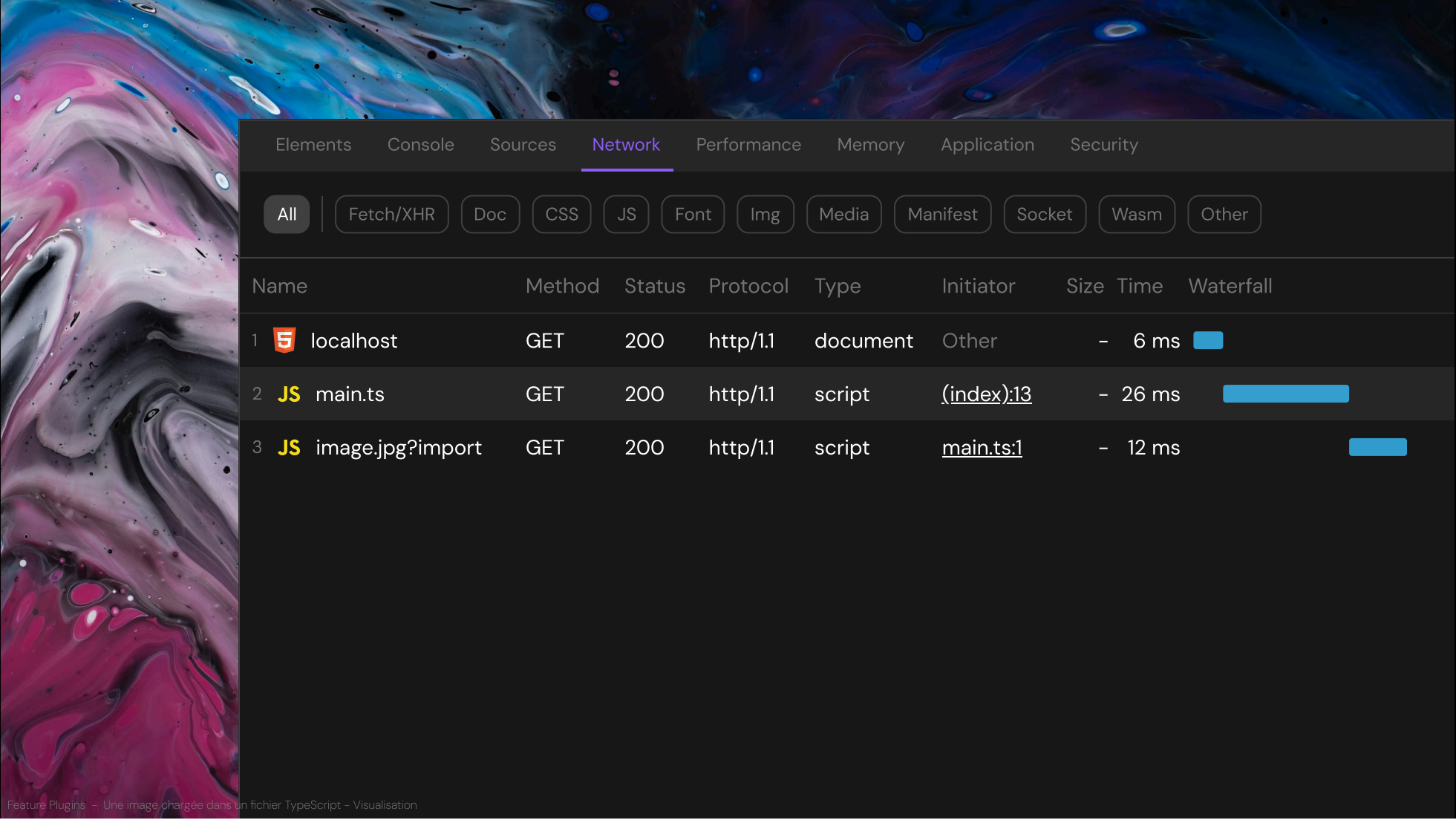
```
import './style.css'
```




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 – Une image chargée dans un fichier TypeScript - Visualisation

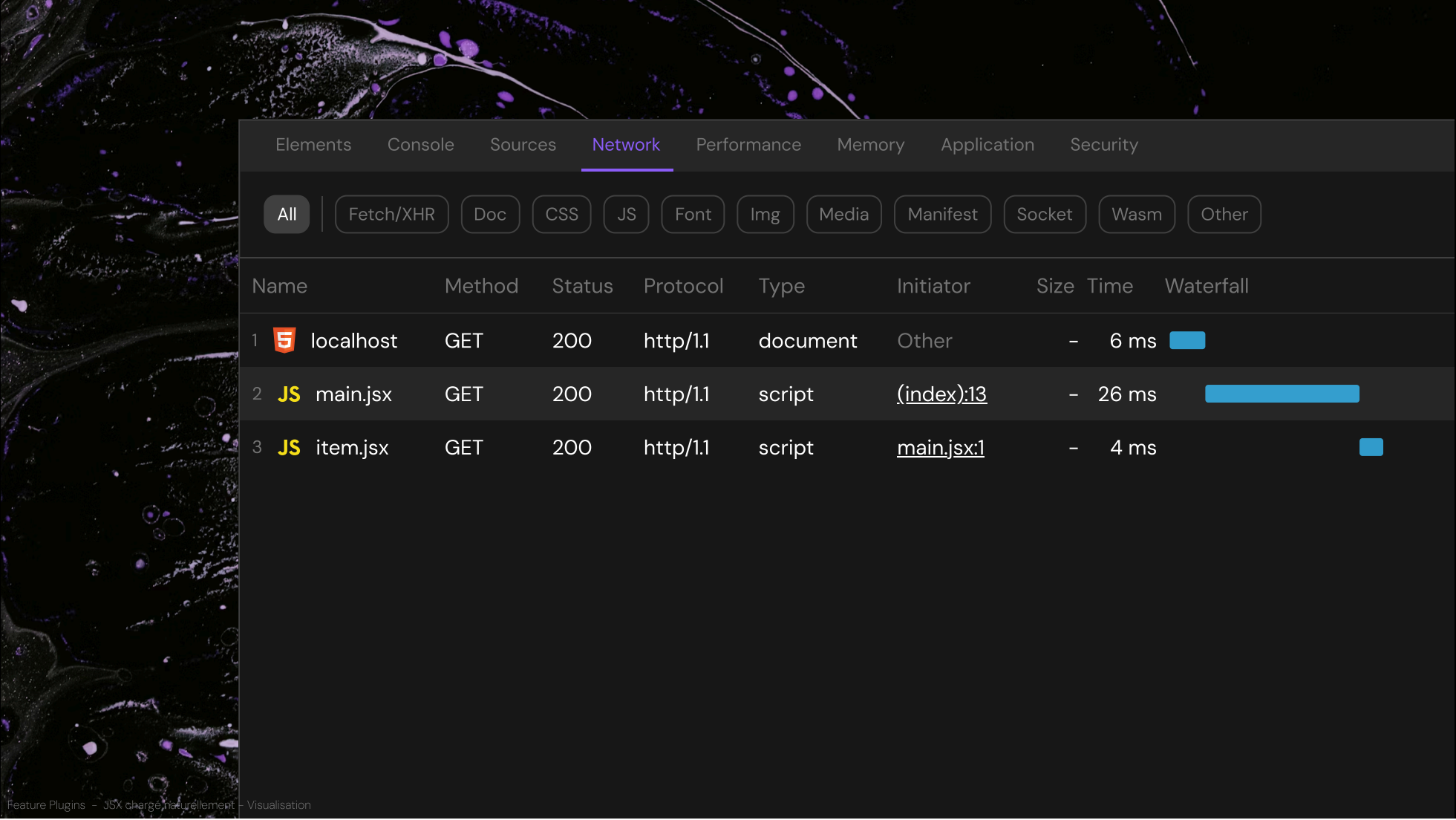

Vite Build Output
dist
index.html
assets
image-DNpSpoYj.jpg
JS
index-CgILWvq0.js

```
import { createRoot } fr
import { Item } from './I

function App() {
  return (
    <div>
      <h1>My App</h1>
      <Item name="Item 1"
      <Item name="Item 2"
    </div>
  )
}

const root = createRoot(
root.render(<App />)
```





Feature Plugins – JSX charge nécurellement – Visualisation

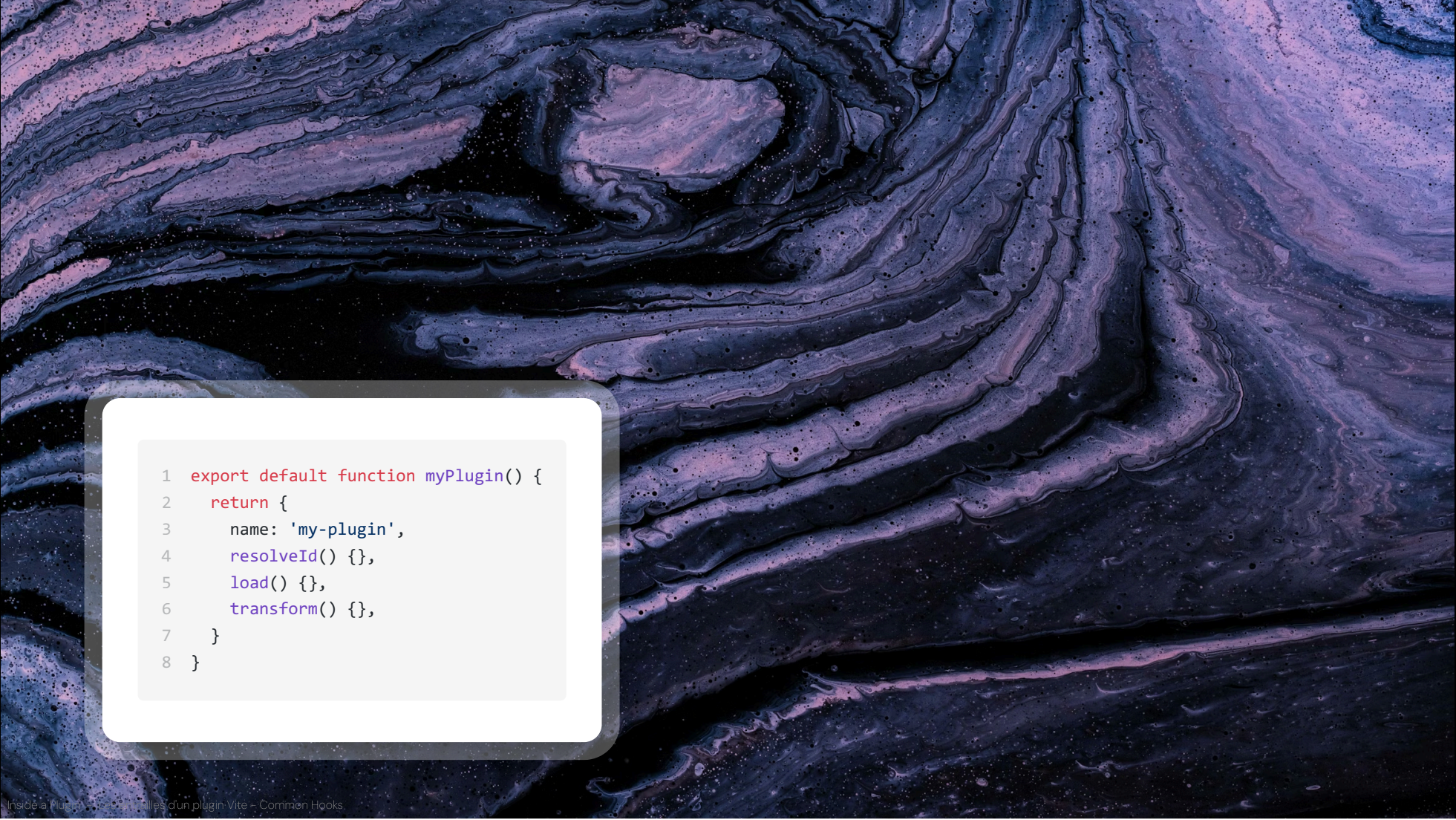
Vite en 3 points



1. **Un serveur web pour le développement**
   Il fait transiter des requêtes qu'on transformera à la volée

2. **Un bundler pour la production**
   Il transforme notre code pour la production

3. **Un système de plugins**
   Pour étendre ses fonctionnalités et faire faire tout ce qu'on veut


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)
}
}
},
}

Plusieurs plugins ?

request

resolveId `resolveId(id)`

VuePlugin

CustomPlugin

load `load(id)`

VuePlugin

CustomPlugin

transform `transform(code, id)`

VuePlugin

CustomPlugin

response

input

output

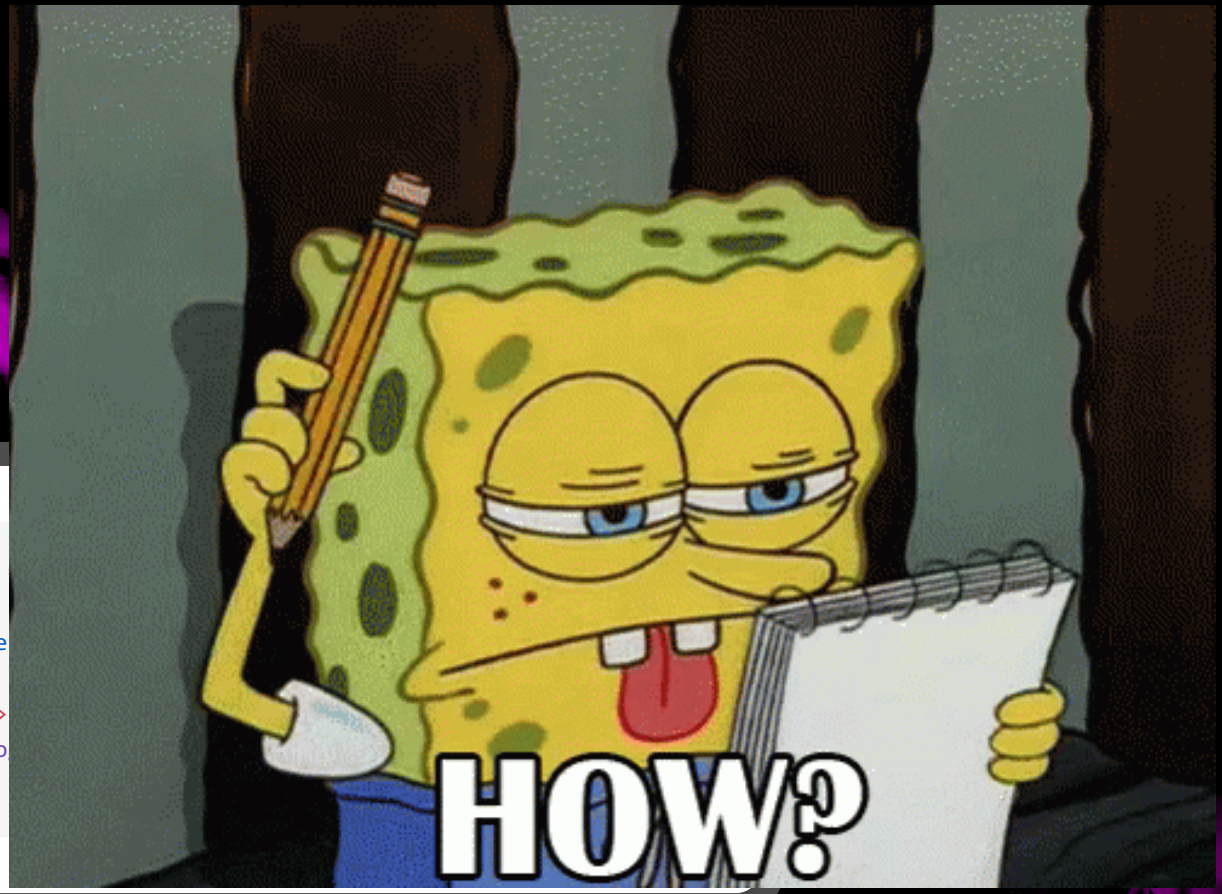
```
export default function myPlugin() {
  let config = null
  return {
    name: 'my-plugin',
    config(config) {
      config.root = 'src'
    },
    configResolved(resolvedConfig) {
      config = resolvedConfig
    },
    async buildStart() {
      await scanDir(config.root)
    },
    async buildEnd()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


Concrètement,
ça donne quoi ?

```
const count

const double

effect(() =>
  console.lo
})
```




MAGIC

HTTP Logs

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



```ts
export default function autoImport(options: Options) {
  let ctx = createContext(options)
  return {
    name: 'unplugin-auto-import',
    async transform(code, id) {
      const s = new MagicString(code)

      await injectImports(s, id)

      return {
        code: s.toString(),
        map: s.generateMap({ source: id }),
      }
    },
  }
}
```

vite-plugin-inspect

Visualiser le pipeline

Le plugin Vite en 3 points



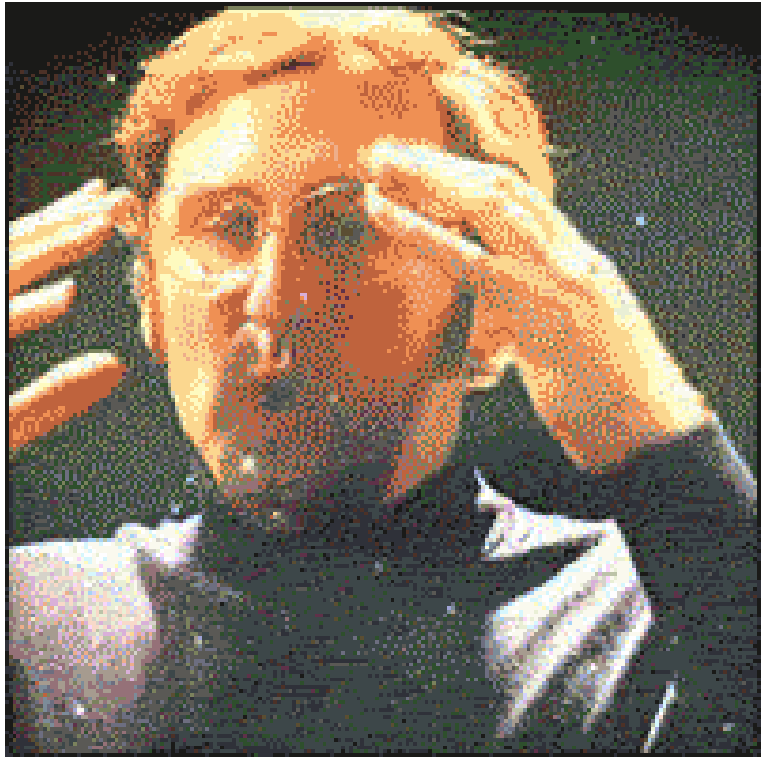
1. **Un plugin Vite, c'est une fonction qui retourne un objet**
   L'objet doit au moins avoir une propriété name

2. **Il y a 3 hooks principaux pour agir sur les modules**
   resolveId, load et transform

3. **Il y a aussi des hooks pour le cycle de vie de Vite**
   config, configResolved, buildStart, buildEnd et closeBundle

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 - Un module virtuel

0 / 3 · 0%

0
VOTES

VitePress - Des data virtuels

0 / 3 · 0%

0
VOTES

Icons Plugin - Des icônes virtuelles

3 / 3 · 100%

3
VOTES

Infos Plugin - Des infos virtuelles

0 / 3 · 0%

0
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 9c9e79dc3fd1f38df0a3e5756e2ef0eedac2b310 at Tue May 19 2026 19:03:22 GMT+0200 (Central European Summer Time)

Virtualization - Infos Plugin - Des infos virtuelles

```ts
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 - Infos Plugin - Des infos virtuelles

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.

Advanced Capabilities - Les autres capacités des plugins

Le Hot Module Replacement (HMR)

1 / 5 · 20%

1
VOTES

Le middleware - Un fichier virtuel

1 / 5 · 20%

1
VOTES

Run Plugin - Un plugin pour exécuter des commandes

1 / 5 · 20%

1
VOTES

Laravel Vite - La communication inter-processus

0 / 5 · 0%

0
VOTES

unplugin-macro - Un plugin pour créer des macros

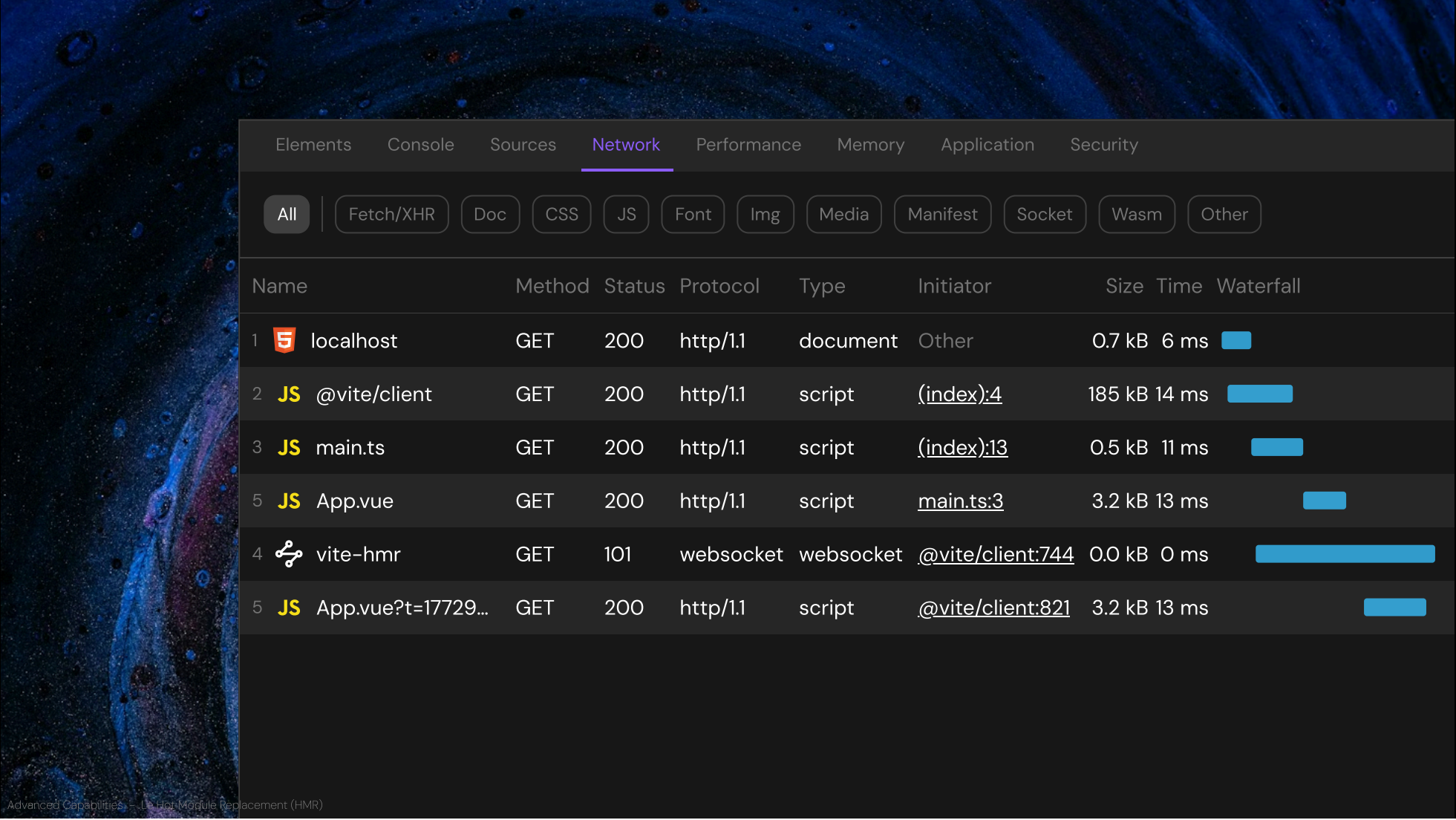
0 / 5 · 0%

0
VOTES

Nitro - Un backend

2 / 5 · 40%

2
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
Advanced Capabilities - ...e Hot Module Replacement (HMR)

```ts
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'

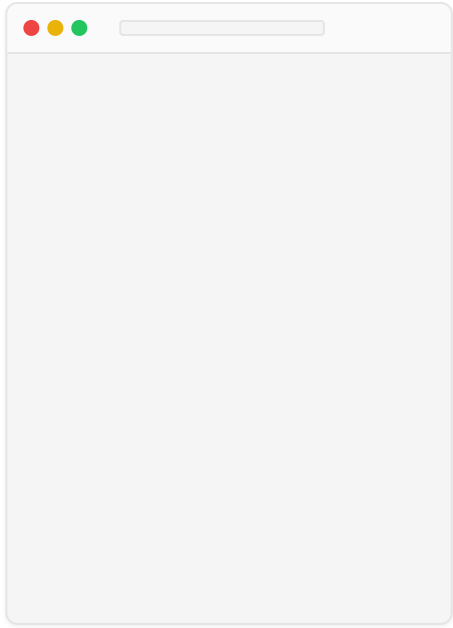
{
  name: 'vite-plugin-inspect',
  configureServer(server) {
    server.middlewares.use(`${base}/__inspect`, sirv(DIR_CLIENT, {
      single: true,
      dev: true,
    }))
  }
}
```

```ts
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$/, /api.php$/],
      }
    ]),
  ],
})
```

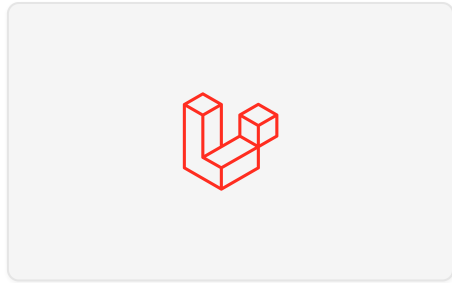
```
import { execSync } from 'node:child_process'
import { defineConfig } from 'vite'

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



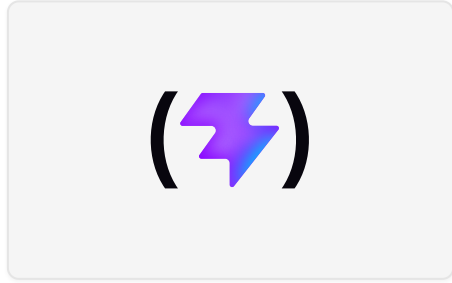
Ask Page →

← 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
```

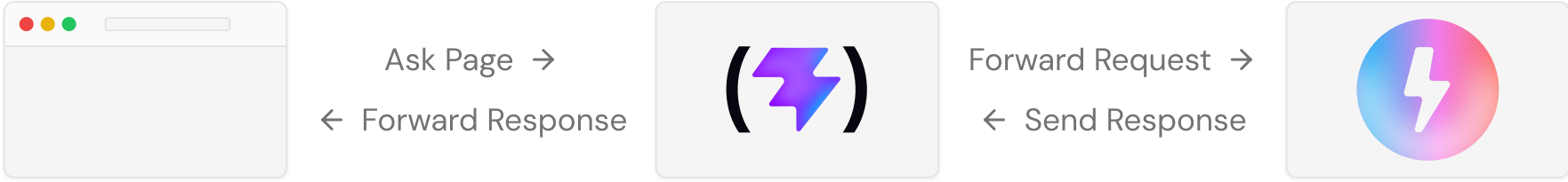
```ts
import { defineConfig } from 'vite'

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()
          }
        })
      }
    }
  })()]
})
```

Ce qu'il faut garder en tête



1

Une pipeline entre vos fichiers et le navigateur

Chaque requête passe par Vite, qui peut transformer les modules à la volée

2

Des plugins pour créer des modules qui n'existent pas

Modules virtuels, routes, icônes, infos de build... tout peut être généré à la volée

3

Tout peut être étendu par un plugin

Transformation de code, HMR, middlewares, side-effects, macros, backend...



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

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

En 45 minutes, vous avez découvert le nécessaire pour réaliser votre plugin Vite.





C'était

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

Pour **L'Atelier** (09 avril 2026)

Présenté par  Estéban Soubiran

⬇️ Votre feedback ⬇️

example.com