

Oct. 17th, 2021

New Ways to Vue

新技術和工具如何影響我們編寫 Vue 專案的方式

ANTHONY FU

Anthony Fu

Vue & Vite 核心團隊成員

Slidev, VueUse, Vitesse, Type Challenges 等項目作者

狂熱的開源愛好者，目前任職於 NuxtLabs。



antfu



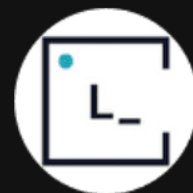
antfu7



antfu.me



Gold Sponsors



Leniolabs_



Vue Mastery



Evan You



Gitpod



Steven Yung

Sponsors



IU



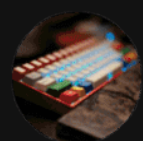
hiroki osameHunter



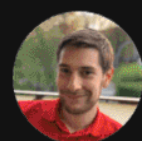
Liu



Ben Hong



PENG Rui



Jan-Henrik



Eric Otto



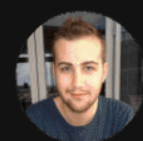
FallDownTh...



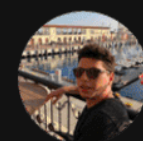
Johann



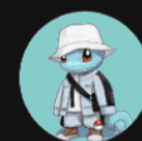
ILPT GmbH



schalk-b



Ivan Que

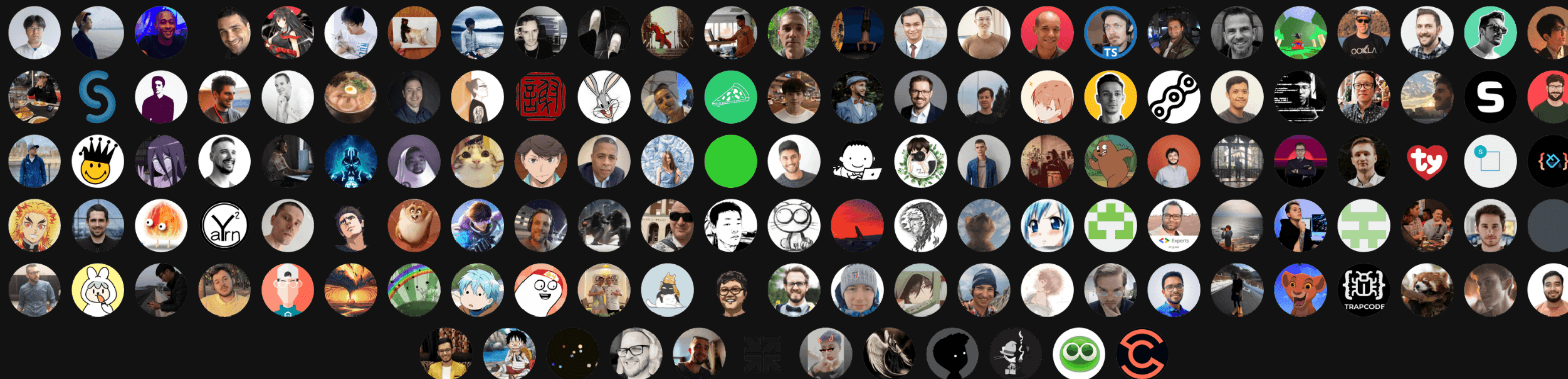


ygj6



Ivan Demchuk

Backers



在 GitHub 上赞助我

New Ways to Vue 

Vue 2 中的編寫方式

```
<script>
import Vue from 'vue'
import Foo from './components/Foo.vue'
import { mixinBar } from './mixins/bar'

export default Vue.extend({
  components: {
    Foo,
    // ...
  },
  mixins: {
    mixinBar,
    // ...
  },
  props: {
    value: {
      type: String
    },
    // ...
  },
  data() {
    return {
      // ...
    }
  }
})
```

存在的問題

- "腳手架程式碼"
- 可擴展性
- TypeScript 支援

組合式 API

物件式 API

```
export default {
  data() {
    return {
      dark: false,
      media: matchMedia('(prefers-color-scheme: dark)')
    }
  },
  methods: {
    toggleDark() { this.dark = !this.dark },
    update() { this.dark = this.media.matches }
  },
  created() {
    this.media.addEventListener('change', this.update)
    this.update()
  },
  destroyed() {
    this.media.removeEventListener('change', this.update)
  }
}
```

組合式 API

```
import { ref, onUnmounted } from 'vue'
export default {
  setup() {
    const media = matchMedia('(prefers-color-scheme: dark)')
    const dark = ref(media.matches)

    const update = () => dark.value = media.matches
    const toggleDark = () => dark.value = !dark.value

    media.addEventListener('change', update)
    onUnmounted(() => {
      media.removeEventListener('change', update)
    })

    return { dark, toggleDark }
  }
}
```


可組合性

```

<!-- DarkToggle.vue -->
<script>
import { useDark } from './useDark'

export default {
  setup() {
    return {
      ...useDark()
    }
  }
}
</script>

```

```

// useDark.js
import { ref, onUnmounted } from 'vue'

export function useDark() {
  const media = matchMedia('(prefers-color-scheme: dark)')
  const dark = ref(media.matches)

  const update = () => dark.value = media.matches
  const toggleDark = () => dark.value = !dark.value

  media.addEventListener('change', update)
  onUnmounted(() => {
    media.removeEventListener('change', update)
  })
  return { dark, toggleDark }
}

```

`<script setup>` 語法

`<script>`

```
<script>
import { ref, computed } from 'vue'
import MyButton from './MyButton.vue'
export default {
  components: {
    MyButton,
  },
  setup() {
    const counter = ref(0)
    const doubled = computed(() => counter.value * 2)

    function inc() {
      counter.value += 1
    }

    return { counter, doubled, inc }
  }
}</script>
```

`<script setup>`

```
<script setup>
import { ref, computed } from 'vue'
import MyButton from './MyButton.vue'

const counter = ref(0)
const doubled = computed(() => counter.value * 2)

function inc() {
  counter.value += 1
}
</script>
```

- 頂層變數和函式宣告
- 變數和組件將直接可以在模板中使用
- Vue 3.2 中已穩定

`<style>` 中的 `v-bind()` 語法

WITHOUT

```
<template>
  <button :style="{ color: buttonColor }">
    My Button
  </button>
</template>

<script>
export default {
  data() {
    return {
      buttonColor: 'green'
    }
  }
}
</script>

<style>
button {
  border-radius: 4px;
}
</style>
```

WITH V-BIND()

```
<template>
  <button>My Button</button>
</template>

<script setup>
const buttonColor = ref('green')
</script>

<style>
button {
  border-radius: 4px;
  color: v-bind(buttonColor);
}
</style>
```

全新的默認開發工具 Vite



Vite 是什麼？

打包器 Bundlers



Webpack

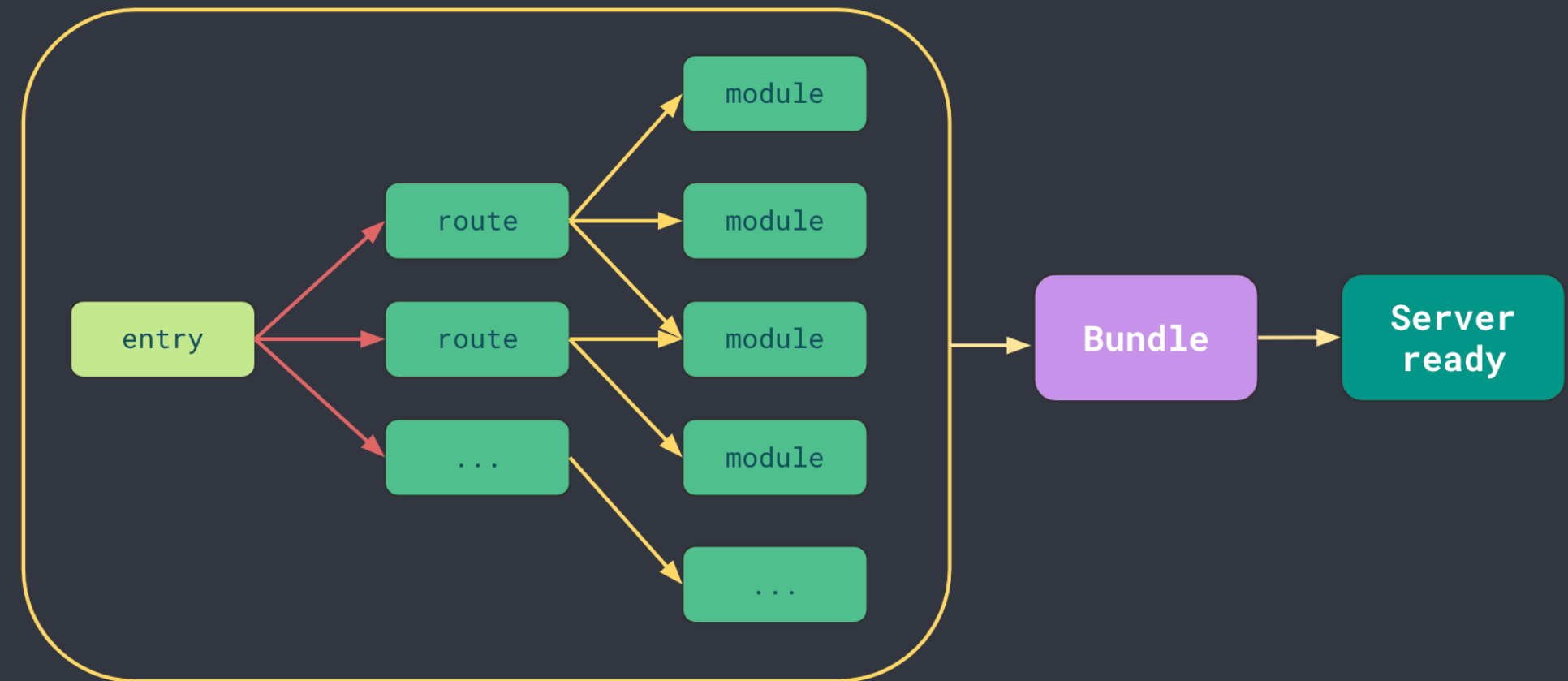


Rollup

構建優先 BUILD FIRST

- 主要為生產環境構建設計
- 需要先打包整個專案才能讓開發伺服器啟動
- 需要複雜的配置
- 模塊熱更新 (HMR) 隨著項目變大而顯著變慢

Bundle based dev server



開發用伺服器 Dev Server



Snowpack

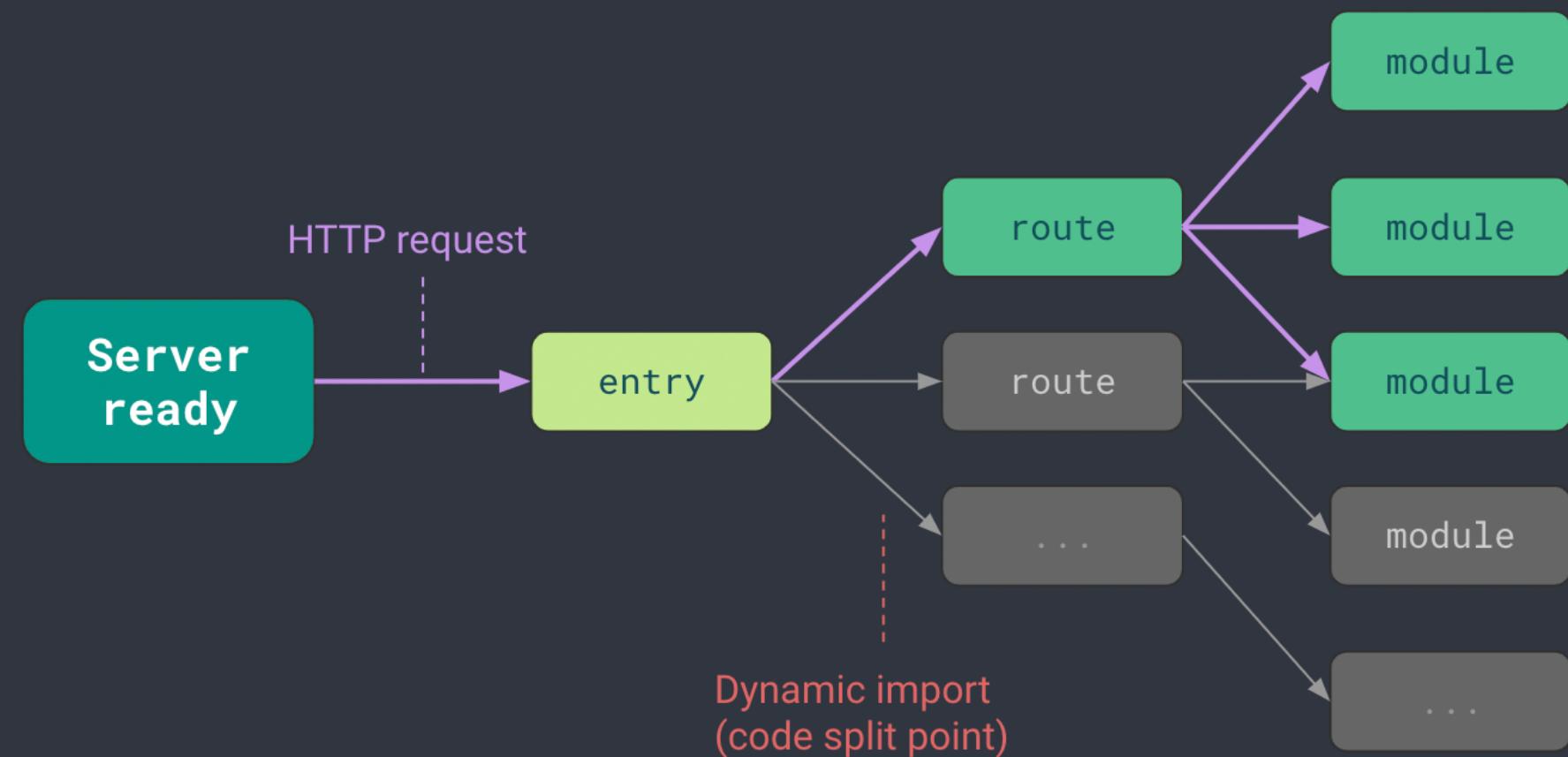


Vite

開發優先 DEV FIRST

- 專為前端開發設計
- 不打包 + 使用原生 ESM
- 按需編譯
- 伺服器瞬間準備完成
- 瞬間的熱更新 HMR
- ...更多新功能

Native ESM based dev server



Vue 3 和 Vite 給我們帶來了什麼？

更快的速度和更好的開發體驗

New Ways to View 🙄🙄

使用組件

```
<template>
  <my-container>
    <my-button />
    <my-input />
  </my-container>
</template>

<script>
import MyContainer from '../components/MyContainer.vue'
import MyButton from '../components/MyButton.vue'
import MyInput from '../components/MyInput.vue'

export default {
  components: {
    MyContainer,
    MyButton,
    MyInput,
  }
}
</script>
```

要使用一個組件

- 使用 `import` 導入，並取一個名字
- 註冊組件
- 在模板中使用它

問題

- 冗余
- 組件的名字至少被重複了四次

使用組件

```
<template>
  <my-container>
    <my-button />
    <my-input />
  </my-container>
</template>

<script setup>
import MyContainer from '../components/MyContainer.vue'
import MyButton from '../components/MyButton.vue'
import MyInput from '../components/MyInput.vue'
</script>
```


使用 `<SCRIPT SETUP>` 語法


- 導入可以直接在模板中使用
- 不再需要註冊組件

但是...

- 組件的名字還是被重複了三次

組件自動導入

 antfu/vite-plugin-components

使用  vite-plugin-components

```
<template>
  <my-container>
    <my-button />
    <my-input />
  </my-container>
</template>
```

收工！

怎麼做到的？

- **編譯期** 組件解析
- 在 `src/components` 資料夾下的組件將被自動導入

不同於全局組件註冊

- Code-splitting
- 不再需要手動註冊組件
- 跳過了運行時組件解析

編譯期解析

```
<template>
  <my-container>
    <my-button />
    <my-input />
  </my-container>
</template>
```

將會被 `@vue/sfc-compiler` 編譯成 (可以通過 <https://sfc.vuejs.org> 查看)

```
import { resolveComponent as _resolveComponent } from "vue"
function render(_ctx, _cache) {
  const _component_my_button = _resolveComponent("my-button")
  const _component_my_input = _resolveComponent("my-input")
  const _component_my_container = _resolveComponent("my-container")

  return (_openBlock(), _createBlock(_component_my_container, null, {
    default: _withCtx(() => [
      _createVNode(_component_my_button),
      _createVNode(_component_my_input)
    ]), _: 1 /* STABLE */
  })))
}
```

編寫一個 Vite 插件

```
// vite.config.ts
export default {
  plugins: [{
    name: 'my-plugin',
    enforce: 'post',
    transform(code, id) {
      if (!id.endsWith('.vue'))
        return

      return code.replace(
        /_resolveComponent\("(.\+?)"/g,
        (_, name) => {
          const component = findComponent(name)
          // 注入導入程式碼 (略)
          return component.path
        })
    }
  }]
}
```

- 使用 `enforce: post` 確保在 Vue 編譯後執行
- 使用 `transform` 鉤子進行程式碼變換
- 過濾掉不是 Vue 的檔案
- 將 `_resolveComponent` 替換成直接導入的組件

請參照 [Vite Plugin API 官方文件](#)

最終結果

```
import { resolveComponent as _resolveComponent } from "vue"

function render(_ctx, _cache) {
  const _component_my_button = _resolveComponent("my-button")
  const _component_my_input = _resolveComponent("my-input")
  const _component_my_container = _resolveComponent("my-container")


  return () => /* ... */
}
```

替換後產物

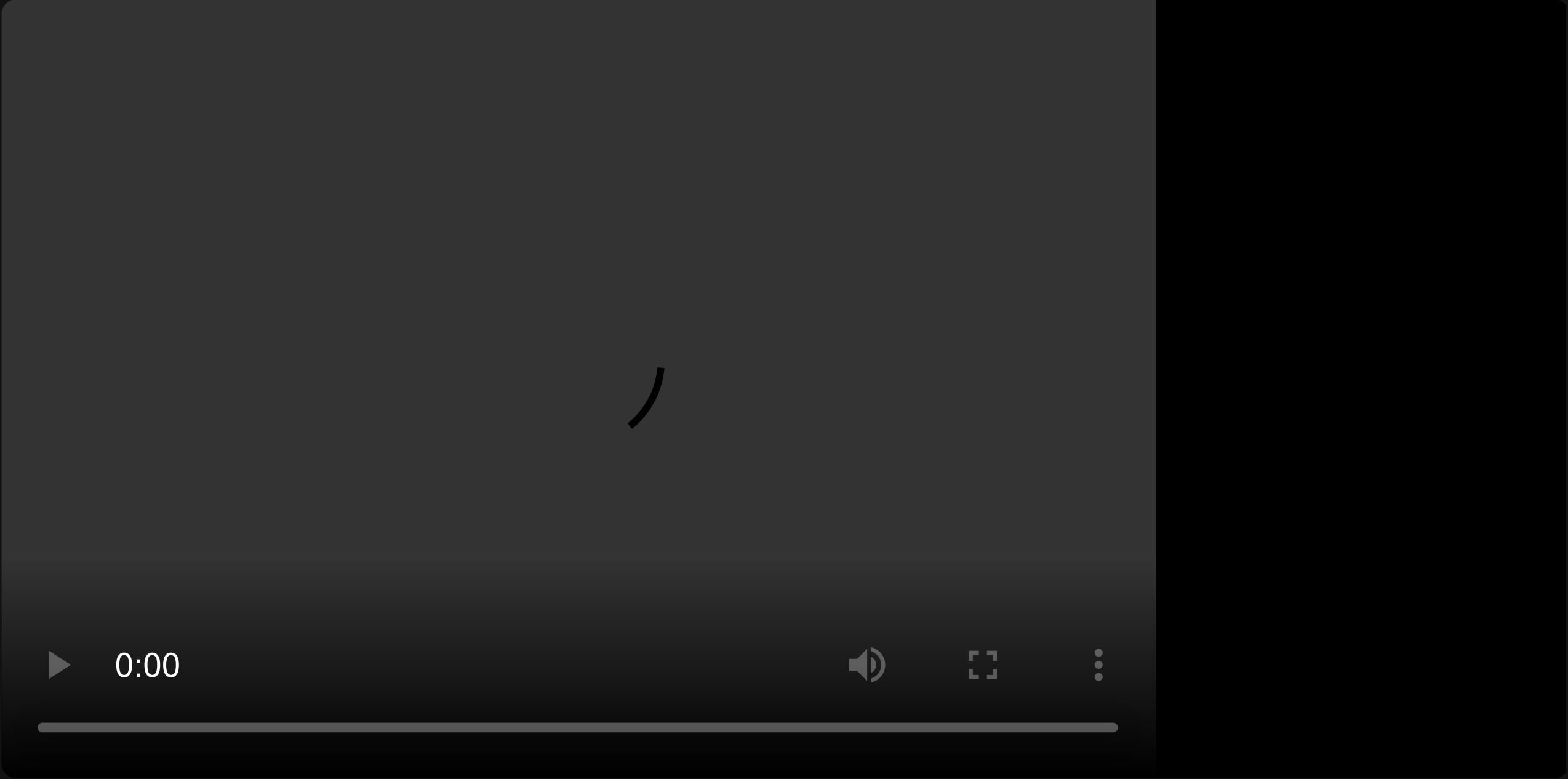
```
import { resolveComponent as _resolveComponent } from "vue"
import _component_my_button from "../components/MyButton.vue"
import _component_my_input from "../components/MyInput.vue"
import _component_my_container from "../components/MyContainer.vue"

function render(_ctx, _cache) {
  return () => /* ... */
}
```

檢閱模組樹 Inspect Module Graph

 antfu/vite-plugin-inspect

獲悉每次變換的中間狀態



API 自動導入

 antfu/unplugin-auto-import

類似的，我們也可以做到 API 自動導入

```
<script setup>
import { ref, computed, watch } from 'vue'
import { debouncedWatch } from '@vueuse/core'

const counter = ref(0)
const doubled = computed(() => counter.value * 2)

debouncedWatch(counter, () => {
  console.log('counter changed')
})
</script>
```

```
<script setup>
const counter = ref(0)
const doubled = computed(() => counter.value * 2)

debouncedWatch(counter, () => {
  console.log('counter changed')
})
</script>
```

使用圖標 Icons

以往的方式

- Icon Fonts
 - 需要打包整圖標集
 - FOUC (無樣式的閃爍)
- SVG
 - 需要手動引入
 - 無法根據上下文上色
- 圖標組件
 - 依賴於圖標集的實現
 - 需要手動註冊

更好的解決方案?

受啟發於 Vite 的“按需分配”理念，我們可以讓圖標在編譯期按需分配

```
<script setup>
import MdiAlarm from '~icons/mdi/alarm'
import FaBeer from '~icons/fa/beer'
import TearsOfJoy from '~icons/twemoji/face-with-tears-of-joy'
</script>

<template>
  <MdiAlarm />
  <FaBeer style="color: orange" />
  <TearsOfJoy/>
</template>
```

透過 Iconify，我們可以讓 100+ 圖標集的 10,000+ 圖標使用相同的語法導入

```
import Icon from '~icons/[collection]/[id]'
```

插件將會按需將虛擬模組編譯成對應的圖標組件

圖標按需導入

- 按需引入，只有用到的圖標才會被打包
- 可以使用幾乎任何圖標
- 可直接使用 `class` 和 `style` 修改樣式
- SSR / SSG 友好
- 配合 `vite-plugin-components`，獲得類似 Icons Fonts 的體驗

```
<template>
  <MdiAlarm />
  <FaBeer style="color: orange" />
  <TearsOfJoy />
</template>
```


Vite 生態系統

-  vite-plugin-components - 組件自動引入
-  vite-plugin-auto-import - API 自動引入
-  vite-plugin-icons - 按需分配的圖標方案
-  vite-plugin-inspect - 檢視 Vite 內部狀態
-  hannoeru/vite-plugin-pages - 檔案系統路由生成器
-  vite-plugin-windicss - Windi CSS (按需生成的 Tailwind CSS)
-  axe-me/vite-plugin-node - 為後端 Node 應用提供 Vite HMR
-  anncwb/vite-plugin-style-import - 按需引入組件樣式

...以及更多



awesome

 vitejs/awesome-vite

One more thing

Vite 的設計思想激發了很多新的工具和開發體驗

把他們帶到你現有的專案中
今天就行！

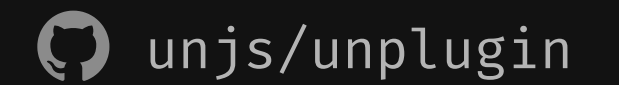
Introducing unplugin

通用化的插件 API，可用於 Webpack, Vite, Rollup 等等...

編寫一次但適用於：



Unplugin



VITE 插件

```
export const VitePlugin = () => {
  return {
    name: 'my-first-unplugin',
    transform (code) {
      return code.replace(
        /<template>/,
        `<template><div>Injected</div>`
      )
    },
  }
}
```

UNPLUGIN

```
import { createUnplugin } from 'unplugin'

export const unplugin = createUnplugin(() => {
  return {
    name: 'my-first-unplugin',
    transform (code) {
      return code.replace(
        /<template>/,
        `<template><div>Injected</div>`
      )
    },
  }
})

export const VitePlugin = unplugin.vite
export const RollupPlugin = unplugin.rollup
export const WebpackPlugin = unplugin.webpack
```








Vite 插件 → Unplugins

``vite-plugin-components`` → ``unplugin-vue-components``

``vite-plugin-auto-import`` → ``unplugin-auto-import``

For  Vue /  React /  Svelte /  JS Vanilla / Any framework








``vite-plugin-icons`` → ``unplugin-icons``

 Vue
 React
 Preact
 Svelte
 SolidJS
 Web Components
 JS Vanilla
...

+

 Vite
 Nuxt
 Next.js
 Rollup
 Vue CLI
 Webpack
...

+

 Carbon Icons
 Material Design Icons
 Unicons
 Twemoji
 Tabler
 Boxicons
 EOS Icons
...

那 Vue 2 如何?

沒問題，替你想好了！

Vue 2

POLYFILLS

- 組合式 API: `@vue/composition-api`
- `<script setup>` 語法和 Ref 語法糖: `unplugin-vue2-script-setup`

VITE 支援

- `vite-plugin-vue2`
- `nuxt-vite`

開發體驗

- `unplugin-vue-components`
- `unplugin-auto-import`
- `unplugin-icons`

打包帶走

今天就能獲得的開發體驗，不管你是用 Vue 2, Vue 3, Vite, Nuxt, 還是 Vue CLI 都可以！

```
<template>
  <button>
    <IconSun v-if="dark" />
    <IconMoon v-else />
  </button>
</template>

<script>
import IconSun from '@some-icon-set/sun'
import IconMoon from '@some-icon-set/moon'

export default {
  components: {
    IconSun,
    IconMoon,
  },
  data() {
    return {
      dark: false,
      media: matchMedia('(prefers-color-scheme: dark)')
    }
  },
  methods: {
```



```
<script setup>
const dark = useDark()
</script>


<template>
  <button>
    <IconSun v-if="dark" />
    <IconMoon v-else />
  </button>
</template>
```

起手專案模板

預置了上面提到的所有插件和配置

 [antfu/vitesse](#) Vue 3 + Vite 專案模板

 [antfu/vitesse-nuxt](#) 在 Nuxt 2 上獲得 Vitesse 開發體驗

 [antfu/vitesse-webext](#) Vitesse 瀏覽器插件版本

現在嘗試！

```
npx degit antfu/vitesse
```

Spoiler: Nuxt 3 將會原生提供許多上面提到的功能！

謝謝！

簡報可以在 antfu.me 下載