New Ways to Vue

How the new tools and techniques affect
the way we **view** and build applications

ANTHONY FU

Vue.js London
Oct. 20th, 2021

Anthony Fu

Vue & Vite core team member.

Creator of Slidev, VueUse, Vitesse, Type Challenges, etc.

Fanatical open sourceror. Working at NuxtLabs.

antfu

antfu7

antfu.me

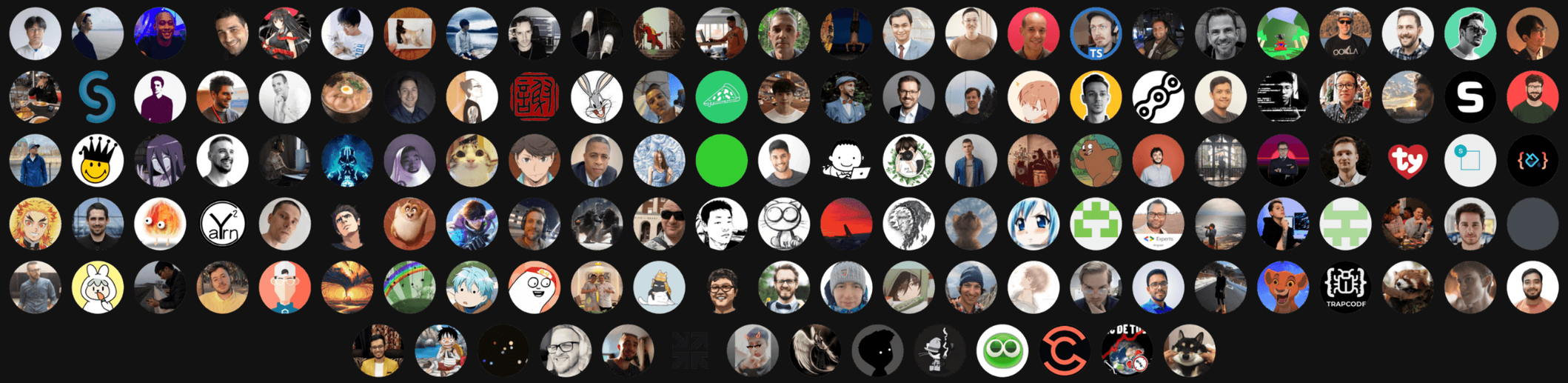
Gold Sponsors

Leniolabs_ Vue Mastery Evan You Gitpod Steven Yung

Sponsors

IU hiroki osame Hunter Liu Ben Hong PENG Rui Jan-Henrik Eric Otto FallDownTh... Johann ILPT GmbH schalk-b Ivan Que ygj6 Ivan Demchuk

Backers



Sponsor me at GitHub

New Ways to Vue

The Vue 2 Ways

```
<template>
  <!--  -->
</template>

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

export default Vue.extend({
  components: {
    Foo,
    // ...
  },
  mixins: {
    mixinBar,
    // ...
  },
  data() {
    return {
      // ...
    }
  },
  methods: {
    //
```

THE PROBLEM

- "Scaffolding code" for each component
- Extensibility
- TypeScript support

Composition API

OPTIONS 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)
  }
}
```

COMPOSITION 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 }
  }
}
```

Composability

```
import { useDark } from './useDark'

export default {
  setup() {
    return {
      ... useDark()
    }
  }
}
```

```
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>` syntax

`<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>
```

- Variables, functions, and components are directly available in the template
- Now stable in Vue 3.2

`v-bind()` in `<style>`

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>
```

The New Default Tooling - Vite

What’s Vite?

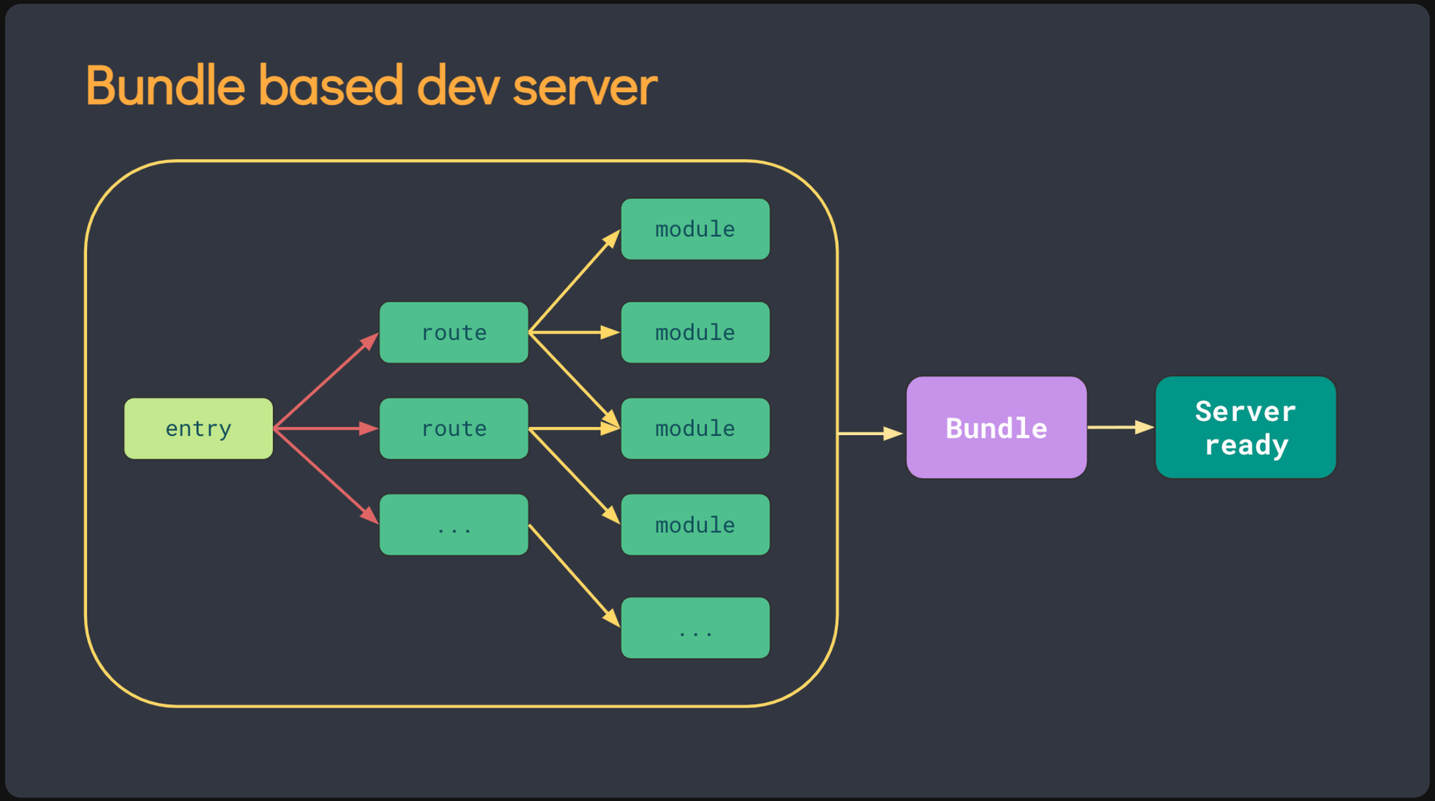
Bundlers


Webpack


Rollup

BUILD FIRST

- Designed for production build first
- Need to bundle the entire project to start the dev server
- Complex configuration
- HMR gets slower as projects grow

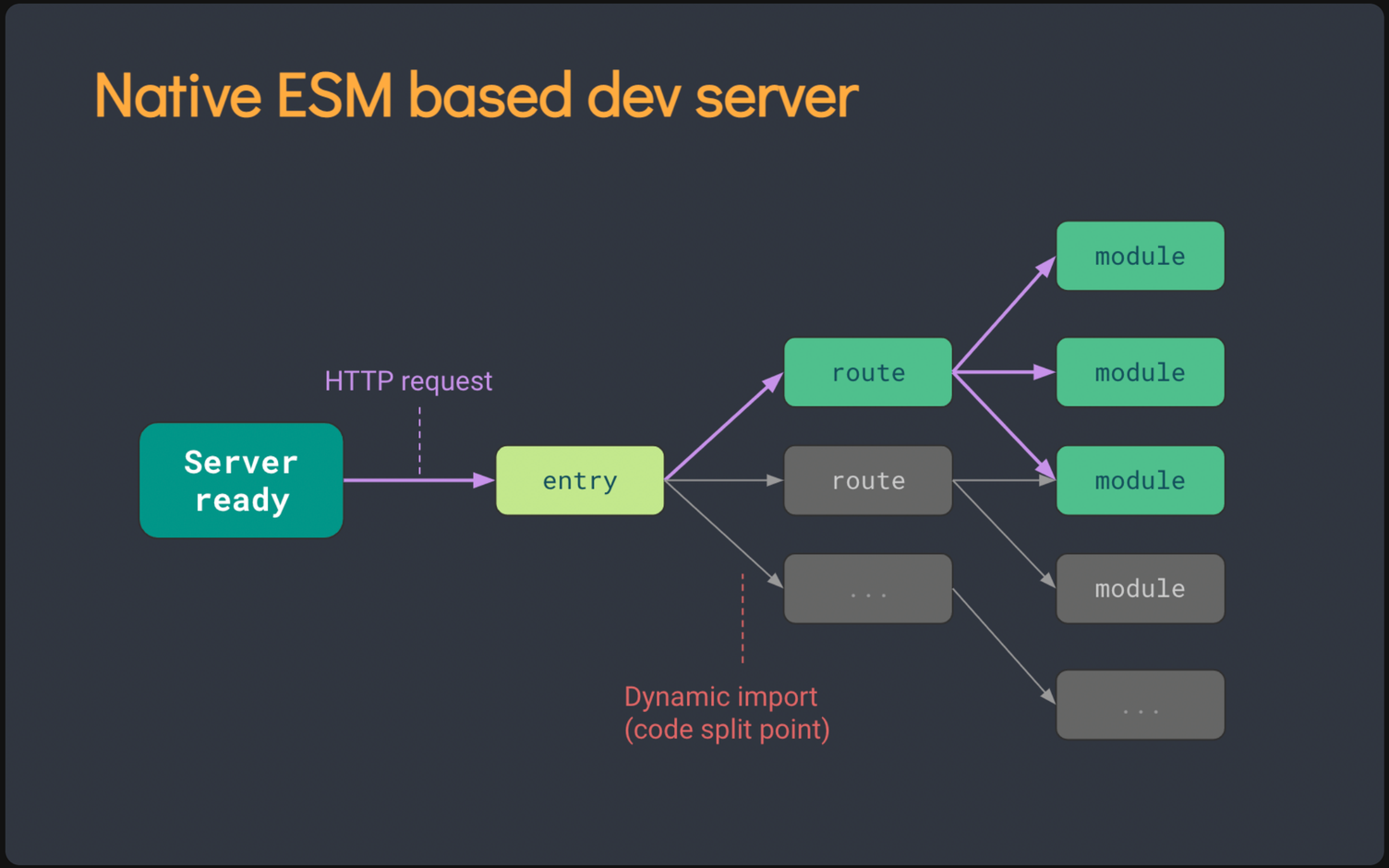


Dev Servers

Snowpack Vite

DEV FIRST

- Design for Web development
- Native ESM + Unbundled
- Server ready immediately
- On-demand
- Instant HMR
- ...much more!



What do Vue 3 and Vite bring to us?

Better performance and better DX

New Ways to View 👀

Using Components

```vue
<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>
```

TO USE A COMPONENT

- Import and name it
- Register the component
- Use it in the template

THE PROBLEM

- Verbose
- Names are repeated at least 4 times

Using Components

```
<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>
```

WITH `<SCRIPT SETUP>`

- Imports will be available directly in the template
- No longer need to register the components

BUT...

- The name is still repeated 3 times

Components Auto Importing

antfu/vite-plugin-components

Using vite-plugin-components

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

That's it!

HOW?

- **Compile-time** components resolving
- Components auto-discovery under `src/components` directory

DIFFERENCES FROM GLOBAL REGISTRATION

- Code-splitting
- No manual registration
- Skipped runtime resolving

How the compilation work

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

Will be compiled by `@vue/sfc-compiler` to (Could inspect via https://sfc.vuejs.org)

```
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 */
  }))
}
```

Write the Vite plugin

```
// 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)
          // inject import for component
          return component.path
        })
    }
  }]
}
```

- Use `enforce: post` to ensure the plugin runs after Vue's compilation
- Use `transform` hook to modify the code
- Filter out files that are not Vue
- Replace the `_resolveComponent` usage to real component import

Read Vite Plugin API Documentation for more

The Result

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

After:

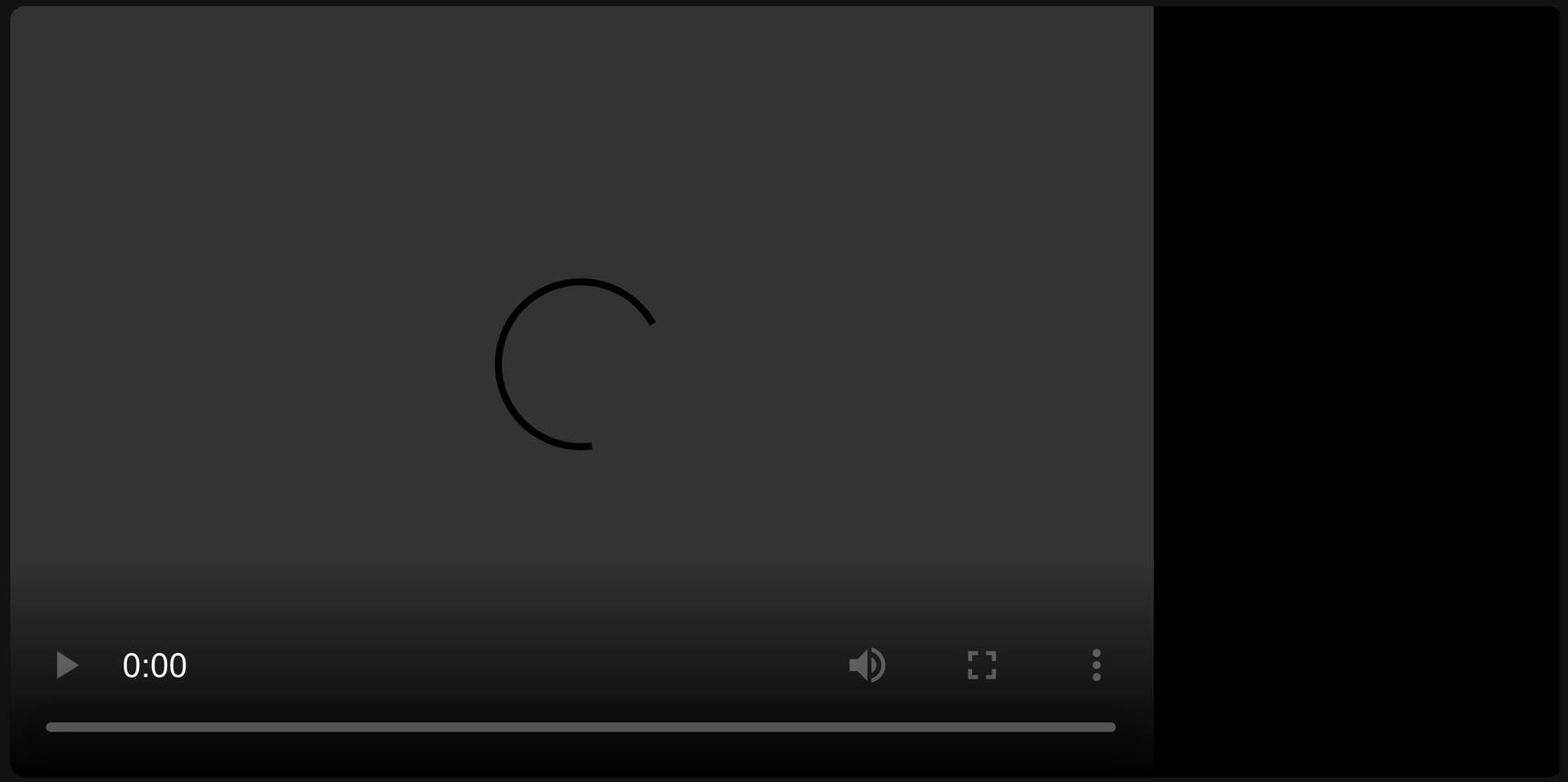
```
import { resolveComponent as _resolveComponent } from "vue"
import _component_my_button from "../components/MyButton.vue"
import _component_my_input from "../components/MyInput.vue"
import _component_my_container from "../components/MyContainer.vue"

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

Inspecting Module Graph

antfu/vite-plugin-inspect

Intermediate state of each transformation



API Auto Importing

antfu/unplugin-auto-import

Similarly, we could do auto importing for APIs.

```
<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>
```

Vite Ecosystem

- `vite-plugin-components` - Components auto-import
- `vite-plugin-auto-import` - API auto-import
- `vite-plugin-icons` - On-demanded icons solution
- `vite-plugin-inspect` - Inspect intermedia state of Vite
- `hannoeru/vite-plugin-pages` - File-based routing
- `vite-plugin-windicss` - Windi CSS (On-demand Tailwind CSS)
- `axe-me/vite-plugin-node` - Vite HMR for backend Node.js app
- `anncwb/vite-plugin-style-import` - On-demand components style importing

...and more

awesome

`vitejs/awesome-vite`

Vite has inspired many new plugins and better ways to improve DX

Bring them to Your Existing Projects

Today!

Introducing **unplugin**

A universal plugin interface for Webpack, Vite, Rollup, and more...

write once and runs on:

Unplugin

unjs/unplugin

VITE PLUGIN

```
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 Plugins → Unplugins

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

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

For Vue / React / Svelte / Vanila / Any framework

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

- Vue
- React
- Preact
- Svelte
- SolidJS
- Web Components
- Vanila
- ...

+

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

+

- Carbon Icons
- Material Design Icons
- Unicons
- Twemoji
- Tabler
- BoxIcons
- EOS Icons
- ...

What about Vue 2?

We got you covered!

Vue 2

POLYFILLS

- Composition API: `@vue/composition-api`
- `<script setup>` & Ref sugar: `unplugin-vue2-script-setup`

VITE SUPPORT

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

DX ENHANCEMENT

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

Sum Up

This is what you could get in Vue 2, Nuxt 2, Vue CLI, Vue 3, Vite:

```vue
<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: {
```

→

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

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

Starter Templates

Project templates that have plugins mentioned previously

`antfu/vitesse`	Opinionated Vue 3 + Vite Starter template
`antfu/vitesse-nuxt`	Vitesse experience on Nuxt 2
`antfu/vitesse-webext`	Vitesse for Web Extensions

TRY IT NOW!

```
npx degit antfu/vitesse
```

Spoiler: Nuxt 3 will have many of these features built-in directly.

Thank You!

Slides can be found on antfu.me