

# Vue开发体验 with Vue

ELONE HOO

Work Talks  
Mar. 18th 2023

# Elone Hoo

JumpIDE, VueHooksForm, Pistachio, LogonTracer 等开源项目作者

任职于 Beneway



🏗 WIP of Elone Hoo

🐱 elonehoo

🐦 elonehoo

👤 elonehoo.me

# 开发者体验

# 响应能力



Vite  
网页中的 HMR



vite-node  
Node 中的 HMR



Spring DevTool  
用于后端 API 重新加载

# 常见做法

- TypeScript / ESM
- SPA / SSR / Static / Hybrid
- 布局、插件、路由中间件...
- 组合实用程序 - `'useState'`、`'useAsyncData'`、`'useFetch'` ...
- Head/SEO - `'useHead'`、`'useSeoMeta'` ...
- 后端集成、serverless 等。

# 惯例

- 基于文件的路由
- 组件自动导入
- Composition API 自动导入
- 端到端类型

# 生态

- 模块
- 轻松整合

## 插件

### 注意

Vite 旨在为常见的 web 开发工作提供开箱即用的支持。在搜索一个 Vite 或 Rollup 兼容插件之前，请先查看 [功能指引](#)。很多场景下，在 Rollup 项目中需要添加插件，而在 Vite 中已经内建支持了。

请查看 [使用插件](#) 一章了解更多插件使用方式。

### 官方插件

#### @vitejs/plugin-vue

- 提供 Vue 3 单文件组件支持。

#### @vitejs/plugin-vue-jsx

- 提供 Vue 3 JSX 支持（通过 [专用的 Babel 转换插件](#)）。

#### @vitejs/plugin-vue2

- 提供对 Vue 2 的单文件组件支持。

#### @vitejs/plugin-react

- 使用 esbuild 和 Babel，使用一个微小体积的包脚注可以实现极速的 HMR，同时提升灵活性，能够使用 Babel 的转换管线。在构建时没有使用额外的 Babel 插件，只使用了 esbuild。

#### @vitejs/plugin-react-swc

- 在开发时会将 Babel 替换为 SWC。在构建时，若使用了插件则会使用 SWC+esbuild，若没有使用插件则仅会用到 esbuild。对不需要标准 React 扩展的大型项目，冷启动和模块热替换（HMR）将会有显著提升。

#### @vitejs/plugin-legacy

- 为打包后的文件提供传统浏览器兼容性支持

本目录

官方插件

社区插件

Rollup 插件

StackBlitz

storyblok

tailwind

replit

VueJobs

RiotJS

Prefect

Cloudinary

Pine View

Adobe Stock



Get 10 Free Images From  
Adobe Stock. Start Now.

ADS VIA CARBON

# Vue中的 - 响应式

- 自动收集依赖 & 更新
- Vue 3 中新的 API
  - ref
  - reactive
  - effect
  - computed

	A	
1		1
2		2
3	=SUM(A1:A2)	
4		

# 响应式 - Reactive

- 使用 Proxy 实现
- track, trigger 进行响应式追踪

```
const reactive = target => new Proxy(target, {
  get(target, prop, receiver) {
    track(target, prop)
    return Reflect.get(...arguments) // get original data
  },
  set(target, key, value, receiver) {
    trigger(target, key)
    return Reflect.set(...arguments)
  }
})

const obj = reactive({
  hello: 'world'
})

console.log(obj.hello) // `track()` get called
obj.hello = 'vue' // `trigger()` get called
```

# 响应式 - Effect

- track 追踪调用它的函数
- trigger 出发绑定的更新
- effect 调用函数并且触发收集依赖

```
const targetMap = new WeakMap()

export const track = (target, key) => {
  if (tacking && activeEffect)
    targetMap.get(target).key(key).push(activeEffect)
}

export const trigger = (target, key) => {
  targetMap.get(target).key(key).forEach(effect => effect())
}

export const effect = (fn) => {
  const effect = function () { fn() }
  enableTracking()
  activeEffect = effect
  fn()
  resetTracking()
  activeEffect = undefined
}
```

更进一步

介绍





git

- 多分支管理
- 多版本控制
- 最佳的 diff 展示

Demo 时间!

[查看 Demo](#)

这就是我们在个项目中是如何使用 Git 的

# Thank You!

Slides on [elonehoo.me](http://elonehoo.me)