Vue.js 技术分享

尤雨溪

2019.06.04

大纲

- Vue 团队和发展现状介绍
- 设计思路及与其它框架的比较
- 生态介绍和方案推荐
- --- 午休 ---
- 3.0 新特性、改动介绍
- Vue 的培训和学习路线 & QA

团队 / 发展现状

发展现状

- Chrome DevTools: 93.2万周活跃用户
  - 对比：React ~160万
- npm 下载量：～400 万次/月
- Jsdelivr CDN：5亿次引用/月
- GitHub stars：13.6万
  - 全 GitHub 第三，实际代码项目第一

全球化的影响力

- 遍布世界各地的线下聚会 https://events.vuejs.org/
- 目前每年在世界各地举办的 Vue 主题会议：
  - VueConf 中国
  - Vue.js Amsterdam（欧洲）
  - VueConf US（美国）
  - Vue Fes Japan（日本）
  - VueConf Toronto（加拿大）
  - VueDay Italy（欧洲）
  - Vue Summit Brazil（南美）

良好的反馈

- State of JavaScript 2018 调查
  - 前端框架满意度第一 (91%)
- StackOverflow 2019 年度调查
  - Most Loved Web Frameworks 第二

使用公司遍布全球


Alibaba Group
阿里巴巴集团


Baidu 百度


Tencent 腾讯


GitLab


Microsoft


IBM


NASA


Bloomberg


中国平安
PING AN


滴滴出行
—滴滴一下 美好出行—


新浪微博
weibo.com


饿了么


Adobe

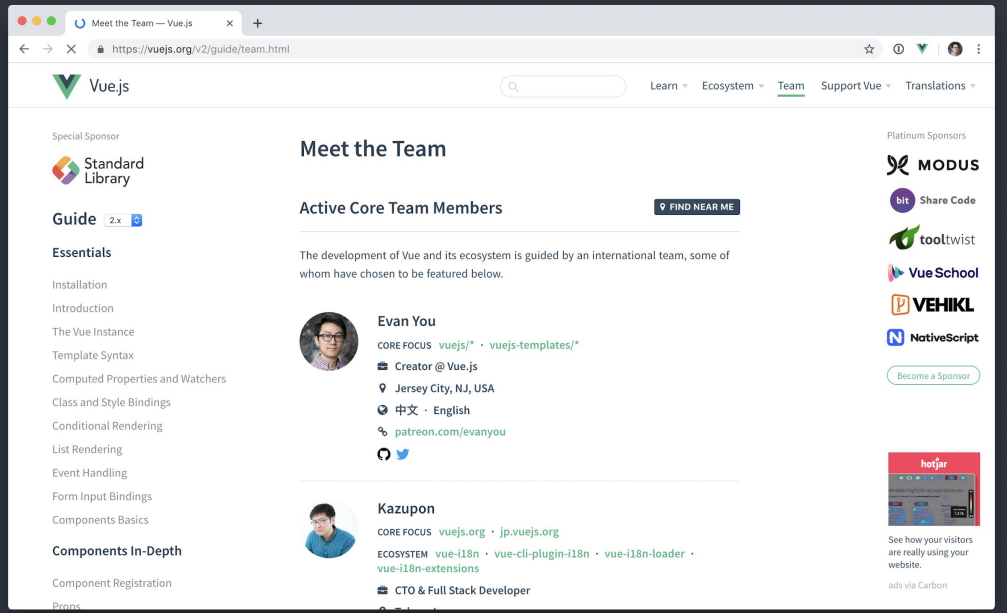

Bēhance


快手


dji
大疆創新

团队

- 20 人的活跃核心团队，来自世界各地，大部分日常工作与 Vue 相关
- 独立运营，资金主要来源于赞助商，三年来稳步增长
- 国内有一位开发者通过 OpenCollective 资金全职维护 CLI 及工具链



https://vuejs.org/v2/guide/team.html

与其它框架的比较

到底是“框架”还是“库”？

“只是一个视图层库”

~~“只是一个~~视图层库”

“渐进式框架”

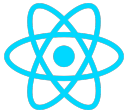
应用复杂度
vs.
框架复杂度

框架功能

Less

More

纯模版引擎



React

Vue

Backbone

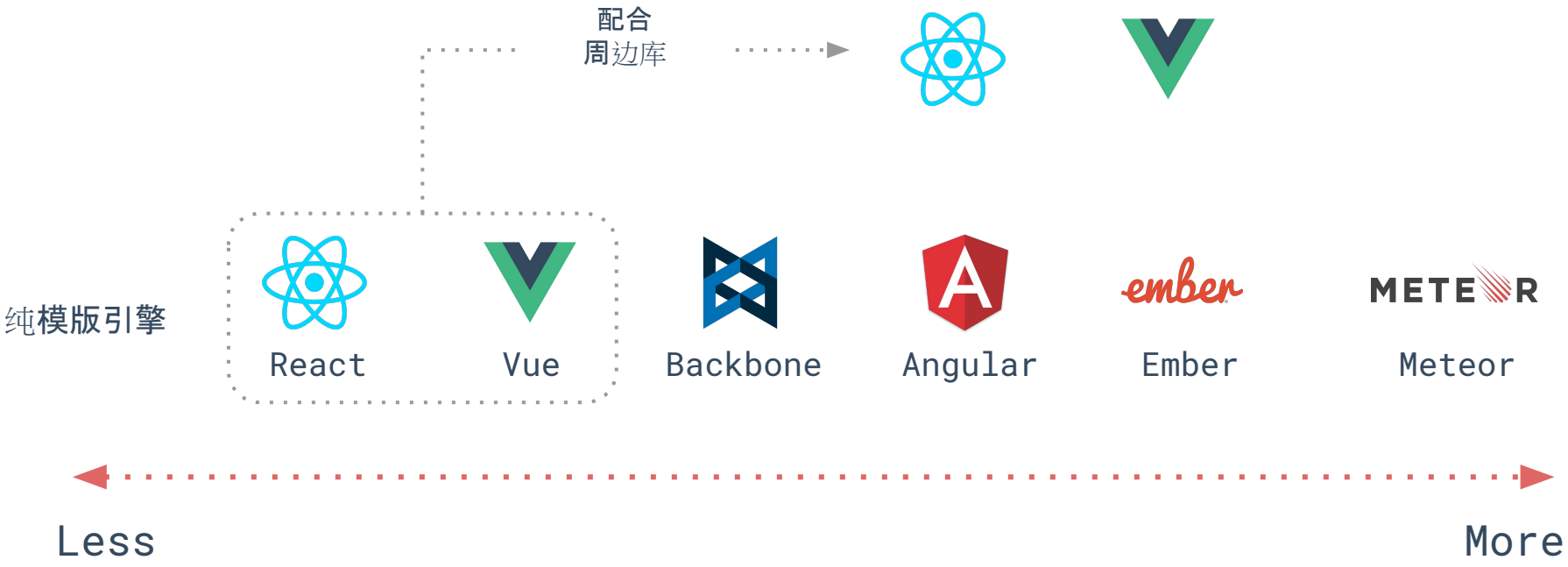
Angular

Ember

Meteor

Less

More


配合
周边库
纯模版引擎
React
Vue
Backbone
Angular
Ember
Meteor
Less
More

Eric Clemmons Follow
Creator of React Resolver, Genesis/Evolution for WordPress. Purveyor of a better Developer Experience…
Dec 26, 2015 · 4 min read

Javascript Fatigue

A few days ago, I met up with a friend & peer over coffee.

> *Saul: "How's it going?"*
> *Me: "Fatigued."*
> *Saul: "Family?"*
> *Me: "No, Javascript."*

"JavaScript 疲劳"

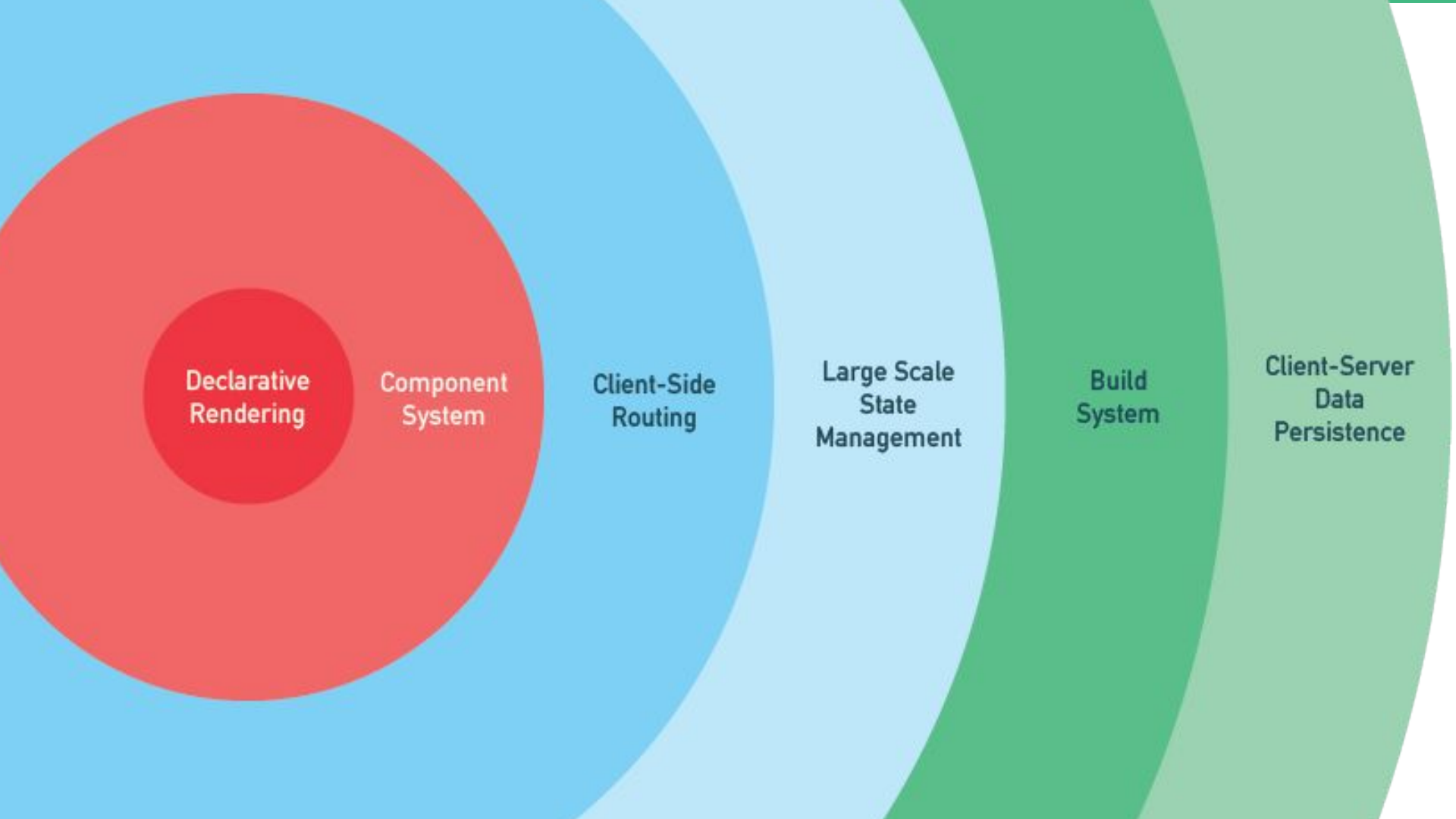
渐进式框架
The Progressive Framework

pro·gres·sive

/prəˈgresiv/

adjective

1. happening or developing gradually or in stages; proceeding step by step.
   "a progressive decline in popularity"
   synonyms: continuing, continuous, increasing, growing, developing, ongoing, accelerating, escalating; More


Declarative Rendering
Component System
Client-Side Routing
Large Scale State Management
Build System
Client-Server Data Persistence

对比 Angular

- 更灵活的适应各种场景
  - 默认 API 适合纯前端背景的开发者 / 小快灵场景
  - 配合 TypeScript 也可以适合传统 Java 后端背景的开发者 / 大型项目
- 更低的培训成本，更快的上手速度
- 底层的 Virtual DOM 在高级场景下提供更多的灵活性
- 大型应用中与 TypeScript 的整合不如 Angular
  - 3.0 中会针对性增强

对比 React

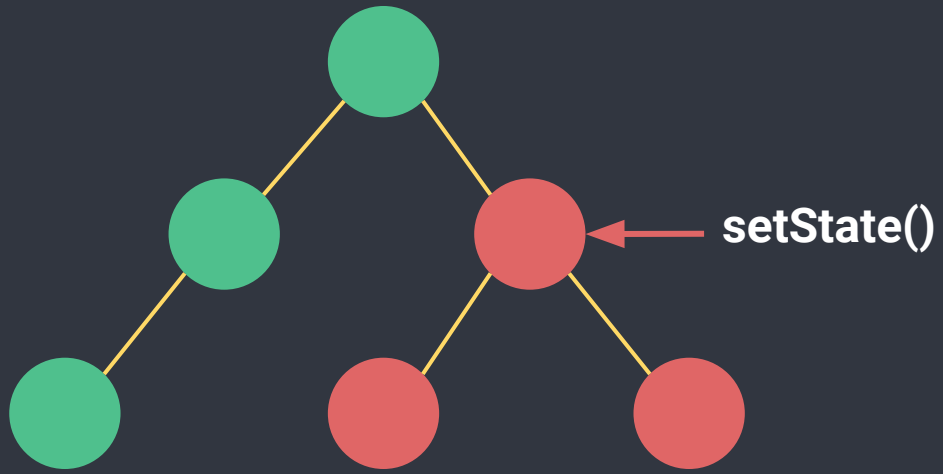
- 对大部分常见场景都提供了事实标准方案
  - 不需要额外自行调研选取方案
  - 在必要情况下也可以 换用自研方案
- 模版提供更友好的学习曲线
  - 同时暴露底层 Virtual DOM 用于高级场景（也支持 JSX）
- 大型应用中与 TypeScript 的整合暂时不如 React
  - 3.0 中会针对性增强，尤其是 TSX
- 对标 React 16+
  - Vue 同样可以实现类似 Hooks 的逻辑复用机制
  - 3.0 支持时间分片

与其它所有框架的区别

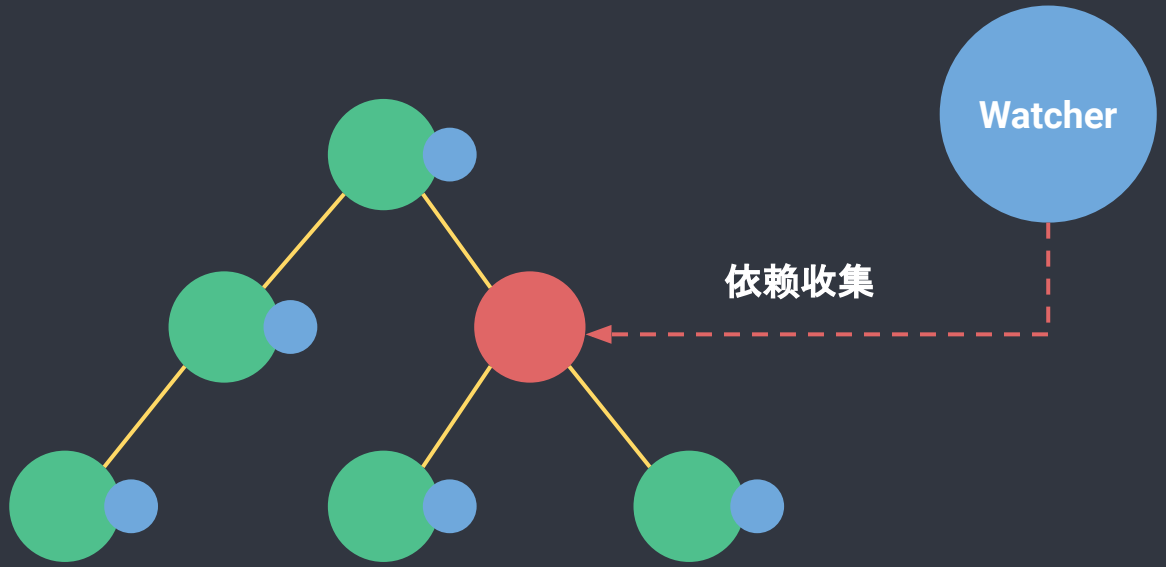
- 自带的响应式系统 (Reactivity System)
  - 类似于 MobX, 但与框架本身的整合更无 缝
  - 在复杂组件树中提供比 React 更精确的更新侦测
  - 3.0 将暴露更多底层响应式 API

- 单文件组件 (Single File Components)
  - HTML 的自然延伸, 符合直觉的代码组织方式
  - 完善的工具链
    - 预处理器支持
    - Scoped CSS
    - webpack 热更新
    - IDE 支持 (VSCode + Vetur)
    - Linter 支持 (eslint-plugin-vue)

React 的组件更新触发



Vue 的组件更新触发



生态介绍/方案推荐

官方工具链

- 路由：vue-router
- 状态管理：vuex
- 构建工具脚手架：vue-cli
- 开发者工具：vue-devtools
- IDE 支持：VSCode + Vetur
- 静态检查：ESLint + eslint-plugin-vue
- 单元测试：Jest + vue-jest + vue-test-utils
- 文档/静态站生成：VuePress

优秀的组件库

- Vuetify
  - 基于 Material Design，功能完整强大，桌面 + 移动
  - 支持 SSR
- Element-UI
  - 知名国产组件库，来自饿了么前端团队
  - 只支持桌面端应用
  - 有现成的控制后台模版
- iView
  - 另一个知名国产库，来自 TalkingData 前端团队
  - 主要支持桌面端，但也有小程序整合
  - 有现成的控制后台模版

优秀的组件库

- Quasar
  - 不仅仅是组件库的全平台解决方案
  - 桌面端 SPA/SSR + 移动端 PWA / Hybrid + 桌面端 Electron。

上层框架

- Nuxt.js
  - 开箱即用的 SSR 封装
  - 也支持 SPA / 静态生成模式
  - 有商业支持

移动端方案(H5 / Hybrid)

- Vant
  - 来自有赞的纯移动端 web 组件库
- Vux
  - 基于微信 UI 风格的移动端 web 组件库
- Onsen UI
  - 来自日本，基于 custom elements 支持多个上层框架的 hybrid 移动端方案
- Ionic 4
  - 知名 hybrid 移动端方案，原本只支持 Angular，4.0 开始通过原生 custom elements 支持 Vue（目前 beta）

移动端方案（Native)

- Weex
  - 来自阿里，现为 Apache 基金会项目
  - 在阿里内部广泛使用，经受过考验
  - 有一定的国内社区
- NativeScript
  - 原本只支持 Angular，但现在大力投入与 Vue 的整合
  - 完善的文档和活 跃的国外社区，但大多 为英文
  - 作为产品，背后有商业投入，提供商业支持
- Uni-app
  - 国产的 Vue 跨端方案，来自 DCloud
  - 同一套代码编译到 iOS, Android, H5 + 多种小程序

3.0 新特性 / 改动

3.0 设计目标

- 更小
- 更快
- 加强 API 设计一致性
- 加强 TypeScript 支持
- 提高自身可维护性
- 开放更多底层功能

更小

- 全局 API 和内置组件 / 功能支持 tree-shaking
- 常驻的代码尺寸控制在 10kb gzipped 上下

更快

- 基于 Proxy 的变动侦测，性能整体优于 getter / setter
  - 长远来看，JS 引擎会继续优化 Proxy，但 getter / setter 基本不会再有针对性的优化
- Virtual DOM 重构
  - 更新速度 / 内存占用均有质的提升
- 编译器架构重构，更多的编译时优化

提高自身可维护性

- 代码采用 monorepo 结构，内部分层更清晰
- TypeScript 使得外部贡献者更有信心做改动

开放更多底层功能

- Custom Renderer

```
import { createRenderer } from '@vue/runtime-core'

const { render } = createRenderer({
 nodeOps,
 patchData
})
```

传统 vdom 的性能瓶颈

- 虽然 Vue 能够保证触发更新的组件最小化，但在单个组件内部依然需要遍历该组件的整个 vdom 树
- 在一些组件整个模版内只有少量动态节点的情况下，这些遍历都是性能的浪费
- 传统 vdom 的性能跟模版大小正相关，跟动态节点的数量无关

动静结合突破瓶颈

- 通过模版静态分析生成更优化的 vdom 渲染函数
- 将模版切分为 block（if, for, slot），每个 block 内部动态节点位置是固定的
- 每个 block 的根节点会记录自己所包含的动态节点（包含子 block 的根节点）
- 更新时只需要直接遍历动态节点

动静结合突破瓶颈

- 新策略将 vdom 更新性能与模版大小解耦，变为与动态节点的数量相关
- 整体比 Vue 2 性能提升 2～5 倍

TypeScript

- 3.0 本身用 TypeScript 重写，内置 typing
- TSX 支持
- **不会影响不使用 TS 的用户**

TypeScript

重要的事情说三遍

- Class API 提案已撤销
- Class API 提案已撤销
- Class API 提案已撤销

Function-based API

```
const App = {
  setup() {
    // data
    const count = value(0)
    // computed
    const plusOne = computed(() => count.value + 1)
    // method
    const increment = () => { count.value++ }
    // watch
    watch(() => count.value * 2, v => console.log(v))
    // lifecycle
    onMounted(() => console.log('mounted!'))
    // 暴露给模版或渲染函数
    return { count }
  }
}
```

Function-based API

- 对比 Class API
  - 更灵活的逻辑复用能力
  - 更好的 TypeScript 类型推导支持
  - 更好的性能
  - Tree-shaking 友好
  - 代码更容易被压缩

关于逻辑复用

- Mixin
  - 混入的属性来源不清晰
  - 命名空间冲突
- 高阶组件 (HOC)
  - Props 来源不清晰
  - Props 命名空间冲突
  - 多余的组件实例造成的性能浪 费
- Scoped Slots
  - 来源清晰
  - 无命名空间冲突
  - 多余的组件实例造成的性能浪 费

关于逻辑复用

- Composition Functions
  - 就是简单的函数组合
  - 无额外的组件实例开销
  - 以 “功能” 而不是 “选项” 或 “生命周期” 切分代码

例子：抽取鼠标位置侦听逻辑

```
function useMousePosition() {
  const x = value(0)
  const y = value(0)
  const update = e => {
    x.value = e.pageX
    y.value = e.pageY
  }
  onMounted(() => {
    window.addEventListener('mousemove', update)
  })
  onUnmounted(() => {
    window.removeEventListener('mousemove', update)
  })
  return { x, y }
}
```

例子:抽取鼠标位置侦听逻辑

```
const App = {
  setup() {
    const { x, y } = useMousePosition()
    const z = useOtherLogic()
    return {
      x,
      y,
      z
    }
  }
}
```

对比 React Hooks

- 同样的逻辑复用能力
- 只调用一次
  - 符合 JS 直觉
  - 没有闭包变量问题
  - 没有内存/GC 压力
  - 不存在内联回调导致子组件永远更新的问题

所有核心改动都已经发布对应 RFC

https://github.com/vuejs/rfcs

培训 / 学习路线

新手向的学习路线

https://zhuanlan.zhihu.com/p/23134551

已有前端基础的学习路线

- 通过文档基础部分上手
- 通过实战类的视频课程或是书籍深化理解
- 再过一遍文档，这一遍着重看细节
- 在中小型项目中继续熟悉
- 阅读一些开源组件代码，了解一些高级用法
- 阅读深入内部实现、源码研究类的书籍
    - 推荐两本电子书：Vue 技术内幕 Vue.js 技术揭秘