Vue相关

# Vue

## 最佳实践

### 性能

1. 体现web 应用性能的两个主要方面

* **页面加载性能：**Google 所定义的一系列 [Web 指标](https://web.dev/vitals/#core-web-vitals) (Web Vitals) 来进行衡量
* **更新性能**

1. 分析

* 生产部署（[PageSpeed Insights](https://pagespeed.web.dev/)、[WebPageTest - Website Performance and Optimization Test](https://www.webpagetest.org/)）
* 本地开发（[Chrome 开发者工具“性能”面板](https://developer.chrome.com/docs/devtools/evaluate-performance/)、[Vue 开发者扩展](https://cn.vuejs.org/guide/scaling-up/tooling.html#browser-devtools)）

1. 加载优化

* 和框架无关（[web.dev 指南](https://web.dev/fast/)）
* 首屏加载缓慢避免使用纯SPA，结合SSR和SSG（如营销相关页面 (落地页、关于页、博客等)
* 引入es module提供tree-shaking和避免引入过重依赖

1. 更新优化

* prop稳定性：子组件只会在其至少一个 props 改变时才会更新，保持较少的prop变更

1. 通用优化

* 大型虚拟列表（[vue-virtual-scroller](https://github.com/Akryum/vue-virtual-scroller)、[GitHub - rocwang/vue-virtual-scroll-grid: A Vue 3 component that can render a list with 1000+ items as a grid in a performant way.](https://github.com/rocwang/vue-virtual-scroll-grid)、

[GitHub - 07akioni/vueuc: Util Components for Vue](https://github.com/07akioni/vueuc)）

* 减少大型不可变数据的响应性开销。Vue 的响应性系统默认是深度的，浅度的使用 [shallowRef()](https://cn.vuejs.org/api/reactivity-advanced.html#shallowref) 和 [shallowReactive()](https://cn.vuejs.org/api/reactivity-advanced.html#shallowreactive)
* 避免不必要的组件抽象，组件实例比普通 DOM 节点要昂贵得多

## 进阶主题

### 使用vue的多种方式

* 独立脚本，无需构建步骤，一般用户服务器渲染的html中的渐进式增强功能
* 作为webcomponent嵌入（暂时还没了解）
* spa
* SSR，纯客户端的 SPA 在首屏加载和 SEO 方面有显著的问题，服务端返回渲染好的html后再被客户端接管交互，框架有[Nuxt: The Progressive Web Framework](https://nuxt.com/)（该框架同时提供SSG能力）
* SSG，用于预渲染只包含静态数据的HTML，分为单页SSG（重交互）和多页SSG（更高性能）

### composition api FAQ

1. 什么是组合式api: 组合式 API (Composition API) 是一系列 API 的集合，涵盖以下：

* [响应式 API](https://cn.vuejs.org/api/reactivity-core.html)：例如 ref() 和 reactive()，使我们可以直接创建响应式状态、计算属性和侦听器。
* [生命周期钩子](https://cn.vuejs.org/api/composition-api-lifecycle.html)：例如 onMounted() 和 onUnmounted()，使我们可以在组件各个生命周期阶段添加逻辑。
* [依赖注入](https://cn.vuejs.org/api/composition-api-dependency-injection.html)：例如 provide() 和 inject()，使我们可以在使用响应式 API 时，利用 Vue 的依赖注入系统。

1. 为什么要有组合式api?

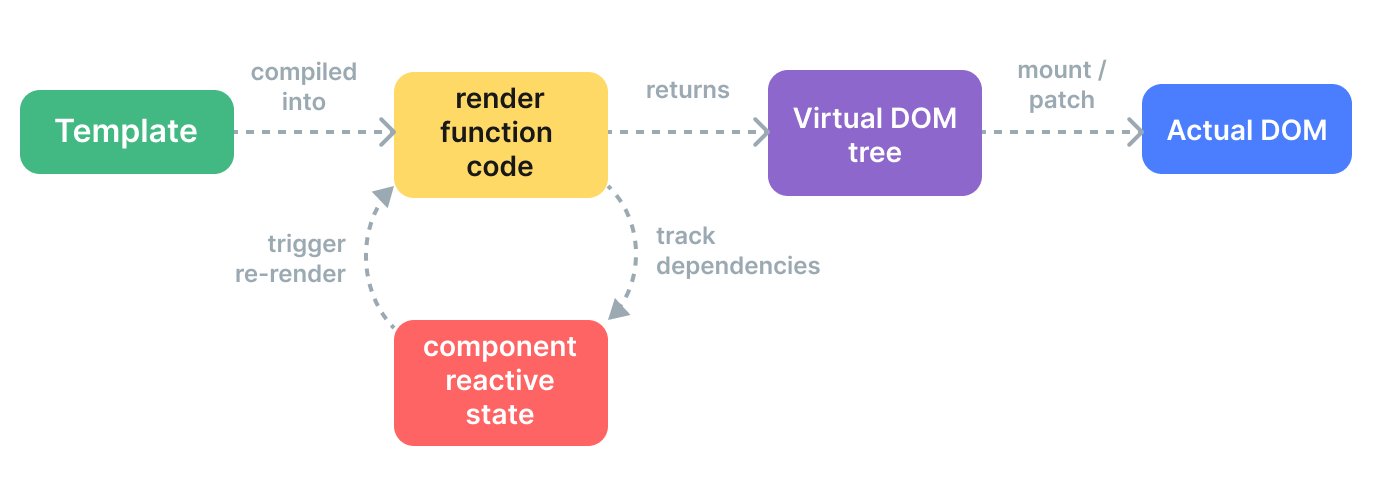
* 更好的逻辑复用：通过组合式函数实现，社区组合式函数集合项目： [VueUse](https://vueuse.org/)。vue2用的是mixins，有来源难以分析的缺陷等
* 更灵活的代码组织：而不是像选项式中需要把逻辑关注点分散在各个选项中
* 更好的类型推导：因为是利用基本的变量和函数
* 更小的生产包体积：<script setup>中的变量和模板中编译为了的内联函数位于同一作用域，不像选项式 API 需要依赖 this 上下文对象访问属性。这对代码压缩更友好，因为本地变量的名字可以被压缩，但对象的属性名则不能。

1. 和React hook的对比：

* 有相同级别的逻辑组织能力
* 可以条件执行
* 不用担心闭包变量问题
* 自动收集依赖

### 渲染机制

1. 虚拟dom: 设计模式
2. 渲染流程



1. 模板 vs. 渲染函数

* 模版优势：

1）模板更贴近实际的 HTML

2）容易对模板做静态分析

* 不带动态绑定的进行缓存静态内容
* 更新类型标记：编译时推断出大量信息，比如需要更新哪些属性。通过这样的更新类型标记，Vue 能够在更新带有动态绑定的元素时做最少的操作。
* 树结构打平：编译的结果会被打平为一个数组，仅包含所有动态的后代节点

# vue-router

## 不同的历史记录模式

### momery模式

Memory 模式不会假定自己处于浏览器环境，因此不会与 URL 交互**也不会自动触发初始导航**。这使得它非常适合 Node 环境和 SSR。

虽然不推荐，你仍可以在浏览器应用程序中使用此模式，但请注意**它不会有历史记录**，这意味着你无法*后退*或*前进*。