**VUE3.0特性**

**vue3.0六大亮点：**

-performance:性能比vue2.x快1.2-2倍

-tree shaking support:按需编译，体积比vue2.x更小

-composition API:组合api（类似于react hooks）

-better ts:更好的ts支持

-custom renderer API:暴露了自定义渲染api

-fragment,teleport,suspense:更先进的组件

**Vue3.0创建项目的方式？**

**-Vue-CLI**

Npm install -g @vue/cli

Vue create projectName

Cd projectName

Vue add vue-next

Npm run serve

**-Webpack**

Git clone

<https://github.com/vuejs/vue-next-webpack-preview.git> projectName

Cd projectName

Npm install

Npm run dev

**-Vite**

vite是vue作者开发的一款意图替代webpack的工具；

原理是利用import会发送请求去加载文件的特性，拦截这些请求，做一些预编译，省去webpack冗长的打包时间；

Npm install -g create-vite-app

Cva projectName / create-vite-app projectName

Cd projectName

Yarn install

Yarn run dev

**Vue 3.0是如何变快的？**

1. **Diff算法优化**

Vue 2.x ：虚拟dom是进行全局比较的

Vue 3.0 ：新增了静态标记（patchflag）

在创建虚拟dom的时候,会根据dom中的内容会不会发生变化，添加静态标记

在与上次虚拟节点进行比较的时候，只对比带有patch flag的节点，并且可通过flag的信息得知当前节点要对比的具体内容（eg: flag=1 动态文本 flag=2 动态class flag=4 动态style....）

1. **hoistStatic 静态提升**

Vue2.x：无论元素是否参与更新，每次都会被创建，然后再渲染

Vue3.0：对于不参与更新的元素做静态提升，只会被创建一次，在渲染时直接复用

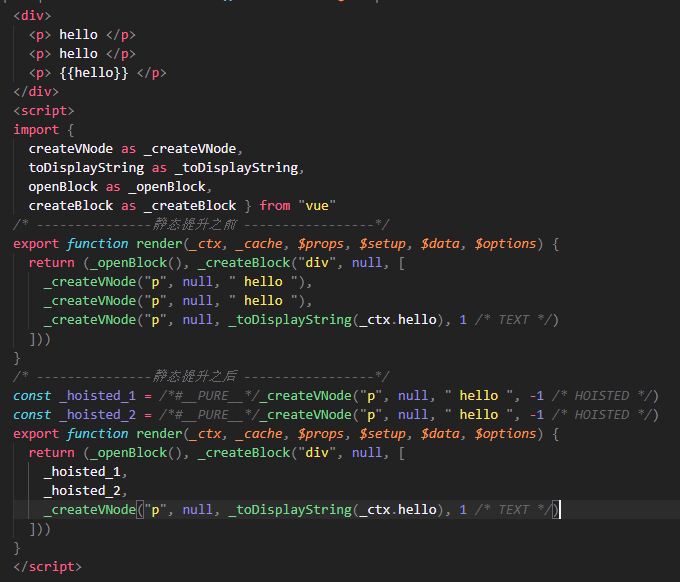
1. **Cache Handlers 事件侦听器缓存**

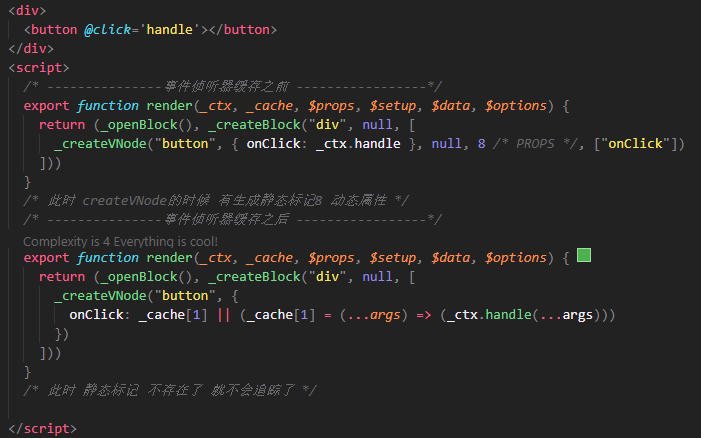
默认情况下on Click会被视为动态绑定，所以每次都会去追踪他，

但是因为是同一个函数，所以可以直接缓存起来复用

1. **ssr 渲染**







**Vue3是如何实现响应式数据的？**

*- reactive是什么？*

*+   reactive是vue3提供的实现响应式数据的方法*

*在vue2中响应式数据通过defineProperty来实现*

*在data中声名一个obj后，vue2.x会利用Object.defineProperty来递归的给data中的数据加上get和set，然后每次set的时候，加入额外的逻辑。来触发对应模板视图的更新*

*而在vue3中响应式数据是通过es6中的proxy来实现的*

*通过proxy实现对data对象的get和set的劫持，并返回一个代理的对象*

*- reactive的注意点？*

*+   reactive的参数必须是对象（json/arr）*

*+   如果给reactive传递的不是自定义的对象 例如 Date*

*+   默认情况下修改对象，log中的值会变，ui中的值不会更新*

*+   如果想要ui中的对象同步更新，需要重新复制*

*-   ref和reactive一样，也是用来实现响应式数据的方法*

*-   ref的本质还是reactive，系统会根据我们传给ref的值将他转化成reactive*

*+   ref(18)->reactive({value:18})*

*+   所以更改ref数据时，需要 变量名.value=新值*

*ref和reactive的区别：*

*+   在template中使用ref类型数据，vue会自动我们添加.value*

*+   在template中使用reactive类型数据，vue不会自动我们添加.value*

*+   vue在解析数据之前会自动判断这个数据是否属于ref类型*

*+   判断方式通过当前数据的私有属性\_\_v\_ref来判断*

*+   如果有这个私有属性且为true，那就代表这是一种ref类型数据*

**递归监听（shallowRef,shallowReactive,triggerRef）**

- ref和reactive都是递归监听（一般情况下可以使用这两种类型）

- 递归监听在数据量比较大的情况下，非常消耗性能

- shallowRef和shallowReactive是非递归监听

- 如果是shallowRef类型数据，可通过triggerRef主动触发页面更新

- shallowRef创建数据，vue监听的是.value的变化，而非第一层数据的变化

- shallowReactive创建数据，vue监听的是第一层数据变化，如果第一层数据有变化，视图就会更新，反之，如果第一层没有变化，深层的数据有变化也不会引起视图更新