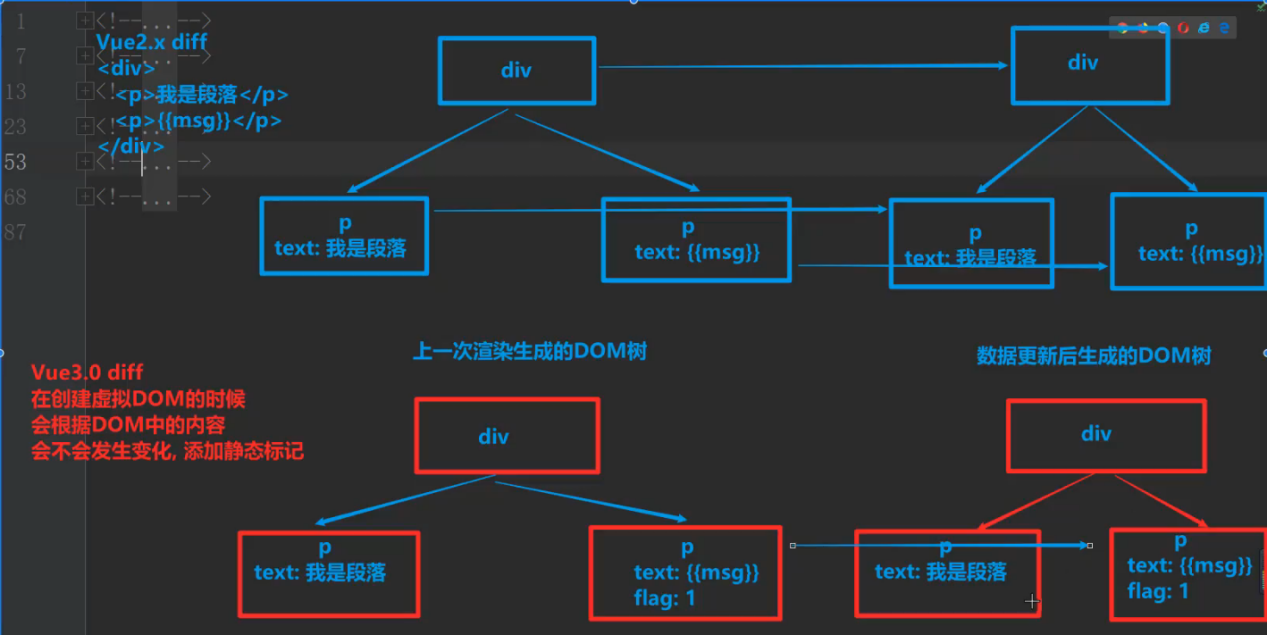
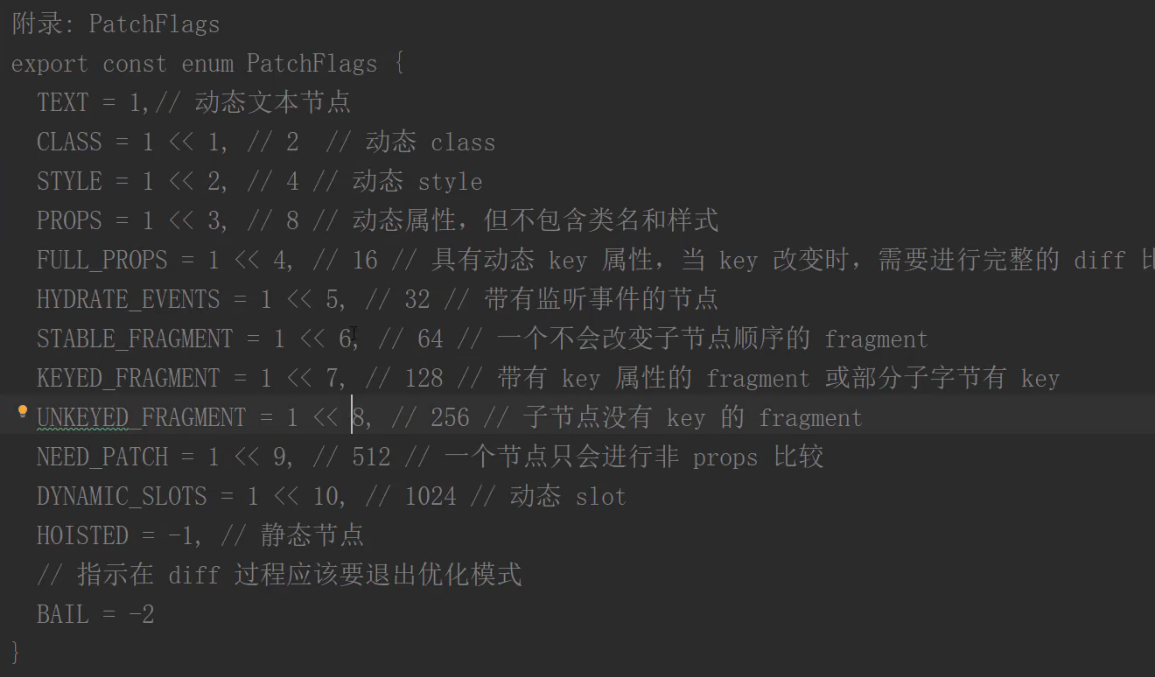
<https://vue-next-template-explorer.netlify.app/> // 代码转为vue3源码格式

**Diff算法**

<1>diff算法

在2x版本中，虚拟dpom是进行全量的比较，diff算法是逐层比较的，div跟div比较，p跟p标签比较

在3x版本中增加了静态标记（patchFlag），在于上次虚拟节点比较的时候，只对比带有patchFlag的节点

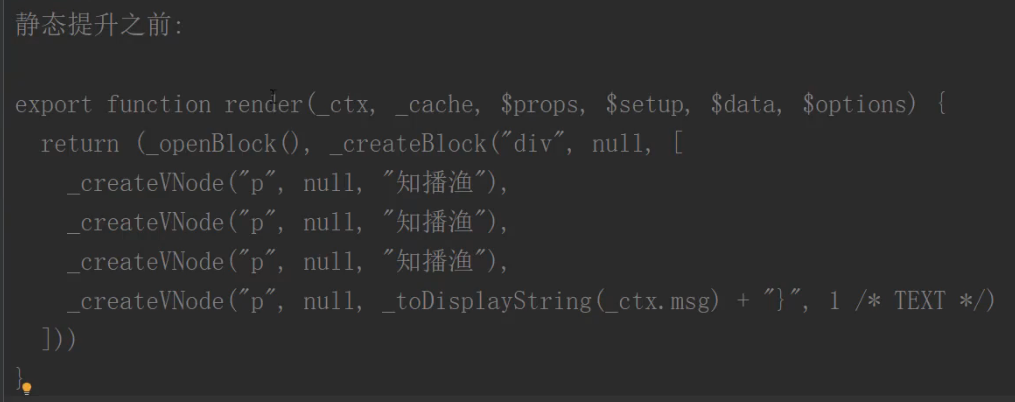
 

<2>静态提升

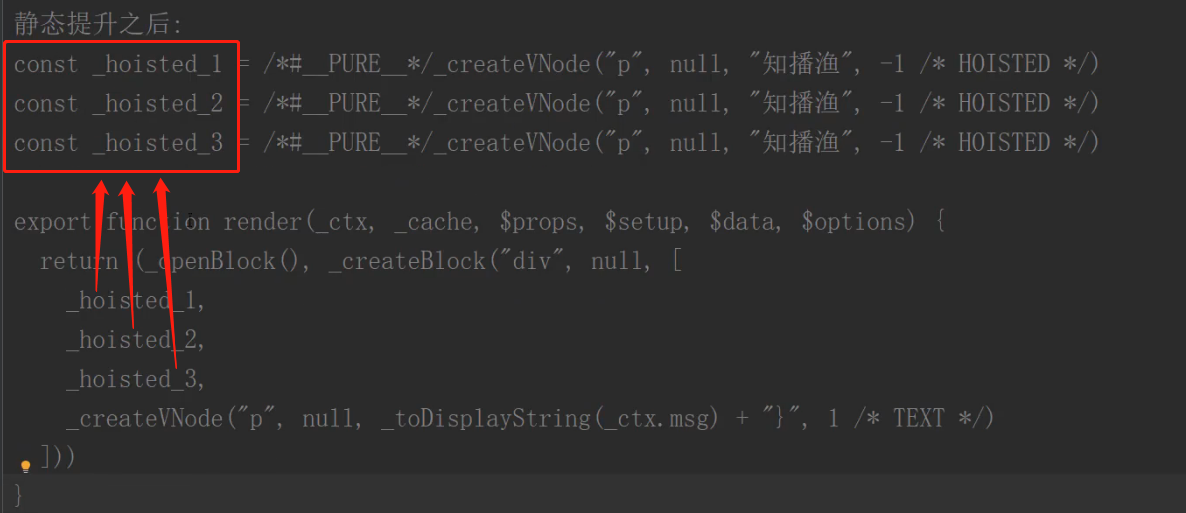
2x中无论元素是否参与更新，每次都会重新创建，然后再渲染

3x中对于不参与更新的元素，会做静态提升，只会被创建一次，再渲染的时候直接复用

静态提升之前：



静态提升之后



<3>事件侦听器缓存

2x中会对onclick视为动态绑定，每次都会追踪它的变化

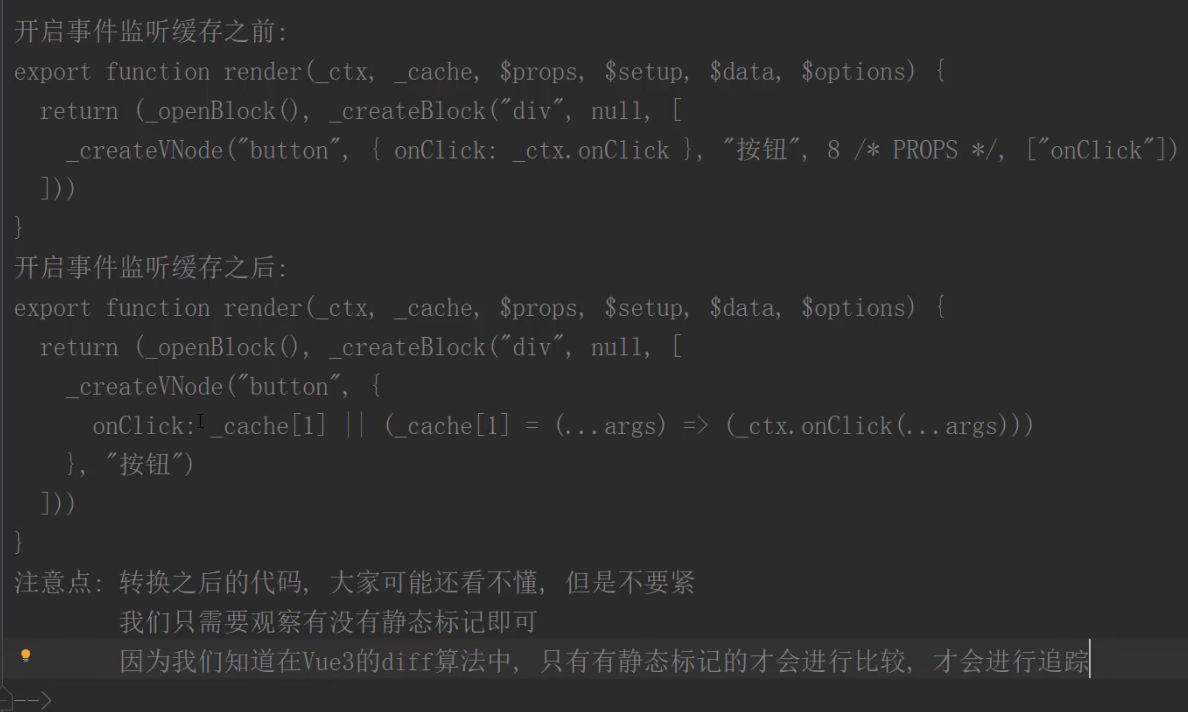
3x中，因为是同一个函数，不需要追踪变化，直接缓存即可

Eg:

Eg： <div>

        <button @click='onClick'></button>

    </div>



**开始创建项目**

**1什么是vite**

Vite是vue 作者开发一款意图取代webpack 的工具

其实现原理是利用es6的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的冗长打包时间

安装vite

npm install -g create-vite-app

利用vite创建v3项目

Create-vite-app projectName

启动项目

npm run dev

**全局 API Treeshaking**

在2x版本，Vue.nextTick() 是一个全局的 API 直接暴露在单个 Vue 对象上，在全局 API中， Vue.nextTick() 不可摇动，将包含在最终捆绑中不管它们实际在哪里使用。



在3x版本，全局 API 现在只能作为 ES 模块构建的命名导出进行访问。例如，我们之前的片段现在应该如下所示：



从而实现了按需编译，体积比vue2.x更小。

**组合api**

在2x中，同一个业务逻辑（eg：新增）可能在methods computed watch中都有代码逻辑编写，数据和业务逻辑分级的问题

在3x中 ，setup是组合api的入口函数，可以将统一个业务逻辑写到同一个方法中，解决了数据和业务逻辑分级的问题



图例



在3x中怎么使用ref

<span ref="box1"></span>

import { ref,onMounted} from 'vue'

 setup(){

const box1 =ref(null)

onMounted(()=>{

this.$refs.box1

})

return {box1}

}