数据劫持

1. 递归 Object.defineProperty(object,key,{ get() {}, set() {} }) Object.defineProperty(obj, prop, descriptor)给对象上定义一个新属性，或修改一个对象的现有属性，并返回此对象。

descriptor：属性描述符

属性描述符两种： 数据描述符，存取描述符，不能同时是两者

两者共享的可选键值： configurable:false enumerable:false

数据描述符： value:undefined writable:false

存取描述符：get,set

Object.defineProperties(obj, { 'property1': { value: true, writable: true }, 'property2': { value: 'Hello', writable: false } // etc. etc. });

- **vue2劫持数据采用defineProperty，会把对象循环给每个属性增加getter，setter** 无法监听对象的新增属性，删除属性，无法监听数组方法的操作，数组length的修改，索引设置数组 需要递归key对应的value(如果value是引用属性) 数组需要劫持方法来单独处理
- **vue3采用proxy不需要去改写属性getter，setter，不需要完全递归，取到某一层递归**

proxy: const p = new Proxy(target,handler) Proxy对象用于创建一个对象的代理，从而实现基本操作的拦截和自定义（如属性查找、赋值、枚举、函数调用等） handler对象包含Proxy的各个捕捉器 getPrototypeOf, setPrototypeOf,isExtensible,preventExtensions,getOwnPropertyDescriptor,defineProperty,has(),get(),set(),deleteProperty，ownKeys，apply(函数调用),constructor(new 操作符)

vue3性能提升

编译阶段

diff算法优化

vue2每个组件实例都对应一个wacher实例，它会在组件渲染的过程中把用到的数据property记录为依赖，当依赖发生改变，触发setter，则会通知watcher，从而使关联的组件重新渲染。 异步渲染 dom diff

vue2 dom diff 算法复杂度o(n),同层级比较，同时采用了双端比较的算法，如果4种比较都没匹配，如果设置了key就会用key进行比较 组件tagName不一致放弃继续比对，key不一致放弃比对

vue2 虚拟DOM是全量的对比，在运行时会对所有节点生成一个虚拟节点树，数据更新时，会遍历判断vnode所有节点有没有发生变化

vue3 dom diff

vue3巧妙结合runtime(patchVNode)与compiler实现靶向更新和静态提升

vue3中，模板编译时，编译器会在动态标签末尾加上/*Text*/PatchFlag。也就是在生成VNode的时候，同时打上标记，在这个基础上再进行核心的diff算法并且patchFlag会标示动态的属性类型有那些。 vue3对于不参与更新的元素，做静态标记并提示，只会被创建一次，在渲染时直接复用。

vue3在创建vnode的时候，会根据vnode的内容是否可以变化，为其添加静态标记patchFlag。diff的时候，只比较有patchFlag的节点，patchFlag是有类型的，比如一个可变化文本节点会将其添加patchFlag枚举类为TEXT的静态标记。这样diff的时候只需比对文本内容，需要比对的内容更少了。PatchFlag还有动态class、动态style、动态属性、动态key属性等枚举值

render阶段的静态提升（render阶段指生成虚拟DOM树的阶段）

在vue2中，一旦检测到数据变化，就会re-render组件，所有的vnode都会重新创建一遍，形成新的vdom树

在vue3中，对于不参加更新的vnode，会做静态提升，只会被创建一次，re-render时直接复用。

静态提升可以理解为第一次render不参与更新vnode节点的时候，保存它们的引用。re-render新dom树时，直接拿它们的引用过来即可，无需重新创建。

事件侦听缓存

在vue2中@click="onClick"会被当作动态属性，但实际上其不会随数据更新而变化，所以vue3中会把事件缓存起来标记为无需更新，这样在render和diff两个阶段事件侦听属性都节约了不必要的性能消耗。

按需编译，体积更小

函数需要按需引入，更好的tree-shaking，没有用到的不会打包，体积更小

支持compositionAPI 逻辑更容易复用，而且逻辑不用像vue2那样分散更好维护

更好的ts支持

模板编译优化

- vue2优化：判断是不是静态节点，如果静态节点不去dom diff
- 编译时生成block tree，对子节点动态节点收集，减少比较，采用了patchFlag标记动态节点

compositionApi 避免了在配置项里反复横跳，优化复用逻辑（mixin带来的数据来源不清晰，命名冲突），类型推断更加方便

增加了Fragment（多根节点），teleport，suspense组件

---

ref（将一个普通类型，转换成一个对象，这个对象有value属性，指向原来的值）和reactive（） toRef（解构reacttive 解构某一个 将一个对象的属性变成ref） toRefs 响应式解构 name.value ）

---

reactive内部用proxy ref内部用proxy

effect中所有的属性都会收集effect，当这个属性发生变化会重新执行effect
