快速入门VUE3
该文档的目的在于快速上手vue3的开发

1. 首先理解data属性，其返回的对象属性会在组件实例被创建的过程中存储在实例上；双向绑定，**区别于react的state；$和_前缀为vue保留关键字，不能作为内部data属性的命名方式；**
2. 其次methods属性包含组件实例所需的方法，并且实现自动绑定this指向实例逐渐，因此需要注意避免好实用箭头函数，这会影响this指向；
3. methods和computed的区别：
   首先，computed是属性，注册的函数是getter，而methods是方法；实际上可以在methods中注册一个同样功能的属性；
二者区别在于：
Computed属性依赖于响应式数据缓存，计算属性只会在相关响应式依赖发生改变时重新求值；计算属性默认着自由getter，也可以设置setter
Methods里面的方法会在触发重新渲染时再次执行函数；

4. **使用计算属性可以满足大部分情景，也可以使用自定义的侦听器在响应数据的变化，watch选项可以实现这个功能，以响应异步操作或者时间开销大的操作，使用属性同名函数完成定义；**

计算属性与侦听器的区别：
都可以实现对data的响应，但是侦听器定义于单个属性，而计算属性可以同时侦听多个属性；

5. **修饰符概念：**

修饰符被附加到事件的末尾，并带有一个点，如下所示：`@event.modifier`. 以下是事件修饰符列表：

- `.stop`：停止传播事件。等效`Event.stopPropagation()`于常规 JavaScript 事件。
- `.prevent`：防止事件的默认行为。相当于`Event.preventDefault()`。
- `.self`：仅当事件是从这个确切的元素分派时才触发处理程序。
- `{.key}`：仅通过指定的键触发事件处理程序。MDN 有一个有效键值列表；多词键只需要转换为 kebab 大小写（例如`page-down`）。
- `.native`：在组件的根（最外层包装）元素上侦听本机事件。
- `.once`：监听事件，直到它被触发一次，然后不再触发。
- `.left`：仅通过鼠标左键事件触发处理程序。
- `.right`：仅通过鼠标右键事件触发处理程序。
- `.middle`：仅通过鼠标中键事件触发处理程序。
- `.passive`：等价于`{ passive: true }`在 vanilla JavaScript 中使用`addEventListener()`.

6. **Vue生命周期**

Vue 允许您使用生命周期方法在此生命周期的各个阶段运行方法。这对于诸如数据获取之类的事情很有用，您可能需要在组件呈现之前或属性更改之后获取数据。生命周期方法列表如下，按照它们触发的顺序。

1. `beforeCreate()` — 在创建组件实例之前运行。数据和事件尚不可用。
2. `created()` — 在您的组件初始化之后但在组件添加到 VDOM 之前运行。这通常是发生数据提取的地方。
3. `beforeMount()` — 在你的模板编译之后，但在你的组件被渲染到实际的 DOM 之前运行。
4. `mounted()` — 在您的组件安装到 DOM 后运行。可以访问 `refs` 这里。
5. `beforeUpdate()` — 只要组件中的数据发生更改，但在将更改呈现到 DOM 之前运行。
6. `updated()` — 每当您的组件中的数据发生更改以及将更改呈现到 DOM 后运行。
7. `beforeDestroy()` — 在从 DOM 中删除组件之前运行。
8. `destroyed()` — 在组件从 DOM 中移除后运行
9. `activated()` — 仅用于包装在特殊 `keep-alive` 标签中的组件。在组件激活后运行。
10. `deactivated()` — 仅用于包装在特殊 `keep-alive` 标签中的组件。在组件停用后运行。

7. **一些api的理解**

toRef：可以将一个对象的某个属性变为响应式的数据，数据会更新，但是不会渲染在视图上，类似深拷贝的效果创建一个全新的响应式副本；

toRefs：将响应式对象进行解构；

toRaw：将响应式对象变为普通对象。

markRaw：向一个实例dom添加一个属性_v_skip，不会为其创建proxy对象，因为一般不会直接对实例dom对象进行操作，可以跳过创建代理，劫持数据的操作。