Vue 李靖学习一周阶段总结(暗号：笔记)

我先确定我的的目标是学习完 Vue 框架后，能够制作出一套开源并且精美的后台管理页面。

第一步我要学习基础结构，有如下基本结构：
el: '#app',挂载点是目的地。

template：模板``。如果 template 中定义了内容，优先加载 template 模板；这里跟入口组件有关系。如果没有定义内容，加载的是#app 的模板。

data：保存数据属性，体现最精彩的数据驱动视图原则，（{{}}体现了 MVVM 中的 ViewModel 部分，而且注意大胡子内部需要有空格）
既可以是对象模式/又可以是函数模式，vue 内部会做判断，把函数的返回值作为 data 渲染到对应大胡子的位置。

render：渲染函数会返回一个 VNode 对象（虚拟节点）
因为 Vue 会根据虚拟节点进行渲染，得到真实的 DOM 节点 ,然后挂在到指定的位置。

第二步什么是指令呢？在 vue 中提供了一些对于页面+数据的更为方便的输出，这些操作就叫做指令，以 v-xxx 表示。指令中封装了一些 DOM 行为，结合属性作为一个暗号，暗号有对应的值，根据不同的值，框架会进行相关 DOM 操作。的绑定。
1、下面我列举一些常用的指令与自定义指令：

首先进行响应式的双向绑定，就是很方便的一点。data=>view=>data=>view2
v-model：就是双向数据绑定的体现，下面举个表单绑定代码例子：
（注意只会体现在 UI 控件中，只能应用在有 value 属性的地方）

{{message}}

然后 v-bind 是控制 class 名字，动态绑定标签属性的灵活使用。

再比如限制减一的按钮减到 1 就不能再减了，可以如下操作 v-bind:disabled="item.count <= 1".
循环遍历 v-for index 的使用场景我不常遇到，下面我用一段代码演示来加深我对这个 index 的理解：优先级最大

- {{index}}.{{item}}

其次是条件判断 v-if 的灵活使用：数据属性对应的值如果为假，则不在页面中渲染反之亦然。即判断是否插入这个元素，相当于对元素的销毁和创建，下面举个例子。

八百上映了
这里要注意 v-show 的区别:一般来说，v-if 有更高的切换开销，因为它是真正的条件渲染，它会确保在切换过程中条件块内的事件监听器和子组件的适当的被销毁和重建。相当于 appendChild()或者 removeChild()。
v-show 是通过控制 dom 元素的显示隐藏 display:none|block，因此有更高的初始渲染开销。
综上，如果需要非常频繁的切换，则使用 v-show 较好；如果在运行时条件很少改变，则使用 v-if 较好。

v-on：事件调用，原生事件名='函数名'简便写法：@

最后我们还可以通过 `Vue` 提供的方法来自定义指令。

`vue` 提供了两种指令注册方式
全局指令注册： Vue.directive('指令名称', {指令配置});
局部指令注册：

```
new Vue({
el: '#app',
directives: {
'指令名称': {指令配置}
}
});
```

注意在使用指令的时候，需要使用 `v-指令名称` 的方式来调用，注册的时候不需要。

2、可复用组件（实现组件化开发）的基础知识我要整理以下几个关键点。

首先组件的概念是对应用可复用的结构、样式、行为的封装。然后还会产生父子组件。

因为每一个组件都可能会有数据：
一是组件内部私有数据-函数内部变量-data，
二是组件外部传入数据-函数的形参-props。

局部组件的使用原则：声明子组件，挂载子组件，使用子组件。

```
//1.声子
var App={
template: <div>我是入口组件</div>
}

new Vue({
el: '#app',
```

```
data(){
return {
msg:'hello
}
},
//2.挂子
components:{

}

//3.用子
template:'',
});
```

首先 data 必须是一个函数，因为可复用组件可能会被复用很多分，如果直接把 data 设置成对象，则会共享，一处改动其他也会跟着改动。

然后 props 又多种不同的形式，为了保证 props 传入数据的正确性，props 还可以定义成对象形式，

下面我举个全局组件例子，Props 中的数据与 data 类似，都会代理到组件实例上。

生命周期：（钩子函数）
指令的运行方式很简单，它提供了一组指令生命周期钩子函数，我们只需要在不同的生命周期钩子函数中进行逻辑处理就可以了

- bind : 只调用一次，指令第一次绑定到元素时调用。在这里可以进行一次性的初始化设置
- inserted : 被绑定元素插入父节点时调用 (仅保证父节点存在，但不一定已被插入文档中)
- update : 所在组件更新的时候调用（但是可能发生在其子 VNode 更新之前）。
- componentUpdated : 所在组件及其子组件更新完成后调用
- unbind : 只调用一次，指令与元素解绑时调用。

不同的生命周期钩子函数在调用的时候同时会接收到传入的一些不同的参数

- el : 指令所绑定的元素，可以用来直接操作 DOM
- binding : 一个对象，包含以下属性：
  - name : 指令名，不包括 `v-` 前缀
  - value : 指令的绑定值（作为表达式解析后的结果）
  - expression : 指令绑定的表达式（字符串）
  - arg : 传给指令的参数，可选
  - modifiers : 传给指令的修饰符组成的对象，可选，每个修饰符对应一个布尔值
  - oldValue : 指令绑定的前一个值，仅在 update 和 componentUpdated 钩子中可用，无论值是否改变都可用

3、 watch 与 计算属性 computed ： watch 支持异步任务,比如 当组件当某个状态（数据）发生变化以后，需要去做一些事情，那么就可以通过 watch 来实现。 而 computed: 根据某些东西的变化产生数据
举个例子

```
watch: {
visible() {
console.log(this.visible);
this.$emit(this.visible ? "open" : "close");
},
},
```