上节说完初始，暂时对vue有了一个浅显的认知 下面正式开始使用

诚如文档所言，每一个vue应用都是通过vue函数来创建一个新的vue实例开始的

var VM = new Vue({

这里面是选项·，可以设置诸如data，el，methods，computed，钩子函数等

})

数据和方法

当一个vue实例被创建时，它向vue的响应式系统中加入了其data对象中能找到的所有的属性。当这些属性的值发生变化时，视图会随之响应，既匹配更新为新的值

但是有一点需要我们注意，能够为响应式的数据或属性的，他们必须在data选项中进行初始化配置，否则是不具备响应式的功能的

（值得注意的是只要被创建时的data中存在的属性才是响应式的。。）

实例生命周期钩子

所谓生命周期就是指的就是vue的实例被创建，挂载以及卸载的一系列过程

每个vue实例在被创建时都要经过的一系列初始化过程--例如设置数据监听，编译模板，将实例挂载在DOM并在数据变化时更新DOM等。同时这个过程中也会运行

一些叫做生命周期钩子的函数，这给我了用户在不同阶段添加自己代码的机会。

beforeCreate

created

beforeMount

mounted

beforeUpdate

Updated

beforeDestory

destoryed

actived

deActived

这些钩子函数对应着vue实例的生命周期过程，创建，挂载，更新，卸载等等；

需要注意的是在调用钩子函数时，我们不要使用箭头函数，因为生命周期的钩子函数中的this指向vue实例，而箭头函数是指向父级上下文的

模板语法

vue.js使用了基于HTML的模板语法，允许开发者声明式的将DOM绑定至底层vue实例的数据。所有vue.js的模板都是合法的HTML,所有能被遵循规范的浏览器和HTML

解析器解析。

在底层的实现上，vue将模板编译成虚拟的DOM渲染函数。结合响应系统，vue能够智能的计算出最少需要重新渲染多少组件，并把DOM操作次数减少的最少。

在模板语法中，我们需要把数据与DOM结合起来渲染出用户可见可操作的美好页面，vue的数据绑定最常见的形式就是双大括号文本插值，也叫作Mustache（{{}}）

Mustache标签会拿到data中初始化数据的值，并把它渲染到页面，如果数据改变，那么页面也会随之刷新

v-once

如果我们希望实现一次性的插值，那么v-once等着你去探索

v-html

还有一点需要注意的是Mustache标签会把数据解析成普通文本，它并不能输出html，此时我们可以用vue指令v-html

v-text

同样的有时我们希望取到html标签中的文本值，并且后台传过来的数据为html时，那么可以用v-text

v-bind 简写：

Mustache语法并不能作用在html上，如果我们需要给html标签绑定某个属性，把他设置成可变的变量，此时v-bind就可以出马了

vue中同样可以使用js表达式，卸载Mustache标签被，但是我们一般不这样做，代码过于杂乱不利于维护，再一个与写在methods方法集合computed相比性能并不好

同样的是表达式会在所属的vue实例的数据作用于下作为js解析，但是，每一个绑定只能包含单个表达式 ，语句，或流式控制也不会显示

换言之，只有单条语句才能被解析，赋值语句不生效，if for等语句也不行

而且模板表达式都被放在沙盒中，只能访问全局变量的一个白名单。如Math，Date，我们并不能在模板表达式中试图访问用户定义的全局变量

vue中的某些指令可以接收一个参数，在指令后以冒号表示；

诸如<a v-bind:href='url'></a>

<li v-on:click='dosome'></li>

修饰符

修饰符是以半角句号.指明的后缀，用于指出一个指令应该以特殊的方式绑定

如：preventDefault（）；

指令缩写

vue针对于指令提供缩写 比如 v-on：click==》@click v-bind：href ==》：href