**Data项：**

组件中data项必须声明为返回一个初始数据对象的函数，防止实例化多个组件后公用同一个数据对象，这样保证每次实例化的组件都能得到一个全新的副本数据对象，每个组件的data的独立（对象的引用关系，变量名保存的是该对象在堆内存中物理地址）

**模板：**  
1.html模板：

{{}},v-text,v-html,js运算表达式

2、字符串模板（template）

3、render函数（createElement返回的是虚拟dom）

**Import和export**

1、一个js文件只能有一个exprot default

但是可以有多个exprot

2、在不用default时，在需要引用的组件中用｛｝包起来，注意此处｛｝中的名字要和外部的js中的方法名要一致

**Better-scroll使用注意**

1、层级关系：

BS的基本结构是：一个wrapper层，一个content层。

wrapper层是整个滚动页面占据的显示空间。content层则包含了全部的页面内容。

<div class="wrapper">

<div class="content">

content...

</div>

</div>

new BS('wrapper');

简单点说，就是：wrapper站住位置，content在wrapper里面滚动。

2、content的高度要比wrapper的高度要高

在这里我们移动端通常要使用父级div高固定，子级两个div上下，上边div高固定，下边自适应的布局 布局方法见小哥博客https://my.oschina.net/u/3407699/blog/1920464

3、在vue中使用时

在计算一些跟dom相关的东西时，一定要保证该dom已经被渲染了，所有要在vue的$nextTick回调函数里面进行BS的实例化

$nextTick：在同一事件循环中的数据变化后，DOM完成更新，立即执行nextTick(callback)内的回调。

4、事件绑定：

//bScroll 自己封装的派发click事件，该事件对象有\_constructed，原生的没有该属性

//在pc端，原生点击会被监听到，没有被阻止，在初始化bScroll的时候又派发了click事件，会执行两次

//为了pc和移动端的效果一样,这里做下判断，不执行原生的click事件

**过渡**

例子：显示：从右边向左边滑动，隐藏：从右侧滑动到左侧

使用translate3d进行动画硬件加速

&.food-enter-active,&.food-leave-active{ //过渡的过程

transition: all 0.3s linear;

}

&.food-enter,&.food-leave-to{ //显示开始和隐藏结束的状态

transform: translate3d(100%,0,0);

}

&.food-enter-to,&.food-leave{ //显示结束的状态和隐藏开始的状态

transform: translate3d(0,0,0)

}

**Flex布局**

flex:1 和flex:2的比值是不包括padding和margin 即flex-grow属性

flex-shrink为0的项在空间不足时不会缩小，其他为1的则会缩小

flex-basis设置项的长宽 ，默认auto即项目本来的大小

水平对齐 align-items

**注意事项：**

**数组：**

1、当你利用索引直接设置一个项时，例如：vm.items[indexOfItem] = newValue

2、当你修改数组的长度时，例如：vm.items.length = newLength

对于第一种可以这么解决：

Vue.set(vm.items, indexOfItem, newValue)

vm.items.splice(indexOfItem, 1, newValue)

vm.$set(vm.items, indexOfItem, newValue)

这三种方式

对于第二种可以这么解决：

vm.items.splice(newLength)

**对象**

Vue 不能检测对象属性的添加或删除：

解决的方案有：（三种）

1、Vue.set(vm.userProfile, 'age', 27)

2、vm.$set(vm.userProfile, 'age', 27)

3、vm.userProfile = Object.assign({}, vm.userProfile, {

age: 27,

favoriteColor: 'Vue Green'

})

需要为已有对象赋予多个新属性，比如使用 Object.assign() 或 \_.extend()。

在这种情况下，你应该用两个对象的属性创建一个新的对象，不要采用如下：

Object.assign(vm.userProfile, {

age: 27,

favoriteColor: 'Vue Green'

})

同一节点上的v-for的优先级比v-if高

**事件修饰符**

<!-- 阻止单击事件继续传播 -->

<a v-on:click.stop="doThis"></a>

<!-- 提交事件不再重载页面 -->

<form v-on:submit.prevent="onSubmit"></form>

<!-- 修饰符可以串联 -->

<a v-on:click.stop.prevent="doThat"></a>

<!-- 只有修饰符 -->

<form v-on:submit.prevent></form>

<!-- 添加事件监听器时使用事件捕获模式 -->

<!-- 即元素自身触发的事件先在此处处理，然后才交由内部元素进行处理 -->

<div v-on:click.capture="doThis">...</div>

<!-- 只当在 event.target 是当前元素自身时触发处理函数 -->

<!-- 即事件不是从内部元素触发的 -->

<div v-on:click.self="doThat">...</div>

<!-- 点击事件将只会触发一次 -->

<a v-on:click.once="doThis"></a>

<!-- 只有在 `keyCode` 是 13 时调用 `vm.submit()` -->

<input v-on:keyup.13="submit">

<!-- 同上 -->

<input v-on:keyup.enter="submit">

<!-- 缩写语法 -->

<input @keyup.enter="submit">

全部的按键别名：

.enter

.tab

.delete (捕获“删除”和“退格”键)

.esc

.space

.up

.down

.left

.right