**Vue**

**常见的框架：**

1. 函数库：jquery和zepto
2. MVVM架构：m是module，v是view，vm是module和view

Angular：模板（套js），表达式，指令，双向绑定

React：组件化，虚拟DOM（局部刷新效率高）

Vue：结合上面两者

**关于Vue：**

1. 动态构建用户界面（动态获取数据展现知道界面）编码简洁, 体积小, 运行效率高, 移动/PC端开发
2. 遵循MVVM（三大框架都是这样）
3. 本身只关注UI界面（后面还会涉及到ajax路由等需要插件）
4. 与其他库的关联：

\* 借鉴angular的模板和数据绑定技术  
 \* 借鉴react的组件化和虚拟DOM技术

5. vue包含一系列的扩展插件(库):  
 \* vue-cli: vue脚手架  
 \* vue-resource(axios): ajax请求  
 \* vue-router: 路由(SPA)  
 \* vuex: 状态管理  
 \* vue-lazyload: 图片懒加载  
 \* vue-scroller: 页面滑动相关  
 \* mint-ui: 基于vue的组件库(移动端)  
 \* element-ui: 基于vue的组件库(PC端)

1. 渐进式：vue.js是特小的，可以实现基本的功能，后面要是要完成比较复杂的可以根据需求，添加第三方插件，这个过程叫做渐进式

**Vue的使用：**

**一、.声明式渲染小例子：**

<div id="app">

{{ message }}

</div>

1. **在body中**

<div id="app-3">  
 <p v-if="seen">现在你看到我了</p>  
</div>

1. **引入js**

<script type="text/javascript" src="../js/vue.js"></script>

1. **js编写**

var app3 = new Vue({  
 el: '#app-3',//元素  
 data: { seen: true } }//如果v-if="seen"是true就执行

)

<script type="text/javascript">  
 var app = new Vue({ **// 配置对象: 包含n个配置选项** el: '#app',  **// element** data: { **// 给页面模板使用**  
 message: 'Hello Vue!'  
 }  
 })  
</script>

**二、条件和列表**

**1.条件：**

**body：**

**js：**

**2.列表**

**body：**

**<div id="app-4">  
 <ol>  
 <li v-for="todo in todos">//类似于for in  
 {{ todo.text }}  
 </li>  
 </ol>  
</div>**

**js**

var app4 = new Vue({  
 el: '#app-4',  
 data: {  
 todos: [  
 { text: '学习 JavaScript' },  
 { text: '学习 Vue' },  
 { text: '整个牛项目' }  
 ]  
 }  
})

**三、处理用户输入（**methods:**）**

**在methods**里面添加事件

用**v-on:click="reverseMessage"**

**v-on:事件="该事件名称"**

例子：

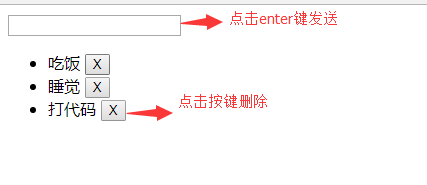
body：  
<div id="app-5">  
 <p>{{ message }}</p>  
 <button v-on:click="reverseMessage">逆转消息</button>  
</div>

js：

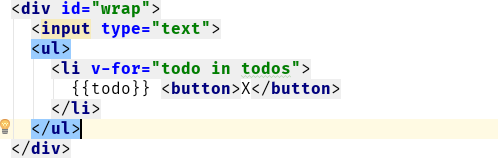
<script type="text/javascript">  
 var vm = new Vue({  
 el: '#app-5',  
 data: { message: 'Hello Vue.js!' },  
 methods: { reverseMessage: function () {}}}

小练习：

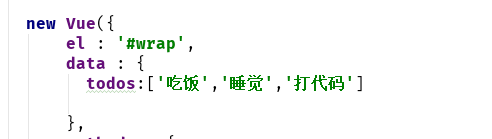
要求：完成如下的操作



1. 首先搭建界面，在li中遍历数组中的值



1. 初始化数据，在data中定义数组放要做的事



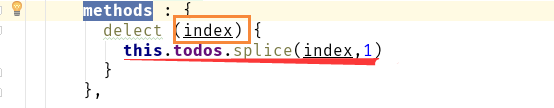
1. 先做删除键，在删除键里面绑定单击事件

1)<**button v-on:click="delect"**>X</**button**>

2)在里面传参数，把index一起传入

<**li v-for="(todo,index) in todos"**>

3)在methods中完成单击函数



1. 完成enter键添加TODO
2. 给input框添加数据绑定，并在data中初始化input框的值，这个绑定**v-model="value"，**会自动收集input框里面的值保存到value中



1. 给input添加按键事件

**<input type="text"v-model="value"v-on:keyup.enter="addTodo">**

**v-on:keyup.enter="addTodo"**

这是键盘enter起来的时候触发的事件，keyup后面是指定是哪个键盘，有关键值，也可以用keycode的数字代码代表键

5．在methods中完成单击函数，此时data中的value已经保存了最新的input的值，将它插入数组中就行了

**完整代码：**

**body：**

**js：**

<**script type="text/javascript"**>  
 **new Vue**({  
 **el** : **'#wrap'**,  
 **data** : {  
 **todos**:[**'吃饭'**,**'睡觉'**,**'打代码'**],  
 **value**:**''** },  
 **methods** : {  
 delect (index) {  
 **this**.**todos**.splice(index,1)  
 },  
 addTodo (){  
 **if**(!**this**.**value**){  
 alert(**'请输入你要添加的TODO'**)  
 **return**;  
 }  
 **this**.**todos**.unshift(**this**.**value**);  
 **this**.**value**=**''** }  
 },  
  
 **computed** : {  
  
 }  
 });  
</**script**>

<div id="wrap">  
 <input type="text" v-model="value" v-on:keyup.enter="addTodo">  
 <ul>  
 <li v-for="(todo,index) in todos">  
 {{todo}} <button v-on:click="delect">X</button>  
 </li>  
 </ul>  
</div>

**Vue模板语法**

<p>{{msg}}</p>//向页面中添加数据  
<p>{{msg.toUpperCase()}}</p>//在里面写js代码

1. **双大括号{{}}**

**2. 指令一: 强制数据绑定【：】得到html的标签属性，又希望是变量的时候就加‘冒号’**

**3. 绑定事件监听**

<a v-bind:href="url">百度一下</a> <br>  
<a :href="url">百度一下2</a> <br>

v-bind:href= :href

v-on:keyup.enter="showValue">

@keyup.enter="showValue2('myvalue')"

v-on=@

**Vue计算属性**

**1.计算属性**  
 在computed属性对象中定义计算属性的方法  
 在页面中使用{{方法名}}来显示计算的结果  
**2. 监视属性:**  
 通过通过vm对象的$watch()或watch配置来监视指定的属性  
 当属性变化时, 回调函数自动调用, 在函数内部进行计算  
**3. 计算属性高级:**  
 通过getter/setter实现对属性数据的显示和监视  
 计算属性存在缓存, 多次读取只执行一次getter计算

4.如果没有用到set，那么get可以省略，直接加函数

fullName3: {  
 get: function () { return 语句 }//get要有return  
 set: function (value) {}//这个要有参数，是监听的时候，发生变化就会触发  
}

用法：

**双向绑定：**

当获取属性值的回调，计算返回结果，改变一次调用一次函数，就像改变fullName的时候要其他两个也改变，就用这个get可以写当前fullName的值，set可以监听上面get的值有没有变化

也可以用用实例对象的方法来监听

vm.$watch('firstName', function (val) {

//这个回调函数是当前返回改变的值  
 this.fullName2 = val + ' ' + this.lastName  
})

双向绑定的方法：

用set和get

用实例对象的方法vm.$watch

**class和style绑定**

**1. 说明**  
 在应用界面中, 某个(些)元素的样式是变化的  
 class/style绑定就是专门用来实现动态样式效果的技术  
**2. class绑定: :class='xxx'**  
 表达式是字符串: 'classA'  
 表达式是对象: {classA:isA, classB: isB}  
 表达式是数组: ['classA', 'classB']  
**3. style绑定**  
 :style="{ color: activeColor, fontSize: fontSize + 'px' }"  
 其中activeColor/fontSize是data属性

**1. class绑定: :class='xxx'**

**2. style绑定**

p class="classB" :class="a">表达式是字符串: 'classA'</p>  
<p :class="{classA: isA, classB: isB}">表达式是对象: {classA:isA, classB: isB}</p>  
<p :class="['classA', 'classC']"> ['classA', 'classB']</p>

<**p :style="{color, fontSize}"**>style="{ color: activeColor, fontSize: fontSize + 'px' }"</**p**>

**data** : {  
 **a**: **'classA'**,  
 **isA**: **true**,  
 **isB**: **false**, **color**: **'red'**, **fontSize**: **'20px'** } **常用指令**

**1. 常用指令**  
 v-if  
 v-else  
 v-show  
**2. 比较v-if与v-show**  
 如果需要频繁切换 v-show 较好

因为v-show是修改样式的，而v-if是换dom的树  
**例子：**

**body：**

<**div id="app"**>  
 <**p v-if="ok"**>成功了</**p**>  
 <**p v-else**>失败了</**p**>  
  
 <**p v-show="ok"**>成功了22222</**p**>  
 <**p v-show="!ok"**>失败了22222</**p**>  
  
 <**button @click="ok=!ok"**>切换</**button**>  
</**div**>

**js：**

<**script type="text/javascript"**>  
 **new Vue**({  
 **el**: **"#app"**,  
 data () {  
 *// 先做处理, 再产生结果* **return** {  
 **ok**: **true** }  
 }  
 })  
</**script**>

列表显示

1. 列表显示  
    数组: v-for / index

**persons**: [  
 {**id**: 1, **name**: **'Tom'**, **age**: 18},  
 {**id**: 4, **name**: **'Jack'**, **age**: 17},  
 {**id**: 3, **name**: **'Bob'**, **age**: 19},  
 {**id**: 2, **name**: **'Lucy'**, **age**: 16},  
 {**id**: 5, **name**: **'Marry'**, **age**: 20}  
]

<**li v-for="(p, index) in persons" @key=index**>  
 {{index}}--{{p.id}}--{{p.name}}--{{p.age}}

</**li**>

对象: v-for / key  
<**ul**>  
 <**li v-for="(value, key) in persons[0]"**>  
 {{ key }} : {{ value }}  
 </**li**>  
</**ul**>

小案例

