**一、初识Vue**

1. 使用双大括号{{message}}将数据渲染进DOM中。



可以使用app.message对渲染后的数据进行修改。

1. Vue常用的指令：

1）v-bind 用来绑定常用属性

主要用法：直接在html标签上添加v-bind:属性名=”vue实例中的message”



2）v-if 条件

主要用法：直接在html标签上添加v-if=”vue实例中的某个属性名”，这个属性对应的属性值应该是布尔值



3）v-for 可以绑定数组的数据来渲染一个列表

主要用法：实例的data对象中添加一个对象，其中的一项的属性名如todos，它的值是个数据数组，在html中要循环的元素上添加v-for=”任意名如todo in todos”，然后通过双花括号取到todos中每一条数据的某个字段值{{todo.text}}



通过app4.todos.push({text: ‘新数据’})新增一项数据

4）v-on 绑定事件，并调用实例中的方法

主要用法：在vue实例中添加methods对象，里面定义方法如reverseMessage，在html的dom元素直接添加v-on:事件名=”该方法名”



注意：方法中的this指代vue实例

5）v-model 主要用于实现表单输入和应用状态之间的**双向数据绑定**

主要用法：常用于input元素，添加v-model=”实例中的数据如message”，同时在另个元素中使用双大括号加载该数据{{message}}，从而实现了input框的value值和另个元素数据的同步联动



1. 组件化 类似于**自定义元素***（demo07 & 24）*

简言之，如果html页面中某个元素需要重复多遍，就可以把它封装成一个组件加载。

如下例：

Vue实例的数据中有重复多个的数据json，若想在html页面实现它，就需要考虑下将每一条数据用组件来定义。

1）定义组件的方法：

Vue.component(‘组件元素名’, {

props: [‘todo’], // 选写。类似于自定义一个属性，属性名叫todo，主要作用是**在注册的组件和根实例之间传递数据**

template: ‘标签元素’ // 必写。决定将要采用哪个标签，待第2）完成后将循环后的每一个item数据中的文本加载进来，形成template: '<li>{{ todo.text }}</li>'

})

2）怎么在html中使用：

直接将上面的组件元素名当作标签使用，并使用v-for指令循环，v-bind指令绑定组件中定义的那个属性todo，它的值为v-for中的每一项如item，v-bind还可以绑定其他属性，如id.



1. **Vue实例**
2. 构造器

每个Vue.js应用都是通过构造函数Vue创建一个Vue的根实例启动的，如：

var vm = new Vue({ //选项 });

这些选项包括：https://cn.vuejs.org/v2/api/#选项-数据

1. el
2. data
3. methods
4. filters 过滤器
5. computed 计算属性

可以对vue实例进行扩展：



1. 属性与方法
2. 每个Vue实例都会代理其data对象里所有的属性，如下例，换句话说vm其实就相当于实例中的data，所以能按照json方式取到data中的key和value



1. 实例属性和方法 https://cn.vuejs.org/v2/api/#实例属性

都有前缀$，如



1. 生命周期
2. created 在实例被创建之后被调用
3. mounted updated
4. destroyed
5. **模板语法**
6. 使用javascript表达式



而且在这种表达式下访问不到用户自定义的变量，只能访问到全局变量的一个白名单，如Math和Date等。

**注意**：太复杂的表达式会使html模板冗余，所以一般写成**计算属性（见第四部分）**！

1. 修饰符 . 指出某些指令应该以特殊方式绑定 ps: demo中的用法好像不对！！！

1）.prevent 相当于event.preventDefault() 

1. 过滤器

1）只能在mustache双花括号和v-bind表达式中使用，右管道符“|”指示，用于文本转换，其中message相当于capitalize这个过滤方法的参数。



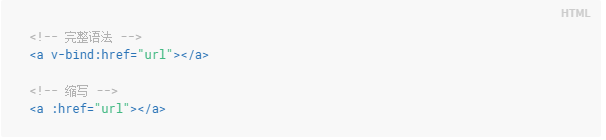
1. 过滤器可以串联



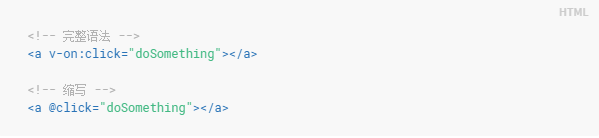
1. 过滤器可以传多个参数



1. 缩写

常用的两个：v-bind:href=”url” => :href=”url” 

v-on:click=”doSth” => @click=”doSth”



1. **计算属性**
2. 通过选项computed添加对象，可以有多个值即计算属性，但在一般需求下只添加一个就好了。

参见下例：

我们可以通过vm.reversedMessage获取到计算后的message，当vm.message发生变化时，vm.reversedMessage也会随之改变。



1. 与methods对比

计算属性基于自己的依赖进行缓存，只要message没发生变化，多次访问该计算属性就会立即返回之前的计算结果，而不必再次执行函数；

而methods调用总会执行该函数。

1. 与watch对比

都是用来处理某些数据随其他数据的变动而变动的，但尽量不要滥用watch，如果代码是命令式和重复的，先考虑下计算属性computed，可使代码更简洁。

**注意**：需要深入学习下watch选项。



1. 计算setter

计算属性默认只有 getter ，不过在需要时你也可以提供一个 setter ：

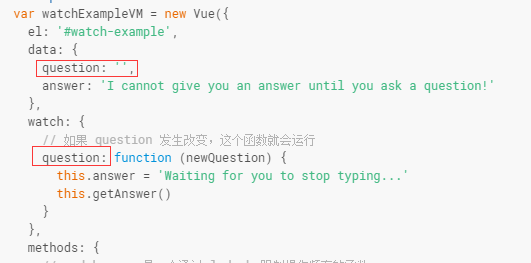


注：setter中的参数newValue即后面设置的vm.fullName—‘Jhon Doe’.

而且没有上面的get方法，下面设置的fullName根本就不生效。

1. 观察Wactchers（demo18）

提供一个watch选项，当想要在数据变化响应时，执行**异步操作或开销较大的操作**时使用。



注意：因为watch作为监听选项，它的某一项属性必须与data中的某一项相对应。

1. class与style绑定—绑定到行间，别忘了加引号
2. class绑定
3. 对象语法

有三种模式，第三种最常用且强大：

1. 直接在行间用v-bind:class或:class绑定一个对象，并在实例的data中设置其布尔值来动态控制。



1. 直接在行间绑定实例data中的一个对象：



1. 在行间绑定一个返回对象的计算属性：\*\*\*\*\*



1. 数组语法

同样有三种模式，其中数组与对象结合的模式最常用：

1. 直接在行间采用v-bind:class或:class绑定数组，数组中的每一项与实例的data中的key值相对应。



1. 用三元表达式进行条件切换。



1. 当有多个class需要进行条件判断时，采用上述三元表达式就显得繁琐，可采用数组与对象结合的方式。



该例中的isActive在实例的data中定义，其值为布尔值。

1. 用在组件上（关于组件化详见第一部分第3小节）

在一个定制的组件上用到class属性时，这些类将被添加到根元素上，这个元素上已经存在的类不会被覆盖。





1. 绑定内联样式style
2. 对象语法

与绑定class的对象语法类似，分为三种：直接绑定style、绑定一个对象、绑定一个返回对象的计算属性，当然，推荐使用后两种。

a）直接绑定style（不推荐）



b）绑定一个对象（让模板更清晰，推荐）



c）绑定一个返回对象的计算属性（推荐）



1. 数组语法

将多个样式**对象**应用到一个元素上：



1. 自动添加前缀（demo中未生效）
2. 多重值*（demo28）*

适用于2.3.0+，当绑定的style含有多个带前缀的值时使用。

