学习网址：<https://cn.vuejs.org/>

2016年12月之前是1.0的版本

2.0和1.0版本的区别：

（1）2.0版本删除了系统自带的过滤器

vue.js简介：

vue是是一套构建用户界面的**渐进式和只关注视图层的框架。**vue.js不支持IE8及以下版

vue.js特点：

1. 易用

基于css、html和javasctipt。简而言之就是好学、门槛低

1. 灵活

vue一样是渐进式的技术栈（由易变难），能够和任何第三方库进行结合不会造成相互的影响，只关注视图层（简而言之就是只帮我们做渲染），采用的是传统的DOM模版

1. 性能

vue.js文件足够小，压缩后大概20KB

vue.js的使用：

1. 引入vue.js文件
2. 下载到本地引入（建议使用）
3. 绝对路径引用

<script src="https://unpkg.com/vue"></script>

1. 创建vue应用。一个DOM模版、一个Vue实例、再在实例配置对象中传一个data属性，组件也是放到DOM处进行渲染。如下：



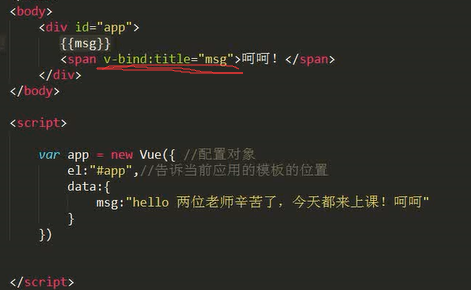
声明式渲染：如何在模版中使用对象中的数据

el：通过id把DOM模板和实例相关联

vue.js的指令：

vue.js的指令都带有前缀 v-，以表示它们是 Vue 提供的特殊属性，会在渲染的 DOM 上应用特殊的响应式行为（指令是用来帮我们做DOM操作的地方）

（1）v-bind：指令，将这个元素节点的 xxx属性和 Vue 实例的xxx属性保持一致”。（简而言之是将数据的变化映射到DOM的行为，想要使属性值是一个变量，就需要在属性名前面加v-bind前缀，如：v-bind:title=”message”）



上图解释：vue.js把data里面的属性全部挂载到实例app中，所以直接打点调用msg（app.msg）的属性。

（2）v-if条件指令（控制切换DOM节点显示与隐藏）

在DOM节点添加一个属性指令（v-if=”变量”），在实例data属性中添加此变量（变量:true），true表示显示，false表示隐藏。如下：



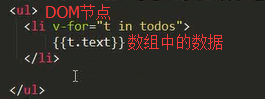
（3）v-for循环指令

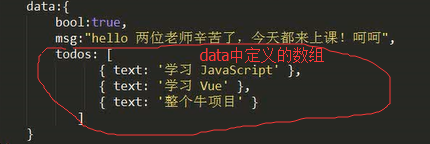
在data属性中定义一个数组todos→在DOM节点添加一个属性指令（if-for=”t in todos”）。如下：

t：表示数组的每一项

in：关键字

todos：所遍历的数组

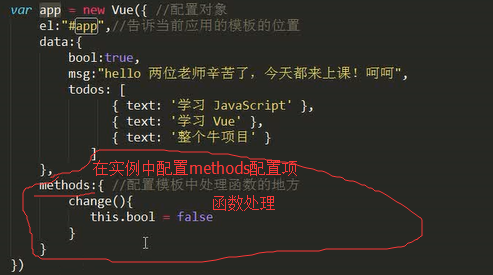




数据驱动：可以往todos中添加一组数据，app.todos.push(text:”学习”)，在页面中同样跟新显示数据，此时会显示4条相对应的数据

（4）v-on:绑定事件指令，处理用户输入

在DOM节点上绑定一个事件（v-on:click=”change”）→再在实例中配置methods配置项→在methods配置项中配置处理函数（methods配置项中是专门用来配置模板中处理函数的地方）

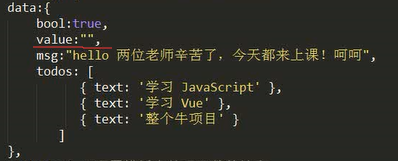


上图this指向的是当前实例。

（5）v-model指令，数据的双向绑定

在input节点上绑定指令（v-model=”value”）→在实例中的data配置项中配置此value属性（value:””），此时data中的value值就与input中的value值绑定。如下：

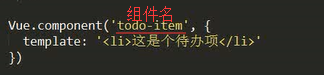




vue.js的组件构建：

最外层始终是一个实例，实例中告诉一个模板。模板中插入的全是组件。

组件的创建，如下：



vue周期函数

1. 使用方法将data配置项挂载到实例之前的beforeCreate函数
2. 实例创建好之后的created函数
3. 插入真实DOM节点之前的beforeMount函数
4. 渲染模板之后的mounted函数

1-4的周期函数是初始化的函数

1. 当有数据改变时，虚拟DOM重新渲染之前的beforeUpdate、updated函数
2. 实例或组件销毁时的beforeDestroy-destroyed函数，删除子组件和事件监听（app.$destroy()）

注意：带$的方法都是vue内部的方法

Vue模板的语法

1. 模板中插入文本，使用双大括号{{msg}}，可以在{{}}使用简单的js表达式，如+-\*/ 三目运算符
2. 纯HTML，使用 v-html 指令，.
3. 指令，指令就是对DOM进行操作

指令的参数，如事件指令v-on:后面所跟的事件类型（click、focus）就是指令的参数

指令的修饰符，如：.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()

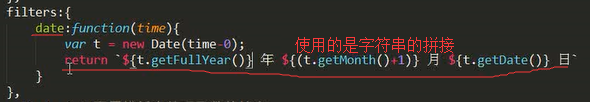


Vue的过滤器

当我们需要格式化显示文本的时候就要用过滤器（文本转换）

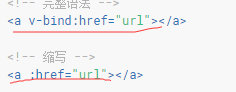
在data配置项中添加time属性（date:new Date().getTime()）→此时就需要格式化转换为我们常用的时间格式，在data配置项中创建filters属性，在次属性中创建一个data的过滤器，创建一个匿名函数（第一个参数是所操作的数据），如下：



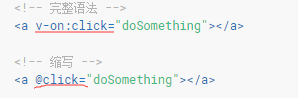


Vue的缩写

（1）v-bind属性的缩写



（2）v-on事件绑定的缩写



Vue的计算属性（重点）

计算属性出现的原因：由于在模板中使用太多逻辑会让代码变得难以维护，同时当你想监听多个数据变化的时候就需要使用计算属性

注意：计算属性计算好的值必须做return处理

使用：在实例中配置computed配置项computed:{}（computed是配置计算属性的），如下：

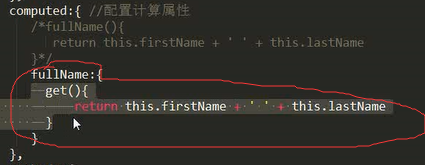


计算属性的特点：

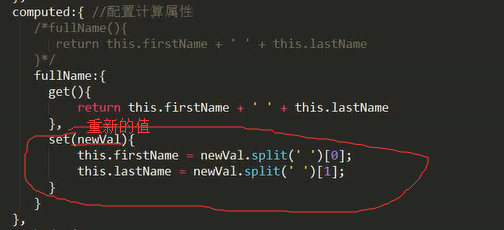
1. 会监听计算属性里面所依赖的任何属性（简而言之同时监听多个属性）
2. 计算属性会被缓存，当我们在DOM模板多次使用时不会再调用函数
3. 当监听依赖的属性发生变化的时候，计算属性的值会被重新计算并重新赋值

计算属性的get（取计算属性的值），如下：



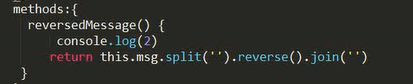


计算属性的set，当计算属性的值发生改变的时候执行一个函数，同时可以改变计算属性所依赖的数据值，如下：



函数调用的方法：



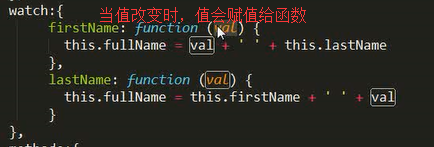


计算属性和函数调用的方法的区别：

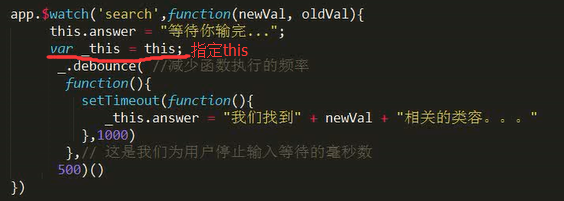
1. 计算属性只调用一次（缓存），函数调用的方法是调用多少次就是多少次（造成资源浪费）
2. 计算属性在数据改变时会从新更新，函数调用的方法在数据改变时无法从新更新

watch属性，当有异步操作的时候就需要用watch属性，首先需要引入lodash.js文件（lodash库是对象操作的拓展，是工具库），当input中的值改变的时候，对应的内容也会跟着改变。有两种方法，具体使用如下：

1. 在实例配置中配置watch属性，如下：



1. 用实例调$watch，首先需要引入，如下：



watch属性的缺点：一次只能监听一个数据的变化

watch属性的优点：可以在watch函数中设置中间状态，可以在watch函数中作异步加载