**正课:**

**1. 原生API vs 函数库 vs 框架**

**2. Vue概述**

**3. MVVM框架  
4. 数据绑定语法**

**5. 指令(directive)**

**1. 原生API vs 函数库 vs 框架**

**DOM jQuery Vue**

**原生API：浏览器/平台已经实现的，咱们可直接使用的API**

**比如: DOM BOM**

**优: 万能**

**缺: 繁琐**

**函数库(library)：基于原生API，进一步封装的，更简化的一组函数的集合。**

**比如: jQuery**

**优: 每个API都极简化**

**缺: 没有改变做事的方式。比如: 依然需要增删改查和事件绑定**

**框架(framework)：前人将多次成功的项目经验总结起来，形成的半成品项目。**

**优：无需重复劳动，后人只要继续开发新功能即可。**

**缺：彻底改变了做事的方法！需要适应**

**2. Vue概述:**

**什么是Vue.js: 一个渐进式的，基于MVVM设计模式的JavaScript框架。**

**渐进式(progressive)：虽然提供了很多功能，但不强迫全部使用。可逐渐使用，逐渐扩大使用范围。**

**MVVM设计模式: 待续**

**何时: 适用于以数据操作为主的项目**

**如何:**

**安装: 官网: https://cn.vuejs.org/**

**最新版本: 2.5**

**特点：不支持IE8及以下版本**

**IE8 无法模拟的 ECMAScript 5 特性**

**2种方式安装和使用:**

**1. 直接下载vue.js文件，引入网页**

**2个版本: 开发版——有错误提示**

**压缩版——去掉了错误提示**

**强调: vue是一个纯前端框架，所有不需要nodejs，apache等服务端软件就可使用。**

**2. 用VUE-CLI命令生成项目脚手架代码，并填充内容**

**3. MVVM设计模式:**

**传统网页组成: 3部分：**

**HTML：定义网页的内容**

**CSS：定义网页样式**

**JS：添加交互行为**

**问题: JS肩负的任务太多太杂了，就产生了大量重复的劳动**

**重新划分3部分:**

**View视图: 包含网页内容(HTML)和样式(CSS)**

**Model 模型数据: 指程序中创建的或从服务器端获取的数据**

**ViewModel 视图模型/控制器: 代替之前的DOM/JQUERY操作，自动把模型中的数据，绑定到View中的指定位置。**

**其实就是new Vue()**

**引入Vue.js时，等于在全局引入了一种新类型Vue**

**包含: 1. 构造函数: 创建该类型的子对象**

**2. 原型对象：包含该类型子对象共用的成员**

**new Vue() 创建Vue类型的一个示例，其实是创建一个用于同步Model和View的ViewModel对象**

**new Vue({**

**el:"选择器", //告诉Vue监听哪个父元素的内容**

**data:{ //作为Model，保存页面需要的所有数据**

**模型数据:值, 模型数据**

**}**

**})**

**new Vue()：2件事:(2大子系统)**

**1. 响应系统：**

**new Vue()将data中的每个模型数据提升为整个Vue对象的访问器属性。在访问器属性的set方法中，只要修改模型数据的值，就触发通知。通知Vue框架修改页面。**

**2. 虚拟DOM树：**

**什么是: 用js程序模拟的一棵简化版的DOM子树**

**为什么：真实DOM树包含的内容太多！极其不便于比较和遍历。**

**如何生成虚拟DOM树:**

**new View({el:"选择器"})会查找指定父元素，并遍历其中所有子内容。仅将可能发生变化的元素和属性用js对象结构集中存储在内存中。**

**何时使用虚拟DOM树:**

**当修改模型数据，触发响应系统时，响应系统通知Vue要修改DOM树。Vue框架就遍历虚拟DOM树，找到发生变化的节点，利用底层的DOM API仅修改发生变化的DOM元素**

**虚拟DOM树的优点:**

**1. 节点个数和属性个数都比真实DOM树少的多，极其便于遍历和比较**

**2. 仅修改受影响的DOM元素，不变的元素不用修改。**

**总结:**

**使用Vue框架步骤:**

**1. 定义HTML和CSS界面，只不过用特殊标识标记可能改变的位置**

**2. 在内存中定义模型数据对象data，其中，页面有几处改变，data中就要有几个模型变量与之对应。**

**3. 创建new Vue()示例对象，充当ViewModel**

**new Vue()中定义el:"选择器"，指向要监视的父元素**

**new Vue()中，还要将data对象包含进来作为Model**

**4. 数据绑定语法:**

**{{}}：双括号语法，外号大胡子语法(Mustache)**

**学名: interpolation 补缺**

**什么是数据绑定: 选择特定的Model变量，绑定到View试图中指定位置。使视图中指定位置的值，随Model变量同步变化。**

**何时: 只要希望用Model中的一个变量，填补View中的一个动态显示位置时。**

**如何: {{模型变量名}}**

**执行时，会用模型变量的当前值，代替{{}}位置**

**{{}}中可包含以下几种情况的内容:**

**1. {{模型变量名}} 比如: {{uname}}**

**2. {{运算}} : 算术/比较/逻辑/三目运算**

**比如: 小计:{{price\*count}}**

**3. {{方法调用}} 比如: {{uname.toUpperCase()}}**

**4. {{对象.属性}} 比如: {{address.city}}**

**5. {{数组[i]}}**

**注意！不能写if else for 等程序解构**

**问题: {{}}只能绑定innerHTML，不能绑定属性和事件**

**解决: 指令**

**5. 指令(directive)**

**什么是: Vue.js提供的，专门用于增强HTML功能的特殊属性**

**为什么: HTML本身是静态的，写死的，无法根据数据动态变化属性值。**

**何时: 只要希望动态绑定HTML属性时,都用指令**

**包括: 13个:**

**1. v-bind: 专门绑定属性值。**

**如何: <ANY v-bind:属性="模型变量">**

**强调: 不用加{{}}**

**简写: v-bind可省略，只写":属性="**

**2. v-for: 根据数组或集合内容，反复生成多个相同的HTML元素**

**何时: 只要根据数组数据反复生成多个相同的HTML元素**

**如何: <ANY v-for="(元素值,下标) in 数组|对象|字符串">**

**结果: 根据in后的数组/集合中元素的个数，反复生成当前HTML元素。**

**每次in都会取出数组/集合中当前元素值和位置下标**

**如果<ANY下还有子元素，则子元素可用for获得元素值和下标继续绑定内容。**

**强调: v-for必须放在要反复生成的HTML元素上，不能放在父元素上！**

**简写:**

**1. 如果不关心下标，只关心内容: v-for="变量 in 数组"**

**2. in可换为of，**

**3. v-if: 根据一个bool类型的模型变量值作为条件，控制是否生成该元素到DOM树（控制一个元素的显示和隐藏）**

**何时: 根据条件，控制一个元素的显示和隐藏**

**如何: <ANY v-if="bool类型的模型变量"**

**如何控制多个元素中选其一显示？**

**<ANY v-if="条件1">**

**<ANY v-else-if="条件2"**

**... ...**

**<ANY v-else>**

**依次判断每个条件，只要条件为true，就生成该元素，其余元素不生成(隐藏)**

**强调: v-if，v-else-if，v-else之间必须紧邻，不能插入其它元素。**

**4. v-show: 用法和v-if几乎完全一样！也可以控制一个元素的显示隐藏。**

**<ANY v-show="bool类型的模型变量">**

**鄙视: v-show vs v-if的区别:**

**v-show，不改变DOM树，通过display:none/block控制显示隐藏**

**优: 效率高！**

**缺: 只适合控制一个元素的显示隐藏。如果控制多个元素的显示隐藏时，代码会很繁琐**

**v-if，通过添加/删除DOM树上元素节点的方式控制显示隐藏**

**缺: 效率低！**

**优: 专门控制多个元素选其一显示，代码精简。**

**如果一个元素频繁显示隐藏切换，首选v-show**

**如果多个元素控制选其一显示，首选v-if, v-else-if, v-else**

**5. v-on: 专门为元素绑定事件处理函数**

**如何:**

**1. 在new Vue()实例对象内添加事件处理函数:**

**new Vue({**

**el:"选择器"**

**data: { 模型变量:值, ... ... }**

**methods:{**

**处理函数(参数列表){**

**//访问当前Vue内自己的模型变量: this.变量名**

**}**

**}**

**})**

**2. 在HTML元素上绑定事件处理函数:**

**<ANY v-on:事件名="处理函数()">**

**获得事件对象e:**

**<ANY v-on:事件名="处理函数($event)">**

**$event是Vue内置的事件对象，不能随便改名**

**简写:**

**1. v-on: 被@替代: <ANY @事件名="处理函数()"**

**2. 事件修饰符: 对常用事件操作的简写！**

**1. 取消冒泡: e.stopPropagation()**

**<ANY @事件名.stop="处理函数()">**

**2. 阻止默认行为: e.preventDefault();**

**<ANY @事件名.prevent="处理函数()">**

**3. 键盘事件:**

**<ANY @事件名.按键号="处理函数()">**