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

1.原生API: 浏览器/平台已经实现的，咱们可直接使用的原生对象和函数

比如: ES, DOM, BOM 优点：万能 缺点: 代码繁琐

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

优: 每步操作的API比原生API简单 例如：jQuery

缺: 并没有改变做事的方法，如jQuery，还是DOM，依然是增删改查，事件绑定。

3.框架(framework): 前人将多次成功项目的经验总结起来，形成的半成品项目

优: 后人继续开发即可，无需重复劳动。

缺: 需要改变原有的做事方法。

比如: vue中根本没有查找方法！数据一样能自动出现在想出现的位置。

2. Vue概述

1.官网: <http://cn.vuejs.org> 最新版本:2.5

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

什么是渐进式: 虽然提供了很多功能，但不强迫必须全部使用。可逐渐使用，逐渐扩大范围

vs AngularJS和React: 必须打包全套使用

什么是MVVM: 稍后讲

何时: 适用于以数据操作(增删改查)为主的项目(Web, APP)。

2.安装:

兼容性: 不支持IE8及以下版本

当前版本: 2.5

两种方式安装使用：

1. 直接下载vue.js文件，引入网页中——前三天

2个版本: 开发版 —— 有错误提示

压缩版 —— 没有错误提示

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

2. 使用Vue-CLI脚手架工具——后两天

3. MVVM设计模式:

传统网页组成: 3部分:

HTML: 定义网页的内容

CSS: 定义网页的样式

JS: 为网页添加交互行为

问题: JS DOM操作，既要加工数据，又要承担大量的反复的增删改查操作。

重新划分上述3部分：

View视图: 指网页中的元素和样式，包括HTML和css

Model模型数据: 指程序中创建的或从远程获取的数据（如ajax请求的数据，放在data中）

**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和视图中的HTML元素"绑定"在一起.

其实就是new Vue()

原理: 添加一种新类型——Vue:

包括: 2部分:

1. 构造函数——创建Vue类型的实例对象

2. 原型对象——保存所有Vue类型子对象共有的成员（API）

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

}

})

//2. ViewModel控制器

var vm=new Vue({

el:"#header",//element:找到并监视header

data:{//Model数据模型: 其实就是普通对象

//模型变量

uname :"dingding",

score:3000

}

})

//两大系统:

//1. 响应系统: 通知Vue有属性发生更改

//2. 虚拟DOM: 负责实际修改DOM树

什么是绑定: 不需要写一行代码，就可让页面元素内容和数据保持联动变化。

Vue类型，提供的就是一种强大的ViewModel对象，可自动同步数据和页面元素。

ViewModel的核心原理: 2部分:

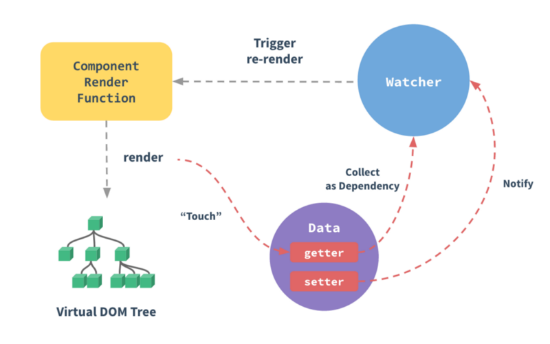
**1. 响应系统**(Reactivity System):

内存中数据变，页面自动变

原理:

（每个内存中的数据，都被Vue自动添加了访问器属性。）

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(Virtual DOM):**

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。

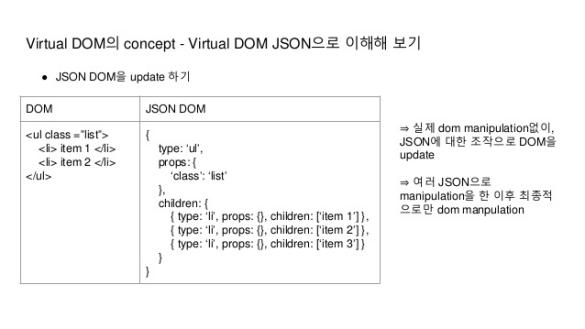
何时使用虚拟DOM树:

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

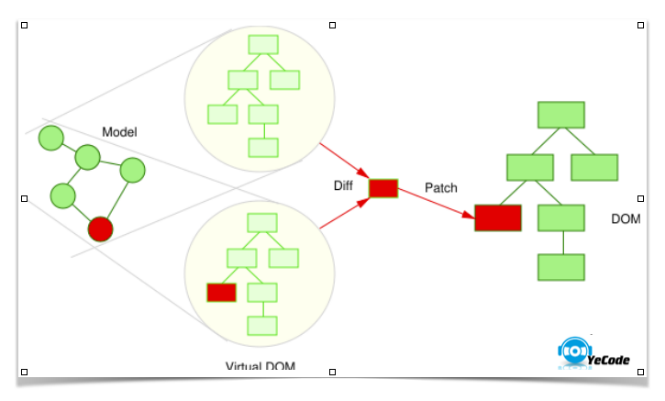
用js面向对象方式，模拟的DOM树。

保存了当前界面中所有DOM节点的描述信息，以及节点间关系。

优点1: 仅保存关注的少部分节点属性和内容，遍历快！



当响应系统通知Vue框架某个属性发生变化，要修改页面时，Vue框架会快速遍历虚拟DOM树，找出发生变化的DOM元素节点，仅修改发生变化的真实DOM树上的节点。没有发生变化的节点，保持不变。



优点2: 仅修改受影响的少量元素，效率高。——无需人工

**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

强调: 即使选择器找到多个父元素，vue也只能监视第一个找到的父元素及其子元素。

4. 数据绑定语法:

{{}}: 双括号语法，也叫大胡子语法(Mustache)

官方名字: interpolation 补缺

什么是: 可选择特定的Model中的数据，绑定到View视图中

何时: 只要希望将Model中的一个数据，显示到页面上，就在页面上用{{模型变量}}引入

如何: {{}}中可包含多种表达式:

1. 直接写{{模型变量名}}: {{uname}}

2. {{运算}}: 算术/比较/逻辑/三目运算:小计 {{price\*count}}

3. {{方法调用}}: {{uname.toUpperCase()}}

4. {{对象.属性}}: {{address.city}}

5. {{数组[i]}}: {{list[2]}}

注意: 不能写if,else,for等程序结构(不止一句话，so不行)

问题: 只能绑定元素的内容(innerHTML)，不能绑定属性和事件

解决:

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，没有任何动态生成内容的能力。

何时: 只要希望根据内存中的数据，动态绑定HTML元素的属性和样式时，都用指令。

包括: 看手册: 官网/学习/API

如何: v-\*, 凡是v-\*开头的自定义扩展属性，都是vue为增强HTML功能而提供的指令。

1. v-bind: 专门绑定属性值:

如何: <any v-bind:属性="模型变量"

强调: 绑定属性，不用{{}}！

简写: v-bind可省略，只保留:

<any :属性="模型变量"

v-bind

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: 只要反复生成多个相同的HTML元素时

如何: <any v-for="(元素,下标) in 数组/对象/字符串"

结果：根据in后的集合中的元素个数，反复生成<any元素

每次in都取出元素的值和下标，保存到in前的变量中

在<any元素及其子元素中，可使用元素和下标，子元素继续动态绑定内容

强调: v-for必须写在想要反复生成的元素上

ul

li v-for

简写: 1.如果只关心内容，不关心下标可省略()

v-for="变量 in 数组

<li v-for="task in tasks">

2.变化: 其实, in可用of代替。（效果一样）

<ul>

<li v-for="(task,i) in tasks">

<span>{{i+1}}</span>-<b>{{task}}</b>

</li>

</ul>

<script>

var vm=new Vue({

el:"ul",

data:{tasks:["吃饭","睡觉","打亮亮","跑步"]}

})

</script>

3. v-if: 根据一个bool数据作为条件，控制是否创建该元素到DOM树（控制显示隐藏）

何时: 根据条件，控制一个元素的显示隐藏时

如何: <any v-if="bool值的模型变量"true、false

<any v-if="bool">条件1

<any v-else-if="bool">条件2

<any v-else-if="bool" >

…..

<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。

**强调: v-if, v-else-if, v-else之间不允许插入任何其他元素**

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

v-if vs v-show:

v-if... 都是用添加删除DOM元素的方式，来控制显示隐藏

v-show: 不改变DOM的内容，通过display:none来控制显示隐藏

1. 如果只控制一个元素的显示隐藏，首选v-show

2. 如果控制多个元素选其一显示时，首选v-if

3. 如果频繁显示隐藏切换时，首选v-show，效率高

5. v-on: 为元素绑定事件处理函数

问题: 通常vue实例对象都是没名的，为了避免冲突和全局污染。所以，在普通事件处理函数中，不可能访问vue中的模型变量。

解决: v-on

何时: 今后只要在vue中绑定事件一律用v-on:

如何: 2步:

1. 添加事件处理函数:

new Vue({

el:...,

data:{ ... },

methods:{

处理函数(){

//this->当前vue实例对象

... this.模型变量 ...

}

}

})

2. 定义绑定: <any v-on:事件名="处理函数()"

简写:

1. v-on:事件名 => @事件名

被@替代: <ANY @事件名="处理函数()"

2. 如果没有参数，可省略() ——不建议

问题: 如果a当按钮, 会擅自修改url

解决: 2种:

1. e.preventDefault()

获取事件对象: 2步:

1. 在绑定时: @click="处理函数($event)"

其中: $event是Vue内置的对象，可直接使用

$event名字不能变

2. 在定义处理函数时: 处理函数(e){ ... }

2. 用事件修饰符代替事件函数:

e.preventDefault() => .prevent

e.stopPropagation() => .stop

如何: @事件名.修饰符="处理函数()"

提示: 如果想获得鼠标位置，就必须用$event

简写: 事件绑定:

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()">

比如: @keyup.13="处理函数()"

课后任务:

(1)复习: Vue绑定和事件指令

(2)作业: 完成课后练习:  
 题目要求:

1. 实现学子商城商品详情页信息加载

2. 实现学子商城页头查找功能事件绑定

运行效果:

提示:

(3)项目: 仿学子商城首页和商品详情页，用vue改造个人项目的首页和详情页数据动态绑定

题目要求:

运行效果: