Vue.js框架专题

1. 什么是Vue.js

|  |
| --- |
| Vue.js是一套构建用户界面的框架，它只关注视图层的内容，它是前端的主流框架之一。  前端三大主流框架：Angular.js、React.js、Vue.js  前端框架主要负责的是MVC中的V层，主要的工作就是和界面打交道，主要用来对页面中的数据进行处理，以及制作前端页面相关的特效及动画。 |

1. 为什么需要学习Vue.js

|  |
| --- |
| 在实际项目开发中，无论是做前端开发还是后端开发，使用框架技术是最佳的提高效率的方式。  另外使用Vue.js来做前端框架，对于处理数据的方面可以完全的替换掉原有的DOM操作的理念。（之前常用的手段为原生JavaScript以及jQuery类库操作DOM对象，通过Vue.js框架提供的指令，前端开发人员不再关心DOM对象是如何渲染的，可以有更多的时间去关注业务逻辑本身。） |

1. Vue.js版本说明

|  |
| --- |
| Vue.js当今应用最普遍的有两个版本，分别是Vue1.js和Vue2.js。版本2对于1版本来说进行了较大的改动。所以二者 从概念到语法上有着很大的区别，随着版本2的成熟以及市场的认可，当今市场中对于新项目的开发以及完全偏向于版本2。  版本3正在测试预备阶段。故版本2仍然是最主流的版本。 |

1. 对于前端MVC和MVVM的区别

|  |
| --- |
| 1. MVC思想      1. 前端MVVM思想   MVVM是前端视图层的分层开发思想，主要是用来将每一个页面分成M、V、VM这三种组件，这三个组件在前端开发中分工明确、分工协作、其中，VM是MVVM中的核心，是因为VM是M和V之间的调度者，我们所需要学习的Vue.js。主要就是学习通过Vue.js框架来创建VM这一组件，为M和V来进行服务。 |

1. Vue.js入门
2. 指令属性的基本使用

|  |
| --- |
| 1. v-cloak   使用v-cloak主要是为了解决插值表达式的闪烁问题  使用插值表达式的问题：  在页面加载的过程中，在页面中的{{}}插值表达式首先会被页面认为是html元素中的实际存在的内容。所以浏览器会首先将{{}}展现在页面上，页面载在完毕后，插值表达式{{}}才会真正转变为动态赋予的值。这就会造成一种现象，如果将来终端在访问服务器的时候，由于网络延迟，那么我们的{{}}会首先展现出来，{{}}展现出来之后，会一闪而过，最终显示的是最终被赋予的值。这就是前端开发中所谓的，插值表达式闪烁问题。  解决办法： |

|  |
| --- |
| 1. v-text 2. v-html     区别：   * 对于元素中已经存在的值，只有插值表达式**能够将原有的值保留**，在原有的存在的值的基础上添加动态数据。   **使用v-text和v-html之所以不能够保留元素标签中原有的内容，是因为在使用以上两个指令属性之前，会提前将标签对中的内容先清空**，在赋予动态的值。如果未来实际项目开发，需求为在原有的内容的基础上，追加动态的值，要选择使用插值表达式。  从另一个方面来看，插值表达式虽然会出现页面闪烁现象。但是对于原有内容的保留只有插值表达式能够完成。所以插值表达式具有不可替代的优势。       * v-text和v-html的对比：   v-text主要用来赋予纯内容的，如果赋予到元素中的内容本身包含html，那么v-text会将html原封不动的展现在页面上，这些内容是得不到浏览器的解析。  v-html：除了能够为前端元素赋予内容之外，更重要的是，如果内容本身包含html代码，那么赋值后最终能够展现出来的结果，html元素会得到浏览器的解析。 |

|  |
| --- |
| 1. v-bind指令属性       对于v-bind在开发中一共有如下几种使用方式：   * 直接使用指令属性v-bind来为元素中的属性进行绑定操作 * 使用简化后的方式将v-bind去除，直接使用：来对元素中的属性进行绑定。在实际项目开发中对于前端元素的绑定是常规操作，v-bind的复用率非常高，所以Vue为v-bind指令属性提供了简写的方式（只针对于v-bind），直接使用：即可。   【注意】：  **使用v-bind对元素中的属性进行绑定的时候，可以直接在操作值的位置进行内容的拼接。** |

|  |
| --- |
| 1. v-on指令属性触发事件      * v-on的基本使用   v-on:click=“fun1”的形式来绑定事件  相当于原生js中的onclick=“fun1()”来绑定事件处理函数。  【v-on的简化形式】：**@**   * 事件修饰符的使用：   stop：使用.stop来对是事件的冒泡机制进行阻止。  js中的事件的冒泡指的是，在触发了内层元素的同时，也会随之继续触发外层元素（外层元素包裹了内层元素），在做点击的过程中，点击了内层元素，认为同时点击了外层元素，所以两个事件都会触发。在实际项目开发中，几乎没有太多需求会使用到这种事件冒泡机制，所以需要阻止事件的冒泡。  prevent：使用.prevent来阻止超链接默认的行为（禁用超链接）。默认行为指的就是超链接中的href属性链接。  原生js：<a href=”javascript:void(0)”></a>  Vue框架：使用.prevent    capture：  使用.capture实现捕获触发事件的机制。实例中是使用外层div套用内层div。内层div没有阻止事件冒泡，在此基础上，点击内层的div，先触发内层div绑定的事件，再触发外层div绑定的事件。当进行.capture机制处理后，点击内层div，会先触发外层div绑定的事件，再触发内层div绑定的事件。  self：  self实现阻止事件冒泡行为的机制。    once  使用.once修饰符，只触发一次事件处理函数 一般结合.prevent来使用。  使用场景：需要同时为@click事件使用多个修饰符。例如我们现在就需要同时使用.once和.prevent修饰符。在使用的时候，通过绑定事件的@click连续的绑定修饰符就可以了。@click.once.prevent |

|  |
| --- |
| 6） 使用v-model指令实现双向数据绑定  v-bind ：单向绑定的概念：    结论：v-bind只能实现数据的单向绑定，从M绑定到V，使用VM将数据去渲染视图，无法通过该形式实现数据的双向绑定。  双向数据绑定：  所谓的双向数据绑定即使，不仅可以通过M取得数据渲染V视图，还可以通过V视图中的变化动态的更新M中的值。 最佳案例：通过M中的数据渲染标题，又可以通过文本框中改变视图中的数据，动态修改M中的数据，进而通过M中值的动态修改又可渲染标题。 |