Vuejs

第一章 Vuejs 概述

1-1 什么是 Vuejs

Vuejs 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 3 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 的这一层 ,主要的工作就是和界面打交道，主要是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。

1-2 为什么要学习 Vuejs

在实际项目开发中，不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 Vuejs 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（之前我们常用的原生 js 以及 jquery 类库），通过 Vuejs 框架提供的指令，前端开发人员不再关心 dom 是

如何渲染的，可以有更多的时间去关注业务逻辑本身。

Vuejs 作为现今最火爆的前端开发框架，学习后可以有效的提高就业的价值。

1-4 vuejs 版本 1 和 2 的区别

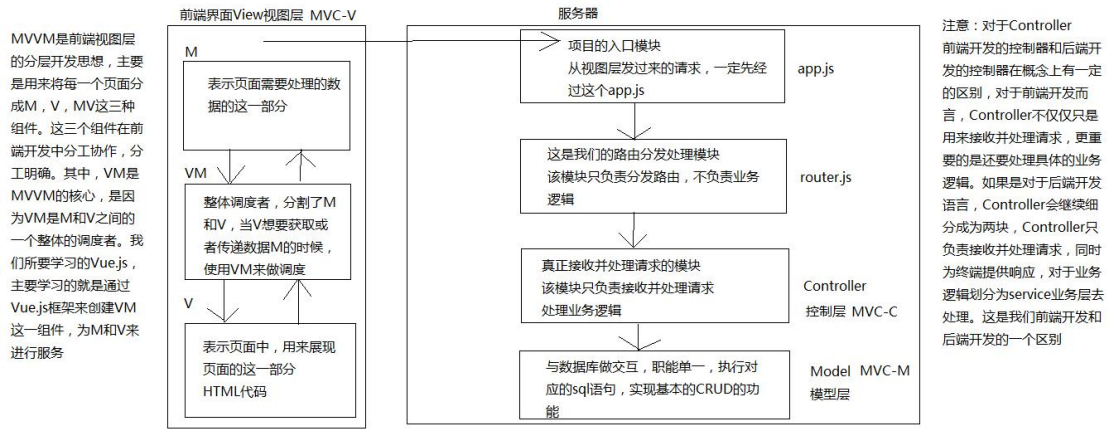
Vuejs 当今应用最普遍的有两个版本，分别是 Vue1.js 以及 Vue2.js。

值得说明的是版本 2 在版本 1 的基础上进行了大量的改动，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，当前市场中对于新项目的开发已经完全偏向于使用版本 2.

教程：版本 2　Vue2.js

另外版本 3 现今是处于测试及预备阶段，所以现在仍然是以 Vue2.js 为最主流的版本。

1-5 MVC 和 MVVM 的区别



第二章 Vuejs 基础语法

2-1 第一个 Vuejs 案例_HelloWorld

2-2 通过基础案例解析 MVVM

2-3 指令属性的基本使用

（1）v-cloak

使用 v-cloak 主要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可为是 html 元素中的实在存在的内容。所以浏览器会首先将{{}}展现在页面上，页面加载完毕后，插值表达式{{}}才会真正的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过程中，网速较慢（网络比较卡），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，插值表达式的闪烁问题。

（2）v-text

（3）v-html

（4）插值表达式与 v-text/v-html 的区别

a.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，在原有的已经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对中原有的内容，是因为在使用以上两个指定属性之前，会提前将标签对中的内容先清空掉，在赋予动态的值。如果未来的实际项目开发，需求为在原有的内容的基础上，追加动态的值，我们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{{}}页面闪烁的现象（v-text 和 v-html 不会出现页面闪烁的情况），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。

对比 v-text 和 v-html

v-text:主要是用来赋予纯内容的，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的。

v-html:除了能够为前端元素赋予内容之外，更重要的是，如果内容本身含有 html 代码，那么赋值后最终展现出来的结果，html 元素会得到浏览器的解析的。

从以上结果判断，v-html 的功能要更加强大，对于前端页面的展

现，不可能让使用该系统的用户看到任何 html 代码，而是应该让用户看到解析后的 html 效果。所以在实际项目开发中，使用 v-html 的概率要高于 v-text。

另外使用插值表达式的效果，与 v-text 是一样的，内容中的 html 代码时得不到浏览器的解析的。

（5）v-bind

v-vind:是 Vuejs 中，提供用于绑定属性的指令

对于 v-bind 在开发中一共有如下几种使用方式

a.直接使用指令属性 v-bind 来为元素中的属性进行绑定操作

b.使用简化后的方式，将 v-bind 去除，直接使用:来对元素中的额属性进行绑定。

因为在实际项目开发中，对于前端元素的绑定是我们的常规操作，v-bind 的使用复用率非常高，所以每一次都直接写 v-bind 会很麻烦，所以 vuejs 为 v-bind 指令属性提供了简写的方式，直接使用:即可。

v-bind:title --> :title

注意：这种简写形式仅仅只是针对于我们的 v-bind，以上所讲解的其他的指令是不具备简写形式的。

在实际项目开发中，我们都是使用这种简写的形式。

c.在使用 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。

2-4 使用 v-on 指令触发事件

（1）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

v:bind-->:

v-on-->@

（2）事件修饰符的使用

a.stop

使用.stop 来对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，也可以认为是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的机制，所以我们需要阻止事件的冒泡。阻止事件冒泡之后的效果是，在我们点击了内层元素之后，内层元素绑定的事件触发，触发完毕后，由于事件冒泡被阻止，就不会继续触发外层元素的事件了。

b.prevent

使用.prevent 来阻止超链接默认的行为

所谓默认的行为指的是超链接中的 href 属性链接

在实际项目开发中，不仅仅只是按钮需要我们绑定事件来控制行为，超链接的使用我们也是要遵循这种自己绑定事件触发行为的方式。所以在 a 标签中的 href 链接往往要被我们以特殊的方式处理掉。

c.capture

使用.capture 实现捕获触发事件的机制

使用的是外层 div 套用内存 div 的例子（其中内层 div 没有选择阻止冒泡），在此基础之上，点击内层的 div，先触发内层 div 的事件，再触发外层 div 的事件。

加入了.capture 修饰符之后，先触发的是外层的 div 事件，后触发的是内层 div 事件。被.capture 修改之后，事件优先触发。

d.self

self 实现阻止事件冒泡行为的机制（之前我们讲过了一个.stop）

使用.self 实现的是阻止自身冒泡的行为（它不会真正的阻止冒泡）

案例：外层 div，内层 div，在内层 div 中加入了 button

测试 1：在内层 div 加入.stop 修改符

观察实验结果，点击最内层的按钮，触发按钮事件，触发了内层的 div，外层 div 没有触发，也就是说，使用.stop 修改符，不会阻止掉自身事件的触发，在自身事件触发完毕之后，阻止事件继续向外扩散。

测试：在内层 div 加入.self 修改符

观察实验结果，点击按钮，触发按钮事件，没有触发内层 div 事件，跨过了内层 div 事件的触发后，继续触发了外层 div 的事件。

表示使用.self 只是将当前元素（测试中的内层 div）的冒泡行为阻止掉了，但是不会影响冒泡行为继续向外扩散（测试中的外层 div 继续触发了）。

e.once

使用.once 修饰符，只触发一次事件处理函数

.once 需要结合.prevent 来使用

在我们使用事件修饰符的时候，有可能会存在这样的需求，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。

语法： @click.prevent.once

测试：通过超链接的触发来观察

在只使用@click 的时候（没有做任何修饰符的修饰），点击超链接，先触发@click，后触发 href 链接。

当为@click 添加上@click.prevent.once 之后，点击超链接。

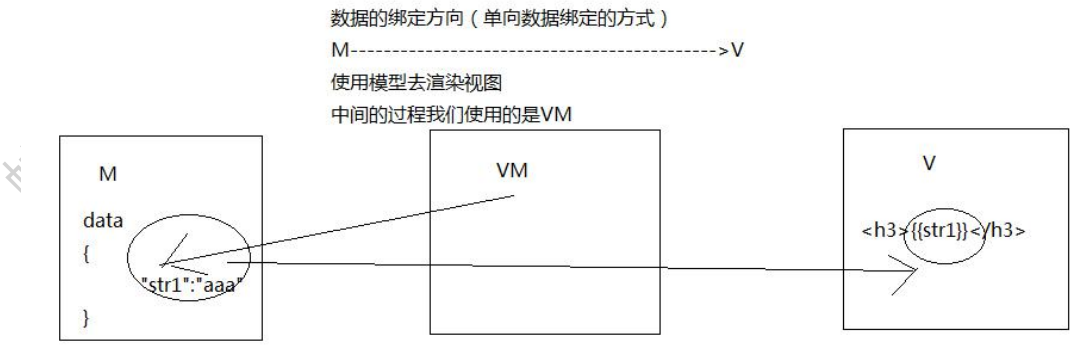
第一次点击：只触发了@click 的事件，href 链接没有触发

第二次点击（已经点击过一次之后再次点击）：没有触发@click，只触发了 href。

2-4 使用 v-model 指令实现双向数据绑定

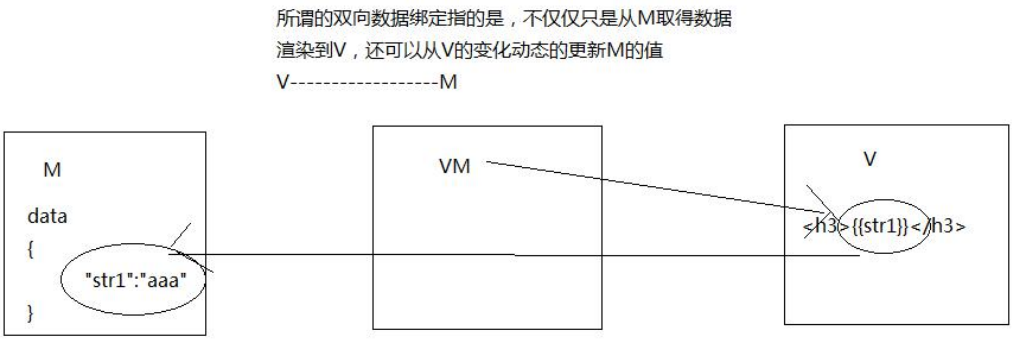
v-bind

单向数据绑定



我们可以对 v-bind 的绑定数据的形式得出一个结论，v-bind 只能实现数据的单向绑定，从模型（M）绑定到视图（V），使用 VM 将数据去渲染视图，但是我们无法通过该形式实现数据的双向绑定。

双向数据绑定



在实际项目开发中，不仅仅只是将模型准确的渲染到视图中，视图中的数据的变化，更需要模型去进行有效的记录。所以我们需要使用双向数据绑定的机制。

（1）v-model 的基本使用

使用 v-model 可以对数据进行双向的绑定操作。

值得注意的是，由于记载页面元素值的需求都是发生在表单元素中，所以 v-model 只能运用在表单元素中。

form

<input>系列 type:text,hidden,checkbox,radio....

<select>

<textarea>

...

（2）使用 v-model 实现简单的计算器功能

1

2-5 使用 class 样式

使用 vuejs 控制样式，会使样式变化的操作更加的灵活

（1）class 样式的使用

案例 1：传递一个 class 样式的数组，通过 v-bind 做样式的绑定

该形式与之前的形式没有太大的区别

:class="[样式 1,样式 2]"

案例 2：通过三目（元）运算符操作以上数组

boolean?true 执行 :false 执行

案例 3：使用对象（json） 来表达以上三目（元）运算符的操作

{样式:flag}

案例 4：以对象引用样式

:class={}

案例 5：通过以直接模型 M 的形式做样式渲染

注意：这样使用必须直接将具体的 boolean 值结果（true/false）赋值，不能以 this.模型的形式来做引用

（2）style 样式补充

在实际项目开发中，对于 style 样式的使用，没有 class 使用的广泛，通常 style 属性仅仅只是对个别指定元素的样式进行的进一步补充使用。

案例 1：引用样式对象

:style="引用样式对象"

注意：在写 color 这样的样式属性的时候，由于仅仅只是一个单词，所以不需要加引号，开发中的大多数的样式都是包含横杠（-）的，例如：font-style，background-color 等等，在使用这样带有-的演示属性的时候，必须套用在引号中。

'font-style'

'background-color'

color

案例 2：引用样式对象数组

:style="[样式对象引用 1,样式对象引用 2]"

2-5 v-for 指令和 v-if 指令的应用

（1）v-for 指令的基本使用

案例 1：遍历字符串数组

案例 2：遍历对象数组

案例 3：遍历对象的属性和属性值

案例 4：遍历整型数字

（2）v-for 指令使用注意事项

对于 key 属性的使用案例

key 属性的作用：

在实际项目开发中，使用 v-for 仅仅只是将元素遍历出来，并展现在页面中。如果在将来的其他需求中，使用到指定的遍历出来的某个元素，那么视图并没有为我们提供一个有效的辨识指定元素的方式。

所以在遍历元素的过程中，需要为每一条遍历出来的元素做一个有效的标识，这个标识就是该元素在页面中的唯一标识，在将来使用到该元素的时候，页面可以通过该标识认准该元素。在 v-for 的使用过程中，添加 key 属性及属性值就是做这个标识的最好的手段。

所以我们需要养成一个在遍历元素的过程中，为 key 属性赋值的好习惯。

对于 key 属性的应用，值得注意的是：

注意 1：key 属性值必须是一个数值或者字符串，对象不能当做属性

值。否则系统会提出警告，将来不能有效的应用。

注意 2：key 属性的应用，必须要搭配绑定 v-bind 指令，在使用的时候必须是以该形式":key"来使用，否则系统不会生效。

注意 3：对 key 属性所赋的值，必须是记录的唯一标识，我们通常使用的是记录的 id。

（3）v-if 指令和 v-show 指令的使用和区别

案例 1：v-if 的使用

案例 2：v-show 的使用

简单的比较 v-if 指令和 v-show 指令，效果是一模一样的

点击浏览器（F12）中的查看器观察显示页面元素信息

如果 flag 为 true，观察到的结果是一致的

如果 flag 为 false，观察到的结果是不同的

其中 v-if 中的元素是本身就没有的，v-show 中的元素是存在的，只是通过 false 属性值，系统为该元素新增了 display:none，表示不展现该元素

通过观察得到结论：

v-if 为 true：创建条件元素 为 false：去除该元素

v-show 为 true：展现条件元素 为 false：隐藏该元素

在实际项目开发中，一般的需求情况下，我们使用 v-if 就可以了。

但是如果对于元素的展现与否需要频繁的切换 ,我们需要使用 v-show 的形式展现或者隐藏元素 ,因为 v-if 在每次切换为 true 的时候都需要重新的创建元素，降低页面展现元素的效率。

综合应用：

完成一个基本信息系统（学生信息管理系统）的查询列表操作，可对该列表进行添加操作，可进行删除操作。