正课：

1. 指令

2. 双向绑定

3. 绑定样式

一.指令:

1. 绑定多个相同结构的有组元素: v-for

(1). <要反复生成元素 v-for="(value, i) of 数组" :key="i">

a. value会获得当前正在遍历的数组元素值

b. i会获得当前正在遍历的下标位置

c. 问题：如果没有:key，则每个重复生成的元素是无差别的。如果将来数组中某一项改变了，v-for就要被迫将所有元素重新生成一遍。——效率低

d. 解决: 只要使用v-for都要同时绑定:key="i"，其实就是给每个重复生成的元素一个唯一的标识。在数组中某一个元素更新时，只需要根据更新元素的下标，找到对应key的元素，只更新这一个元素就可以。

(2). of会遍历data中指定数组中每个元素，每遍历一个元素就创建一个当前HTML元素的副本。且of前的(value, i)两个变量可在当前HTML内用于绑定语法。

|  |
| --- |
| 易错: v-for必须写在要重复生成的元素上，而不是写在父元素上。比如: <ul>下要生成多个<li>，则v-for一定要写在<li>上。如果写在<ul>上是反复生成多个<ul>的意思 |

|  |
| --- |
| 坑: 任何时候修改vue中数组某一个元素的值绝不能用下标。因为数组下标不受vue监控。今后只要想修改数组中一个元素的值，必须用数组函数操作。因为数组函数的执行是受vue监控的。  比如: this.tasks[1]="跑步" 界面是不会跟着变的  应该: this.tasks.splice(1,1,"跑步") 界面会跟随自动变化 |

(3). v-for可以遍历：

a. 所有数字下标的数据: 数组，类数组对象，字符串 ——同js中的for of。

b. v-for还会数数:

<元素 v-for="i of 10" :key="i">

结果: of后的数字是几，就反复创建几个HTML元素。of前的i可依次获得一个递增的数字。从1开始。1 2 3 4 5 6 7 8 9 10

c. v-for 还可遍历对象属性:

<元素 v-for="(value,key) of 对象" :key="key">

结果：对象中有几个属性，就会反复创建几个HTML元素。其中，value获得的是属性值，key获得的是属性名

2. 事件绑定:

(1). <元素 v-on:事件名="js语句或一个函数名">

(2). 其实可以简写:

<元素 @事件名=" js语句或一个函数名">

(3). 事件处理函数，按规定必须写在new Vue()中的methods属性对象内：

new Vue({

el:"#app",

data:{ ... },

methods:{

处理函数(形参){

//如果用到data中的变量，必须加this.

}

}

})

结果：无论包含在data中的变量，还是包含在methods中的的方法，new Vue()时都会被打散，并直接隶属于new Vue()。所以方法中才能用 this.变量 方式访问data中的变量

|  |
| --- |
| 绑定时是否加()：  如果需要向处理函数中传入自定义参数，就要加()，()里还能写实参值或变量  <元素 @事件名="处理函数(实参值或变量)"  比如:  <button @click="change(-1)">-</button>  <button @click="change(+1)">+</button>  ...  methods:{  change(i){ ... }  }  如果不需要向处理函数中传入自定义参数，就可省略()  <元素 @事件名="处理函数" |

(4). 事件对象:

a. 如果绑定时，不需要传入自定义参数，则在省略()情况下，事件对象始终默认作为第一个参数传入处理函数——同DOM

<元素 @click="doit">

methods:{

doit(e){ ... }

}

|  |
| --- |
| 坑: 一旦擅自加上()，自己又没有传入参数，则vue不会再自动传入事件对象e。形参e就是undefined  结论: 只要不需要传参时，都省略() |

b. 如果既要传入自定义参数值，又要传入事件对象:

<元素 @click="doit($event, 自定义实参值)">

methods:{ ↓ ↓

doit( e, 形参){ ... }

}

|  |
| --- |
| $event：是vue内置的专门封装DOM事件对象e的一个关键词。不能改名。使用$event等于使用DOM事件对象e |

3. 代替{{}}绑定HTML片段：

(1). {{}}的问题：如果要绑定的变量值是一段HTML片段，{{}}会将HTML片段原样显示，不会编译为页面元素。

(2). 用 v-html代替{{}}绑定HTML片段

4. 避免看到{{}}:

(1). 暂时隐藏元素: 2步:

a. 在要暂时隐藏的元素上添加v-cloak属性

<要隐藏的元素 v-cloak>

其中: v-cloak不需要属性值

但是v-cloak本身不带任何隐身属性

b. 程序员需要在当前页面中手动添加[v-cloak]{ display:none }，通过属性选择器找到所有带有v-cloak属性的元素，暂时隐藏他们。

原理: 每当new Vue()加载完成后，会自动查找带有v-cloak属性的元素，移除上边的v-cloak属性，该元素就显示出来了。

(2). 用v-text代替{{}}：

<元素 v-text="变量或js表达式">

结果：v-text就是代替{{}}，将变量或表达式的值放入元素的内容中。但是，因为v-text是属性，没有直接写在元素内容中，所以，即使new Vue没有加载完，用户也不会看到元素的属性中的内容。

5. 只在页面首次加载时，绑定一次。之后，即使变量值改变，页面也不跟着变: v-once。

原理: 首次加载时，会绑定页面中的内容。而当首次加载后，凡是标有v-once的元素都被从虚拟DOM树中移除。不再受vue的监控。

6. 保持内容中的{{}}不被vue编译

(1). 刚巧元素内容中有{{}}，不想被vue编译，只想原样显示时。就用v-pre.

(2). <元素 v-pre> ... {{xxx}} ... </元素>

二. 双向绑定:

1. 什么是单向绑定: 只能将模型对象中的数据，绑定到页面显示。无法将页面上的更改，反向更新回模型对象中。（M->V）

2. 什么是双向绑定: 既能将模型对象中的数据，绑定到页面显示。又能将页面上的修改，反向更新回模型对象中。（M->V, V->M）

3. 何时：只要绑定表单元素的值或状态时，才用双向绑定。

4. 如何：v-model:属性="变量"

5. 结果：只要页面上表单元素的值或状态改变，就会自动将表单元素中新的内容，更新回模型对象里绑定的变量上。

6. 原理: 凡是用v-model: 绑定的表单元素，都会自动添加DOM事件绑定:

比如: <input type="text" v-model:value="keywords">

会被翻译为：

<input type="text" oninput="function(){ vm.keywords=当前文本框的value }"

7. Vue中实现只要变量值变化，就自动执行操作:

(1). 其实，放入data中的每个变量，都被添加了一个监听函数

(2). 在监听函数中所写的代码，会在每次变量值改变时自动触发

(3). 如何使用变量的监听函数:

new Vue({

el:"#app",

data:{ ... },

methods:{ ... },

watch:{

要监视的变量名(){

//这里的代码，只要同名的变量值发生改变时，就自动执行。

}

}

})

(4). methods vs watch

a. methods中的方法都是自定义名称的。不调用不执行。

b. watch中的方法的名称必须和要监视的变量保持一致。且只要变量值变化，就自动执行。

补: 键盘按键事件:

keydown: 键盘按键按下。问题: 只要按下不抬起，就会反复执行。所以，为了防止反复操作，通常使用keyup代替keydown

keyup: 键盘按键抬起。好处: 只要不抬起，就不执行操作！

如何获得按的是哪个键:

DOM中: e.keyCode 获得每个按键的按键号

回车：13号

左, 上，右，下，: 37, 38, 39, 40

vue中，如果限制只有按回车时，才能触发事件：

@keydown.13="处理函数"

↑

事件修饰符

8. 不同表单元素的双向绑定:

(1). 绑定文本框和文本域元素:

<input type="text" v-model:value="变量">

<textarea v-mode:value="变量"></textarea>

结果: 元素内容改变，元素的value属性就改变，v-model就会把新的value值更新回data中的变量上。

(2). 绑定select:

a. 特点: v-model:value="变量"写在父元素select上

而每个<option>上都有一个备选的value等待比较

b. 比如: <select v-model:value="变量">

<option value="值1"></option>

<option value="值2"></option>

<option value="值3"></option>

c. 结果:

1). 加载数据到页面时(M->V)：用绑定的变量值和每个option的value比较，哪个option的value等于变量的值，就选中哪个option

2). 改变选中项时(V->M): 自动将新选中项的value值，更新回data中对应变量中保存。

(3). 绑定单选按钮:

a. 特点: value是在每个radio上提前写死的，所以v-model不能绑定在value属性上。

每次选择时，其实改变的是checked属性，所以v-model应该绑定在checked属性上。

b. 如何:

<input type="radio" value="1" v-model:checked="sex">男

<input type="radio" value="0" v-model:checked="sex">女

c. 原理:

1). 加载数据到页面时(M->V): 用变量的值和value值做比较，如果变量值等于value值，就选中，否则，就不选中

2). 如果选中状态发生改变，会将当前选中的radio的value值作为新值更新到绑定变量中保存。

(4). 绑定复选框: 单用一个checkbox

a. 特点：选中与不选中修改的只是checked属性。且checked属性是bool类型，所以绑定的变量也应该是bool类型

b. 如何:

<input type="checkbox" v-model:checked="isAgreen">同意

c. 原理:

1). 将数据显示到页面上(M->V): 如果绑定的变量值是true，则当前checkbox是选中的。否则，不选中。

2). 当切换选中与不选中时: v-model会将当前是否选中的状态bool值，更新回data中绑定的变量上。

9. 简写: 其实只需要写v-model即可，不用写:属性。v-model会自动根据不同的表单元素，切换不同的绑定方式。——信任

三. 绑定样式:

1. 绑定内联样式:

|  |
| --- |
| 何时: 如果希望精细控制某个css属性的变化时，就要修改内联样式——同DOM |

(1). 不好的方法: 将style属性当做一个普通的字符串绑定

<元素 :style="变量">

new Vue({

el:"#app",

data:{ 变量:"left:100px; top:50px" }

})

结果: <元素 style="left:100px; top:50px ">

问题: 不便于单独修改一个css属性的值

(2). 好的方法: 用对象语法:

<元素 :style="变量">

new Vue({

el:"#app",

data:{

变量:{

left:"100px",

top:"50px"

}

}

})

结果: vue会自动将对象翻译为css语法字符串

<元素 style="left:100px; top:50px ">

好处: 可以非常容易的仅修改一个css属性，而不影响其他css属性： this.变量.left="150px"

(3). 如果同时包含不变的style和可能变化的css属性:

其实，一个元素中不带冒号的普通style和带:动态绑定的style可以并存。运行结果，两个style中的样式会合为一个style字符串

2. 绑定class:

|  |
| --- |
| 绑定:  1. 绑定元素内容: {{}}  2. 绑定元素属性值: :  3. 一个元素控制显示隐藏: v-show  4. 两个元素二选一显示隐藏: v-if v-else  5. 多个元素多选一显示隐藏: v-else-if  6. 反复生成结构相同的一组元素: v-for  7. 只要绑定事件: @  8. 要绑定的内容是HTML片段: v-html  9. 防止用户短暂看到{{}}：v-cloak 或 v-text  10. 只在首次加载时绑定一次: v-once  11. 防止内容中的{{}}被vue编译: v-pre  12. 只要绑定表单元素的值或状态: v-model |

作业：

1. xz\_public项目中，首页的一楼后三个商品以及详情页的规格列表

2. 去掉8,9,10三个例子v-model后的属性，检查是否运行正常。