正课:

1. 指令:

2. 双向绑定:

3. 绑定class和style

4. 计算属性:

5. 自定义指令——了解

6. 过滤器:

1. 指令:

1**. v-html:** 使用模型数据替换当前元素的innerHTML

何时: 要用HTML代码片段替换内容

2. **v-text:** 使用模型数据替换当前元素的innerText

何时: 要用纯文本内容代替元素内容

{{}} vs v-text vs v-html

1. {{}}可和其它内容拼接形成新的内容

v-text和v-html只能完整替换内容

2. 如果绑定纯文本内容，则首选v-text，原样显示，不转换

如果绑定HTML片段，则首选v-html，会被浏览器转换为正文，不保持原样。

<p v-html ="`￥${price.toFixed(2)}`"></p>

<p v-text="`￥${price.toFixed(2)}`"></p>

<p >￥{{price.toFixed(2)}}</p>

3. **v-cloak:** 可让未加载完的元素，暂时不显示

问题: 当vue加载慢时，页面会暂时显示{{}}内容

解决: 如果vue没有加载完，则元素不显示！

如何: 2步:

1. 为vue监视的元素指定v-cloak属性

2. 手动在css中为v-cloak指定隐藏样式:

[v-cloak]{ display:none }

原理: 当vue加载完成后，会自动移除v-cloak属性！

总结: 今后被vue监视的父元素都要用v-cloak默认隐藏，等vue加载好后，才显示出来。

4. v-pre: 保留当前元素中的{{}}，不进行绑定渲染

何时: 如果正文中有{{}}，不想被vue渲染，可用v-pre保护

<li v-pre>Vue使用{{模型变量}}语法绑定元素的内容</li>

5. v-once: 设置一个{{}}，仅在首次加载时渲染一次，之后不再被监视。

何时: 如果希望一个元素，只在首次加载时绑定一次

原理:

其实在Vue中有一个巨大的数组，保存着所有绑定的元素

每次模型数据有修改时，都通过遍历数组找到绑定的元素，修改元素。

v-once可在第一次加载后，将元素从监视数组中移除！之后遍历，无法遍历到该元素，则值不会被改变！

<li v-once>开始时间: {{now}}</li>

data:{ now:new Date().toLocaleString()},

2. 双向绑定:

什么是: 即可把Model数据绑定到页面的元素中——M => V

又可把元素的内容修改，反向更新到Model中——V =>M

为什么: {{}}和v-bind，仅是单向绑定，只能将Model中的数据绑定到页面显示。无法将页面上的更改，反向更新到Model中

何时: 专门用于绑定表单元素

如何: <表单元素 v-model:value="模型变量"

可简写为: <表单元素 v-model="模型变量"

总结: 表单元素的双向绑定:

1. 文本框，文本域text, textarea: v-model="模型变量"

2. radio: name="分组名" value="写死值" v-model="模型变量"

性别：<label ><input type="radio"name="gender"value="1"v-model="gender">男</label>

当用户单击一个radio时，用选中的value赋值给v-model指定的模型变量。

3. select: <select v-model="模型变量"

<option value="写死的值"

...

当用户选择一个option时，option的value会给select，select将选中的value反向更新到模型变量上

4. checkbox: <input type="checkbox" v-model="模型变量"

默认: v-model绑定的是checked属性，值为bool类型

自定义监视函数:

什么是: 模型变量发生变化时，自动执行的函数

何时: 希望在每次模型变量发生变化时，都执行一项操作

如何:

new Vue({

el:"#app",

data:{ ... },

methods:{ ... },

watch:{

模型变量名(){ ... this.模型变量 ...}

}

})

3. 绑定class和style属性:

2种:

方式1: 把class和style作为普通字符串绑定

方式2: 把class和style作为对象绑定:

data:{

xxxclass:{

class1:true,

class2:false,

... : ...

}

}

绑定时: <any class="固定不变的class" :class="data中的xxxclass变量"

style的绑定:

data:{

xxxStyle:{

css属性名:值,

... : ...

}

}

绑定时: <any style="固定不变的属性" :style="data中的xxxStyle变量"

4. 计算属性:

什么是: 不实际存储属性值，而是根据其他数据属性的值，动态计算获得。

为什么: 有些属性的值，不能直接获取到，需要经过其他属性计算后，才能获得。

比如: 订单总价=每个商品的数量\*每个商品的单价，再累加求和。

FullName=FirstName+ · +LastName

何时: 今后，只要一个属性的值，无法直接获得，依赖于其他属性的值，就用计算属性临时计算获得。

如何:

1. 定义计算属性:

new Vue({

el: "selector",

data:{ ... },

methods:{ ... },

watch:{ ... },

computed:{

属性名(){ return 属性值 }

}

})

2. 绑定: 同普通属性绑定完全一样！

{{属性名}} 强调: 不加()!

vs methods:

计算属性的值可被vue缓存！只要以来的其他属性不变，则始终使用缓存中的值，避免重复计算

methods不会被缓存。每调用一次，就重复执行一次。

5. 自定义指令——了解:

何时: 如果除13种Vue指令外，还想为HTML元素添加更多的功能

如何:

1. 创建自定义指令:

Vue.directive("自定义指令名",{

inserted(el){ //当元素被加载到DOM树时触发

//el: 写有当前指令的当前DOM元素

//对el执行DOM操作

}

})

强调: 定义指令名时, "自定义指令名"不加v-前缀

在HTML中使用时，要加v-前缀！

2. 在HTML元素上使用自定义指令:

<any v-指令名

强调: 必须加v-

6. 过滤器(filter):

什么是: 在接收到原始数据后，执行再加工

强调: vue官方没有提供任何预定义的过滤器，都要自己定义。

何时: 有些数据，需要经过加工后才能给用户看

比如: ms->日期, 1,0->"男","女", 钱->¥xx.00

10,20,30->订单状态名

如何: 2步:

1. 创建自定义过滤器:

Vue.filter("过滤器名", function(val){

//val收到传入的原始值

return 修改后的新值

})

2. 在绑定时使用过滤器:

2种:

1. {{ 模型变量 | 过滤器名 | ... }}

2. <any :属性="模型变量 | 过滤器名 | ..."

多个过滤器可连续使用，但后一个过滤器时基于前一个过滤器的加工结果继续加工。

带参过滤器:

1. 定义时:

Vue.filter("过滤器名", function(val, 其它参数...){

return ...

})

2. 使用时:

{{模型变量|过滤器(参数值)}}