正课:

1. 自定义指令—
2. 过滤器(filter):
3. 组件:
4. Axios—了解directive inserted
5. Vue.js实例/组件生命周期——重点原理
6. 组件化开发——重点,难点

1. 自定义指令——了解: directive

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

}

})

<script>

Vue.directive("focus",{

inserted(elem){

elem.focus();

}

})

</script>

<div id="app">

<input type="text"v-focus><button >百度一下</button>

</div>

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-

2. 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。

比如: 性别(1,0->男，女)

钱(345.5-> ¥345.50)

时间(ms->字符串格式)

状态(10,20,30,40->未付款，已付款，已发货，已签收)

如何:

Vue官方没有提供任何预定义的过滤器，都要自己定义

1. 创建过滤器：

Vue.filter("过滤器名",function(value,形参1,形参2,...){

//value接收的是原始值

return 加工后的新值！

})

2. 使用过滤器:

只要绑定数据，都可用过滤器：

模型变量|过滤器名|......

模型变量|过滤器名(值1,值2)|......

**强调:**

1. 多个过滤器可连续使用后一个过滤器获得的原始值，是前一个过滤器加工后的

2. v-text不能使用过滤器

<style>

[v-cloak]{display: none}

</style>

</head>

<body>

<script>

Vue.filter("genderConvert",function(val,lang="zh-cn"){

if(lang=="zh-cn")//过滤器可以加载参数

return val==1?"男":"女";

else

return val==1?"Male":"Female"

})

Vue.filter("genderIcon",function(val){

return val+(val=="男"||val=="male"?"♂":"♀")

})

</script>

<div id="app"v-cloak>

性别：<span>{{gender|genderConvert("en-us")|genderIcon}}</span>

</div>

<script>

new Vue({ el:"#app",data:{ gender:1}})

</script>

3. 组件(Component):

什么是: 用有专属HTML，CSS，数据的，可重用的独立页面区域

为什么: 重用！松耦合！便于团队协作和分工

何时: 今后，所有页面都是由多个组件组合而成。

凡是网页中，重要的，可重用的区域都要先封装为组件，再反复使用组件。

如何:

1. 创建一个组件:

Vue.component("组件名",{

template:`<div> html+css

html代码片段  
 </div>`,//强调: 组件模板必须有一个唯一的父元素包裹

data:function(){ //为每个组件实例创建专属的data

return {

模型变量

}

},

methods:{},

watch:{},

computed:{}

})

2. 在html视图中反复使用组件

组件其实就是一个可重用的标签: <组件名></组件名>

3. 也需要new Vue() 来监视页面

组件只能用在被Vue监视的父元素中

问题: HTML和js又掺杂在一起，不便于维护

解决: 用html5: <template>

<template>是默认不显示的元素

专门给框架使用，用于临时隐藏一段代码

何时: 今后组件的HTML片段都要先放在<template>中隐藏

如何: 2步:

1. 在<body>中用<template>封装组件的HTML片段

<template id="tplXXX">

<div>

html片段  
 </div>

</template>

强调: 1. <template>不能放在被Vue监视的父元素内

2. <template>中也必须使用唯一父元素包裹

2. 在Vue.component中用template:"选择器"找到<template>

Vue.component("组件名",{

template:"#tplXXX",

... ...

})

<div id="bpp">

<ul>

<li><xz-counter></xz-counter> </li>

<li><xz-counter></xz-counter> </li>

<li><xz-counter></xz-counter> </li>

<li><xz-counter></xz-counter> </li>

</ul>

</div>

<!--1.定义组件<template>元素，包含组件的HTML片段隐藏-->

<template id="tplXzCounter"><!--组件的独立视图-->

<div>

<button @click="change(-1)">-</button><span v-text="n">1</span><button @click="change(+1)">+</button>

</div>

</template>

<script>

//2.定义组件，找到template：

Vue.component("xz-counter",{

template:"#tplXzCounter",//选择器找到template

data:function(){//每次都function return 一次

return{n:1}

},

methods:{

change(i){this.n+=i;

if(this.count<1) this.count=1

}

}

})

new Vue({ el:"#app"})

new Vue({ el:"#bpp"})

4. Axios:

什么是: 一个基于promise的HTTP函数库

可在任意平台发起http请求

何时: 今后在没有jquery的情况下，想要发起http/ajax请求，都用axios

如何：axios是一个单独的js文件，与平台无关。

即可用在浏览器中，也可运行在node.js中

为什么:

发送ajax请求: 4种:

1. 自己封装ajax函数——麻烦

2. jQuery的$.ajax——必须引入jQuery，大材小用

3. 官方曾经提供了VueResource插件——官方废弃

4. axios库，官方推荐的代替VueResource的——本身与jQuery或Vue没有关系

如何操作:在浏览器中：

1. 下载并引入axios.min.js文件，辅助下载并引入qs.min.js。

2. 调用axios函数，发起异步请求

get请求:

var res=await axios.get("url",{

params:{

//随url发送的参数

}

}).then(function(res){

//res不是直接的响应结果

//res.data才是！

})

await和then选其一即可。

post请求:

先引入qs.min.js

axios.post(

var res=await axios.post("url",Qs.stringify({

请求参数1:值,

... : ...

"url","查询字符串" //uname=dingding&upwd=123456

).then(function(res){

//你要的数据: res.data

})

await和then选其一

var vm=new Vue({

el:"#app",

data:{ res:{} },

mounted(){

(async ()=>{

var res=await axios.get("http://localhost:3000/products",{

params:{

kwords:"i5",

pno:0

}

})//.then((res)=>{ ... })

this.res=res.data;

})()

/\*axios.post("http://localhost:3000/users/signin",Qs.stringify({

uname:"dingding",

upwd:"123456"

})).then(res=>{

console.log(res);

})\*/

5. Vue.js实例/组件生命周期 鄙视题

生命周期: 从创建到销毁的过程

Vue.js实例/组件生命周期:

四个阶段:

1. 创建(create)阶段:

new Vue实例或组件对象创建完成

但尚未挂载DOM树

$el: undefined 没有虚拟DOM树

data: { ... }

2. 挂载(mount)阶段:

组件加载DOM树完成

$el: DOM元素

data: { ... }

3. 更新(update)阶段: 组件所绑定的model数据发生变化

4. 销毁(destroy)阶段: 调用$destroy()方法，组件被销毁。

为了在每个阶段，触发执行一项任务，Vue为每个阶段定义了生命周期钩子函数

8个钩子函数:

1. beforeCreate(){}

2. created(){} ——可ajax请求获取数据，不可DOM操作

3. beforeMount(){}

4. mounted(){} ——可ajax请求，又可DOM操作

5. beforeUpdate(){}

6. updated(){}

7. beforeDestroy(){}

8. destroyed(){}

6. 组件化开发: Component重点难点

组件化开发: ——重点&难点

将来拿到网页，先划分组件。

组件化开发，就是将大的网页划分为若干组件区域，每个组件区域都有专门的HTML，CSS和数据。

自定义组件，分为3种:

1. 根组件:

new Vue({

el:"父元素选择器",

data:{ 模型变量 }

})

2. 全局组件：可用在页面任何位置的组件

Vue.component("组件名", {

template:"template元素的选择器",

data:function(){

return { 模型变量 }

}

})

3. 局部组件: 子组件: 只能用于特定的父组件内

何时: 只要规定一个组件只能出现在指定的父组件内

如何:

1. 将全局组件降级:

var 子组件对象={

template:"template元素的选择器",

data:function(){

return { 模型变量 }

}

}

2. 在父组件中，添加components属性，包含子组件对象，并为子组件起名！

var 父组件={

... ...

components:{

"子组件名":子组件对象,

子组件名: 子组件对象

//子组件名和子组件对象名都要用驼峰命名

//如果子组件名和子组件对象名相同，可只写一个

//Vue会自动将驼峰命名的子组件名，转化为横线分割: todoAdd自动转为<todo-add>; todoList 自动转为 <todo-list>

}

})

一个页面，都是由多块区域，多个组件

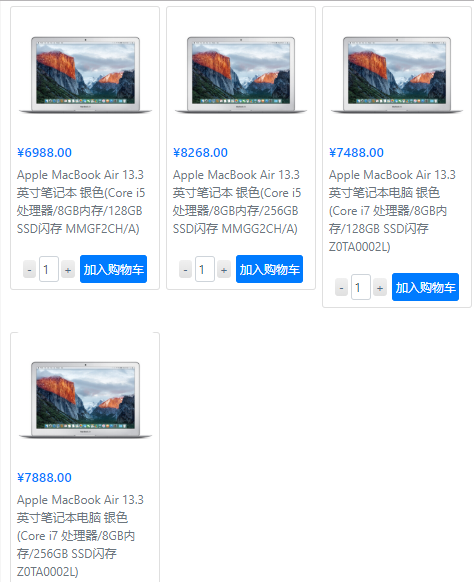
课后任务:

(1)复习: Vue组件化开发

(2)作业: 完成课后练习:  
 题目要求:

使用Vue实现学子商场商品列表页面的绑定

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: