提前准备:

1. 今天的历届师兄师姐问过的经典的高频的问题清单都已经整理在小程序->首页->VUE->day03里

2. 今天的课程中所需的单词列表都已经整理在小程序->单词->VUE->day03里

3. 今天的重点视频、扩展高频鄙视题视频已经整理在小程序->在线-> VUE ->day03里

如果不知道问什么: 谁问的越多！谁培训费花的越值！

1. 笔记中或画图中那句话或者那个位置不明白，立刻问！

2. 案例中哪行代码或注释不明白，立刻问！

3. 课下自己做项目，功能不会做，立刻问！

4. 刷笔试题时遇到不会的笔试题，立刻问！——公众号: 前端大全 定期总结大厂笔试题

5. 看小程序问题清单时，遇到看不懂的问题，立刻问！

正课:

1. 过滤器

2. axios

3. 组件生命周期

4. 组件化开发  
5. SPA

一. 过滤器:

1. 什么是过滤器: 对变量的原始值进行加工后再显示出来的一种特殊函数

2. 为什么: 有些值不能直接给人看，必须加工后才能给人看。

比如: sex:1 时间ms

3. 何时: 今后只要一个变量的原始值必须加工后才能给人看时，都要定义过滤器

4. 如何定义:

Vue.filter("过滤器名",function(oldVal){//要求1: 必须有一个形参，来接住变量的原始值

//要求2: 必须返回加工后的新值

return 新值

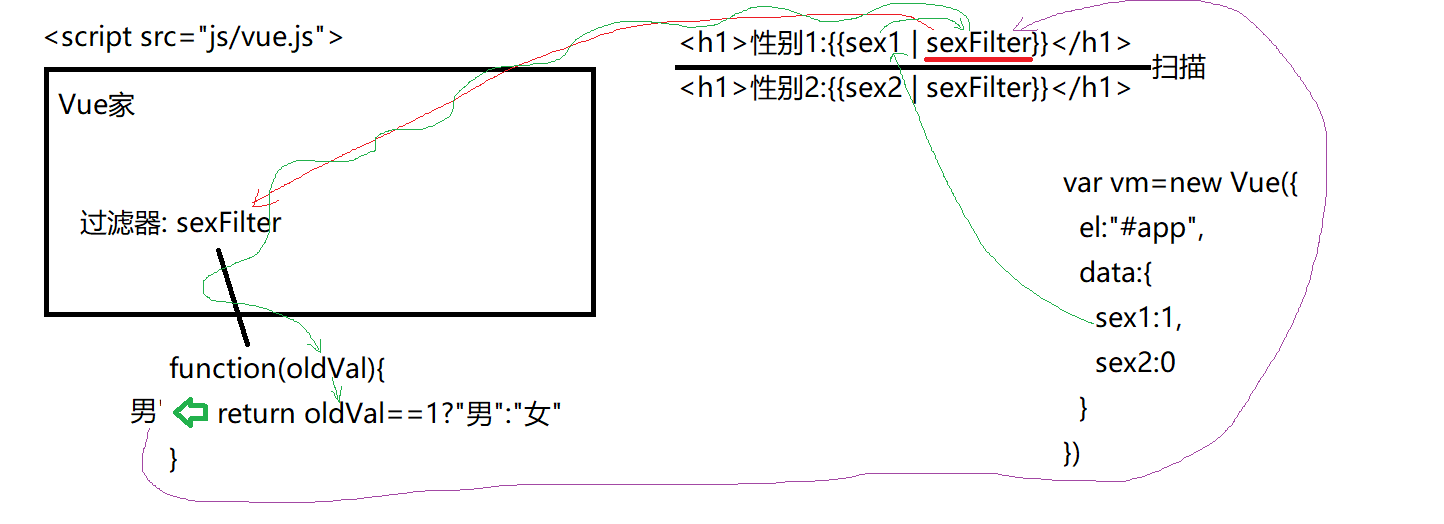
})

5. 使用:

<元素>{{ 变量 | 过滤器}}</元素>

连通

6. 原理: 变量的原始值先传给过滤器的形参oldVal，在过滤器内部加工后，返回新值，显示在绑定的位置上给人看。



7. 过滤器可以传参:

(1). 定义时:

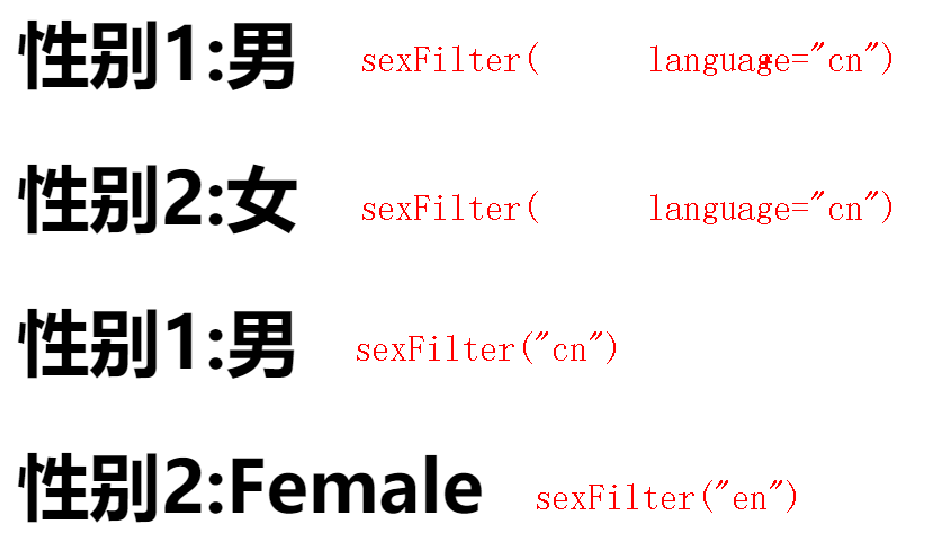
Vue.filter("过滤器名",function(oldVal, 自定义形参,...){

return 加工后的新值

})

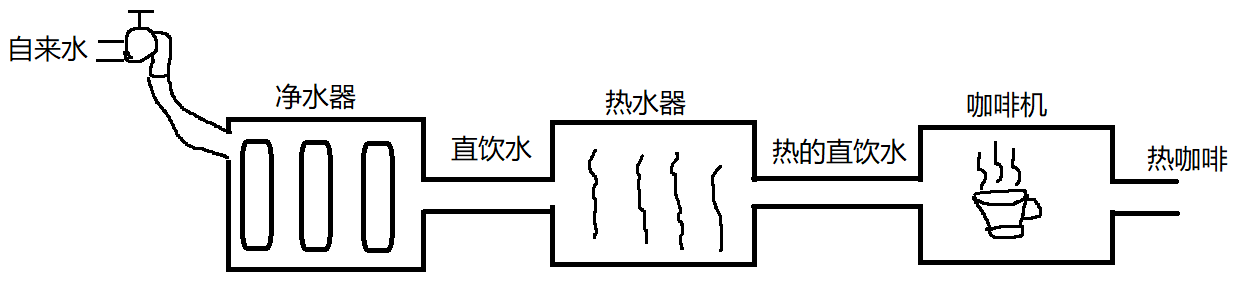
(2). 使用时:

{{变量 | 过滤器名(实参值, ...)}}



8. 过滤器还可连用:

强调: 后一个过滤器获得的值已经不是原始值了，而是前一个过滤器加工后的中间产物



9. 示例: 使用过滤器加工性别的1和0

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <meta http-equiv="X-UA-Compatible" content="ie=edge">    <title>Document</title>    <script src="js/vue.js"></script>    <script>                                //语言:en  cn      Vue.filter("sexFilter",function(oldVal,language="cn"){        if(language=="cn"){          return oldVal==1?"男":"女"        }else{          return oldVal==1?"Male":"Female"        }      })      Vue.filter("sexIcon",function(oldVal){        if(oldVal==1){          return "♂";        }else if(oldVal==0){          return "♀";        }else if(oldVal=="男"||oldVal=="Male"){          return oldVal+" ♂";        }else{          return oldVal+" ♀";        }      })    </script>  </head>  <body>  <div id="app">    <h1>性别1:{{sex1 | sexIcon}}</h1>    <h1>性别2:{{sex2 | sexIcon}}</h1>    <h1>性别1:{{sex1 | sexFilter | sexIcon}}</h1>    <h1>性别2:{{sex2 | sexFilter | sexIcon}}</h1>    <h1>性别1:{{sex1 | sexFilter("cn") | sexIcon}}</h1>    <h1>性别2:{{sex2 | sexFilter("en") | sexIcon}}</h1>  </div>  <script>  var vm=new Vue({    el:"#app",    data:{      sex1:1,      sex2:0    }  })  </script>  </body> |

二. axios:

1. 什么是: 专门在各种平台发送ajax请求的基于Promise的函数库

(1). 各种平台: axios不仅可用于vue中，而且还可用于网页中、nodejs服务端都可用

(2). 发送ajax请求: axios只专注于做好发送ajax请求这一件事

(3). 基于Promise: axios请求回来自动触发的回调函数要放在Promise的.then()中，不再使用回调函数的方式。

2. 为什么:

(1). 今后只要发送ajax请求，都只用一句话就够了！再不会写xhr4步或6步

(2). jQuery虽然有$.ajax()很好用，但是$.ajax这一个函数是混在整个jQuery巨大的函数库内的，没有独立出来。

(3). Vue中几乎不用执行DOM操作，自然也就用不到jQuery

(4). 如果只以为$.ajax()一个函数就要把整个jQuery全部函数库引入，有点儿得不偿失，小题大做！

(5). 所以才需要第三方的专门发送ajax请求的独立的函数库

3. 何时: 今后在任何平台(框架、网页、服务器端nodejs...)发送ajax请求，都可用axios函数库

4. 如何:

(0). 引入axios.min.js: <script src="js/axios.min.js">

(1). 发送get请求:

axios.get("服务器端接口地址",{

params:{ 参数1:值1, ...:... }

}).then(result=>{

//result.data 里获得的就是服务器端返回的结果(已经被json.parse()编译成对象了)

})

(2). 发送post请求:

axios.post("服务器端接口地址",{ 参数1:值1, ...:... }).then(result=>{

//result.data 里获得的就是服务器端返回的结果(已经被json.parse()编译成对象了)

})

(3). 提前配置统一的服务器端接口域名地址:

axios.defaults.baseURL="http://xzserver.applinzi.com";

结果:

axios.get("/index").then(result=>{ ... })

5. 示例: 发送axios请求，获取服务器端学子商城数据

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>Document</title>    <script src="js/axios.min.js"></script>  </head>  <body>  <script>  /\*回顾:  $.ajax({    url:"服务器端接口地址",    data:"get或post",    dataType:"json",    success:function(result){//请求成功后自动调用    }  })\*/  axios.defaults.baseURL="http://xzserver.applinzi.com";  //获得学子商城首页6个商品  axios.get("/index")  .then(result=>{//请求成功后自动调用    console.log(result.data);  })  //获得5号商品的详细信息:  axios.get("/details",{    params:{ lid:5 }  }).then(result=>{    console.log(result.data);  })  //发送登录请求  axios.post("/users/signin",{uname:"dingding", upwd:"123456"})  .then(result=>{    console.log(result.data);  })  </script>  </body> |

三. 组件生命周期:

1. 什么是: 一个组件或一个"页面"加载过程中所经历的每个阶段

2. 为什么: 因为有些初始化操作，是应该在页面或组件加载完成后，就自动执行的！不需要人为手工操作

3. 何时: 今后只要希望在页面或组件加载完成后自动执行一些初始化任务时就需要用到组件或页面的生命周期

4. 一个"页面"或组件加载过程中需要经历哪些阶段: (仅以new Vue()举例) 4个阶段

(1). 创建(create)阶段: (必经)

a. 创建new Vue()对象

b. 加载data对象到new Vue()中

c. 暂时没有虚拟DOM树，可能无法执行DOM操作

(2). 挂载(mount)阶段: (必经)

a. 根据el所指扫描页面，创建虚拟DOM树, 并首次将data中的变量值渲染到页面显示

b. 既又了data，又有了虚拟DOM树，既可以操作data中的变量，又可以对页面执行DOM操作

(3). 更新(update)阶段: (只有在data中的变量被修改后才自动触发)

(4). 销毁(destroy)阶段: (只有手动调用$destroy()函数，销毁当前组件时才自动触发)

5. 生命周期钩子函数:

(1). 什么是: 在每个生命周期阶段自动触发的特定的回调函数(类似于Dom中的事件处理函数)

(2). 何时: 只要希望在特定的生命周期阶段执行一些操作时，都要把操作写在对应的生命周期钩子函数内，才能自动触发

(3). 包括: 8个 (每个阶段前后都有一个钩子函数)

beforeCreate()

创建(create)

created() //可发送axios请求，因为此时已经有data了，如果请求回来数据，可以放在data中，并自动更新到页面上

//小风险: 万一axios请求回来的很快，而后续mount阶段没有执行完，此时是不能执行DOM操作的。

beforeMount()

挂载(mount)

mounted() //最稳妥的发送axios请求的时机：因为此时既有data，又有虚拟DOM树。

beforeUpdate()

更新(update)

updated()

beforeDestroy()

销毁(destroy)

destoryed()



6. 示例: 使用生命周期钩子函数发送ajax请求

|  |
| --- |
| <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <meta http-equiv="X-UA-Compatible" content="ie=edge">    <title>Document</title>    <script src="js/vue.js"></script>    <script src="js/axios.min.js"></script>  </head>  <body>  <div id="app">    <ul>      <li v-for="(p,i) of products" :key="i">        {{p.pid}} | {{p.title}} <a href="javascript:;" @click="del(i)">×</a>      </li>    </ul>    <a href="javascript:;" @click="destory">销毁当前new Vue()</a>  </div>  <script>  var vm=new Vue({    el:"#app",    data:{      products:[],//先定义一个空数组等着接axios请求回来的6个商品对象    },    methods:{      del(i){        this.products.splice(i,1);//复习数组函数      },      destory(){        this.$destroy()      }    },    beforeCreate(){      console.log(`创建data对象前自动触发...`)    },    created(){      console.log(`创建data对象后自动触发...`)    },    beforeMount(){      console.log(`创建虚拟DOM树前自动触发...`)    },    mounted(){      console.log(`创建虚拟DOM树后自动触发...`)      console.log(`    发送axios请求`);      axios.get("http://xzserver.applinzi.com/index")      .then(result=>{        console.log(`收到响应结果，并放入data中的products变量`)        this.products=result.data;//修改了data中的products，就会触发update阶段      })    },    beforeUpdate(){      console.log(`更新data中的变量前自动触发...`)    },    updated(){      console.log(`更新data中的变量后自动触发...`)    },    beforeDestroy(){      console.log(`在销毁当前组件或页面前自动触发...`)    },    destroyed(){      console.log(`销毁当前组件或页面后自动触发...`)    }  })  </script>  </body> |

总结:

MVVM: 界面(View), 数据模型(Model), 视图模型(ViewModel)

Vue绑定原理: 访问器属性+虚拟DOM树

new Vue({

el:"#app",

data:{ ... },

methods:{ ... }, //调用时加()，不缓存结果，会反复计算

computed:{ ... },//调用时不加(), 缓存结果，不会反复计算

watch:{ ... }

})

绑定和指令:

1. 如果元素内容可能发生变化: {{}} v-text

2. 如果元素的属性值发生变化: :

3. 控制一个元素显示隐藏：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-if v-else-if v-else

6. 反复生成多个相同结构的元素: v-for :key

7. 绑定事件: @ $event

8. 绑定HTML内容: v-html

9. 防止用户短暂看到双花括号: v-cloak v-text

10. 只在开始时绑定一次: v-once

11. 防止内容中的{{}}被编译: v-pre

12. 获取表单元素的值: v-model

绑定样式:

1. 只操作一个元素的一个css属性值:

<元素 style="不变的css属性" :style="变量"

data:{

变量:{

Css属性: "属性值",

... : ...

}

}

2. 批量修改一个元素的多个css属性值: class

<元素 class="不变的class" :class="变量"

data:{

变量:{

Class名1: true或false,

... : ...

}

}

自定义指令:

Vue.directive("指令名",{

inserted(domElem){

}

})

作业:

1. (扩展)看小程序视频学习组件间传参与ref:

小程序->在线->VUE->day03 7 组件间传参 todo 父给子 子给父 兄弟间 ref $refs

2. (扩展)看小程序视频学习SPA单页面应用与路由嵌套

小程序->在线->VUE->day03 8 SPA 川剧 变脸 单页面应用 all in one

3. (扩展)看小程序视频学习JS模块化开发

小程序->在线->VUE->day03 模块化(7个视频) 1-手工实现 2-CommonJS/AMD ... ...