# Vue开发工具

## 使用vscode开发，插件安装如下：

Vetur:

—— 语法高亮、智能感知、Emmet等  
包含格式化功能， Alt+Shift+F （格式化全文），Ctrl+K Ctrl+F（格式化选中代码，两个Ctrl需要同时按着）

# 支持储备

## es6语法

## babel

# Vue脚手架

## Vue cli:

脚手架简单来说就是创建项目

**// 安装脚手架**

**npm install -g @vue/cli**

**-g代表全局安装脚手架，在任何目录下都可以使用，不会提示命令找不着。**

**//安装cnpm**

**npm install –g cnpm**

**// 脚手架生成 vue 项目，同时会自动安装依赖**

**vue create vue-cli-demo**

**vue项目入口页面:./public/index.html**

**vue项目的入口模块：./src/main.js**

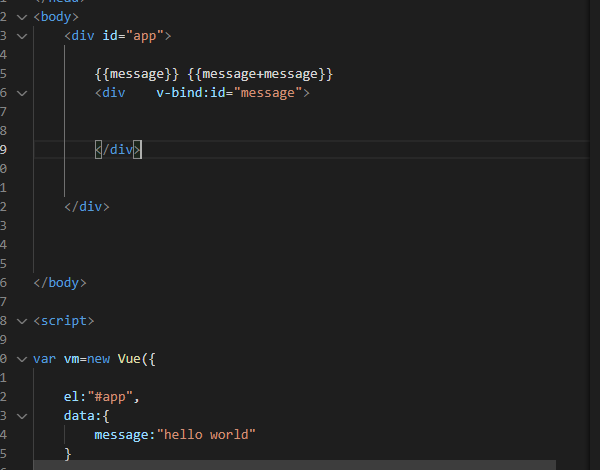
# Vue知识点:

## Vue项目结构:

* Package.json

cnpm i 根据Package.json重新下载依赖

* {{}}中只支持变量和表达式,



## Vue基础

### 模板语法:

模板中的内容会被当做javascript来处理

* 插值

纯文本格式为:{{}}（防止xss攻击），假设用户在输入参数时，输入了一段javaScript代码，后台将参数存储到数据库，前端页面加载内容时，将javascript片段渲染到用户的页面上。

{{}}插值语法会自动过滤特殊字符，对于后端请求的标签，浏览器不会去解析。

纯html：使用v-html输出html片段，浏览器会自动解析html标签。

* 指令

**指令中内容会被当做javascript来处理。**

v-bind

 <div v-bind:id="myClass">测试v-bind指令</div>

v-bind:会将元素的内容使用vue中的状态值替换

v-if:动态创建和删除。

  <div v-if="isCreated">动态创建和删除</div>

isCreated为false，则元素不会被创建

v-show:动态显示和隐藏

  <div v-show="isShow">动态显示和隐藏</div>

isShow对应vue中状态值，通过true和false判断元素是否显示和隐藏。

v-model：数据双向绑定。

* 缩写

v-bind可以省略为:元素名

 <div :id="myClass">测试v-bind指令缩写</div>

### 数据双向绑定

1. 数据双向绑定v-model,表单中的值和状态中的值会互相影响

表单的input type=text，textArea,radio(设置value)，绑定对象

表单的checkbox（设置value）,绑定到数组上。

注意：v-model默认是和input事件同步的，在表单中，可能仅仅在失去焦点时，才需要绑定数据，可以添加一个v-model.lazy，将数据绑定延迟到change事件中，减少数据交互。

### Class绑定

* 使用三目运算符绑定class

changeClass改变状态值（变量值），通过isActive的true和false改变背景样式。

<button @click="changeClass">点击切换class</button>

<div :class="isActive?'red':'yellow'">三目写法</div>

var vm = new Vue({

    el:"#app",

    data(){

        return {

            message:"我是湖北人",

            isActive:true

        }

    },

    methods:{

         changeClass(){

            this.isActive=!this.isActive;

        }

    }

});

* 动态绑定对象写法

  <div :class="classObj">绑定对象写法</div>

 data(){

        return {

            message:"我是湖北人",

            isActive:true,

            classObj:{a:true,b:true}

        }

    },

默认情况下，div会存在；两个class属性，可以通过修改classObj对象的a,b属性值为trye和false来显示和隐藏对应的class。

注意：假设后期我们希望通过给对象添加属性的形式增加样式，给classObj动态添加c属性，则c属性不会加到class属性中，因为vue本质是对状态的get set方法拦截，进行页面渲染，新家的属性c未生成get和set方法，所以不会生效。可以使用**Vue.set(dataobj,name,value)的方式来解决此问题。**

* 动态绑定数组形式

  <div :class="classarray">绑定数组方式</div>

   classarray:["a","b","c"]

可以通过给数组添加元素（push()）,减少元素(pop),来动态切换样式。

### Style绑定

* 使用三目运算符绑定class

<div :style="'background-color:'+(isActive?'red':'yellow') ">绑定style-正常写法</div>

* 动态绑定对象写法

            <div :style="styleObj">绑定style-绑定对象写法</div>

Vue状态值:

   styleObj:{

                'backgroundColor':'red',

                'fontSize': '100px'

            }

Css带有横线的属性，可以使用驼峰命名法的形式书写。

* 动态绑定数组形式

            <div :style="stylearray">绑定style-绑定数组写法</div>

            stylearray:[{backgroundColor:'red'},{fontSize:'20px'}]

### 条件渲染

* v-if 和v-else，

可以类比jstl标签库的<c:if> <c:else>标签。

使用场景：

列表页面默认是空值，则默认走v-else分支（列表没有数据），

用户添加数据，使用ajax传递到后台，则会渲染列表走v-if页面。

<div id="app">

        <button @click="addItem">点击添加数据</button>

        <div v-if="listItem.length>0">

            <ul v-for="data in listItem">

                <li>{{data}}</li>

            </ul>

        </div>

        <div v-else>列表没有数据</div>

    </div>

var vm=new Vue({

    el:"#app",

    data:{

        listItem:[]

    },

    methods:{

        addItem(){

            this.listItem.push("aaa");

            this.listItem.push("bbb");

        }

    }

});

默认数组数据为空，走else分支，点击按钮后，数据被赋值，走if分支。

* v-if 、v-else-if、v-else

可以类比，jstl标签库的c:choose、c:when、c:else。

        <span v-if="transactionType==1">银行付款</span>

        <span v-else-if="transactionType==2">银行收款</span>

        <span v-else="transactionType==3">其他业务类型</span>

通过交易类型的不同的值，页面显示不同的内容。通过改变vm中的transactionType状态来改变数据。

### 列表渲染

* v-for 遍历列表，类比jstl标签库的<c:for>标签

可以遍历数组，遍历对象。**建议为数组对象设置合适的key值，告诉vue怎么去对比虚拟dom节点。**

格式为:

v-for=”(tempItem,tempIndex) in 列表名称或者对象名称”，

或者

v-for=”item of 列表名称”， 列表名称或者对象名称”。

如果是遍历数组，则tempItem为数组值、tempIndex为数组下标。

如果是遍历对象，则tempItem为对象的value值，tempIndex为对象的名称(key)

<div id="app">

        <ul v-for="(tempItem,index) in itemArray">

            <li>{{tempItem}}---------------{{index}}</li>

        </ul>

        <ul v-for="(tempItem,index) of itemArray">

            <li>{{tempItem}}---------------{{index}}</li>

        </ul>

        <ul v-for="(tempValue,tempKey) in itemObj">

            <li>{{tempValue}}---------------{{tempKey}}</li>

        </ul>

        <ul v-for="(tempValue,tempKey) of itemObj">

            <li>{{tempValue}}---------------{{tempKey}}</li>

        </ul>

    </div>

* 列表key值设置

Vue数据拦截之后，需要重新生成虚拟dom节点，设置key值告诉vue怎么去对比虚拟dom节点，为了复用虚拟dom，减少不必要的对比。Key值建议设置为后端对象的唯一性标识，通常设置为data.id。

* 列表数组监测

操作数组方法时，需要判断对象是否是同一个对象，如：数组的pop,push、**splice**等方法是直接改变数组，会被拦截，从而影响原数组，视图被更新。

Filter、concat、slice和map等方法，是又生成了一个新数组，原数组元素保持不变，从而未被拦截，视图未被更新。

**注意：通过数组的索引更新数组，视图不会被更新。解决方式如下：**

1. **使用数组的splice方法**
2. **使用Vue.set(datalist,index,value)方法。不仅可以操作数组，也可以解决给对象新加属性，视图未被更新。**

* **列表过滤应用**

**百度的autocomplete，自动补全功能，会根据用户输入的内容，自动检索匹配数据。**

**注意：需要关注原数组数据被修改而造成的数据丢失的情况。V-model是数据双向绑定。**

**两种方式解决此问题：**

1. **计算属性（本质是一个函数）**
2. **数据复制**

**数据复制实现方式如下:**

    <div id="app">

       请输入用户名称: <input name="userName" value="" v-model="searchInputName" @input="handleInput" />

       <ul v-for="tempItem in userNameAray">

            <li>{{tempItem}}</li>

       </ul>

    </div>

var vm=new Vue({

    el:"#app",

    data:{

        searchInputName:"",

        userNameAray:['aaa','bbb','ccc','ddd','aabced'],

        userNameArayOrig:['aaa','bbb','ccc','ddd','aabced']  //数据拷贝

    },

    methods:{

        handleInput(){

            if(this.userNameAray==null ){

                return ;

            }

            this.userNameAray= this.userNameArayOrig.filter((tempName)=>{

                console.log("tempName=====>"+tempName);

                console.log("searchInputName=====>"+this.searchInputName);

              return  tempName.indexOf(this.searchInputName)>-1;

            });

        }

    }

});

**主要是每次过滤filter时，使用原数组进行过滤（userNameArayOrig），显示时，使用过滤后的数组（userNameAray）显示。**

### 事件处理器

* 绑定事件

**绑定监听事件使用@事件名称。**

    <div id="app">

        <button @click="handleClick(index)">

            <label  v-show="isShow"  >显示</label>

            <label  v-show="!isShow">隐藏</label>

        </button>

    </div>

    var vm=new Vue({

        el:"#app",

        data:{

            index:1,

            isShow:true

        },

        methods:{

            handleClick(indexParam){

                console.log(indexParam);

                this.isShow=!this.isShow;

            }

        }

    });

### 事件修饰符

**javaScript的冒泡事件，子类的事件父类也触发事件。比如在子组件中定义**

**事件，父类中如果也定义了click事件，点击子类，则父类事件也会被触发。**

**格式为：**

**Click.事件修饰符**

**怎么防止冒泡事件？**

1. **使用event的StopPro…阻止**

        <ul  @click='handleUlClick($event)'>

            <li @click="handleLiClick($event)">我是li</li>

        </ul>

        methods:{

            handleClick(indexParam){

                console.log(indexParam);

                this.isShow=!this.isShow;

            },

            handleUlClick(){

                console.log("ul click事件被触发了!");

            },

            handleLiClick(ev){

                ev.stopPropagation();

                console.log("li click事件被触发了!");

            }

        }

**绑定click事件，并传递$event参数，方法中调用stopPropagation阻止冒泡。**

1. [**使用@click.stop**](mailto:使用@click.stop)**事件修饰符阻止。**

        <ul  @click='handleUlClick1()'>

            <li @click.stop="handleLiClick1()">我是li,使用事件修饰符的阻止冒泡</li>

        </ul>

1. **父类设置**self将事件绑定到自身

        <ul  @click.self='handleUlClick2()'>

            <li @click="handleLiClick2()">我是li,将事件绑定到自身阻止冒泡</li>

        </ul>

**常用的事件修饰符:**

stop：阻止冒泡

prevent: 阻止默认事件的发生,如超链接的跳转

self: 将事件绑定到自身，只有自身才能触发，通常用于避免冒泡事件的影响

. once：事件只触发一次

### 按键修饰符

使用场景：

输入框摁下回车键，自动跳转到每个页面，或者显示什么内容，可以使用按键修饰符来过滤指定的键盘事件。

  <input @keyup.enter="handleKeyUp($event)">

  handleKeyUp(ev){

                if(ev.keyCode==13){

                    console.log("嗯下了回车键");

                }

                console.log(ev.keyCode);

            }

**按键修饰符，用来过滤指定的事件。**

使用event来过滤键值

使用按键修饰符，格式为:@keyup.enter或者:@keyup.键盘按键的键值（自定义按键），

**常用的按键修饰符:**

### 表单控件绑定

传统页面提交表单的方式，会根据表单控件的name属性，拼接参数传递到后台，后台根据request.getParameter(“参数名称”)来获取值。

使用vue时，

表单的值可以绑定到vue中的状态，提交状态到后端。

1. 数据双向绑定v-model,表单中的值和状态中的值会互相影响

表单的input type=text，textArea,radio(设置value)，绑定对象

表单的checkbox（设置value）,绑定到数组上。

### 计算属性

复杂的逻辑程序难以处理，可以理解为el表达式的自定义el表达式方法，通过方法直接渲染处理数据.

计算属性在vue中使用computed定义，调用格式为:{{计算属性名称}}，注意没有小括号，和方法的区别之一，区别之二，是方法每次都会重新计算，计算属性只会结算一次，理解为缓存 。**计算属性中的状态值发生改变后，会再次执行计算属性。**

总结:

计算属性只会计算一次，方法会计算多次。

方法调用需要加括号，计算属性不加括号。

### 虚拟dom diff

## Vue组件

### 数据请求方式（fetch、axios）

传统web开发中，通常使用jquery或者自己封装的实现ajax请求，在vue中，有现成的ajax请求组件。

* fetch

fecth是w3c的一个标准库，谷歌浏览器内置。Fetch用法如下:

格式为:

Fetch(url).then(response=>{return response.json()}).then(()=>{箭头函数});

fetch("./json/testJsonData.json").then(response=>response.json()).then(result=> this.fileArray=result.data.films);

使用匿名函数:

           fetch("./json/testJsonData.json").then(function(response){

            return response.json()

           }).then(function(result){

            vm.fileArray=result.data.films

           });

* axios（建议）

第三方的库，是作者推荐使用的库。

使用js进行引用

    <script src=" https://cdn.bootcdn.net/ajax/libs/axios/0.21.1/axios.js"></script>

  axios.get("./json/testJsonData.json").then((result)=>{

                vm.fileArray=result.data.data.films;

            }).catch(err=>{

                console.log(err);

            });

注意：axios多了一个data属性，返回的数据外层套了一套data属性。

### 组件介绍

组件的目的是为了扩展html元素，从而实现代码复用。

组件具有样式(css)、javascript、dom结构(html代码)

Vue实例创建之后，有一个根组件。

* 全局组件注册

使用Vue.component注册，方法需要定义在Vue.component中，不能定义在Vue实例中。

        <div id="app">

            <navbar></navbar>

        </div>

<script>

    Vue.component('navbar',{

        template:`

        <div style='background:red'>

            <button @click='handleHomeClick'>主页</button>

            导航栏

            <button @click='handleBackClick'>返回</button>

        </div>

        `,

        methods:{

            handleHomeClick(){

                console.log("主页按钮被点击了");

            },

            handleBackClick(){

                console.log("返回按钮被电击了1");

            }

        }

    });

    var vm=new Vue({

        el:"#app"

    });

</script>

* 局部组件注册

局部组件注册在组件中使用component属性注册。

  Vue.component('navbar',{

        template:`

        <div style='background:red'>

            <button @click='handleHomeClick'>主页</button>

            导航栏

            <navbarchild></navbarchild>

            <button @click='handleBackClick'>返回</button>

        </div>

        `,

        components:{

            navbarchild:{

                template:`

                <div>navbarchild</div>

                `

            }

        },

        methods:{

            handleHomeClick(){

                console.log("主页按钮被点击了");

            },

            handleBackClick(){

                console.log("返回按钮被电击了1");

            }

        }

    });

* Vue组件和实例的区别

1）自定义组件需要一个root element，仅仅只有一个。

2）父子组件的data无法共享

3）组件可以有data、method、computed，但是data必须是一个函数。

* 组件通信

# 单页应用

## Vue渲染介绍