# 1、VUE的基本开发步骤

## 1.1、导入js

<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

## 1.2、创建Vue实例对象，设置el和data属性

**var** vue = **new** Vue({  
 el:"#app",  
 data:{  
 msg:"lalala"  
 }  
})

## 1.3、使用模板语法进行数据渲染

<div id="app">  
 {{msg}}  
</div>

# 2、el挂载点

## 2.1、作用范围

作用于el选项命中的元素及其后代元素

## 2.2、选择器

el可使用id选择器(建议)、class选择器、标签选择器等

注：挂载点不适用于单标签，不能用在<html>与<body>上，建议用在<div>上

# 3、data数据对象

## 3.1、作用

Vue中用到的数据定义在data中

## 3.2、数据定义

data中可写复杂类型的数据

data:{  
 msg:"lalala",  
 stu:{  
 name:"cxd",  
 age:21  
 },  
 stus:["cxd1","cxd2","cxd3"]  
}

## 3.3、数据获取

数据获取遵循js基本语法，示例如下：

<div id="app">  
 {{msg}}  
 <h2>{{stu.name}}</h2>  
 <ul>  
 <li>{{stus[0]}}</li>  
 </ul>  
</div>

# 4、Vue本地应用

## 4.1、常用指令信息

### 4.1.1、v-text

替换标签文本值，数据来源为data，且可实现拼接字符串操作

<h2 v-text="msg">啦啦啦</h2>

### 4.1.2、v-html

除普通文本操作外，还可实现页面标签元素的追加

<h2 v-html="myurl"></h2>

---------------------------------------------------------------------

data:{myurl:"<a href='http://www.baidu.com'>百度</a>"....}

### 4.1.3、v-on

为元素绑定事件，使用方式为v-on:事件名或@事件名，方法来源为methods，也可以进行参数传递

<input type="button" value="v-on1" v-on:click="test1"/>  
<input type="button" value="v-on2" @click="test1"/>  
<input type="button" value="v-on3" @dblclick="test1"/>

---------------------------------------------------------------------

methods:{test1:**function** () {alert("v-on");}  
}

注：在方法的内部可使用this直接访问定义在data中的元素

### 4.1.4、v-show

可隐藏或显示标签，本质是更改display属性，指令内容可为表达式，最终解析为bool值，true显示false隐藏

<img v-show="isShow" src="./test.png"/>

### 4.1.5、v-if

条件成立标签生效，操作dom树，本质上不同于v-show

<img v-if="1>0" src="./test.png"/>

注：除v-if外，还有v-else-if、v-else，使用相似

### 4.1.6、v-bind

用于设置元素的属性，如v-bind:src=”...”，可简写为:src=”...”，示例如下

<img v-bind:src="imgUrl" v-bind:title="title"/>  
<img :src="imgUrl" :title="title"/>

注：三元运算符的使用：”1>0?active:’’”等价于{active:1>0}

### 4.1.7、v-for

遍历数据，并且可获取下标值index，格式为v-for=”a in b”/”(a,index) in b”

<ul>  
 <li v-for="(temp,index) in addrs" v-text="index+'、'+temp"></li>  
</ul>

<ul>  
 <li v-for="(temp,index) in addrs">{{index}}、{{temp}}</li>  
</ul>

---------------------------------------------------------------------

data:{  
 addrs:["北京","上海","广州","深圳"]  
}

注：v-for可用于遍历数组，也可以用于遍历对象v-for=”(value,key,index) in object”，值分别为当前值，所属key，索引下标

### 4.1.9、v-model

获取和设置表单元素的值-->双向数据绑定，即表单元素值与data值同步

<input v-model="msg"/>  
<h2>{{msg}}</h2>

## 4.2、其余常用信息

### 4.2.1、事件修饰符

@事件名.修饰符=”方法”，示例如下

<input @keyup.enter="whenEnter"/>

### 4.2.2、过滤器filters

格式为{{msg | filter}}，且过滤器配置在filters中，示例如下：

<p>{{ message | capitalize }}</p>  
<div v-bind:id="message | capitalize"></div>

---------------------------------------------------------------------

filters: {  
 capitalize: **function** (value) {  
 **if** (!value) **return** ''  
 value = value.toString()  
 **return** value.charAt(0).toUpperCase() + value.slice(1)  
 }  
}

### 4.2.3、监听属性watch

对某个Vue属性的监听，当发生改变时做出的操作，可在watch中写具体的监听事件，也可以在外部调用$watch方法，如app.$watch(msg,function(old,new))，示例如下：

千米：<input type="text" v-model="kilometers"/>  
米：<input type="text" v-model="meters"/>  
<p id="info"></p>

--------------------------------------------------------------------------------------------------------------

watch:{  
 kilometers:**function** (val) { **this**.kilometers=val; **this**.meters=val\*1000;},  
 meters:**function** (val) {**this**.kilometers = val/ 1000;**this**.meters = val;}  
}

--------------------------------------------------------------------------------------------------------------

// $watch 是一个实例方法  
app.$watch('kilometers', **function** (newValue, oldValue) {  
 // 这个回调将在 app.kilometers 改变后调用  
 document.getElementById ("info").innerHTML = "修改前值为: " + oldValue + "，修改后值为: " + newValue;  
})

## 4.3、自定义指令

### 4.3.1、自定义全局指令

通过Vue.directive(指令名,对应操作)来自定义指令，示例如下：

<input v-focus><!--页面载入时，input 元素自动获取焦点-->

-------------------------------------------------------------------------------

Vue.directive('focus', {  
 inserted: **function** (el) {// 当绑定元素插入到 DOM 中  
 el.focus()// 聚焦元素  
 }  
});

### 4.3.2、自定义局部指令

使用类似局部组件的创建，涉及的vue属性为directives，示例如下：

<input v-focus><!--页面载入时，input 元素自动获取焦点-->

-------------------------------------------------------------------------------

**new** Vue({  
 el: '#app',  
 directives:{  
 focus:{  
 inserted:**function** (el){//当绑定元素插入到 DOM 中，inserted为钩子函数  
 el.focus();// 聚焦元素  
 }  
 }  
 }  
})

### 4.3.3、钩子函数(了解)

**bind**-->只调用一次，指令第一次绑定到元素时调用，可用于初始化动作  
**inserted**-->被绑定元素插入父节点时调用  
**update**-->被绑定元素所在的模板更新时调用

**componentUpdated**-->被绑定元素所在模板完成一次更新周期时调用  
**unbind**-->只调用一次，指令与元素解绑时调用

### 4.3.4、钩子函数参数(了解)

**el**-->指令所绑定的元素，可以用来直接操作DOM  
**binding**-->一个对象，包含name、value、oldValue等属性  
**vnode**-->Vue编译生成的虚拟节点  
**oldVnode**-->上一个虚拟节点，仅在update和componentUpdated钩子中可用

# 5、Vue网络应用

## 5.1、axios

### 5.1.1、引入axios

axios是一个功能强大的网络请求库，需先引入axios.js，如下：

<script src="https://unpkg.com/axios/dist/axios.min.js"> </script>

### 5.1.2、发送get请求

使用方式为axios.get(url).then(function(res){},function(error){})，示例如下：

getMethod:**function** () {  
 axios.get("https://autumnfish.cn/api/joke/list?num=6")  
 .then(**function** (response) {console.log(response);},

**function** (error) {console.log(error);})  
}

### 5.1.3、发送post请求

使用方式为axios.post(url,params).then(function(res){},function(error){})

postMethod:**function** () {  
 axios.post("https://autumnfish.cn/api/user/reg",{username:"cxd"})  
 .then(**function** (response) { console.log(response);},

**function** (error) {console.log(error); })  
}

## 5.2、vue+axios

在请求发送后页面上显示响应的回调数据

注：axios的回调函数中无法使用this，需在外部对this进行保存

<input type="button" value="get请求" @click="getMethod" />  
<p v-text="jokes"></p>

---------------------------------------------------------------------

getMethod:**function** () {  
 **var** info = **this**;  
 axios.get("https://autumnfish.cn/api/joke/list?num=6")  
 .then(**function** (response) {info.jokes = response.data.jokes;},

**function** (error) {info.jokes = "";})  
},

# 6、使用vue-cli来创建项目

## 6.1、安装nodejs

node -v-->检查node安装版本

npm -v-->检查npm版本

## 6.2、设置cache与global

在安装目录下新建node\_cache与node\_global文件夹，再配置路径即可，执行命令如下：

npm config set prefix "D:\nodejs\node\_global"  
npm config set cache "D:\nodejs\node\_cache"

npm list -global-->查看全局文件夹信息

## 6.3、配置淘宝镜像cnpm

npm install -g cnpm --registry=https://registry.npm.taobao.org

注：若提示cnpm命令不存在，则需在path变量中添加D:\nodejs\node\_global

npm config list-->查看配置信息

npm install npm -g-->更新或安装npm，-g指定为全局文件中

## 6.4、开始配置vue脚手架

npm config get registry-->查看镜像地址

npm config set registry url-->设置镜像地址

cnpm install vue -g-->安装vue库

cnpm install --global vue-cli-->安装vue脚手架

## 6.5、创建并运行vue项目

vue init webpack my-project-->创建基于webpack的vue项目

cd my-project-->进入项目  
cnpm install-->安装依赖项  
cnpm run dev-->运行项目，默认为：localhost:8080

注：vsCode执行cnpm install报错-->set-ExecutionPolicy RemoteSigned

## 6.6、项目目录信息



# 7、组件Component

## 7.1、注册全局组件

Vue.component(tagName,options)-->tagName为组件名，options为配置选项。注册后可在任意实例下通过<tagName>来调用全局组件，示例如下：

<div id="app"><runoob></runoob></div>

---------------------------------------------------------------------

Vue.component('runoob', { // 注册全局组件  
 template: '<h1>自定义组件!</h1>'});  
**var** app = **new** Vue({el: '#app'});

## 7.2、注册局部组件

局部组件写在实例对象中，只能在该父模板下使用，示例如下：

<div id="app"><runoob></runoob></div>

---------------------------------------------------------------------

**var** app = **new** Vue({  
 el: '#app',  
 components:{  
 'runoob': { //局部组件只能在父模板中使用  
 template: '<h1>自定义组件!</h1>'}  
 }  
});

## 7.3、原始创建组件的步骤（了解）

基本创建并使用组件的步骤如下（7.1与7.2是简化创建方式）：

<div id="app">  
 <runoob></runoob><!--3.使用组件-->  
</div>

-------------------------------------------------------------------------------

**const** cpnC = Vue.extend({ //1.创建组件构造器对象  
 template:'<h1>自定义组件!</h1>'  
});  
Vue.component('runoob', cpnC);//2.注册全局组件  
**var** app = **new** Vue({el: '#app'});

## 7.4、组件的分离写法

当组件中的模板较为复杂时，js中写模板并不友好，此时可将模板抽离出来，有两种方式：script/template，均需指明id，示例如下：

<div id="app"><cpn></cpn></div>  
<!-- 1.抽离组件：使用script标签 -->  
<script type="text/x-template" id="cpn1"><h1>标题1</h1></script>  
<!-- 2.抽离组件：使用template标签 -->  
<template id="cpn2"><h1>标题2</h1></template>

-------------------------------------------------------------------------------

Vue.component('cpn',{template:'#cpn2'});  
**var** app = **new** Vue({el: '#app'});

## 7.5、组件数据的存放

组件中数据也是存于data中，但是组件的data只能是一个函数，函数中再返回一个对象，示例如下：

Vue.component('cpn',{template:'#cpn2',data(){**return** {title:'标题'}}});

**组件的data为什么只能是函数**：在函数中再返回对象，是为了保证在创建多个组件实例时，data数据不会共享，各个组件之间不会相互影响

## 7.6、参数传递父->子

### 7.6.1、普通props参数传递

父组件的数据需通过 props 传给子组件，在调用子组件时传入，示例如下：

<div id="app"><child message="hello!"></child></div>

Vue.component('child', {  
 props: ['message'],// 声明 props  
 //同样也可以在 vm 实例中像 “this.message” 这样使用  
 template: '<span>{{ message }}</span>'  
})

---------------------------------------------------------------------  
**new** Vue({ el: '#app'})

### 7.6.2、动态props参数传递（重点）

父组件在调用子组件时可使用v-bind:msg=”...”来进行传参，父组件值的变化会同步到子组件

<div id="app">  
 <input v-model="parentMsg">  
 <child v-bind:message="parentMsg"></child>  
</div>

Vue.component('child', {  
 props: ['message'],  
 template: '<span>{{ message }}</span>'  
})  
**new** Vue({el: '#app',data: {parentMsg: '父组件内容'}})

### 7.6.3、props的申明方式

自定义组件申明变量的同时可指定类型及默认值，且当参数为对象或数组时，默认值必须是函数，示例如下：

props: {  
 message:String,// 声明 props的同时指明类型  
 name:{ // 声明 props的同时指明类型、默认值、是否必传  
 type:String,  
 **default**:"cxd",  
 required:**false** },

movies:{ //当参数类型为对象或数组时，默认值必须是函数  
 type:Array,//Object  
 **default**(){**return** [];}  
 }

},

### 7.6.4、props变量的驼峰问题（了解）

在子组件props变量申明时，若变量为驼峰命名，则父传入时应有驼峰映射，因为v-bind不支持驼峰，示例如下：

<child v-bind:my-school-name="parentMsg"></child>

-------------------------------------------------------------------------------

props: {mySchoolName:String}

## 7.7、参数传递子到父-->自定义事件

### 7.7.1、定义子组件事件

子组件事件中使用$emit()来指明自定义事件名与参数，示例如下：

<button v-for="item in catalogs" @click="btnClick(item)">{{item.name}}</button>

-------------------------------------------------------------------------------

//将子组件的值传递到父组件，自定义事件名+参数

btnClick(item){**this**.$emit('itemclick',item);}

### 7.7.2、监听自定义事件

父组件中通过v-on来监听子组件的自定义事件，并写明监听后的对应事件操作，示例如下：

<div id="app"><cpn @itemclick="cpnClick"></cpn></div>

-------------------------------------------------------------------------------

cpnClick(item){//可直接使用子组件传过来的参数，必须显式传入  
 console.log(item);  
}

子传父完整demo示例源码：[vue-子传父](demo/vue02-子传父.html)

# 8、路由Router

## 8.1、路由的基本使用

### 8.1.1、定义组件

此处的组件可以是定义的，也可以是其它文件中导入的

// 1. 定义（路由）组件，可以从其他文件 import 进来  
**const** Foo = { template: '<div>foo</div>' }  
**const** Bar = { template: '<div>bar</div>' }

### 8.1.2、创建router实例

router实例中配置了url与组件的对应关系

// 2. 创建 router 实例，然后传 `routes` 配置  
**const** router = **new** VueRouter({  
 routes:[ //3.定义路由，每个路由应该映射一个组件  
 { path: '/foo', component: Foo },  
 { path: '/bar', component: Bar }  
 ]  
})

### 8.1.3、创建并挂载根实例

// 3. 创建和挂载根实例  
**const** app = **new** Vue({ router:router }).$mount('#app');

### 8.1.4、页面使用路由

在页面中使用<router-link>与<router-view>即可

<div id="app">  
 <!-- router-link 导航.to 属性指定链接，默认会被渲染成一个<a>标签 -->  
 <router-link to="/foo">Go to Foo</router-link>  
 <router-link to="/bar">Go to Bar</router-link>  
 <!-- 路由出口，路由匹配到的组件将渲染在这里 -->  
 <router-view></router-view>  
</div>

完整demo示例源码：[vue-路由](demo/vue03-路由.html)

## 8.2、使用路由的注意事项

### 8.2.1、设置默认路径

当访问ip:port时，默认加载某个路由信息，示例如下：

routes: [// 配置路径与组件间的映射关系

    {

      path: '',// 配置路由的默认路径

      redirect: '/home'// 重定向

},

### 8.2.2、更改项目模式

Vue项目默认为hash模式，即路径中带#号，可在创建路由时更改为history模式，示例如下：

export default new Router({

  mode: 'history',// 默认值为hash，即路径中带#号

## 8.3、<router-link>的属性介绍

### 8.3.1、tag

<router-link>默认会渲染为<a>标签，通过tag属性可指定渲染为其它标签，示例如下：

<router-link to="/home" tag="button">首页</router-link>

### 8.3.2、replace

replace不会留下history记录(默认为push)，即不能返回上一个页面，示例如下：

<router-link to="/about" replace>关于</router-link>

### 8.3.3、active-class

默认为active-class=”router-link-active，”当某个路由被点击时，会添加该样式，也可自己指定样式，示例如下：

<router-link to="/home" tag="button" replace active-class="active">首页</router-link>

也可以在路由配置中指定linkActiveClass属性，示例如下：

export default new Router({

  linkActiveClass: 'active',

### 8.3.4、其它标签替代<router-link>

可使用其它标签来代替<router-link>也能达到相同的效果，但是需在单击事件中进行处理

<button @click="homeClick">首页</button>

<button @click="aboutClick">关于</button>

-----------------------------------------------------------------------

methods:{

    homeClick(){ this.$router.push('/home') },

    aboutClick(){ this.$router.push('/about') }

 }

## 8.4、动态路由

及在url后面有动态拼接内容，如/user/userid，userid为具体值，使用步骤如下:

### 8.4.1、配置动态路由

routes: [{

      path: '/user/:userid',

      component: User

    }

### 8.4.2、<router-link>的to属性传入

<router-link v-bind:to="'/user/'+userid" >用户</router-link>

-----------------------------------------------------------------------

data(){ return{userid:'cxd'}}

### 8.4.3、组件中$route.params获取

具体的参数名需与路由配置中的参数一致

<h1>{{userid}}</h1>

<h1>{{$route.params.userid}}</h1>

-----------------------------------------------------------------------

userClick(){//也可以单击按钮触发路由

      //拼接传参，可通过this.$route.params.userid来获取

      this.$router.push({path: '/user/'+this.userid})

}

//获取传参

computed:{userid(){return this.$route.params.userid}}