# Vue上

复习一下指令:

v-text：简写为双大括号（渲染DOM文本）

v-html:可以渲染DOM文本（可以将标签形式字符串转换为标签）

v-if：通过标签上树、下树形式控制节点显示与隐藏

v-show：通过display这个属性，控制标签显示与隐藏

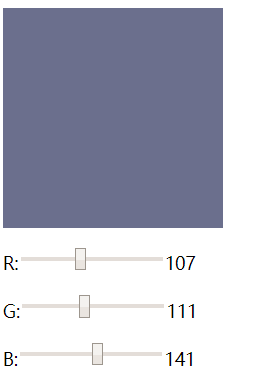
v-model：数据双向绑定（表单元素使用）

v-for:遍历

v-on：简写为@可以给元素绑定事件

v-bing：动态绑定属性值(简写为:)

## 1.1调色板



HTML5中新增的一些表单元素：

HTML5【即为超文本标记语言第五次重大修改，这次修改的时间为2014】新增了很多标签元素：

如下：

<!DOCTYPE html>

<html lang="en">

<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>

</head>

<body>

    <form>

        <p>

            <!-- placeholder:占位符属性 -->

            文本框: <input type="text" placeholder="请你输入用户名">

        </p>

        <p>

            单选按钮:

            <input type="radio" name="hobby">吃饭

            <input type="radio" name="hobby">睡觉

            <input type="radio" name="hobby">打豆豆

        </p>

        <p>

            复选按钮

            <input type="checkbox">飞机

            <input type="checkbox">大炮

            <input type="checkbox">坦克

        </p>

        <hr>

        <p>

            <!-- min：设置范围条最小数值 max最大数值 -->

            范围条: <input type="range" min="0" max="255" value="0">

        </p>

        <p>

            颜色表单元素: <input type="color">

        </p>

        <p>

            日历表单元素： <input type="date">

        </p>

        <p>

            第几周日历:<input type="week">

        </p>

        <p>

            搜索的表单元素： <input type="search">

        </p>

        <p>

            地址格式表单元素:(表单元素输入文本务必符合URL格式)

            <input type="url">

        </p>

        <p>

            邮箱格式：（表单元素的文本务必符合邮箱格式）

            <input type="email">

        </p>

        <p>

            输入密码的表单元素

            <input type="password">

        </p>

        <p>

            <input type="submit">

        </p>

    </form>

</body>

</html>

//调色板代码

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

</head>

<body>

    <!-- 模板 -->

    <div id="app">

        <!-- 颜色变化元素 -->

        <div :style="{width:'400px',height:'100px',background:`rgb(${R},${G},${B})`}"></div>

        <!-- 三个范围条 -->

        <p>

            R: <input type="range" min="0" max="255" value="0" v-model="R"><span>{{R}}</span>

        </p>

        <p>

            G: <input type="range" min="0" max="255" value="0" v-model="G"><span>{{G}}</span>

        </p>

        <p>

            B: <input type="range" min="0" max="255" value="0" v-model="B"><span>{{B}}</span>

        </p>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        //配置对象

        // 挂载点设置[设置Vue可以管理区域]

        el: "#app",

        data: {

            R: 0,

            G: 0,

            B: 0

        }

    })

</script>

## 1.2微博发布框效果

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        .cur {

            animation: donghua 2s linear 0s infinite;

        }

        @-webkit-keyframes donghua {

            from {

                transform: rotate(0deg);

            }

            to {

                transform: rotate(360deg);

            }

        }

    </style>

</head>

<body>

    <!-- 模板 -->

    <div id="app" class="cur">

        <!-- 多行文本框 -->

        <textarea cols="30" rows="10" v-model="info"></textarea><span>{{info.length}}/140字</span>

        <p>

            <button :disabled='info.length==0||info.length>140'>发布</button>

            <button @click="clear">清除</button>

        </p>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        // 配置对象

        el: "#app",

        data: {

            info: ''

        },

        methods: {

            //事件处理函数

            clear() {

                this.info = '';

            }

        },

    })

</script>

## 1.3animation \*\*\*\*\*\*

概述:animation是CSS3中新增的动画样式，它的使用类似于函数，分为两步：声明和调用。

Animation这个属性可以让程序员在样式当中完成动画效果。

注意：以前让元素运动一般都是需要通过JS【比如：JQ】

浏览器私有前缀：

|  |  |  |  |
| --- | --- | --- | --- |
| 谷歌 | 微软 | 火狐浏览器 | 欧朋 |
| webkit | ms | Moz | o |

<!DOCTYPE html>

<html lang="en">

<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>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        div {

            position: relative;

            width: 300px;

            height: 150px;

            background-color: cyan;

            /\* 调用动画 \*/

            /\*

              第一个参数：动画名称

              第二个参数：每一次动画执行时间（S||MS）

              第三个参数：动画执行速率

              第四个参数：第一次动画延迟时间

              第五个参数：动画执行次数（infinite:代表动画执行无数次）

             \*/

            animation: erha 3s linear 0s infinite;

        }

        /\* animation动画声明部分 \*/

        /\* webkit代表谷歌浏览器私有前缀：这个动画只能在谷歌浏览器中使用 \*/

        @-webkit-keyframes erha {

            from {

                left: 0px;

                background-color: cyan;

                border-radius: 0px 0px 0px 0px;

                font-size: 16px;

            }

            to {

                left: 600px;

                background-color: red;

                font-size: 30px;

                border-radius: 0px 0px 100px 0px;

            }

        }

    </style>

</head>

<body>

    <div>我爱你</div>

</body>

</html>

## 1.4watch监听属性

概述：Vue框架当中给我们的配置对象当中可以进行监听属性值变化。

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

</head>

<body>

    <!-- 模板 -->

    <div id="app">

        <h1>{{msg}}</h1>

        <input type="text" v-model="msg">

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        // 挂载点设置

        el: "#app",

        data: {

            msg: ""

        },

        // 监听实例属性值变化地方

        watch: {

            //watch可以监听实例属性值变化

            //watch这里书写函数名字务必和监听属性名字要统一

            msg() {

                console.log("你发生变化了");

            }

        },

    })

</script>

### 1.4.1百度预搜索效果

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

</head>

<body>

    <div id="app">

        {{url}}

        <h1>请你输入想搜索的内容: <input type="text" v-model="url"></h1>

        <ul>

            <li v-for="item in arr">{{item}}</li>

        </ul>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        //挂载点设置

        el: "#app",

        data: {

            url: "",

            arr: []

        },

        //监听url属性值变化

        watch: {

            //函数名字务必和监听属性名字一直

            url(a) {

                //这个函数在监听的属性值发生变化的时候回立即执行一次

                //会将最新的数值给我们作为实参注入

                //请求网址拼接

                var path = "https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=" + a + "&cb=a";

                //JSONP跨域拉去数据

                //实现原理：

                //利用script标签src属性

                //利用函数声明+调用放在两个不同服务器上面

                //动态创建script

                var script = document.createElement("script");

                //设置script标签的src属性值

                script.src = path;

                //标签上树将请求发出去

                document.body.appendChild(script);

            }

        },

    });

    //声明一个同名函数

    function a(obj) {

        console.log(obj);

        //修改vue实例的arr属性值

        vm.arr = obj.s;

    }

</script>

<!-- JSONP跨域原理：利用script标签src属性（天生就跨域） -->

# Vue下

重要知识点：

1：生命周期函数

2：Vue当中使用AJAX技术

3：Vue组件

4:Vue当中通信方式

5：明星小案例

## 2.1生命周期函数

概述：在Vue实例和模板DOM标签进行关联的时候，其实这个阶段经历很多种状态，

可以通过Vue框架提供生命周期函数，可以在不同的状态下，书写自己业务代码即可。

注意：在特定的条件下执行函数而已。（面试常问到：八个）

<!DOCTYPE html>

<html lang="en">

<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.min.js"></script>

</head>

<body>

    <!-- 模板 -->

    <div id="app">

        <h1>生命周期函数-----{{msg}}</h1>

        <button @click="add">单击我我加上1</button>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        //配置对象

        //挂载点

        el: "#app",

        // 数据

        data: {

            msg: 1

        },

        //事件处理函数

        methods: {

            //事件处理函数

            add() {

                this.msg++;

            }

        },

        //监听属性地方

        watch: {},

        //生命周期函数

        beforeCreate() {

            //在创建实例之前执行一次，这里访问不到实例data数据

            console.log("在创建vm之前执行一次");

            // console.log(this.msg);

        },

        created() {

            //在实例创建完毕以后立即执行一次

            console.log("在创建vm之后执行一次");

            // console.log(this.msg);

        },

        beforeMount() {

            //在实例创建完毕且和模板进行关联之前执行一次

            console.log("我是VM即将和模板进行关联");

        },

        /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

        mounted() {

            //实际工作当中经常从这个生命周期函数拉去服务器数据

            console.log("我是VM我已经和模板关联完毕");

            // console.log(this.msg);

        },

        beforeUpdate() {

            console.log("当数据发生变化时候立即执行一次");

            if (this.msg % 2 == 0) {

                this.msg += 6;

            }

        },

        updated() {

            console.log("当数据发生改变以后立即执行一次");

            if (this.msg < 10) {

                this.msg = "哈哈哈";

            }

        },

    })

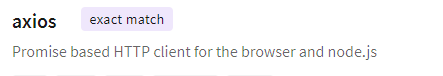
</script>

## 2.2Vue当中AJAX技术实现

概述：AJAX：它是前端当中一门技术，客户端可以悄悄的想服务器端发起上行请求，服务器端接收到请求做出响应的响应，

在页面没有重新加载的情况下实现页面局部更新。

Vue2.0版本以后，当时Vue当中AJAX技术，经常使用axios第三放实现AJAX技术（底层使用promise进行封装）。



axios:它是Vue当中经常使用第三方JavaScript函数库，它主要作用是可以向服务器端发起请求。

下载axios源码方式：

CDN：百度静态资源库（百度公司给提供网站：各种常用前端源码文件都有）

<https://www.bootcdn.cn/>

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

    <script src="./js/axios.js"></script>

    <script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.js"></script>

</head>

<body>

    <div id="app">

        <h1>{{msg}}</h1>

        <button @click="fetchData">单击我发起GET</button>

        <button @click="psotData">单击我法七POST</button>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        //挂载点

        el: "#app",

        data: {

            msg: "默认文本"

        },

        methods: {

            //通过GET请求拉去服务器数据

            fetchData() {

                //发起GET请求，拉去服务器新闻数据

                axios.get("./data.txt").then((obj) => {

                    this.msg = obj.data;

                });

            },

            psotData() {

                //发起POST请求，拉去服务器数据

                // axios.post("./data.txt").then((obj) => {

                //     this.msg = obj.data;

                // })

                $.post("./data.txt", (data) => {

                    this.msg = data;

                });

            }

        },

    })

</script>

## 2.3生命周期函数结合axios一起使用

<!DOCTYPE html>

<html lang="en">

<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>

    <!-- 引包 -->

    <script src="./js/axios.js"></script>

    <script src="./js/vue.min.js"></script>

</head>

<body>

    <div id="app">

        <ul>

            <li v-for="item in postList">{{item.name}}</li>

        </ul>

    </div>

</body>

</html>

<script>

    //创建Vue实例

    var vm = new Vue({

        //挂载点

        el: "#app",

        data: {

            postList: []

        },

        //生命周期函数:实例已经和模板挂在完毕

        mounted() {

            //向服务器拉去数据

            axios.get("./jieyong.txt").then(obj => {

                //修改实例postList属性值

                this.postList = obj.data.postList;

            })

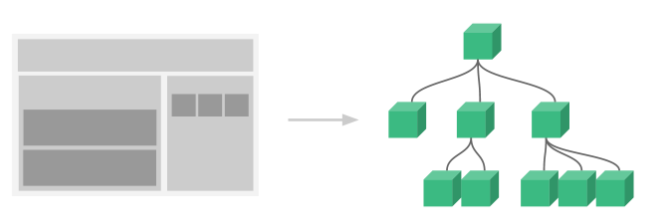
        },

    })

</script>

## 2.4组件开发

组件：component



概述：React、Vue框架开发离不开组件是开发。

到底什么是组件啊？

答：在程序当中经常出现功能类似结构且重复结构，就可以利用组件进行开发；





<!DOCTYPE html>

<html lang="en">

<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>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        p {

            float: left;

            width: 100px;

            height: 100px;

            background-color: cyan;

            margin-right: 10px;

        }

    </style>

    <!-- 引包 -->

    <script src="./js/vue.min.js"></script>

</head>

<body>

    <!-- 模板 -->

    <div id="app">

        <!-- 用标签形式使用组件 -->

        <jch></jch>

        <gen></gen>

        <say-call></say-call>

        <go></go>

    </div>

    <hr>

    <!--下面这里是组件模板 -->

    <template id="erha">

        <div>

            <p @click="add">{{msg}}</p>

        </div>

    </template>

    <template id="gen">

        <div>

            <ul>

                <go></go>

                <li v-for="item in arr">{{item}}</li>

            </ul>

        </div>

    </template>

    <template id="memeda">

        <div>

            <go></go>

            <h1>我是么么打</h1>

        </div>

    </template>

    <template id="go">

        <div>

            <a href="http://www.sina.com">新浪</a>

        </div>

    </template>

</body>

</html>

<script>

    //全局组件

    Vue.component("go", {

        template: "#go"

    })

    //创建Vue实例

    var vm = new Vue({

        //挂载点

        el: "#app",

        //根组件数据

        data: {},

        methods: {},

        watch: {},

        //局部组件需要注册

        components: {

            //组件名字

            jch: {

                //jch组件模板

                template: "#erha",

                //jch组件数据

                data() {

                    console.log(123);

                    //组件数据务必要有返回值

                    return {

                        msg: 1

                    }

                },

                //事件处理函数地方

                methods: {

                    //jch组件事件处理函数地方

                    add() {

                        this.msg++;

                    }

                }

            },

            gen: {

                // gen组件的模板

                template: "#gen",

                data() {

                    return {

                        arr: [1, 2, 3, 4, 5]

                    }

                }

            },

            // 组件名字不能是以后标签名字

            div: {

                template: "#gen"

            },

            sayCall: {

                template: "#memeda"

            }

        }

    });

    //Vue当中组件分为两种：

    //局部组件务必需要在compoents里面进行注册

    //使用的时候当中一个自定义标签即可

    //注意组件名字不能瞎写：不能是以后标签名字

</script>