```
<body>
<!--
模板界面：html+js
js以什么形式存在？
   指令：Vue自定义属性标签(v-开头)，标签属性值是一个js的表达式
   插值：双大括号表达式，动态显示数据（数据双向绑定，内部关联事件）{{msg}}
理解cue的mvvm的实现
-->
   <div id="app">
<!--双向数据绑定： v-model 显示数据：{{ }}-->
      <input type="text" v-model="msg">{{ msg }}</input>
      <p>Hello {{msg}}</p>
   </div>
<!--引包-->
   <script src="./vue.js"></script>
   <script>
      // 初始化
      const app = new Vue({ //配置对象：属性名为特定名称（看API文档）
                            // 每一个属性就是一个选项
        el:'#app', //选择器定位
          data:{
            msg:'hello',
          }
      });//.$mount('#test') 和el效果相同
   </script>
</body>


<body>
   <div id="app">
      <h1>{{msg}}</h1>
      <h1>{{msg.toUpperCase()}}</h1>
<!--       下面两种方式等价-->
      <h1><a v-bind:href="url">跳转</a></h1>
      <h1><a :href="url">跳转</a></h1>
<!--       <h1><a href="{{url}}">跳转</a></h1> 不行-->

<!--       绑定事件，下面两种方式等同-->
      <button v-on:click="test">点击</button>
      <button @click="test">点击</button>
   </div>
   <script src="./vue.js"></script>
   <script>
      const app = new Vue({
        el:'#app', //选择器定位
          data:{
            msg:'hello',
              url:'http://www.mzitu.com/',
          },
          methods:{
            test (event) {
               alert(event.target.innerText)
            }
          }
      });
   </script>
</body>
```

```
<script type="text/javascript">
    /*
    需求：
        1.给P添加一个新属性，fullname=firstname+'-'+lastname
        2.如果修改了firstname或者lastname，fullname也要随着改变
        3.修改了fullname，firstname和lastname也要随之改变
    */
    /*
    * 关于回调函数的三个问题：
    * 1.函数什么时候执行？
    * 2.用来做什么的？
    * 3.this是谁？
    * */
    /*
    * Object.defineProperty()方法会直接在对象上定义一个属性，或者修改一个现有属性
    * get(){return}  //也叫做getter  指属性的get方法
    * set(value){}  //也叫做setter  指属性的set方法
    * */
    const p = {
        firstName:"A",
        lastName:"B"
    }
    Object.defineProperty(p, 'fullName', { //配置对象 ==>属性描述符
        get(){ //1.当读取属性值的时候自动调用 2.将函数返回值作为属性值  3.此时this是属性所在对象
            return this.firstName+'-'+this.lastName;
        },
        set(value){ //1.修改了属性值的时候调用 2.监视属性值的变化 3.this是此时this是属性所在对象
            //value的值是修改后属性的最新值
            const name = value.split('-')
            this.firstName = name[0];
            this.lastName = name[1]
        }
    })

</script>


<body>
    <!--
    1.计算属性
        在computed属性对象中定义计算属性的方法
        在页面中使用{{}}来显示计算结果
    2.监视属性
        通过vm对象的$watch()方法或watch配置来监听指定的属性
        当属性变化时，回调函数启动，在函数内部进行计算
    3.计算属性高级：
        通过getter和setter实现对数据属性的显示和监视
        计算属性存在缓存，多次读取也只调用一次getter计算
    -->
<div id="demo">
    Firstname:<input type="text" placeholder="Firstname" v-model="firstName"><br>
    Lastname:<input type="text" placeholder="Lastname" v-model="lastName"><br>
    Fullname1(单向):<input type="text" placeholder="Fullname1" v-model="fullName1"><br>
    Fullname2(单向):<input type="text" placeholder="Fullname2" v-model="fullName2"><br>
    Fullname3(双向):<input type="text" placeholder="Fullname3" v-model="fullName3"><br>
</div>
```

```
<script src="./vue.js" type="text/javascript"></script>
<script type="text/javascript">
    const vm = new Vue({
        el: '#demo',
        data: {
            firstName:'A',
            lastName:'B',
            fullName2:'A-B'
        },
        //计算属性
        computed:{
            fullName1(){ //相当于属性的get方法，相当于下方因为只有get方法的一种简写
                return this.firstName + '-' + this.lastName  //1.初始会触发一次，后依赖数据发生变化
            },
            fullName3:{
                get(){
                    return this.firstName + '-' + this.lastName
                },
                set(value){ //监视当前属性值变化
                    const name = value.split('-');
                    this.firstName = name[0];
                    this.lastName = name[1]
                }
            }
        },
        //  语法 1
        watch:{
            //监视firstName  firstName的值发生变化的时候调用
            firstName (value) {
                //更新fullName2
                this.fullName2 = value + '-' + this.lastName
            },
        }
    })
    //  语法 2
    vm.$watch('lastName', function (value) {
        //更新fullName2
        this.fullName2 = this.firstName + '-' + value
    })
/*
* 1.Vue控制的所有回调的this都是vm/组件对象
* 2.vue给vm定义一些与data中的属性对应的属性
*   同名
*   getter方法：当通过vm读取属性值的时候触发，读取data对象中同名的属性值
*   setter方法：当通过vm.xxx = value制定新的值时触发，值就会保存在data对应的属性上
*   {上面的方式叫做：数据代理  vm_data.xxx ==>vm.xxx 通过vm代理对vm内部中的data属性的操作}
* */
</script>
</body>


<body>
<!--
    1.理解
        在应用界面中，某些（个）元素的样式是变化的
        class和style绑定就是专门实现动态央视效果的技术
    2.class绑定    :class=""
        class内部可以是对象，字符串，数组
```

```
    3.style绑定   :style="{ color:activeColor , fonSize: fontSize + 'px'}"
       其中activeColor，fontSize为data的属性值
    -->

<div id="demo">
    <h2>class绑定：xxx</h2>
    <p :class="myClass">xxx</p>
<!--什么时候使用对象形式？类名确定，但不确定有咩有   -->
<!--   动静类名可以同时存在，并集-->
    <p class="classC" :class="{ classA: hasA, classB: hasB}">yyy</p>
    <p :class="['classA', 'classB']">yyy</p>

    <h2>style绑定</h2>
    <p :style="{color:activeColor, fonSize:fontSize + 'px'}"></p>
<!--<button v-on:click=""></button>-->
<button @click="myButton" id="update">更新</button>

</div>
<script src="./vue.js"></script>
<script>
    const vm = new Vue({
        el: "#demo",
        data: {
            myClass: "classA",
            hasA:true,
            hasB:false,
            activeColor:'red',
            fontSize:20,
        },
        methods:{
            myButton(){
                // $('#update').attr('class').toggle('classB') 是错的
                this.myClass = 'classB';
                this.hasA = !this.hasA;
                this.hasB = !this.hasB;

            }
        }
    })
</script>
</body>


/*
vue内部是如何监视数据的变化的？
1.对象中的属性数据   （响应式属性）
        setter方法
2.数组中的元素数据
    重写数组中一系列更新元素数组的方法
        1）调用原生函数，对数组进行处理
        2）添加更新页面的措施

*/

axios(config)
    // Send a POST request
    axios({
      method: 'post',
```

```
    url: '/user/12345',
    data: {
      firstName: 'Fred',
      lastName: 'Flintstone'
    }
  });

  // GET request for remote image
  axios({
    method:'get',
    url:'http://bit.ly/2mTM3nY',
    responseType:'stream'
  })
    .then(function (response) {
      response.data.pipe(fs.createWriteStream('ada_lovelace.jpg'))
    });


/*
组件间数据通讯
  props
    父子组件之间的相互通讯


  Vue自定义事件
    给子组件绑定事件监听
    子组件向父组件的通讯方式
    功能类似于function props
    不适合隔层组件和兄弟组件的相互通讯

  全局事件总线
    利用vm对象的$on()/$emit()/$off()
    利用vm对象是组件对象的原型对象
    创建vm对象作为全局事件总线对象保存到vue的原型对象中
      Vue.prototype.$bus = new Vue()
        new Vue()    一般new Vue()使用的是最外层的Vue实例对象  this
          在beforecreate内部设置
        $bus是最随意取名  常用$bus/$eventBus/$globalEventBus
          绑定之后任意组件A可以通过this.$bus.$on()绑定事件监听
          任意组件B都可以通过this.$bus.$emit()分发事件，传递数据

  slot
    父组件向子组件通讯
    通信是带数据的标签
    注意：标签在父组件中解析好

  vuex

  如何与后台通讯
  1）使用什么发ajax请求？
    vue-resource / axios
  2）在什么时候发请求
    mounted()中    //用户打开界面能够看到
    事件监听回调函数或者相关函数中    //用户操作之后数据更新看到
*/

/*
跨域问题？
```

浏览器不能执行其他网站的脚本，由浏览器的同源策略造成的
是浏览器是JS施加的安全限制

同源？
域名，协议，端口皆相同

解决跨域问题？
1）CORS（跨资源共享）

*/

/*
代理服务器的理解和使用
1）代理服务器
能对请求进行转发的工具包
开发环境中：web-dev-server ==>http-proxy-middleware
生产环境中：ngix

2）配置
```
devServer:{
    proxy: {
        'api' : {
            target:'https://...',
            pathRewrite : {
                '^/api' : ' '
            },
            changeOrigin: true
        }
    }
}
```
*/

/*
VueRouter(): //用于创建路由的构建函数
new VueRouter({})

配置项
```
routes:[
    {
        path:'/index',
        component:Index
    },
    {
        ...
    }

    ]
```

在main.js中使用路由模块
```
import VueRouter from 'vue-router'  //引入路由模块
import {routes} from './routes'  //引入静态路由表

Vue.use(VueRouter)  模块使用

const touter = new VueRouter({
    //路由器配置
```

```
        routes : routes
    })

    new Vue({
        el: '#app',
        router,                 //把router实例放入到vue实例中
        render:h =>h(App)
    })


router-link ==> route, component ==> router-view

路由对象的生命周期
    路由组件对象在访问对应路由路径时才会创建
    从A组件跳转到B组件：销毁A组件，创建B组件
    同理，反向跳转时，也有发生销毁和创建组件对象行为

    但是在A组件跳转到A组件（只是改变参数），A组件不会重新创建
*/

/*
    mode: hash
刷新：http://localhost:8080/#/about
    请求后台：http://localhost:8080/返回index.html
    注意：#后边的部分不会交给服务器
    浏览器得到index页面之后就回得到关联的js文件
    js中的路由代码就会将#后边的路由进行解析为前台路由路径

    mode: history
刷新：刷新：http://localhost:8080/#/about
    请求后台：http://localhost:8080/#/about ==>后台无法处理
    返回404
    解决？
        在某个路由路径下刷新，服务器能够返回index页面
        配置：devServer:{
            historyApiFallback：true
        }
*/


/*
移动端适配？
    1）手机出厂网页设置宽度为980*1743

*/
```