一、 ajax笔记

1. ajax介绍

ajax是（asynchronous javascript and xml）的缩写。是一种可以通过js异步操作数据接口的技术。其中的xml统指数据，包括xml、json等。

ajax具有如下优缺点：

- 优点
  - 局部刷新
  - 减少白屏时间，用户体验好
  - 节约带宽资源
- 缺点
  - 没有浏览历史，不能回退
  - 存在跨域问题
  - SEO不友好

2.ajax的 api

1. 核心对象--XMLHttpRequest

XMLHttpRequest为ajax的核心对象，所有的操作都是通过核心对象进行的。在使用ajax时，第一步便是new一个实例化的核心对象。

```
    const xhr = new XMLHttpRequest();
```

2. 设置请求信息--xhr.open(method，URL)

- method:为请求方式，常见的有get、post、put、delete
- URL：为请求的地址，可以写绝对路径；如果服务端设置了静态资源，也可以写相对位置。

```
xhr.open('get','http://127.0.0.1/reg');
xhr.open('post','/reg');
```

3. 传递数据--xhr.send()

- .send（）当中只允许传递字符串
- 最常见的两种传递数据的类型

1. application/json
   当传递的数据为json时，需要用JSON.stringify将传递的json对象转换成字符串。

```
        const hobby = [];
          for(let i=0;i<document.regForm.hobby.length;i++){
               if(document.regForm.hobby[i].checked){
                   hobby.push(document.regForm.hobby[i].value)
                }
          }
        xhr.send(JSON.stringify({
                userName:document.regForm.userName.value,
                sex:document.regForm.sex.value,
                isJieHun:document.regForm.isJieHun.value,
                hobby

            }));
```

2. application/x-www-form-urlencoded

   当传递的数据为x-www-form-urlencoded格式时，需要提前将数据拼接成**a=1&b=2** 这样的字符串格式

```
        let sendStr = "userName="+document.regForm.userName.value+
                      "&isJieHun="+document.regForm.isJieHun.value+
                      "&sex="+document.regForm.sex.value;

        for(let i=0;i<document.regForm.hobby.length;i++){
               if(document.regForm.hobby[i].checked){
                   sendStr += "&hobby="+document.regForm.hobby[i].value;
                }
          }

        xhr.send(sendStr);
```

4. 设置传输数据的请求类型 --xhr.setRequestHeader()

当传递的数据为json时

```
    xhr.setRequestHeader('content-type','application/json')
```

当传递的数据为x-www-form-urlencoded时

```
    xhr.setRequestHeader('content-type','application/x-www-form-urlencoded')
```

5. 获取数据 -- xhr.onload = function(){}

获取从服务端返回的数据，当数据加载完毕后，执行函数体中的内容。

```
    xhr.onload = function (){
      <!-- 获取响应体中的num对应的数据，并将其赋值给页面中btn元素作为文本内容 -->
      const {num} = JSON.parse(xhr.responseText)
      btn.innerText = num;
    }
```

6. xhr.onreadystatechange = function(){}

这是一个监控整个ajax生命周期的函数，数生命周期的状态发生改变时会自动执行；同时也可以接收数据，如果仅仅用于接收数据并对数据进行处理的话，相当于xhr.onload。

```
        xhr.onreadystatechange = function () {
            console.log(xhr.readyState);//获得执行状态
              <!-- 对应的状态表示为：  -->
                // 0 ：初始状态
              // 1 执行了open
              // 2 执行了send
              // 3. 开始接收响应给我的数据
              // 4. 数据接收完毕

            // 当数据接收完毕，执行的操作
            if (xhr.readyState === 4) {
                // 状态码
                console.log('status', xhr.status);
                // 状态字符串
                console.log('statusText', xhr.statusText);

                // 得到响应的内容，根据responseType的类型，决定自身的类型
                console.log(xhr.response);

                // 得到响应体格式为文本的内容
                console.log(xhr.responseText);

                // 获得整体响应头的信息
                console.log(xhr.getAllResponseHeaders());
                // 获得指定响应头的信息
                console.log(xhr.getResponseHeader('conent-type'));
                div.innerText = xhr.responseText
            }
        }
```

7. 指定响应的格式--xhr.responseTtpe

```
    xhr.responseType = 'text';//默认格式为text格式，可以修改成json等其他格式。
```

8. 设置请求时间--xhr.timeout

单位为ms，当请求的时间超过了设定的值，就自动取消请求。

```
xhr.timeout = 1000;
```

9. xhr.ontimeout = function(){}

这是一个事件函数，当请求超时时，自动执行。

```
    xhr.ontimeout = function (){
      div.innerText = '请求超时，请稍后再试'
    }
```

10. xhr.onerror = function(){}

这是一个函数，当ajax发生异常时，自动执行。注意这儿的异常指的是当ajax不能顺利跑通前后端的时候，才算异常；返回404不能算异常。

```
xhr.onerror = function (){
  console.log('发生异常了！')
}
```

11. 取消请求--xhr.abort()

12. xhr.onabort = function (){}

当请求被取消时，自动执行的函数

```
xhr.onabort = function (){
  console.log('请求已被取消')
}
```

3. 跨域

0. 同源策略

同源策略（Same origin policy）是一种约定，它是浏览器最核心也最基本的安全功能，如果缺少了同源策略，则浏览器的正常功能可能都会受到影响。同源策略限制了从同一个源加载的文档或脚本如何与来自另一个源的资源进行交互。

1. 何为跨域

只要请求接口的协议、域名、端口号有一个和当前页面不同，则产生跨域。

2. 如何解决

1. CORS

CORS（Cross-Origin Resource Sharing），跨域资源共享。CORS是官方的跨域解决方案，它的特点是不需要在客户端做任何特殊的操作，完全在服务器中进行处理，支持get和post请求。
CORS是通过设置一个响应头来告诉浏览器，该请求允许跨域，浏览器收到该响应以后就会对响应放行。

```
<!-- 全局设置-->
//不限制路径、和请求方式
app.all("*",function (req,res,next) {
res.set("Access-Control-Allow-Origin","*");
next();
})

<!--解决某个路径的跨域问题  -->
//允许多个地址访问的方式；如果全局设置了这个就不用设置，在这儿只是为了做练习使用。
// 所有请求的地址都在req.headers.origin中
// 将允许的地址放在一个数组中，形成白名单
const originArr = [
  "http://127.0.0.1",
  "http://localhost"
]
// 如果请求的地址在给定的白名单中，就将其设置给Access-Control-Allow-Origin
```

```
    if(originArr.includes(req.headers.origin)){
      res.set('Access-Control-Allow-Origin',req.headers.origin)
    }
```

2. JSONP
   jsonp是利用script的自带跨域能力来发送请求的，只支持get请求。

```
<!-- 在HTML中的处理 -->
//动态创建script标签
  var scrupt = document.createElement('script');
//设置script的src
  script.src = 'http://127.0.0.1/test?callback = abc'
  //设置回调函数
  function abc(data){
    console.log('data.name')
  }
  //将script添加到body中
  document.body.appendChild(script)


  <!-- 在服务端中的设置 -->
  web.get('/test',(req,res) =>{
    //获取传递过来的回调函数名
    var callback = req.query.callback

    var obj = {name:'孙悟空'}

    //将需要传回的数据放在回调函数中返回
    res.send(`${callback}(${obj})`)
  }
  )
```

4. jquery中的ajax

```
    <button>get</button>
    <button>post</button>
    <button>getJSON</button>
    <button>jsonp</button>


    <script>
    // 第一种方式
        $("button").eq(0).click(function () {
        // 发送get请求，第一个参数是地址，第二个参数是传递的参数，第三个参数是回调函数，第四个是指定类型
        $.get("http://127.0.0.1:8082/jquery",{
            a:1,
            b:2
        },function (res) {
            console.log(res);
        },"text")
        })
        // 第二种方式
      $("button").eq(1).click(function () {
```

```
    $.ajax({
        type:"post",// 如果省略该属性，默认为get
        url:"http://127.0.0.1:8084/jquery",
        headers:{
           "content-type":"application/json"
        },
        data:JSON.stringify({
            a:1,
            b:2
        }),
        success(res){
            console.log(res);
        },
        error(){
            console.log("异常啦")
        },
        dataType:"text",
        timeout:1000
    })
  })
</script>
```

5. jquery中的jsonp

```
//通过getJSON实现
// abc=? 会自动生成一个随机的回调函数名
$.getJSON("https://www.baidu.com/sugrec?abc=?",{
    prod:"pc",
    wd:"三"
},function (res) {//最后传回来的数据在这个回调中执行。
    console.log(res);
})

// 服务端的设置
app.get("/jsonp",function (req,res) {
  console.log(1111111,req.query.abc);
  res.send(req.query.abc+"("+JSON.stringify({
      ok:1
  })+")")
})


// 第二种方式
$("button").eq(3).click(function () {
    $.ajax({
        type:"get",
        url:"https://www.baidu.com/sugrec",
        data:{
            prod:"pc",
            wd:"三"
        },
        jsonp:"cb",// 更改默认的参数名callback
        jsonpCallback:"abcdefg",// 指定名字
        dataType:"jsonp",
        success(res){
            console.log(res);
        }
```

```
        })
    })

  // 定义函数
    function abcdefg(res) {
        console.log(res);
    }
```

二、杂记

1. xml和json的区别

xml：可扩展标记语言，用来传输和存储数据，没有预定义标签（HTML都为自定义标签），全都是自定义标签。

> 定义一个男性

```
json:
{
    "username":"张三",
    "age":45,
    "sex":"男"
}

xml:
<student>
    <username>张三</username>
    <age>45</age>
    <sex>男</sex>
</student>
```

2. IE缓存问题

```
1. 问题描述
ie会从缓存中读取请求结果。
2. 解决办法
在请求的路径上传递一个例如时间戳的随机数的参数。
```

```
xhr.open("get","http://127.0.0.1/test?t="+Date.now());
```

3. 数组的map方法

map时es6提供的一个关于数组扩展的方法，通过该方法可以映射出来一个新的数组。

```
const hobbyEnum = {
"1":"学习",
"2":"抽烟",
"3":"喝酒",
"4":"烫头",
}
const hobby = [ '1', '2',"3" ]
const one = hobby.map( v => hobbyEnum[v]).join('
、')
consle.log(one);//学习、抽烟、喝酒
<!-- 最后处理的之所以不是一个数组，是因为.join将其转化成字符串了。 -->
```

4. 数组的filter方法

返回一个符合筛选条件的数据组成的新的数组。

```
const arr = [1, 2, 3, 4];
console.log(arr.filter(v => v !== 1));//[2,3,4]
console.log(arr);//[1,2,3,4]
```

5. 字符串的补全功能

```
const a = "4";
console.log(a.padStart(4, 0));// 0004 左侧补全功能
console.log(a.padEnd(4, 0));// 4000 左侧补全功能
```