Ajax

主讲：崔译

一、简介

> Asynchronous Javascript And XML
>
> 异步的javascript 和 xml

通过JS 代码，异步的， 向后台发送请求，并且不会刷新页面

后台响应数据后，使用JS代码 操作 、更新 DOM

二、使用JS代码发送ajax请求

朗读并背诵全文，实际开发不使用

```
let doSend = ()=> {
  // 定义xhr对象,用于发送ajax请求对象
  var xhr ;
  // 解决浏览器兼容性问题
  if(XMLHttpRequest)
  {
    // Chrome FF 高版本的IE
    xhr = new XMLHttpRequest();
  }else{
    // 对应低版本ie6
    xhr = new ActiveXObject("Microsoft.XMLHttp");
  }
  // 初始化请求对象
  //xhr.open("请求方式","请求地址");
  //xhr.open("get",'${pageContext.request.contextPath}/ajax?id=3&name=abc');
  xhr.open("post",'${pageContext.request.contextPath}/ajax');
  //如果是post请求，首先要设置一个请求头
  xhr.setRequestHeader("Content-type","application/x-www-form-urlencoded");
  // 发送请求
  //xhr.send(null);
  xhr.send("id=3&name=abc");
  xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200)
    {
      var text = xhr.responseText;
      document.getElementById("d").innerHTML = text;
    }
  }
}
```

三、jquery-ajax

1、`$.get()`

```
function doSend()
{
  // 发送get请求
  //url 请求地址，
  //data 请求参数
  //success function，响应成功执行的方法
  //dataType 响应的数据类型 text 默认值 文本 json 响应json对象 xml 响应xml格式对象
  //$.get(url,data,success,dataType);
  $.get('${pageContext.request.contextPath }/ajax',
        "id=3&name=abcccc",
        function(data){
            // 方法会在响应成功后自动调用
            // 方法参数data 就是后台响应的数据
            $("#d").html(data.message);
            console.log(data);
            console.log(typeof(data))
        },
        'json');
}
```

2、`$.post()`

参数和使用方式与`$.get()`一致，发送的是post请求

```
function doSend()
{
  // 发送post请求
  //url 请求地址，
  //data 请求参数
  //success function，响应成功执行的方法
  //dataType 响应的数据类型 text 默认值 文本 json 响应json对象 xml 响应xml格式对象
  //$.get(url,data,success,dataType);
  $.post('${pageContext.request.contextPath }/ajax',
        "id=3&name=abcccc",
        function(data){
            // 方法会在响应成功后自动调用
            // 方法参数data 就是后台响应的数据
            $("#d").html(data.message);
            console.log(data);
            console.log(typeof(data))
        },
        'json');
}
```

3、`$.getJSON()`

与`$.get()`相比，少了最后一个参数，只能响应JSON格式字符串

```
function doSend()
{
  // 发送get请求
  //url 请求地址，
  //data 请求参数
  //success function，响应成功执行的方法
  //响应的数据类型 json 响应json对象
  $.getJSON('${pageContext.request.contextPath }/ajax',
        "id=3&name=abcccc",
        function(data){
            // 方法会在响应成功后自动调用
            // 方法参数data 就是后台响应的数据
            $("#d").html(data.message);
            console.log(data);
            console.log(typeof(data))
        });
}
```

4、`$(selector).load()`

```
// 将后台相应的数据append到选择器对应的标签中
$("#msg").load('${pageContext.request.contextPath }/checkName',
                    "name="+val);
```

5、`$.ajax()`

```
$.ajax({
  url:'',//请求地址
  //data:'id=3&name=a',//请求参数  可以是字符串，也可以是JSON对象
  data:{
    id:3,
    name:"a"
  },
  type:'post',//请求方式get/post
  dataType:'',//相应数据类型 text/json/xml
  success:function(data)
  {
    // 请求成功的回调函数
    // data 相应的数据
  },
  error:function(e)
  {
    // 请求失败的回调函数
    // e 事件对象
  }


});
```

四、使用`fastjson`

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.9</version>
</dependency>
```

```
String jsonStr = JSON.toJSONString(user);//任意类型的对象
```

五、在MVC框架中返回JSON

```
@RequestMapping("/f")
@ResponseBody
public Map f(HttpServletRequest request,HttpServletResponse response)
{
  Blog b = new Blog(2, "222", "aaaaa", 2, "34e4");
  Map<String, Object> map = new HashMap<String, Object>();
  map.put("data", b);
  map.put("success", true);
  return map;
}
```

六、Ajax 跨域请求

1、URL

> Unique Resource Location 统一资源定位符
>
> 唯一的表示一个网络资源（请求，文件，....）

一个完整的url

> 协议://ip:端口/resource..../xx?xxx=yyy
>
> http://localhost:8080/ajax/load?id=3
>
> jdbc:mysql://localhost:3306/mybatis?use.....&...

2、域 和 跨域

- 域：

  域是由 URL 中的 协议 ip 端口 组成

- 跨域：

  两个URL 的 域中的三个组成部分 至少有一个不相同

> 源
>
> http://localhost:8080/mm
>
> 目标
>
> http://localhost:8080/nn (false)
>
> http://localhost:8081/mm （ true)
>
> https://localhost:8080/mm （ true)
>
> http://127.0.0.1:8080/mm （ true)

3、浏览器的同源策略

是一种浏览器的 安全策略，具体指：

浏览器 限制 XMLHttpRequest 向另一个域发送请求

即：ajax 本身 无法发送跨域请求

> No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'http://127.0.0.1:8080' is therefore not allowed access.

4、作用

获取共享数据

聚合数据平台

百度API Store

5、CORS 实现跨域

> Cross-Origin Resource Sharing 跨域资源共享

5-1 实现方式

- 浏览器 (JS)

  整个跨域过程，不需要程序员介入，由浏览器自动完成

  当浏览器发现在发送跨域请求时，会在请求头中添加一行信息

  > Origin: http://127.0.0.1:8080
  >
  > 告诉服务器，该请求是跨域请求，发出域是： http://127.0.0.1:8080

- 服务器

  在Servlet中添加下面代码

```
// 添加响应头，允许 http://127.0.0.1:8080域发送跨域请求
//response.addHeader("Access-Control-Allow-Origin", "http://127.0.0.1:8080");
// 添加响应头，允许 所有域发送跨域请求
response.addHeader("Access-Control-Allow-Origin", "*");
```

5-2 特点

- 支持HTTP所有请求，并且实现方式简单
- 降低服务器的安全性，受到浏览器版本限制（到目前为止，基本不存在不支持跨域的浏览器）
- **当使用 `共享数据平台` 时，服务器代码不可控**

6、JSONP 实现跨域

> JSON with padding
>
> JSON的填充

```
function doSend()
{
  $.ajax({
    url:'http://v.juhe.cn/weather/index',
    data:{
      "cityname":"南京",
      "key":""
    },
    type:'get',
    dataType:'jsonp',
    success:function(data)
    {
      var temp = data.result.today.temperature;
      var weat = data.result.today.weather;
      $("div").html(temp+","+weat)
    }
  });
}
```

7、script标签实现跨域

JSONP 实现跨域的 底层原理

浏览器中，很多标签是可以跨域的(script/link/a/img/form。。。)

```
function doSend()
{
  // 发送ajax跨域请求
  var $sc = $("<script>");
  $sc.attr("src","http://127.0.0.1:8080/10-ajax/cos?callback=abc");
  $("head").append($sc);
}

function abc(data)
{
  console.log(data.name);
}
```

```
String cb = request.getParameter("callback");

User user = new User(1, "zhangsan", "123", 44);

PrintWriter out = response.getWriter();
String str = JSON.toJSONString(user);
str = cb + "(" + str + ")";
out.print(str);
```