Ajax（异步javascript和xml）

服务器软件：

apache：开源的web服务器软件，免费，目前使用人数最多

IIS：微软的web服务器，之前不开源，目前开源

tomcat：web服务器软件

Nginx：web服务器软件

ajax也分get和post

后端决定ajax提交方式（get/post）、数据变量名

前端写请求报文

ajax前端部分

1. 注册事件函数
2. 创建XMLHttpRequest对象
3. 设置请求行
4. 设置请求头
5. 注册回调函数
6. 发送请求

ajax后端部分

1. 接受提交的数据
2. 处理请求
3. 返回响应报文

ajax本质：使用js和xml异步发送http请求和处理返回的数据

通过触发js事件函数，创建xmlhttprequest对象（该对象模拟http协议的get和post请求）发送该模拟请求报文到服务器，服务器处理完数据发送响应报文回浏览器，再通过js回调函数处理返回的数据展示到浏览器上

ajax流程

前端：

发送ajax请求

后端：

读取

返回格式化的数据（如xml、json）

前端：

回调函数解析并展示数据

XMLHttpRequest对象

//创建xmlhttprequest对象

var xhr=XMLHttpRequest();

//设置请求行，参数1：提交方式：get/post，参数2：url，参数3：是否为异步，默认为true异步

xhr.open(‘get/post’, ‘url?key=value/url&key2=value2’, true/false);

//回调函数用onreadystatechange**①**所有浏览器都支持

xhr.onreadystatechange = function() {

if (xhr.readyState==4&&xhr.statu==200) {

数据处理代码;

}

}

//发送请求报文，若方式为post则写入键值对，否则为空或null

xhr.send(null/ ’key=value&key2=value2’);

注：①onreadystatechange事件所有浏览器都支持，但是需要写判断，readyState状态码分为五种：0/1/2/3/4，其中4为完成状态，所以判断if xhr.readyState==4，但是只判断readyState还不够，还需要判断xhr.status==200（页面存在且成功连接）再执行数据处理代码

ajax发送get请求的前端部分

document.querySelector(‘xxx’).onclick = function() { //请求触发事件函数

var xhr=new XMLHttpRequest(); //创建异步对象

xhr.open(‘get’, ’url?name=xxx&age=20’); //设置请求行，get，get请求将数据拼接 在url中

xhr.setRequestHeader(‘cookie’, ‘xxxxxxxx’); //请求头setRequestHeader，‘参数1’， ‘值1’，get请求可以省略设置请求头

xhr.onload=function() { //注册回调函数，响应回来后触发，处理返回的数据

console.log(‘请求响应回来了’);

xhr.responseText; //取用返回的数据，字符串格式

}

xhr.send(null) //请求主体发送，不发送数据直接空或者null

}

ajax发送post请求的前端部分

document.querySelector(‘xxx’).onclick = function() { //请求触发事件函数

var xhr=new XMLHttpRequest(); //创建异步对象

xhr.open(‘post’, ‘url’); //设置请求行，post，url

xhr.setRequestHeader(‘Content-type’, ‘application/x-www-form-urlencode’);

//设置请求头，让ajax像html表单那样发送数据

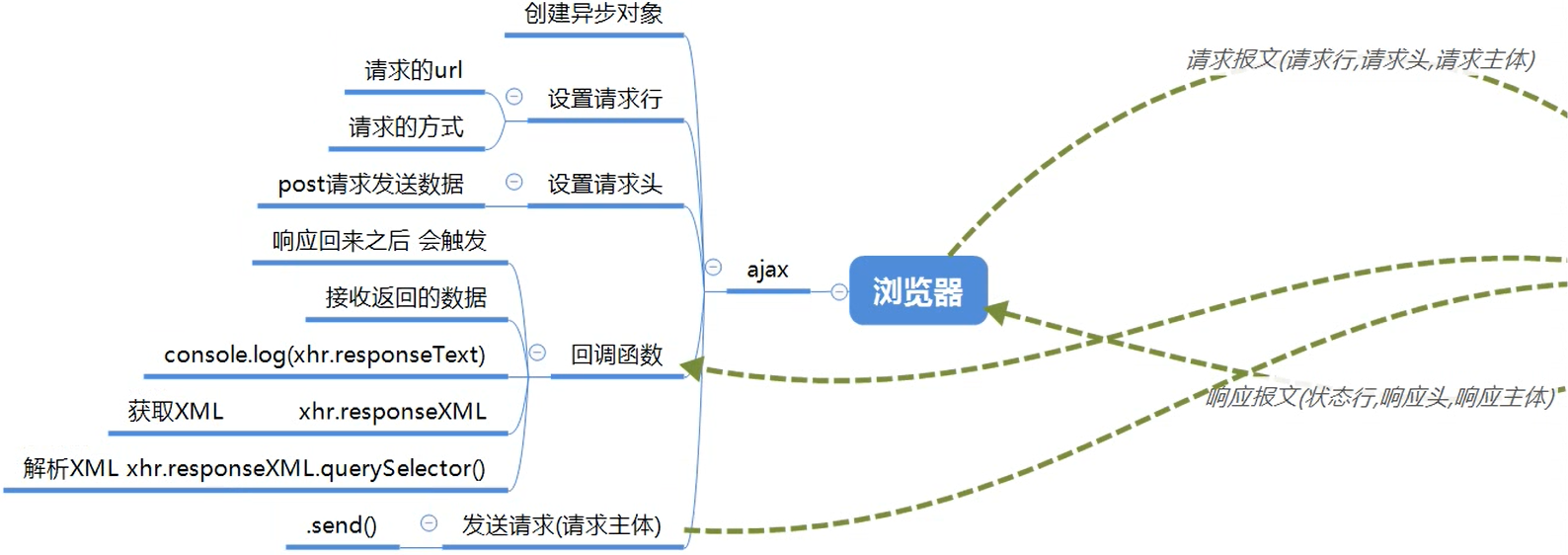
xhr.onload = function() { //注册回调函数，处理返回的数据

console.log(‘xxx’);

}

xhr.send(‘key=value&key2=value2’); //请求主体发送，将数据写在send 中，格式为字符串‘key=value&key2=value2’内容不要加空格

}



ajax流程图

服务器端返回复杂数据

1. xml格式：xxx.xml，相当于一个document

语法：

<?xml version=’1.0’ encoding=’UTF-8’> //声明，可不写

<root> //根标签，必须有，名字自定义

<name>xxx</name>

<age>20</age>

...

</root>

注：

1. xml内标签必须成对
2. 标签名随意，但是最好有意义，类似于编程语言内的变量名

前端解析xml文件：

1. xhr.responseXML.querySelector(‘xxx’)
2. var xxx=标签.innerHTML //对于大量数据需要使用循环进行解析
3. 创建标签来装解析出来的数据

XML缺点：

传输的数据量大，但有效数据比不高

解析复杂

1. json格式：xxx.json

json简介：

1. json是一种数据格式
2. json跟编程语言没关系
3. json的载体是字符串，解析数据用
4. 基本上所有的编程语言都支持json
5. 语法简洁，基本上所有编程语言都提供了方法解析json
6. json格式的字符串转化完毕之后会变成数组对象

json写法1：对象格式

字符串格式：用 ’ ’包裹

对象使用 { }

键名必须使用 ” ”包裹（类似python的字典）

属性值必须使用 ” ”包裹

键值对形式 ”key”: ”value”

var jsonobject=’{“key1”: “value1”, “key2”: ”value2”}’;

转化为对应的对象或数组

var obj=JSON.parse(jsonobject);

取用数据

obj.key1;

json写法2：数组格式

var jsonarr = ’[“value1”, ”value2”, “value3”]’;

转化为对应的数组

var arr = JSON.parse(jsona rr);

取用数据

arr[x];

json写法3：对象数组混合

var jsonobjarr = ‘{“key1”: [“value1”, “value2”, “value3”]}’;

var objarr = JSON.parse(jsonobjarr);

或

var jsonobjarr = ‘[{“key1”: “value1”}, {“key2”: “value2”}]’;

var objarr = JSON.parse(jsonobjarr);

json文件：xxx.json

语法：

[

{

“xxx”: “value”,

“xxx2”: “value2”,

},

{

“xxx”: “value”,

“xxx2”: “value2”,

}

]

获取其他网站的ajax数据

开发者界面→network→XHR选项→挨个点开看看→复制到json格式化网站检验

ajax封装

将相同的部分写死，不同的部分写成参数：url、data、type、回调函数。

只传递一个参数：将参数写成对象格式：option.url、option.data、option.type、option.回调函数

判断数据类型：通过xhr.getResponseHeader(‘Content-Type’)判断回传的数据是何种格式（xml、json、普通字符串）工具函数：字符串. indexOf(‘xxx’)，没有包含’xxx’，则返回-1

var type = xhr.getResponseHeader(‘Content-Type’);

// 是否为json

if (type.indexOf(‘json’) != -1) {

回调函数(JSON.parse(xhr.responseText));

}

// 是否为xml

else if (type.indexOf(‘xml’) != -1) {

回调函数(xhr.responseXML);

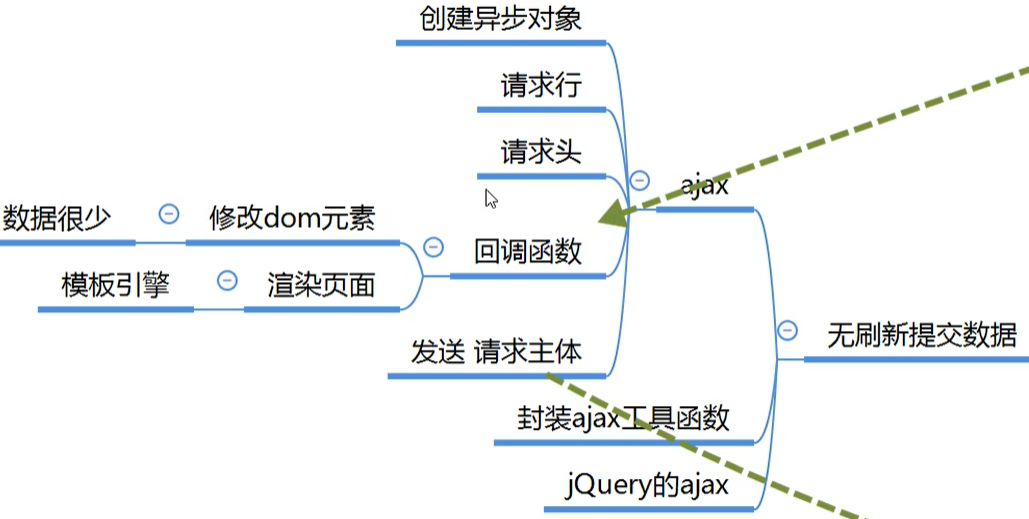
}

// 普通字符串

else {

回调函数(xhr.responseText);

}



ajax数据多时用模板

跨域问题（跟ajax无关）

同源：协议名一样，地址一样，端口号一样称之为同源，即同一服务器内的同一文件夹下

不同源：上述三条存在一条不一样即为不同源

跨域：不同源的网站之间发送请求称为跨域，浏览器默认限制跨域访问

解决办法：

1. cors（cross origin resource sharing）html5才支持，最新方法，未来将普及，需在接收请求方的服务器端设置请求头’Access-Control-Allow-Origin: \*’
2. jsonp（json with padding）dom元素的src属性支持跨域获取资源，jsonp就是利用<script>标签的src属性支持跨域获取资源，跟ajax没关系。老方法，只能用get，存在安全问题，未来将被取代
3. 先写好一个用来处理返回的数据的callback函数
4. 前端：<script src=’url?callback=xxx’></script>
5. 服务器收到请求后拼接数据和函数名字符串返回给前端

fcname=get(‘callback’)

return fcname+’(’+data+’)’

4>前端收到响应报文后会将返回的数据按js代码解析执行