**AJAX**

[1. Ajax 概述 1-1](#_Toc19549465)

[1.1. Ajax 是什么 1-1](#_Toc19549466)

[1.2. Ajax 解决了什么问题 1-2](#_Toc19549467)

[1.3. Ajax 应用模型 1-2](#_Toc19549468)

[2. Ajax 编程实现 2-3](#_Toc19549469)

[2.1. Ajax 编程步骤 2-3](#_Toc19549470)

[2.2. Ajax编程实现（了解） 2-4](#_Toc19549471)

[2.2.1. Ajax 异步GET请求 2-4](#_Toc19549472)

[2.2.2. Ajax 异步post请求 2-6](#_Toc19549473)

[3. JQuery中的Ajax应用 3-6](#_Toc19549474)

[3.1. ajax 函数应用 3-6](#_Toc19549475)

[3.2. load函数应用 3-7](#_Toc19549476)

[3.3. getJson函数应用 3-8](#_Toc19549477)

[3.4. post 函数应用 3-8](#_Toc19549478)

[3.1. 。。。。 3-8](#_Toc19549479)

[4. 手写Ajax框架 4-8](#_Toc19549480)

[4.1. 基本设计思路 4-8](#_Toc19549481)

[4.1. 简易JS框架实现 4-9](#_Toc19549482)

[5. 总结 5-10](#_Toc19549483)

[5.1. 重点和难点分析 5-10](#_Toc19549484)

[5.2. 常见FAQ 5-10](#_Toc19549485)

# Ajax 概述

## Ajax 是什么

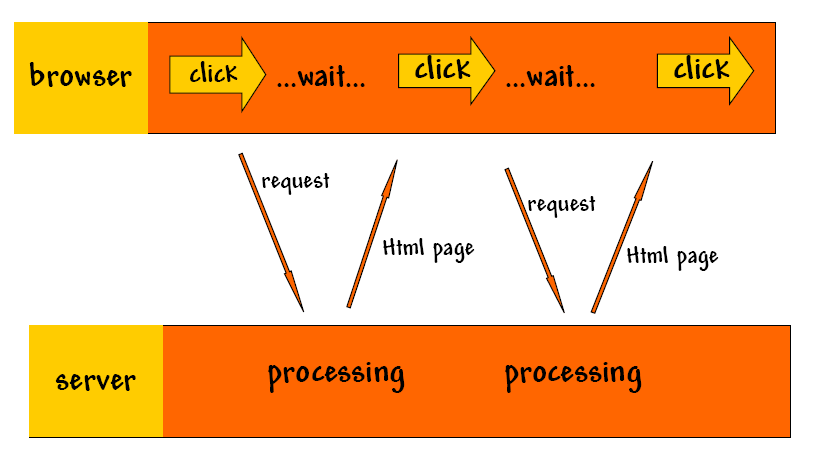
Ajax 即 Asynchronous Javascript And XML，是WEB应用程序开发的一种方法，它使用客户端脚本与web 服务器交互数据，并由最初的全局页面同步加载刷新升级为异步局部加载刷新方式动态刷新页面。

## Ajax 解决了什么问题

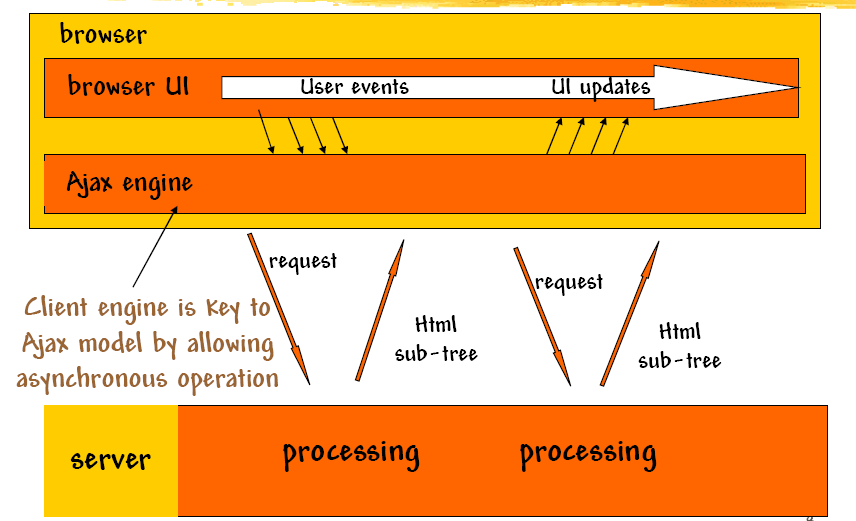
* 页面加载性能问题(刷新整个页面快还是刷新局部页面快?)
* 用户体验问题(页面加载速度快了)

## Ajax 应用模型

传统的web同步模型(多个请求要顺序执行)。



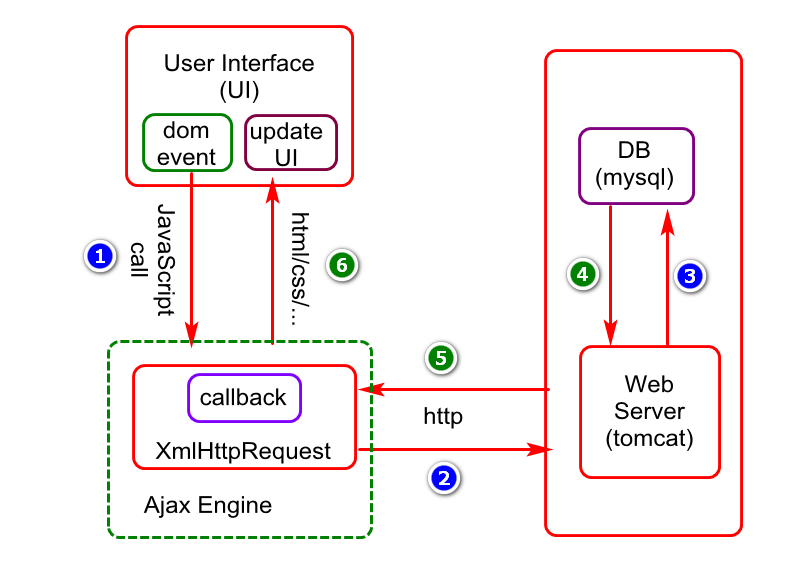
Ajax异步Web应用模型：(多线程并发,尽量减少客户端的阻塞)



# Ajax 编程实现

## Ajax 编程步骤

AJAX 请求基本架构



AJAX请求基本步骤：

Step01：获取Ajax 请求对象(由浏览器端的Ajax引擎提供)

Step02：设置Ajax对象的状态监听(通讯和响应状态)

Step03：创建Ajax请求连接（与服务端建立连接）

Step04：发送Ajax异步请求(发送到服务端)

## Ajax编程实现（了解）

### Ajax 异步GET请求

通过ajax请求异步获取服务端日志信息，并局部刷新页面

Step01：设置dom状态监听

window.onload=**function**(){

doGetObjects();

console.log("hello ajax");

}//dom事件

Step02：定义事件处理函数通过ajax获取数据

**function** doGetObjects(){

//1.创建ajax请求对象(是实现ajax应用的一个入口对象)

**var** xhr=**new** XMLHttpRequest();

console.log(xhr);

//2.设置ajax对象的状态监听(响应是否结束)

xhr.onreadystatechange=**function**(){

//基于请求响应状态处理结果

//4表示服务端响应结束,200表示服务端响应OK(正确)

**if**(xhr.readyState==4&&xhr.status==200){

//responseText为服务端响应的文本

console.log(xhr.responseText);//string

doHandleResponseResult(xhr.responseText);

}

}

//3.发送ajax请求

doSendGetRequest(xhr)

}

发送GET请求

//发送GET请求

**function** doSendGetRequest(xhr){

//3.1.建立与服务端的连接

**var** url="doFindPageObjects.do?pageCurrent=1"

xhr.open("GET",url,**true**);//true表示异步(底层会启动工作线程)

//3.2.发送请求

xhr.send(**null**);//get请求此位置不传数据

}

数据处理函数定义

**function** doHandleResponseResult(resultStr){

//1.将json格式的字符串转换json格式的JavaScript object

**var** result=JSON.parse(resultStr);

console.log(result);

//2.处理数据

**if**(result.state==1){

//将记录显示到tbody中

doSetTableBodyRows(result.data.records);

}**else**{

alert(result.message);

}

}

显示数据方法定义

//在tbody中显示记录信息

**function** doSetTableBodyRows(records){

//1.获取tbody对象，并清空内容

**var** tBody=document.getElementById("tbodyId");

tBody.innerHTML="";

//2.将records数据追加tbody中

**for**(**var** i=0;i<records.length;i++){//一条记录一个tr对象

//1.构建tr对象

**var** tr=document.createElement("tr");

//2.构建多个td对象,并追加到tr中

doCreateTds(tr,records[i]);

//3.将tr追加到tbody中

tBody.appendChild(tr);

}

}

//创建当前的td对象

**function** doCreateTds(tr,row){

**var** td=document.createElement("td");

td.innerText=row.id;

tr.appendChild(td);

td=document.createElement("td");

td.innerText=row.username;

tr.appendChild(td);

td=document.createElement("td");

td.innerText=row.method;

tr.appendChild(td);

}

### Ajax 异步post请求

发送POST请求，可以将此方法替换发送GET请求的方法

**function** doSendPostRequest(xhr){

//3.1.建立与服务端的连接

**var** url="doFindPageObjects.do"

xhr.open("POST",url,**true**);//true表示异步(底层会启动工作线程)

//post请求必须设置此请求头

xhr.setRequestHeader("Content-Type",

"application/x-www-form-urlencoded");

//3.2.发送请求

xhr.send("pageCurrent=1");//post请求此位置传数据

}

# JQuery中的Ajax应用

## ajax 函数应用

juqery 中的ajax函数用于向服务端发起一个异步的ajax请求.

例如:

$.ajax({ url: "address",

type:”GET”,

data: { param1: "foo bar", param2: "baz“ },

dataType: "json",

success: successHandlerFunction,

error: errorHandlerFunction,

cache: false,

})

Ajax 函数应用案例

$(function() {

$("#data-button-2").click(processAjaxRequest);

});

function processAjaxRequest () {

var params ={ param1: $("#field3").val(),param2: $("#field4").val() };

$.ajax({ url: "doFindPageObjects.do",

data: params,

success: function(result){

……..

} );

}

## load函数应用

jquery 中的load函数一般用于在某个位置异步加载某个

url页面,

语法:load(url,[data],[callback])

其中:

1. url 表示远程一个地址
2. data 表示要传递数据(key/value)
3. callback 表示一个回调函数

例如: 案例1

$("#time-result-1").load("listUI.do");

例如: 案例2

$("body").load("listUI.do",{color:red},function(){

…..

});

## getJson函数应用

Jquery中的getJSON函数用于向服务端发起Ajax GET请求

获取json格式的数据

语法:

$.getJSON("url",[data], [callback])

例如:

$.getJSON(“doFindPageObjects.do”,{pageCurrent:1},function(result){

//处理返回结果

})

## post 函数应用

POST方法为Jquery中的发送Ajax post请求的一种方法.

语法:post(url,[data],[callback])

例如:

$.post("doFindPageObjects.do",{pageCurrent:1},function(result){

…..

})

## 。。。。

# 手写Ajax框架

## 基本设计思路

通过原生js实现对ajax操作的封装

## 简易JS框架实现

关键代码参考如下：

(**function**(){

**var** jsFramework=**function**(){}

//动态为jsFramework添加一个getJSON函数

//prototype 动态添加函数时使用

jsFramework.prototype.getJSON=**function**(url,params,callback){

//1.构建XHR对象

**var** xhr=**new** XMLHttpRequest();

//2.注册事件监听

xhr.onreadystatechange=**function**(){

//readyState=4表示服务端响应结束

//status=200表示服务端响应ok

**if**(xhr.readyState==4&&xhr.status==200){

**var** jsonObj=JSON.parse(xhr.responseText);

callback(jsonObj);

}

}//callback

//3.建立连接

**let** getUrl=url+"?"+params;

xhr.open("GET",getUrl,**true**);

//4.发送请求

xhr.send(**null**);//get请求send方法不传数据

}

**var** obj=**new** jsFramework();

//将局部变量赋值一个全局变量

window.$=obj;

})()

# 总结

## 重点和难点分析

## 常见FAQ

1. XMLHttpRequest对象中的状态含义？
2. readyState状态说明(了解s)
3. 0状态:表示xmlHttp对象刚创建，还没有调用open()方法。
4. 1状态：请求开始，调用了open方法，但还没有调用send方法
5. 2状态：调用完了send方法
6. 3状态：服务器已经开始响应，但不表示响应结束了。
7. 4状态：服务器响应结束(通常我们只关心这个状态)
8. Status状态说明(了解)
9. 200：表示服务端响应OK
10. 500：表示服务端响应异常
11. XMLHttpRequest对象如何获取服务端响应的数据？
12. responseText 获取服务端返回的文本内容
13. responseXML 获取服务端返回的xml内容(包括html)