# Ajax

Asynchronous JavaScript And XML：异步JavaScript和XML，不是一门编程语言，指的是一种前后端的交互方式

：客户端与服务端交换数据并更新在局部页面的技术，不需要重新加载整个页面。

Ajax的核心就是异步加载或者叫局部刷新。

需求：

1.点击提交按钮，向服务端发请求，等待响应。

2.同时在input框输入信息。

3.服务端返回“Hello”字符串，在页面进行展示。

传统方式会刷新整个页面：

点击提交按钮后，正在input框输入内容的时候，服务端响应返回，会以重新加载整个页面的方式展示结果，同步加载，所以input框内已经输入的内容被全部清空。

Ajax局部刷新：

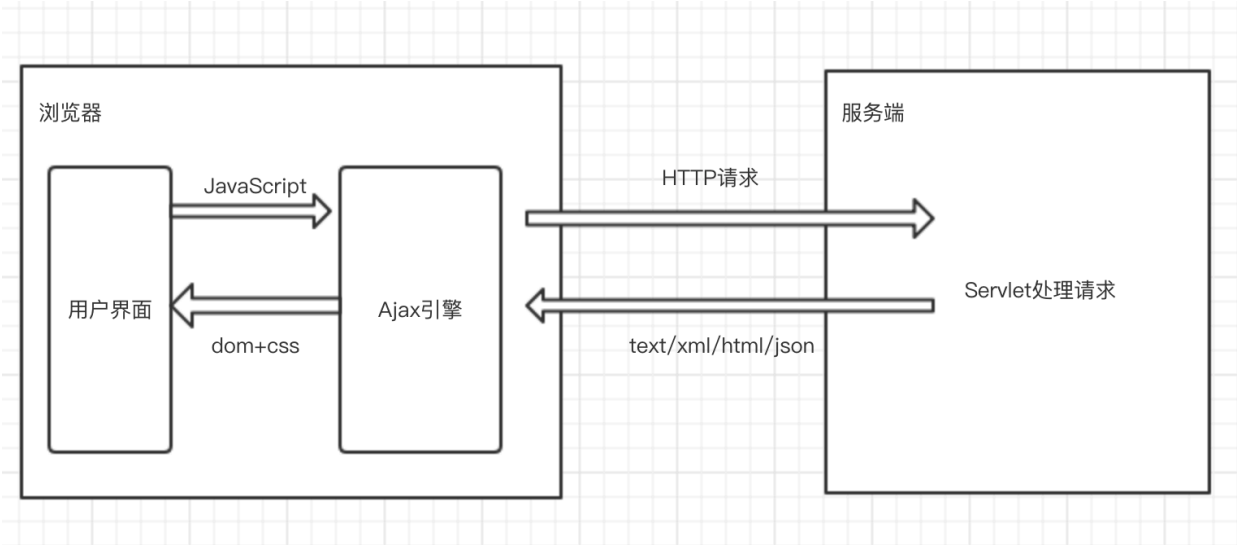
点击提交按钮后，在input框输入内容，服务端响应返回，只是动态的刷新要展示结果的局部网页，并不会影响到input框的输入，结果展示和输入是同时在进行的，互不干扰，异步加载。

传统web数据交互和Ajax数据交互的区别：

- 客户端请求方式不同：
  - 传统：浏览器发送同步请求。
  - Ajax：异步引擎对象发送异步请求。
- 服务端响应方式不同：
  - 传统：响应一个完整页面（JSP）。
  - Ajax：只响应需要的数据。
- 客户端处理方式不同：
  - 传统：需要等待服务端响应完成并重新加载整个页面之后，用户才可以进行后续的操作。
  - Ajax：动态更新页面中的局部内容，并不会影响到用户在当前页面的其他操作。

Ajax原理：

Ajax的工作原理相当于在客户端和服务端之间加了一个中间层，使用户操作与服务端响应异步化。并不是所有的请求都之间提交给服务端，类似一些数据验证和数据处理等都交给Ajax引擎自己来完成，只有确定需要从服务端读取新数据时再由Ajax引擎代为向服务端提交请求。




Ajax技术的核心：XMLHttpRequest

常用方法：

1.open() 创建一个新的HTTP请求。

2.send() 将请求发送到服务端。

常用事件：

onreadystatechange：指定回调函数。

常用属性：

readyState：XMLHttpRequest的状态信息。

状态码表示的意义：

0:XMLHttpRequest对象未完成初始化。

1:XMLHttpRequest对象开始发送请求。

2:XMLHttpRequest对象的请求发送完成。

3:XMLHttpRequest对象开始读取响应。

4:XMLHttpRequest对象读取响应结束。

JavaScript传统方式调用Ajax有很多不足，步骤繁琐，方法，属性多不便记忆，存在浏览器兼容性问题等。jQuery对Ajax进行了封装，屏蔽了很多底层代码，让开发者可以更加方便快捷的进行Ajax开发。

```
function validation() {
    var name = document.getElementById("name").value;
    //创建XMLHttpRequest对象
    xmlHttpRequest = createXMLHttpRequest();
    //设置回调函数
    xmlHttpRequest.onreadystatechange = callback;
    //初始化XMLHttpRequest组件
```

```
    var url = "user.do?name="+name;
    xmlHttpRequest.open("POST",url);
    xmlHttpRequest.send(null);
}

//创建XMLHttpRequest对象
function createXMLHttpRequest(){
    if(window.XMLHttpRequest){
        return new XMLHttpRequest();
    }else{
        return new ActiveXObject("Microsoft.XMLHTTP");
    }
}

//定义回调函数
function callback(){
    if(xmlHttpRequest.readyState == 4 && xmlHttpRequest.status == 200){
        var data = xmlHttpRequest.responseText;
        if(data == "true"){
            document.getElementById("info").innerHTML = "用户名已存在";
        }else{
            document.getElementById("info").innerHTML = "用户名可以使用";
        }
    }
}
```

语法：

$.ajax([settings]);

常用的属性参数：

- url：要请求的服务端资源。
- type：请求类型，默认是GET。
- data：发送到服务端的参数。
- dataType：服务端返回的数据类型（text，json，html，xml）

常用函数参数：

- success：请求成功的回调函数。
- error：请求失败的回调函数。
- complete：请求完成的回调函数（无论成功或失败都会调用）。

```
function validation() {
    var name = $("#name").val();
    $.ajax({
        url:"user.do",
        type:"POST",
        data:"name="+name,
```

```
            dataType:"text",
            success:function(data){
                if(data == "true"){
                    $("#info").html("用户名已存在");
                }else{
                    $("#info").html("用户名可以使用");
                }
            }
        });
    }
```

Ajax返回JSON格式数据

var user = {

id:1,

name:"张三",

pwd:123

}

```
$(function () {
    $.ajax({
        url:"json.do",
        type:"POST",
        data:"",
        dataType:"JSON",
        success:function (data) {
            // $("#id").text(data.id);
            // $("#name").text(data.name);
            // $("#pwd").text(data.pwd);
            $("#num").text(data.num);
            $("#name2").text(data.name);
            $("#age").text(data.age);
            $("#address").text(data.address);
        }
    });
});
```

```
resp.setContentType("text/html;charset=UTF-8");
User user = new User();
user.setId(1);
user.setName("张三");
```

```
user.setPwd(123);

Student student = new Student();
student.setNum(11);
student.setName("小明");
student.setAge(22);
student.setAddress("科技路");

JSONObject jsonObject = JSONObject.fromObject(student);
resp.getWriter().print(jsonObject);
```