目录

[小知识 1](#_Toc105076327)

[Ajax 1](#_Toc105076328)

[Ajax运行原理 2](#_Toc105076329)

[实现步骤 2](#_Toc105076330)

[服务器端响应的数据类型 2](#_Toc105076331)

[Post 请求 3](#_Toc105076332)

[获取服务端的响应 过时 做了解 5](#_Toc105076333)

[onload 监听ajax状态码 5](#_Toc105076334)

[Ajax错误处理 返回的ajax对象返回的 6](#_Toc105076335)

[Ajax请求在 低版本ie浏览器的缓存问题 7](#_Toc105076336)

[同步和异步的概念 8](#_Toc105076337)

[Ajax封装 避免代码的重复 8](#_Toc105076338)

[模板引擎 8](#_Toc105076339)

[验证邮箱合理性 10](#_Toc105076340)

[搜索框自动提示 10](#_Toc105076341)

[省市区 10](#_Toc105076342)

[Fromdeata 作用 11](#_Toc105076343)

## 小知识

onbulr---模糊，漏出 失去焦点时

oninput 输入时

## Ajax

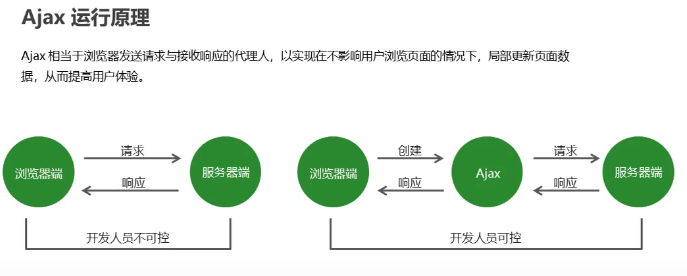
浏览器提供以方，可以实现无刷新更新数据

应用场景

1. 页面上拉加载跟多数据
2. 列表无刷新分页
3. 表单项离开焦点数据验证
4. 搜索框提示文字下拉列表

Ajax技术需要运行在网站环境中才能生效

## Ajax运行原理



## 实现步骤

Xml 服务器返回的格式 很少使用 作为了解性知识点

1.创建Ajax 对象

var xhr=new XMLHttpRequst();

2.Ajax 请求地址和请求方式

xhr.open(‘get’,’http://www.example.com’)

3.发送请求

xhr.send()

4.获取服务器端给客户端的响应数据

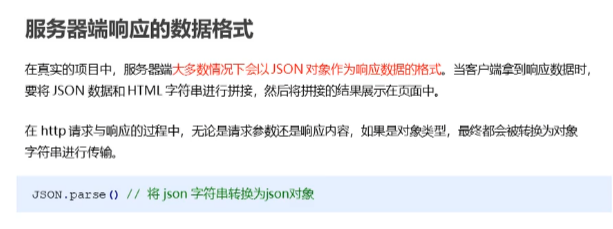
xhr.onload=function(){

console.log(xhr.responseText);

}

## 服务器端响应的数据类型

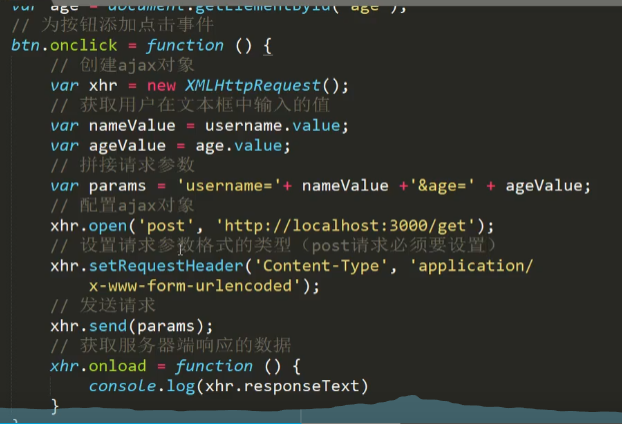
Json数据类型的



## Post 请求

Post 第一个是报文属性 第二个是报文属性的参数





Localhost 本地服务器

Lication 应用程序

Response 回答

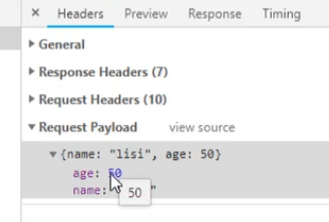
ocation-地方  parser解析器

application 应用

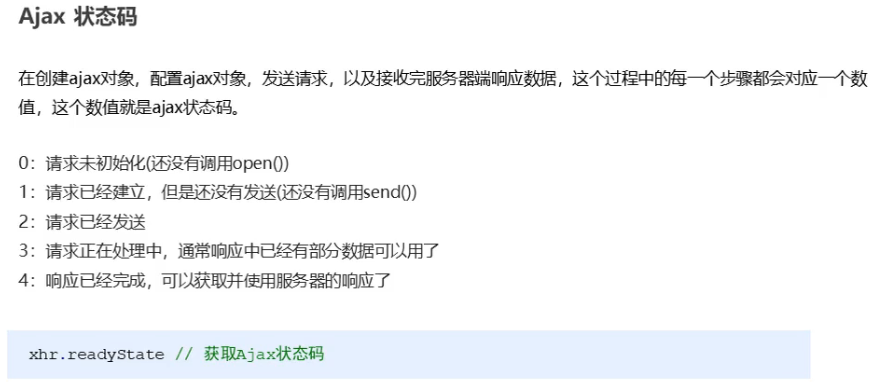








## 获取服务端的响应 过时 做了解



onreadystatechange ajax状态码改变时触发

readystate 事件码

change 改变

## onload 监听ajax状态码

onload事件响应次数只会是一遍 ，但不兼容低版本ie浏览器



## Ajax错误处理 返回的ajax对象返回的

http状态码是服务器端返回的

Status//地位 ---能获取返回状态

网络畅通，服务端能接受请求，服务端返回的结果不是预期结果

判断服务器返回的状态码，分别进行处理 xhr.status 获取http状态码

btn.onclick = () => {

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

    // console.log(xhr);

    xhr.open('get', 'http://127.0.0.1:3000/ajax/errer\_07')

    xhr.send();

    xhr.onload = () => {

        console.log('?');

        // console.log(xhr.responseText)

        // status获取状态码

        console.log(xhr.status)

        console.log(typeof xhr.status)

        if (xhr.status === 400) {

            alert('请求出错!  400')

        } else if (xhr.status === 404) {

            alert('未找到该请求! 404 found')

        } else if (xhr.status === 500) {

            alert('服务器错误! 500')

        } else {

            alert('成功！！')

        }

        // ajax状态码:表示Ajax请求的过程状态 ajax对象返回的

        //http状态码:表示请求的处理结果 是服务器端返回的

    };

    // 但断网时 会触发 onerror事件

    xhr.onerror = () => {

        alert('网络中断!检查网络!')

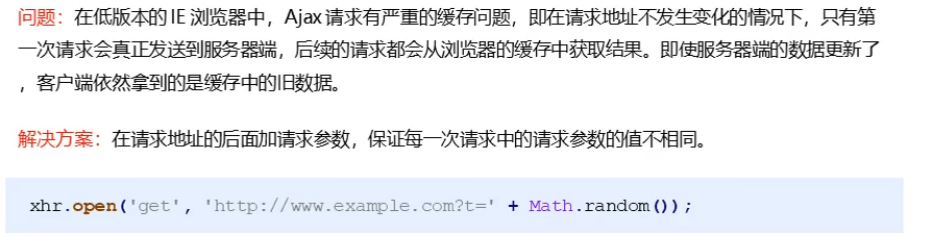
    };

}

## Ajax请求在 低版本ie浏览器的缓存问题

会重复获得

问题:在低版本的ie浏览器中Ajax请求有严重的缓存问题，即在请求地址不发生变化的情况下，只有第一次请求会正真的发送到服务器端，后续的请求都会从浏览器的缓存中获取结果，即便服务器的数据更新了，客户端依然拿到的是缓存中的旧数据



btn.onclick = () => {

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

    // console.log(xhr); //Math.random() 返回一个随机 1到0 的值

    //低版本的ie浏览器需要每次请求不一样  不然不会发送请求至服务器，而是读取本地缓存

    xhr.open('get', 'http://127.0.0.1:3000/ajax/cache\_08?+t=' + Math.random())

    xhr.onreadystatechange = function() {

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

            alert('响应成功!')

            alert(xhr.responseText) //响应内容

            console.log(xhr.responseText);

            // 这个只会返回 ok  错误 statusText

        };

        // console.log(xhr.readyS  tate)

    };

    xhr.send();

};

## 同步和异步的概念

同步 顺序执行代码

异步 在发出请求后，不会等会请求结果 而是跳过继续执行代码，等待请求结果出来后再获取结果 如settimeout 函数等

## Ajax封装 避免代码的重复



xhr.getResponseheader() 获取相应头的type

string.includes(“查找字符串”) 可以获取响应响应头的类型

## 模板引擎

npm installart-template ---save 安装

aui.guthub.io 网站

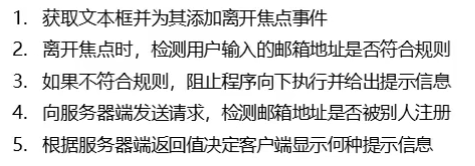
封装好函数函数之后调用 大概也是这样的一个形式



type=text/html 以html方式解析

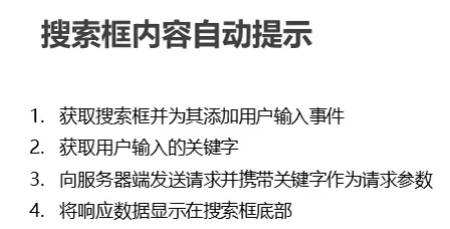


## 验证邮箱合理性



placehoder [ˈpleɪshəʊldə(r)] 占位符 input中属性 预显示

## 搜索框自动提示

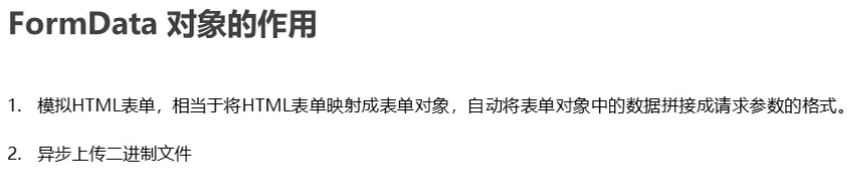


oninput 输入事件

## 省市区



## Fromdeata 作用



使用步骤

