DAY25 node环境和AJAX

2023年5月22日 18:19

nodejs以及前后端交互

一.前端和后端的关系

前端：客户端(浏览器端,依赖浏览器)

后端：服务器端(依赖服务器，安装服务器环境)

二.前端的核心工作

渲染 - 数据驱动视图(渲染视图) - 需要数据(后端)

用户体验

性能优化

三.前后端的通信

1.接口(接口地址)：其实就是后端或者第三方公司提供的一个完整的url地址，里面包含需要的数据。

2.前端发送信息给后端：ajax，form表单，地址栏(地址栏携带数据)

3.前端获取后端的信息：ajax

4.前端获取第三方接口数据：ajax，jsonp

四.地址栏的完整的组成

http://www.baidu.com/about/list/index.html?a=1&b=2&c=3#/home

1.http、https：协议，两台电脑要通信，必须使用协议

2.ip地址：电脑的唯一标识，ip地址由0-255之间的数字，分为四段，10.31.12.13

域名：因为ip地址不方便记忆，产生域名，域名需要申请购买

3.端口：服务器是一台电脑，服务器里面可以存放多套程序，将端口分配给不同的程序，端口的范围是从0 到65535

http默认端口：80

https默认端口：443

4.路径：访问程序的路径

5.携带参数：数据，利用?和&符合进行拼接

6.哈希值：实现单页面应用，#拼接的数据

五.服务器

1.服务器就是一台昂贵的电脑，存放源码的地方。

2.服务器需要环境配置(Nodejs)

3.安装nodejs，电脑里面就有服务器

4.安装nodejs软件的步骤

一路next，不要修改路径

5.安装完成，如何检测是否安装成功

`win+r -- 启动运行窗口 -- 输入cmd回车启动命令行 -- 输入 node -v 输出版本号视为安装成功`

nodeJS以及server目录介绍

一.nodeJS介绍

1.概念:

Node.js 是一个开源的、跨平台的 JavaScript 运行时环境。

Node.js 是基于google浏览器v8引擎的JavaScript的运行环境。

服务器可以脱离浏览器的

2.安装Node.js软件 - 提供共享了

2.1.选择版本

LTS长期支持和维护的版本，成熟的版本，大部分用户都是选择这个。

CURRENT当前最新的版本，测试版本 - 不建议初学者使用。

2.2.在 node 中文网下载的时候，选择安装包，不要选择 二进制文件

2.3.安装Node.js

一路next安装，不要修改路径，自动配置环境变量

环境变量：可以在任意位置启动这个软件服务器，需要配置环境变量

配置环境变量的步骤：此电脑 - 右键属性 - 高级系统设置 - 环境变量 - 系统变量 - path - 编辑

2.4.检测安装环境

通过命令行测试打印软件对应的版本，才表示安装成功，后面的很多环境都是这样的

window win+r cmd node -v或者 node --version 输出对应的版本，视为node安装成功

window win+r cmd npm -v或者 npm --version 输出对应的版本，视为npm安装成功(npm是nodejs的包管理工具，安装nodejs自动安装npm)

如果版本都ok，安装完成。

二.server目录介绍

这是nodejs编写的服务器端接口目录

我们的前后端交互利用这个目录

将目录放到电脑的桌面，一旦涉及到前后端通信，启动servr目录里面的服务器(win点我启动.bat)

接下来的开发必须依赖于这个目录。

ajax的概述

一.ajax的概述

ajax 全名 async javascript and XML 异步的JavaScript和xml

- 是前后台交互的语法
- 也就是我们客户端给服务端发送消息的工具，以及接受响应的工具
- 是一个默认异步执行机制的功能

1.async:异步

javascript是单线程的，只有一个主线程，同步和异步指的是在主线程上执行的顺序

同步：阻塞模式，后面的任务等到前面的任务完成才能执行，进入主线程执行的代码。

异步：非阻塞模式，进入队列，等到主线程上面的同步任务完成才会被通知执行。

2.xml：可扩展的标记语言，标签都是自定义的。

早期json对象还不存在，都是采用xml做数据格式

json完全取代xml,比xml操作更方便。

二.AJAX 的特点

1.不需要插件的支持，原生 js 就可以使用

2.用户体验好（不需要刷新页面就可以更新数据）

3.减轻服务端和带宽的负担

4.缺点： 搜索引擎的支持度不够，因为数据都不在页面上，搜索引擎搜索不到

三.AJAX 的使用

在 js 中有内置的构造函数来创建 ajax 对象(XMLHttpRequest)

创建 ajax 对象以后，我们就使用 ajax 对象的方法去发送请求和接受响应

注意：

发送请求：前端给后端 - request

接收响应：后端给前端 - response

ajax对应的步骤获取接口数据

1.创建ajax对象(利用XMLHttpRequest构造函数)

```
let xhr = new XMLHttpRequest();
```

2.配置请求信息，利用open方法

语法：xhr.open(请求方式,接口地址,是否异步)

请求方式：get/post/put/delete 常见的四种请求方式，当前关注的是get和post

接口地址：后端或者第三方提供的完整的url地址

是否异步：true异步，false同步，默认是异步

```
xhr.open('get', 'http://localhost:8888/goods/list', true);
```

3.发送请求

这一步会产生就绪状态码(数字编码)证明当前这一步是否完成

xhr.readyState:就绪状态码

xhr.readyState===0 表示未初始化完成，也就是open方法还没有执行

xhr.readyState===1 表示配置信息已经完成，也就是执行完open之后

xhr.readyState===2 表示 send 方法已经执行完成

xhr.readyState===3 表示正在解析响应内容

xhr.readyState===4 表示响应内容已经解析完毕，可以在客户端使用了数据

```
xhr.send();
```

4.监听的事件 readystatechange 监听就绪状态码是否改变，只要改变，事件就会触发。

利用xhr.responseText属性获取响应的数据，获取的数据是json格式的字符串

```
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4) {//响应解析全部完成，可以使用数据
    // 利用xhr.responseText属性获取响应的数据，获取的数据是json格式的字符串
    // console.log(xhr.responseText);
    console.log(JSON.parse(xhr.responseText).list);
    重点：渲染商品列表
    let shoppingList = JSON.parse(xhr.responseText).list;//渲染的数据(长度12的数组，里面的每一项都是对象)
    let strHtml = '';//拼接的初始值
    for (let res of shoppingList) {//遍历数组
      strHtml += `
        <li>
          <img src="${res.img_big_logo}" alt="">
          <p>${res.title}</p>
          <span>${res.current_price}</span>
        </li>
      `;
    }
    document.querySelector('.goods ul').innerHTML = strHtml;
  }
};
```

ajax异步操作

一.ajax获取数据的四步曲

1.创建ajax对象(同步)

```
let xhr = new XMLHttpRequest();
```

2.配置请求信息(同步)

```
xhr.open('GET', 'http://localhost:8888/test/first', true);
```

3.发送请求(发送请求是同步的，响应接收是异步的)

```
xhr.send();
```

4.监听就绪状态码，只要改变就会触发

```
xhr.onreadystatechange = function () {//异步的
  if (xhr.readyState === 4) {//响应完成
    console.log(xhr.responseText);
  }
}
```

二.onload

只有处于状态码4，请求已完成，响应已就绪的情况下，才会进入onload。

获取响应的数据

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:8888/test/first', true);
xhr.send();
xhr.onload = function () {
  console.log(xhr.responseText);
}
```

三.同步

浏览器本质上不支持同步的。

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:8888/test/first', false);
xhr.onload = function () {//代码执行过程中，只要就绪状态码为4，触发onload事件
  console.log(xhr.responseText);
}
xhr.send();//同步发送请求，异步响应接收(如果设置为同步这里必须完成)
```

总结：获取响应数据的时候，根据上面的步骤进行，先onload再send，为了同步时也能获取响应数据。

对比：onload和onreadystatechang区别

onload 就绪状态码为4触发，获取响应数据

onreadystatechange 就绪状态码为1，2，3，4都会触发，每个步骤进行细分。如果获取响应数据，需要判断获取

事件很多时候也是异步的，当然不是全部，看具体的事件类型

```
// console.log(1);
// document.onclick = function () {
//   console.log(2);
// }
// document.onclick();//让事件主动触发
// console.log(3);
```

http的状态码

直接通过ajax对象下面的status属性进行获取

概念：是用以表示网页服务器超文本传输协议响应状态的3位数字代码。

1开头 消息类(代表请求已被接受，需要继续处理)

2开头 成功类(这一类型的状态码，代表请求已成功被服务器接收、理解、并接受)

3开头 重定向(这类状态码代表需要客户端采取进一步的操作才能完成请求)

4开头 客户端错误(这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理)

5开头 服务器错误(这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生)

具体的数字代码

如何查看：控制面板 - network - status

101:服务器已经理解了客户端的请求，并将通过Upgrade 消息头通知客户端采用不同的协议来完成这个请求。

200:请求成功

304:如果客户端发送了一个带条件的GET请求且该请求已被允许，而html文档的内容并没有改变，则服务器应当返回这个状态码。

403:服务器已经理解请求，但是拒绝执行它。

404:请求失败，请求所希望得到的资源未被在服务器上发现。

501:服务器不支持当前请求所需要的某个功能。当服务器无法识别请求的方法，并且无法支持其对任何资源的请求。

503:由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:8888/test/third', true);
xhr.onload = function () {
  // if (xhr.status === 200) {
  if (/^2[0-9]{2}$/.test(xhr.status)) {
    console.log(xhr.responseText);
  } else {
    throw new Error('接口地址错误')
  }
}
xhr.send();
```

使用 ajax 发送请求时携带参数

请求方式：

GET(查)

POST(增)

PUT(改)

DELETE(删)

...

1.常用的请求方式：get和post

2.get和post的区别

2.1.语义化：get获取 post传输，发送

2.2.发送数据：get通过地址栏(接口地址)传输数据，利用?和&符合进行拼接，post通过send传输数据，需要设置请求头(内容类型)

```
xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');
```

2.3.数据长度：get仅支持传输2000多个字符(2kb)，post理论上没有上限，可以配置

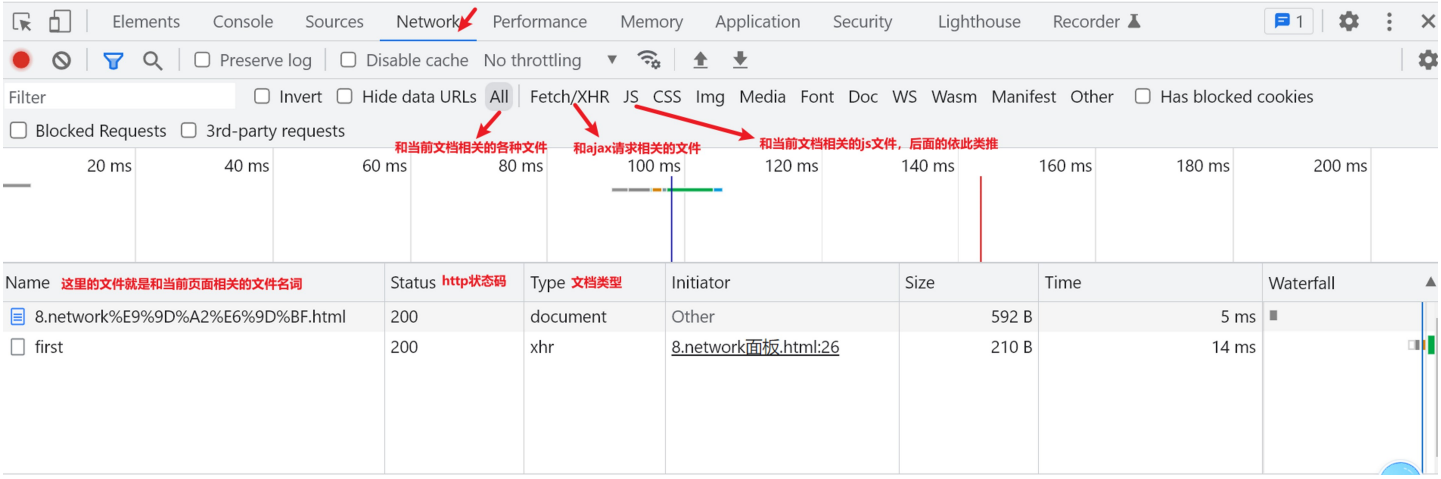
2.4.安全型：get发送数据是明文传输，post是不是明文，安全性比get要高

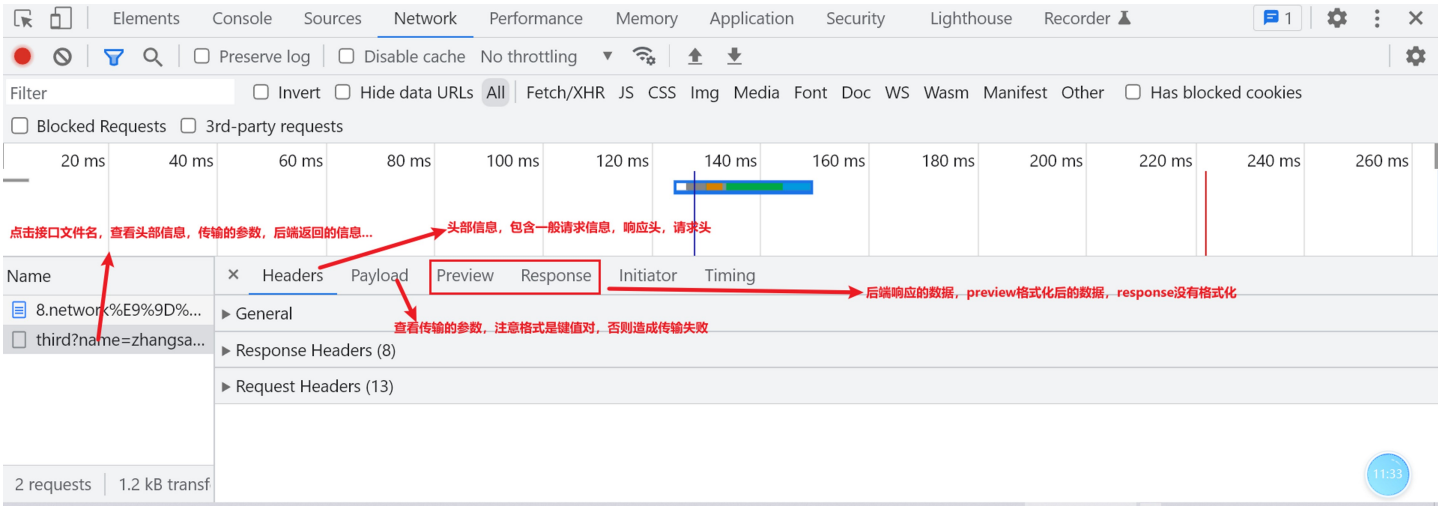
2.5.缓存问题：get有缓存 post没有缓存

```
// get发送数据
let xhr = new XMLHttpRequest();
xhr.open('GET', 'http://localhost:8888/test/third?name=zhangsan&age=18', true);
xhr.onload = function () {
  if (/^2[0-9]{2}$/.test(xhr.status)) {
    console.log(JSON.parse(xhr.responseText));
  } else {
    throw new Error('接口地址错误')
  }
}
xhr.send();

// post发送数据
let xhr = new XMLHttpRequest();
xhr.open('POST', 'http://localhost:8888/test/fourth', true);
xhr.onload = function () {
  if (/^2[0-9]{2}$/.test(xhr.status)) {
    console.log(JSON.parse(xhr.responseText));
  } else {
    throw new Error('接口地址错误');
  }
}
// post发送数据必须设置的
xhr.setRequestHeader('content-type', 'application/x-www-form-urlencoded');
xhr.send('name=wangwu&age=25');
```

network面板





利用ajax测试服务器接口的使用

注意：

1.注意请求方式

2.注意接口地址不能出错

3.注意携带的参数以及参数的格式

4.注意响应的结果

```
// 测试
// let xhr = new XMLHttpRequest();
// xhr.open('GET', 'http://localhost:8888/goods/category');
// xhr.onload = function () {
//   if (xhr.status === 200) {
//     console.log(JSON.parse(xhr.responseText));
//   }
// };
// xhr.send();

// 测试xml接口
// ajax获取任意文件内部的数据

// let xhr = new XMLHttpRequest();
// xhr.open('GET', './data.xml');
// xhr.onload = function () {
//   if (xhr.status === 200) {
//     console.log(xhr.responseXML);
//     let xmlDoc = xhr.responseXML;
//     console.log(xmlDoc.querySelector('style_citynm').innerHTML);
//   }
// };
// xhr.send();
```