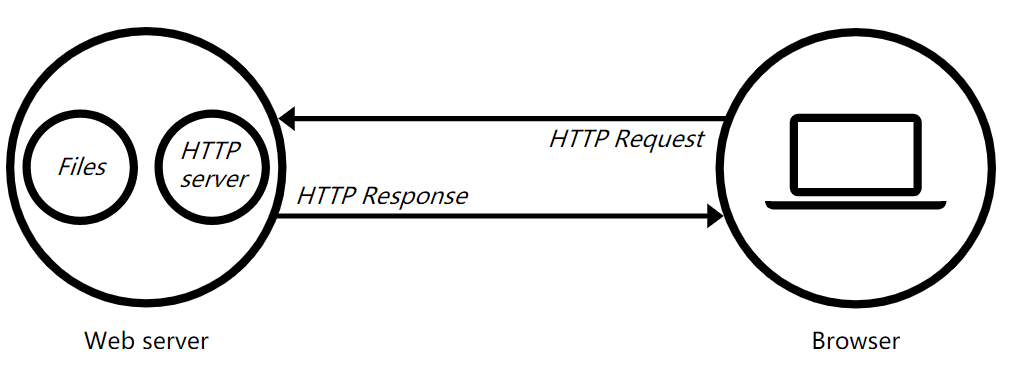
# 从服务器端获取数据

|  |  |
| --- | --- |
| **先决条件:** | JavaScript 基础 |
| **目标:** | 学会如何从服务器获取数据并使用它更新网页内容. |

## 什么是网络服务器？

网络服务器（Web server）”可以代指硬件或软件，或者是它们协同工作的整体。



静态网络服务器（**static web server**），把它托管文件的 “保持原样” 地传送到你的浏览器。

动态网络服务器（**dynamic web server**）通过 HTTP 服务器把托管文件传送到浏览器之前会对这些托管文件进行更新。

扩展阅读：

<https://developer.mozilla.org/zh-CN/docs/Learn/Common_questions/What_is_a_web_server>

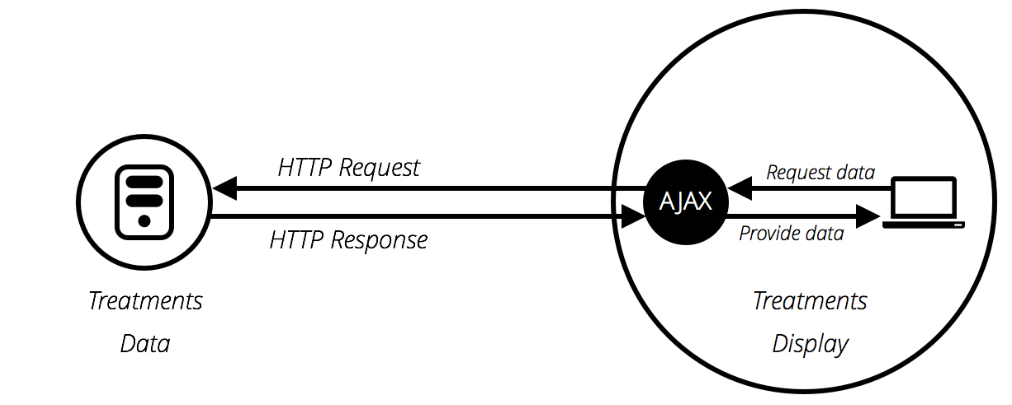
XMLHttpRequest API/ Fetch API技术允许网页直接处理对服务器上可用的特定资源的 HTTP 请求，并在显示之前根据需要对结果数据进行格式化。

## 在VSCode中建立本地测试服务器

(1)安装Live Server插件

(2)并选择Open with Live Server 浏览指定网页

## AJAX技术--局部更新数据



Ajax 即“Asynchronous Javascript And XML”（异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术。（用于局部更新数据）

[**XMLHttpRequest**](https://developer.mozilla.org/zh-CN/docs/Learn/JavaScript/Client-side_web_APIs/Fetching_data#XMLHttpRequest)通常缩写为XHR

（1）创建请求对象

var request = new XMLHttpRequest();

（2）设置请求对象

2-1用open() 方法来指定从网络请求资源的 HTTP request method , 以及URL。

request.open('请求类型', 请求地址);

2-2设置响应类型 （返回什么类型的数据）

request.responseType = '响应数据类型'

2-3注册请求成功事件，事件发生后请求的响应数据response可用

request .onreadystatechange = function() {

if(request.readyState === 4 && request.status ==200)

request.response;//request.response为响应数据

};

（3）发送请求

request.send();

例子1：xhr-basic.html

request.readyState

0：初始化，XMLHttpRequest对象还没有完成初始化

1：载入，XMLHttpRequest对象开始发送请求

2：载入完成，XMLHttpRequest对象的请求发送完成

3：解析，XMLHttpRequest对象开始读取服务器的响应

4：完成，XMLHttpRequest对象读取服务器响应结束

request.status

1xx - 信息提示

2xx - 成功

　· 200 - OK 一切正常，对GET和POST请求的应答文档跟在后面。

　· 201 - Created 服务器已经创建了文档，Location头给出了它的URL。

　· 202 - Accepted 已经接受请求，但处理尚未完成。

　· 203 - Non-Authoritative Information 文档已经正常地返回，但一些应答头可能不正确，因为使用的是文档的拷贝，非权威性信息（HTTP 1.1新）。

　· 204 - No Content 没有新文档，浏览器应该继续显示原来的文档。如果用户定期地刷新页面，而Servlet可以确定用户文档足够新，这个状态代码是很有用的。

　· 205 - Reset Content 没有新的内容，但浏览器应该重置它所显示的内容。用来强制浏览器清除表单输入内容（HTTP 1.1新）。

　· 206 - Partial Content 客户发送了一个带有Range头的GET请求，服务器完成了它（HTTP 1.1新）。

3xx - 重定向

4xx - 客户端错误

## Fetch技术

Fetch是XHR的一个替代技术

fetch(地址).then(function(响应对象) {

响应对象.数据类型转换().then(function(响应数据) {

使用响应数据

});

});

fetch(地址).then(响应函数) 如果地址请求响应成功，则执行相应响应函数

响应对象.数据类型转换().then（）若数据类型转换成功，则执行相应响应函数

测试失败案例：

1. 地址错误---http 404 (Not Found)
2. 数据类型错误

如：

fetch(url).then(function(response) {

response.text().then(function(text) {

poemDisplay.textContent = text;

});

});

};

例子2：fetch-basic.html

## 进阶

罐头小店