# Adv Web Dev Arch
# Lecture 5

Amir Amintabar, PhD

AJAX Calls

# 高级Web开发架构讲座第5讲

阿米尔·阿明塔巴尔，博士

AJAX调用

## Outline (Building a Typical Web API, AJAX, intro to DB)

- Review
- - AJAX calls
  - Browser uses http methods to send request to server
- - AJAX calls demos
- - AJAX call to remote server
- --- self study for next week
- - Working with a Relational Database (CRUD)
- - Using XAMPP package installer
- - Connecting to mySQL DB using nodejs
- - Hosting Node js, MySQL on remote server

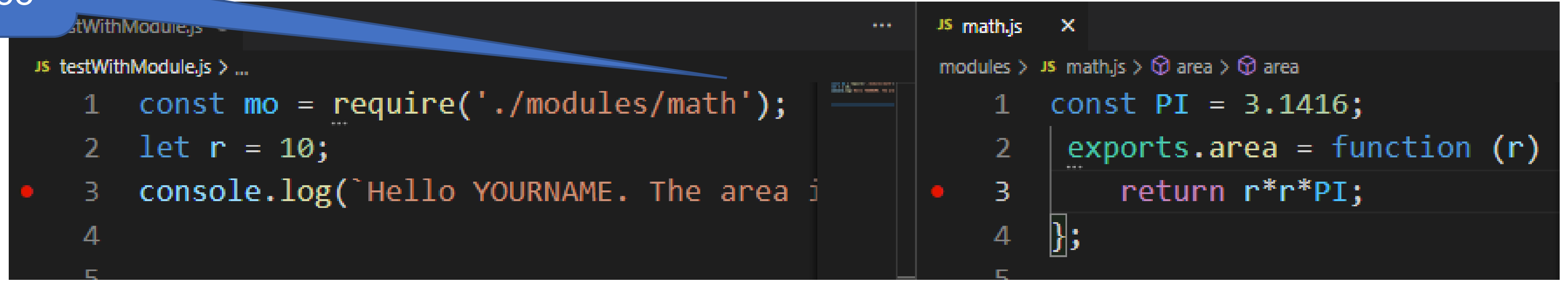
## 课程大纲（构建典型 Web API、AJAX 简介、数据库入门）

- 复习
- - AJAX 请求
  - 浏览器使用 HTTP 方法向服务器发送请求
- - AJAX 请求演示
- - 向远程服务器发起 AJAX 请求
- —— 下周自学内容
- —— 使用关系型数据库（增删改查操作）
- —— 使用 XAMPP 套件安装程序
- —— 使用 Node.js 连接 MySQL 数据库
- —— 在远程服务器上部署 Node.js 和 MySQL

## review

- Week 1: **general challenges when developing web app for various devices.** review material ( over 120 slides) provided to review HTML, CSS, JS ) create objects etc
- Week 2: we defined the term web app architecture, then group-discussed various architecture types in web development, learned about local storage
- Week 3: more intro to JS, timedout, order of execution, event loop, hoisting, var vs let, Anatomy of WebAPIs, intro to RESTful APIs, Microservices, GET vs POST
- Week 4: in js every function returns something nodejs modules

math.js
Works too



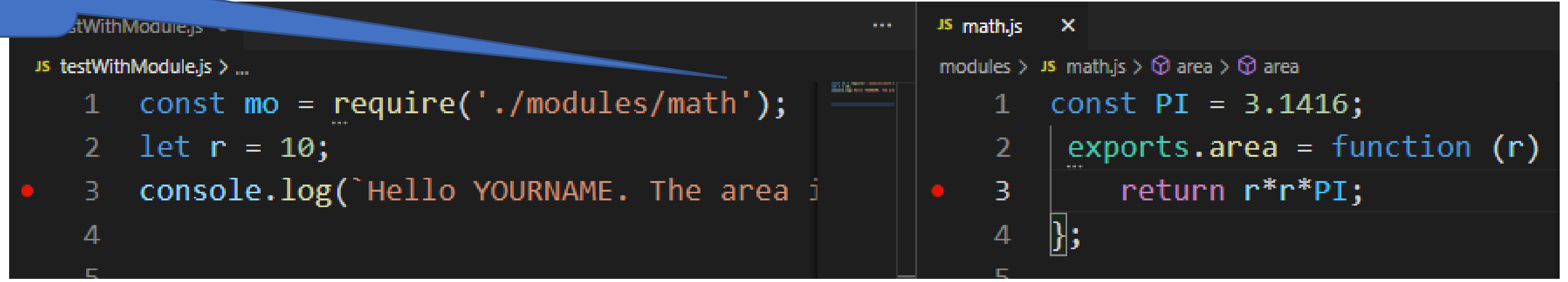
```
JS testWithModule.js > ...
const mo = require('./modules/math');
let r = 10;
console.log(`Hello YOURNAME. The area i
```

```
modules > JS math.js > area > area
const PI = 3.1416;
exports.area = function (r)
    return r*r*PI;
};
```

## 复习

- 第 1 周：**面向多种设备开发 Web 应用时所面临的一般性挑战**。复习资料（逾 120 张幻灯片）已提供，用于复习 HTML、CSS、JavaScript 等内容，并学习如何创建对象等知识。
- 第 2 周：我们定义了“Web 应用架构”这一术语，随后以小组讨论形式探讨了 Web 开发中各类架构类型，并学习了本地存储（Local Storage）相关知识。
- 第 3 周：进一步介绍 JavaScript，包括超时机制（timeout）、执行顺序、事件循环（Event Loop）、变量提升（hoisting）、var 与 let 的区别、Web API 的构成原理、RESTful API 入门、微服务（Microservices），以及 GET 与 POST 方法的对比。
- 第 4 周：在 JavaScript 中，每个函数都会返回某个值；同时介绍了 Node.js 模块机制。

数学.js
同样适用



```
JS testWithModule.js > ...
const mo = require('./modules/math');
let r = 10;
console.log(`Hello YOURNAME. The area i
```

```
modules > JS math.js > area > area
const PI = 3.1416;
exports.area = function (r)
    return r*r*PI;
};
```

1

## AJAX calls

1

## AJAX 调用

## **So far** we sent our requests to the server via the **browsers' address bar**!

What if we wanted send requests to servers from within our HMTL pages and not browser address bar?
e.g. to press a button and send something to the server?

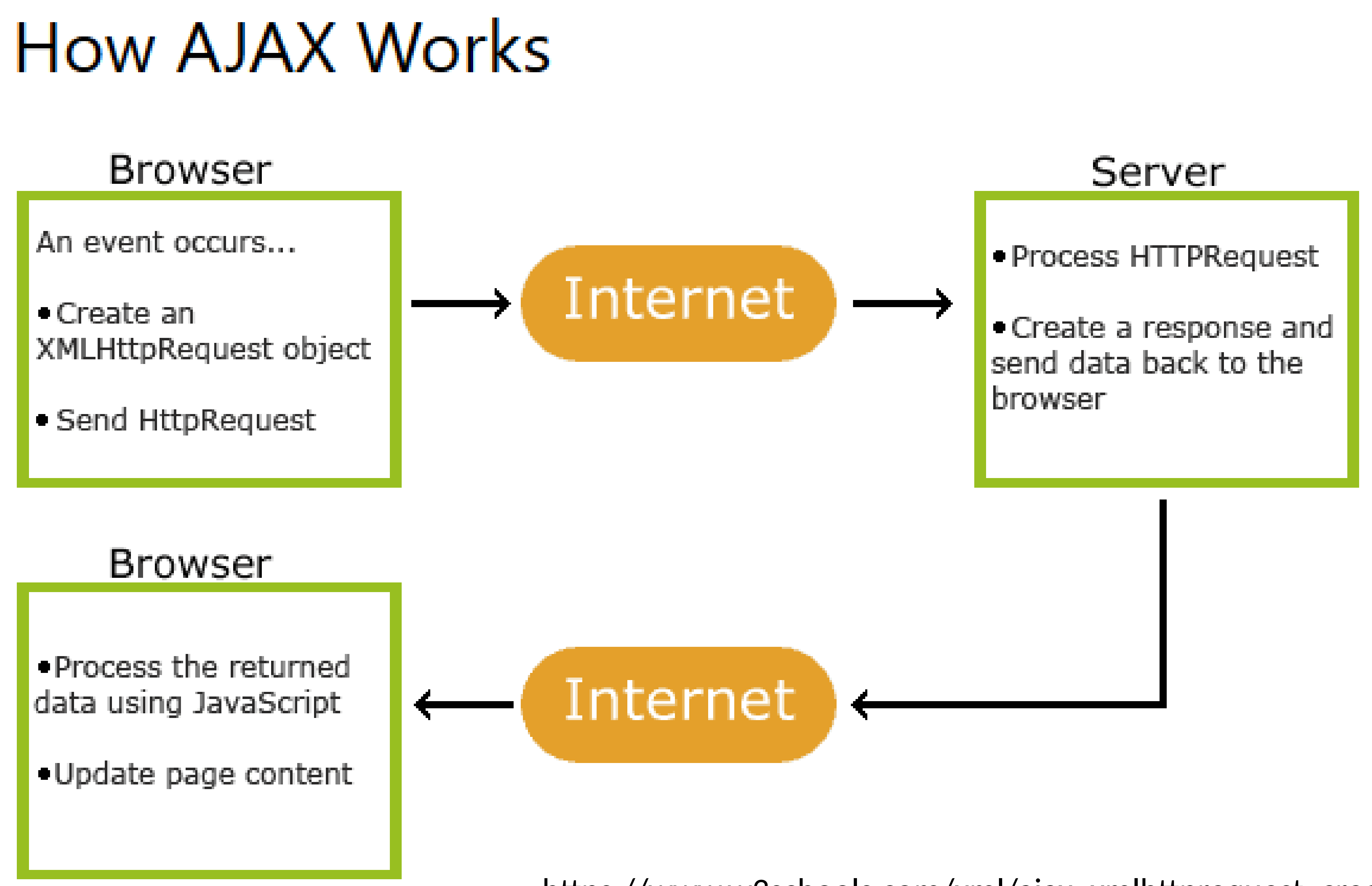
You could do that either via forms
or
AJAX calls

## **截至目前，**我们都是通过服务器发送请求 通过**浏览器的地址栏**！

如果我们希望在 HTML 页面内部而非浏览器地址栏中向服务器发送请求，该怎么办？
页面内而非浏览器地址栏？
例如，单击一个按钮即可向服务器发送某些数据？

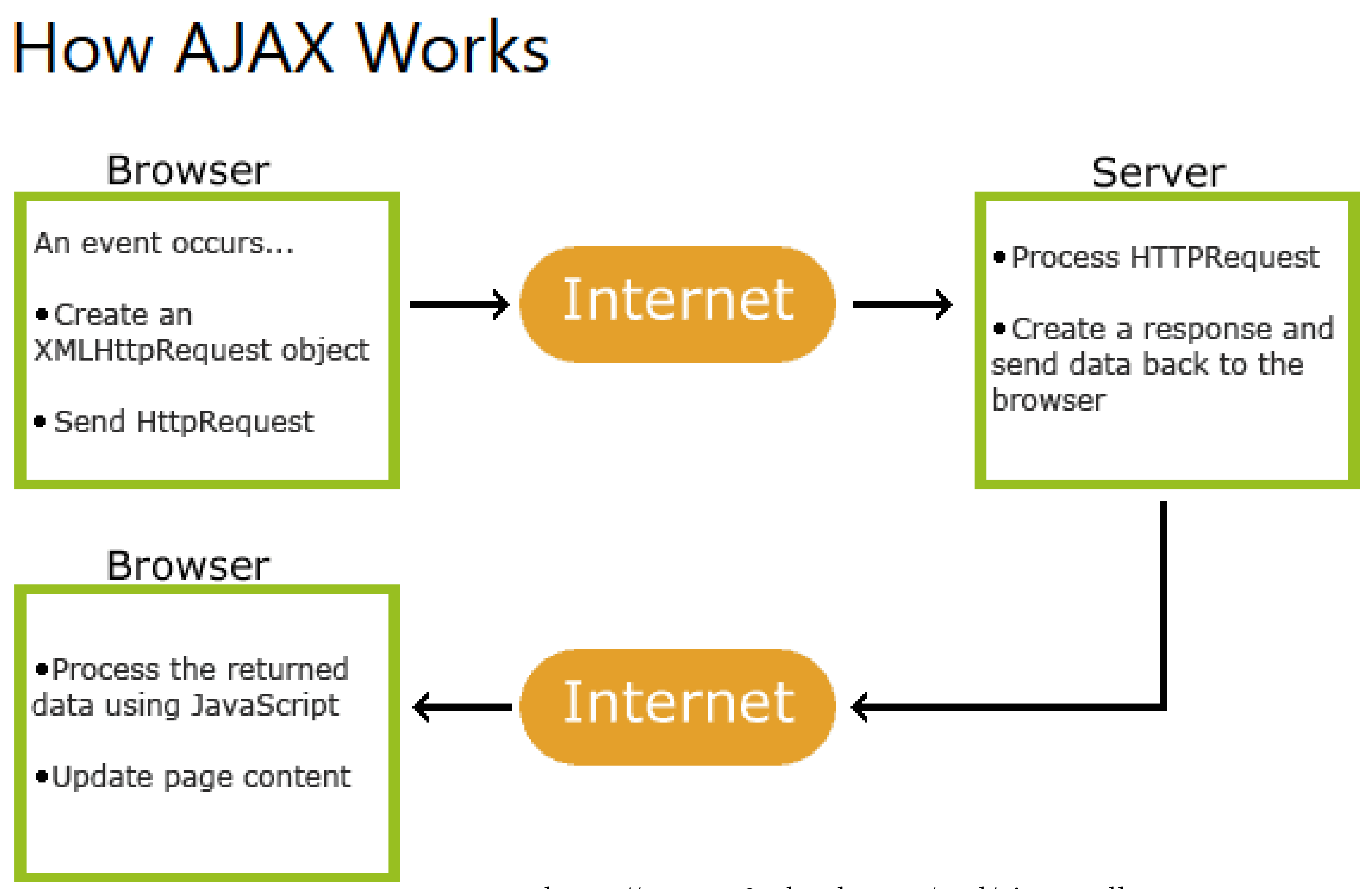
您可以通过表单来实现这一点
or
AJAX 调用

- AJAX allows sending request to server


How AJAX Works
Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp

- AJAX 允许向服务器发送请求


How AJAX Works
Browser
An event occurs...
•Create an XMLHttpRequest object
•Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Browser
•Process the returned data using JavaScript
•Update page content
Internet

https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp

- Here is how we  do it at the client side :
- **1- Create an XMLHttpRequest Object:**
  - ( just like the way we created Date object)
  - let xhttp = new XMLHttpRequest();

- 我们在 客户端 是这样实现的:
- **1- 创建一个 XMLHttpRequest 对象:**
  - （方式与创建 Date 对象类似）
  - 声明 xhttp = 新建 XMLHttpRequest();

- **2- specify the function that handles response :**

```
xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML =
        this.responseText;
    }
};
```

Q: Who determines what status code to return?

status of the XMLHttpRequest :
0 (UNSENT): The XHR object has been created, but open() has not been called yet.
1 (OPENED): open() has been called.
2 (HEADERS_RECEIVED): send() has been called, and the headers of the response are available.
3 (LOADING): The response is being received. As data comes in, the responseText property is updated.
4 (DONE): The operation is complete, and either the request has been successfully completed (status code 2xx) or an error occurred.

**3(LOADING) code be updated multiple times to this status if the server is sending a large size of response in multiple chunks**

http response type your server script returns:
200: "OK“
3xx: generally used for redirection purposes
4xx: client error (something wrong with your request
403: "Forbidden"
404: "Not Found“ etc)
5xx: server error ( something wrong with the server)

- **2- specify the function that handles response :**

```
xhttp.onreadystatechange = function () {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("demo").innerHTML =
        this.responseText;
    }
};
```

问：由谁决定返回何种状态码?

status of the XMLHttpRequest :
0（未发送）：XHR 对象已创建，但尚未调用 open() 方法。
1（已打开）：已调用 open() 方法。
2（已接收响应头_）：已调用 send() 方法，且响应的头部信息已可用。
3（加载中）：正在接收响应。随着数据陆续到达，responseText 属性将被持续更新。
4（已完成）：操作已结束，请求已成功完成（状态码为 2xx）或发生了错误。

**3（加载中）状态码可能被多次更新为该状态，前提是服务器正以多个数据块的形式发送较大体积的响应。**

http response type your server script
返回值：
200: "OK“
3xx：通常用于重定向目的4xx：客户端错误（您的请求存在某些问题
403: "Forbidden"
404: "Not Found“ etc)
5xx: server error ( something wrong with the server)

- **3- Open AJAX request**
- xhttp.open(method, url, Async or sync);
  - Method type could be GET or POST etc
  - url is the url of the file we want to handle our request
- So what else is left?
- Everything is in place! We just send it and carry on (we don’t wait if we sent it asynchronously )

.4-xhttp.send();

- **3- 打开 AJAX 请求**
- xhttp.open(方法, URL, 异步或同步);
  - 方法类型可以是 GET、POST 等
  - URL 是我们希望处理该请求的文件地址
- 那么，还剩下什么呢?
- Everything is in place! We just send it and carry on (we don’t wait if we sent it asynchronously )

.4-xhttp.send();

2

AJAX call demos

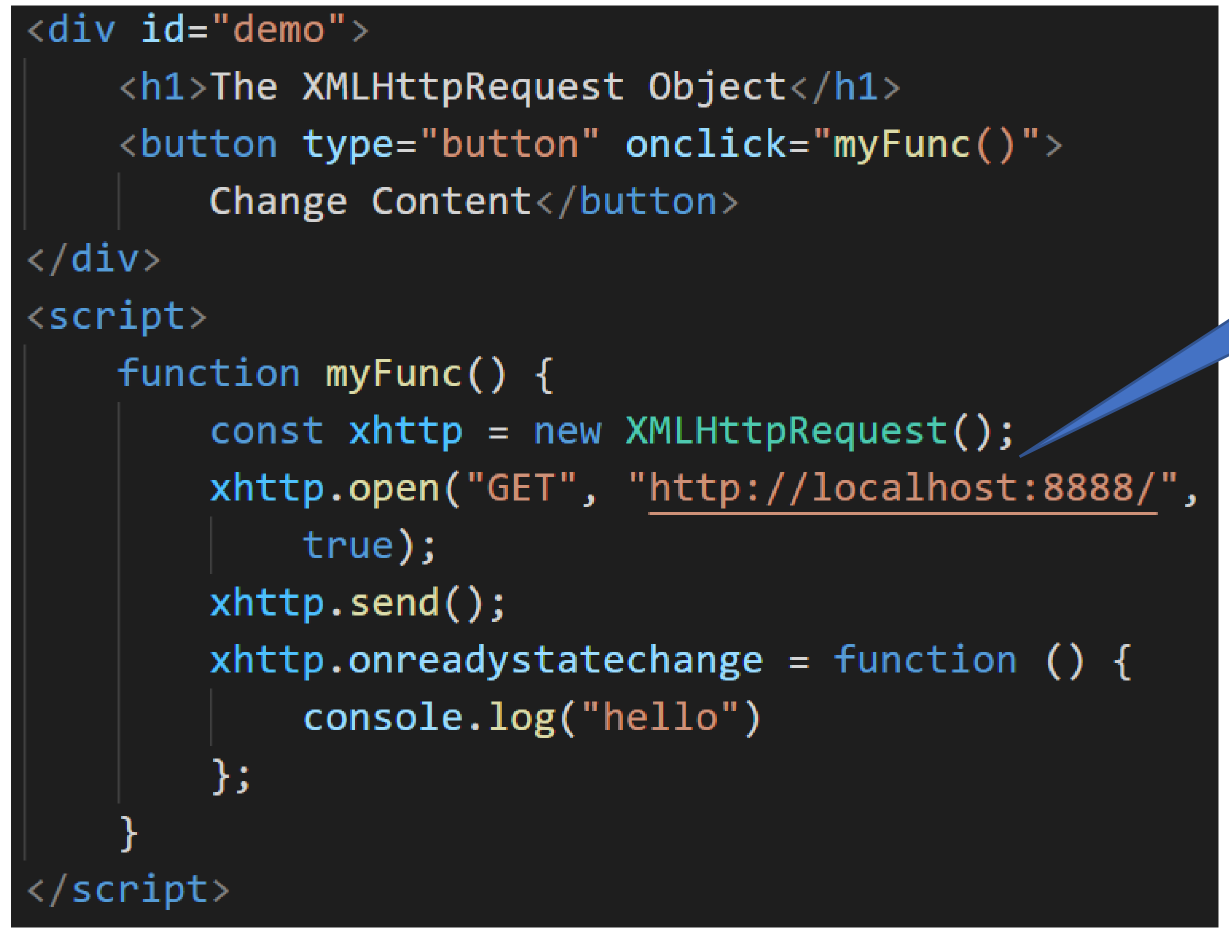
AX 调用
示例

Demo 1

sending AJAX call without any arguments !

演示 1

正在发送不带任何参数的 AJAX 请求!

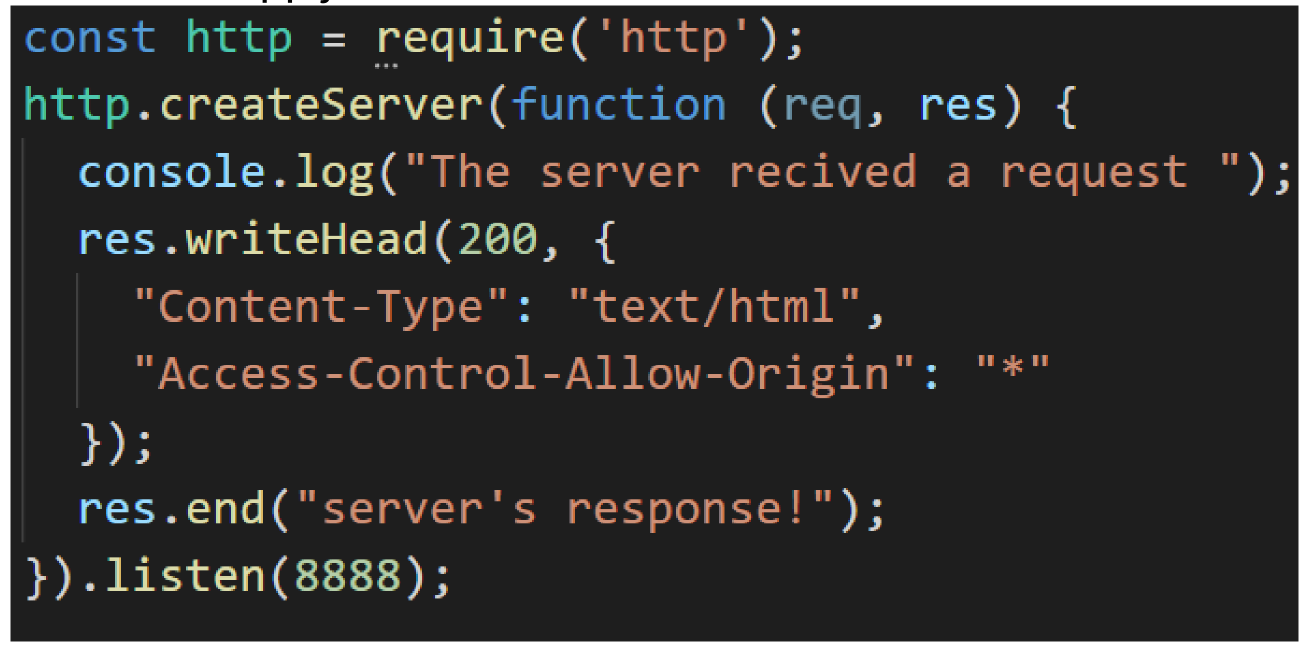


```
<div id="demo">
    <h1>The XMLHttpRequest Object</h1>
    <button type="button" onclick="myFunc()">
        Change Content</button>
</div>
<script>
    function myFunc() {
        const xhttp = new XMLHttpRequest();
        xhttp.open("GET", "http://localhost:8888/",
            true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            console.log("hello")
        };
    }
</script>
```

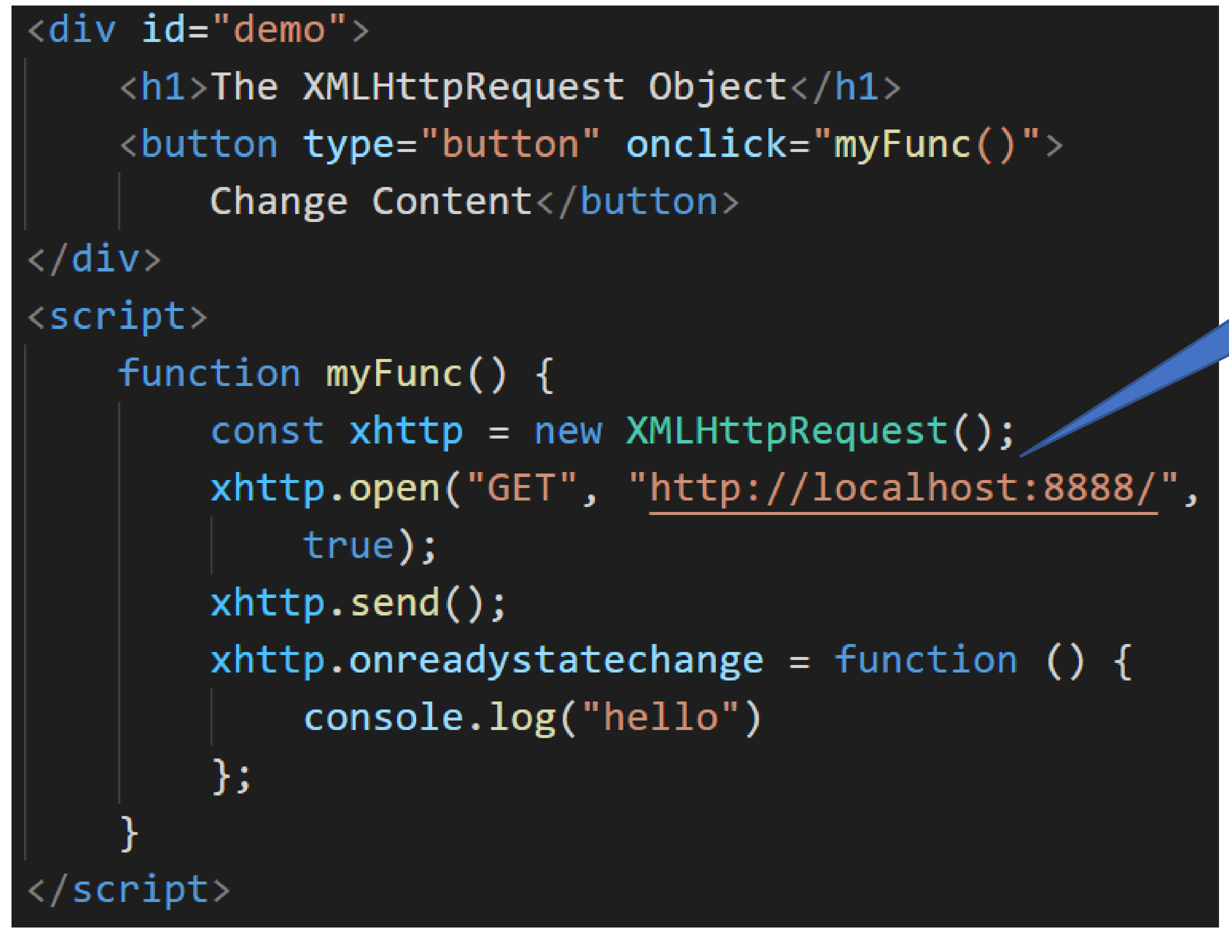
Client: ajax.html

You can also type this url in the browser address bar. Since it's a Get request

Server: app.js



```
const http = require('http');
http.createServer(function (req, res) {
  console.log("The server recived a request ");
  res.writeHead(200, {
    "Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"
  });
  res.end("server's response!");
}).listen(8888);
```

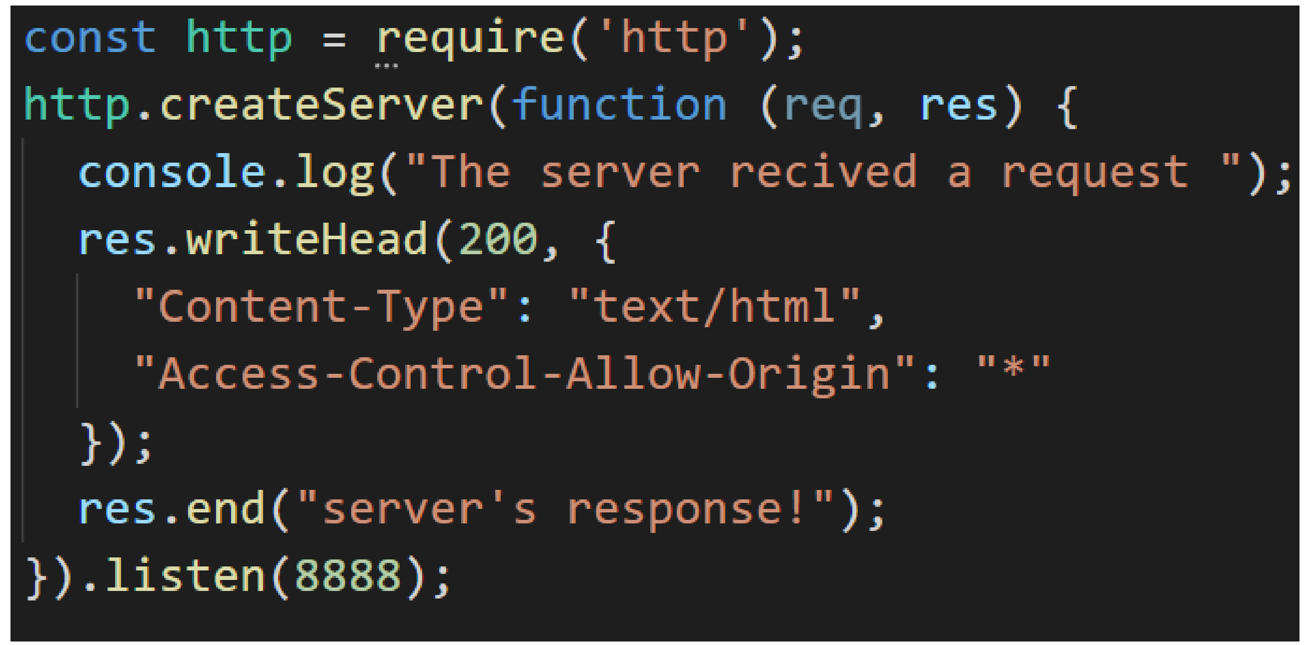


```
<div id="demo">
    <h1>The XMLHttpRequest Object</h1>
    <button type="button" onclick="myFunc()">
        Change Content</button>
</div>
<script>
    function myFunc() {
        const xhttp = new XMLHttpRequest();
        xhttp.open("GET", "http://localhost:8888/",
            true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            console.log("hello")
        };
    }
</script>
```

客户端：ajax.html

您也可以在浏览器地址栏中直接输入该 URL。由于这是一个 GET 请求,

服务器：app.js

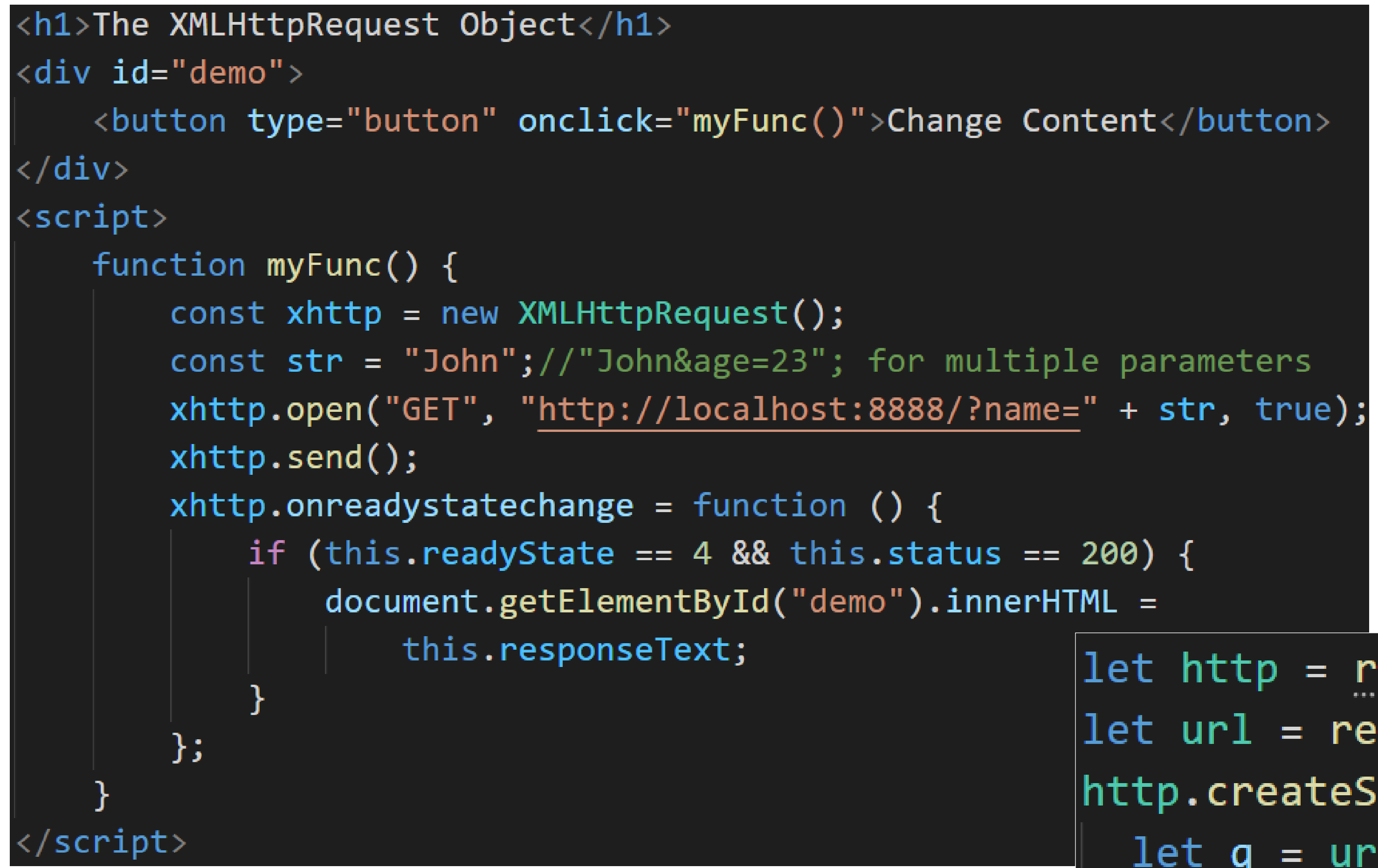


```
const http = require('http');
http.createServer(function (req, res) {
  console.log("The server recived a request ");
  res.writeHead(200, {
    "Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"
  });
  res.end("server's response!");
}).listen(8888);
```

Demo 2

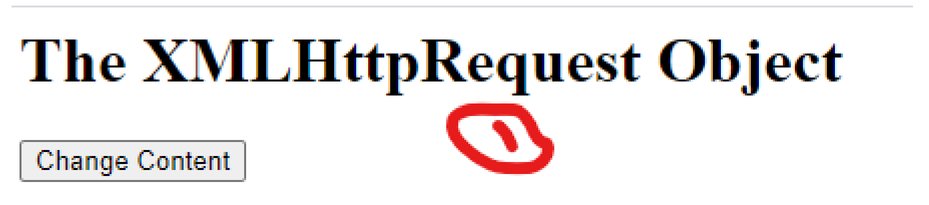
sending AJAX call with passing arguments !

正在发送 AJAX 请求并传递参数参数！



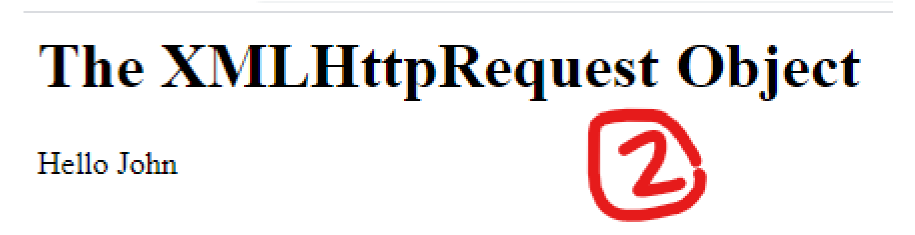
```
<h1>The XMLHttpRequest Object</h1>
<div id="demo">
    <button type="button" onclick="myFunc()">Change Content</button>
</div>
<script>
    function myFunc() {
        const xhttp = new XMLHttpRequest();
        const str = "John";//"John&age=23"; for multiple parameters
        xhttp.open("GET", "http://localhost:8888/?name=" + str, true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
</script>
```

Client: ajax.html



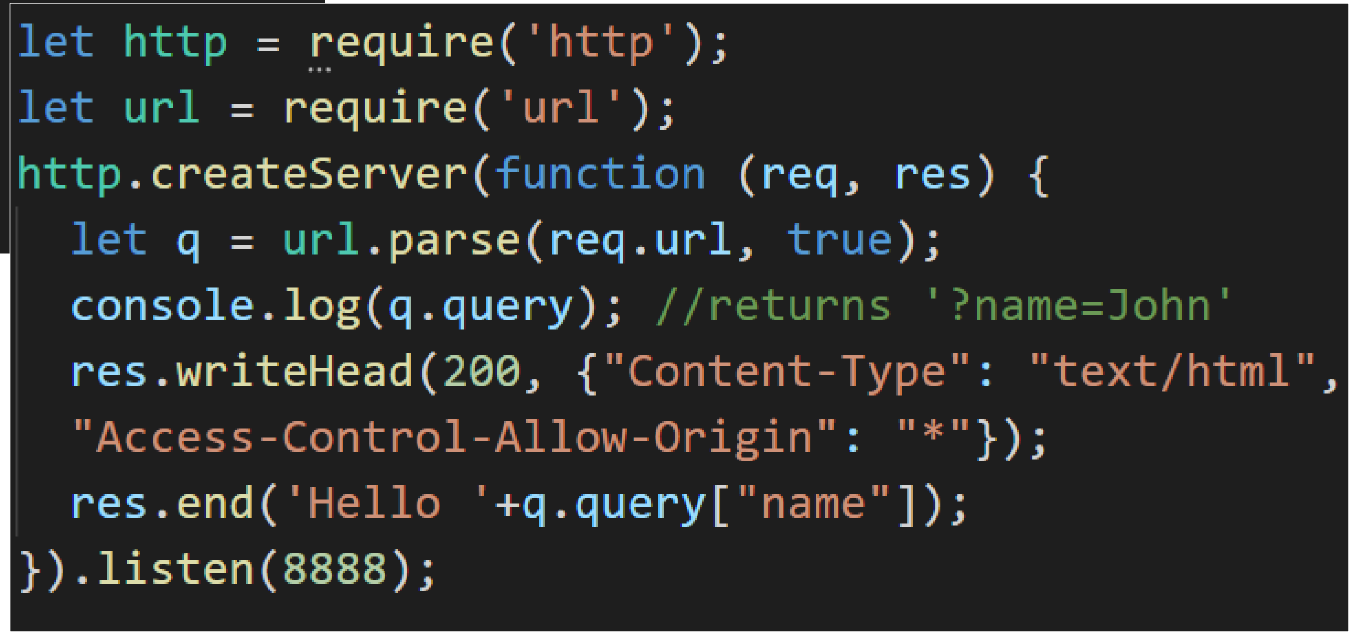
# The XMLHttpRequest Object

Change Content



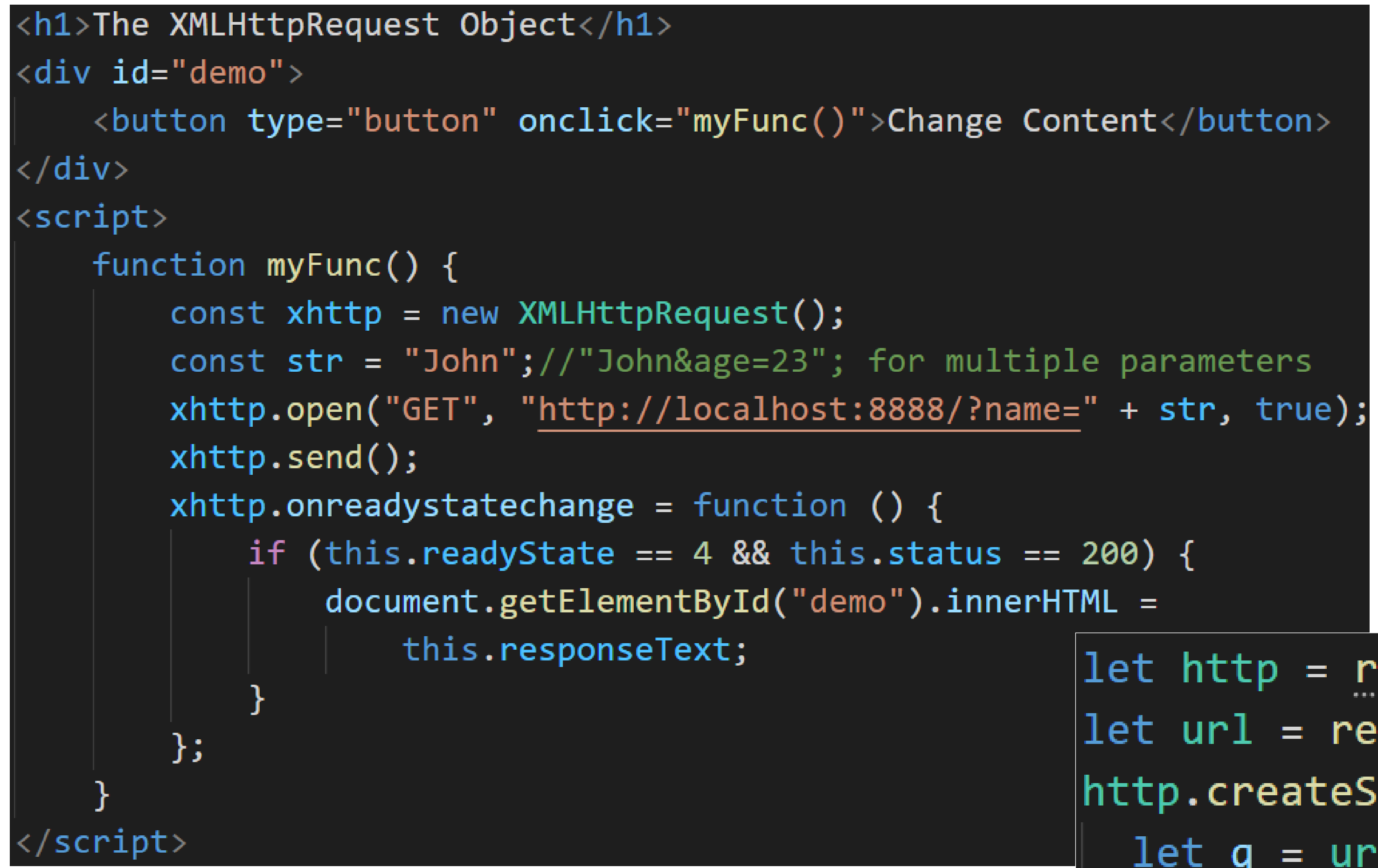
# The XMLHttpRequest Object

Hello John



```
let http = require('http');
let url = require('url');
http.createServer(function (req, res) {
  let q = url.parse(req.url, true);
  console.log(q.query); //returns '?name=John'
  res.writeHead(200, {"Content-Type": "text/html",
  "Access-Control-Allow-Origin": "*"});
  res.end('Hello '+q.query["name"]);
}).listen(8888);
```

Server: app.js



```
<h1>The XMLHttpRequest Object</h1>
<div id="demo">
    <button type="button" onclick="myFunc()">Change Content</button>
</div>
<script>
    function myFunc() {
        const xhttp = new XMLHttpRequest();
        const str = "John";//"John&age=23"; for multiple parameters
        xhttp.open("GET", "http://localhost:8888/?name=" + str, true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
</script>
```

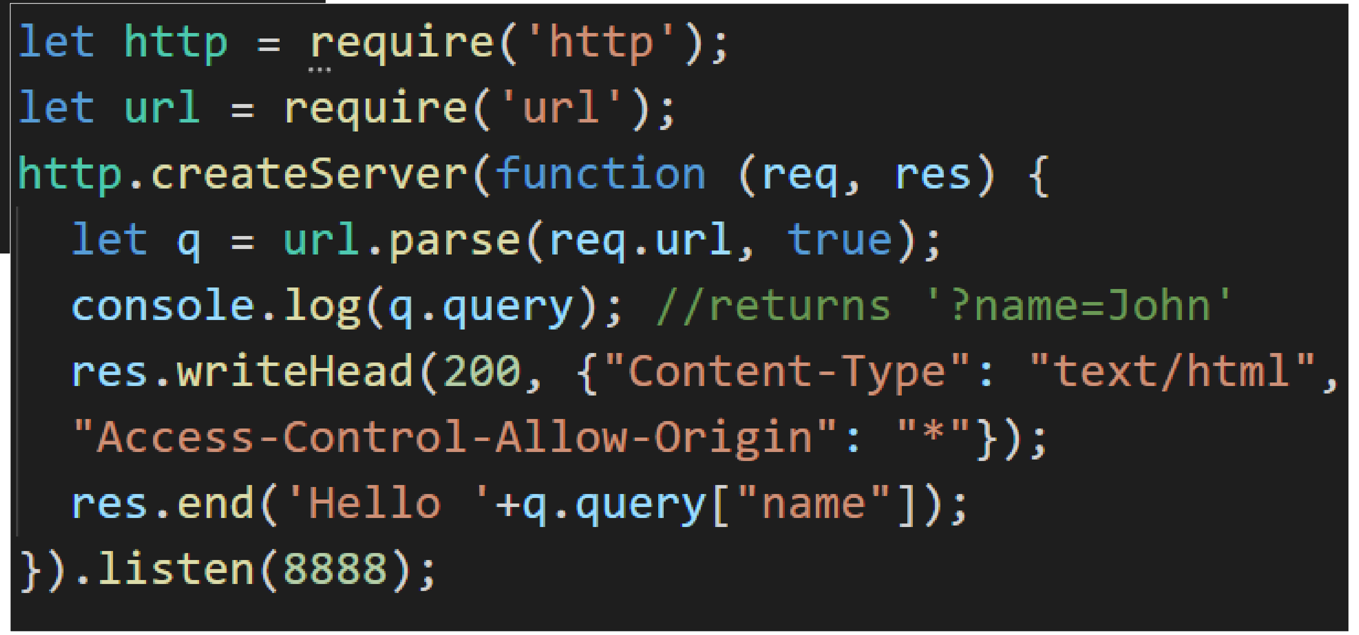
客户端：ajax.html

The XMLHttpRequest Object

Change Content

The XMLHttpRequest Object

Hello John



```
let http = require('http');
let url = require('url');
http.createServer(function (req, res) {
  let q = url.parse(req.url, true);
  console.log(q.query); //returns '?name=John'
  res.writeHead(200, {"Content-Type": "text/html",
  "Access-Control-Allow-Origin": "*"});
  res.end('Hello '+q.query["name"]);
}).listen(8888);
```

服务器：app.js

Demo 3

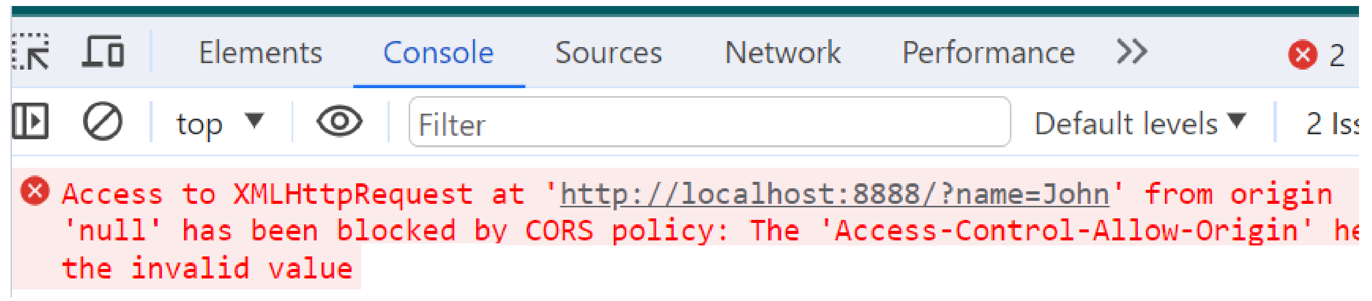
## AJAX call to remote server

Configuring CORS

## 向远程服务器发起 AJAX 请求

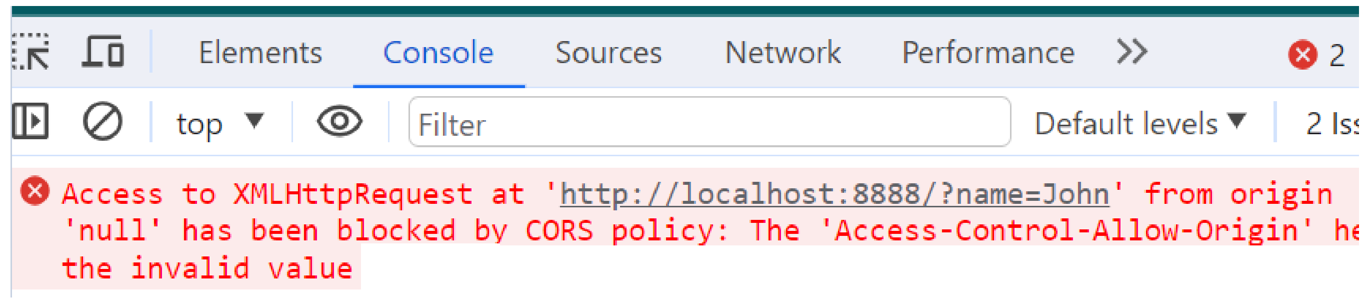
配置 CORS

- Who has seen a similar message?



```
Access to XMLHttpRequest at 'http://localhost:8888/?name=John' from origin
'null' has been blocked by CORS policy: The 'Access-Control-Allow-Origin' he
the invalid value
```

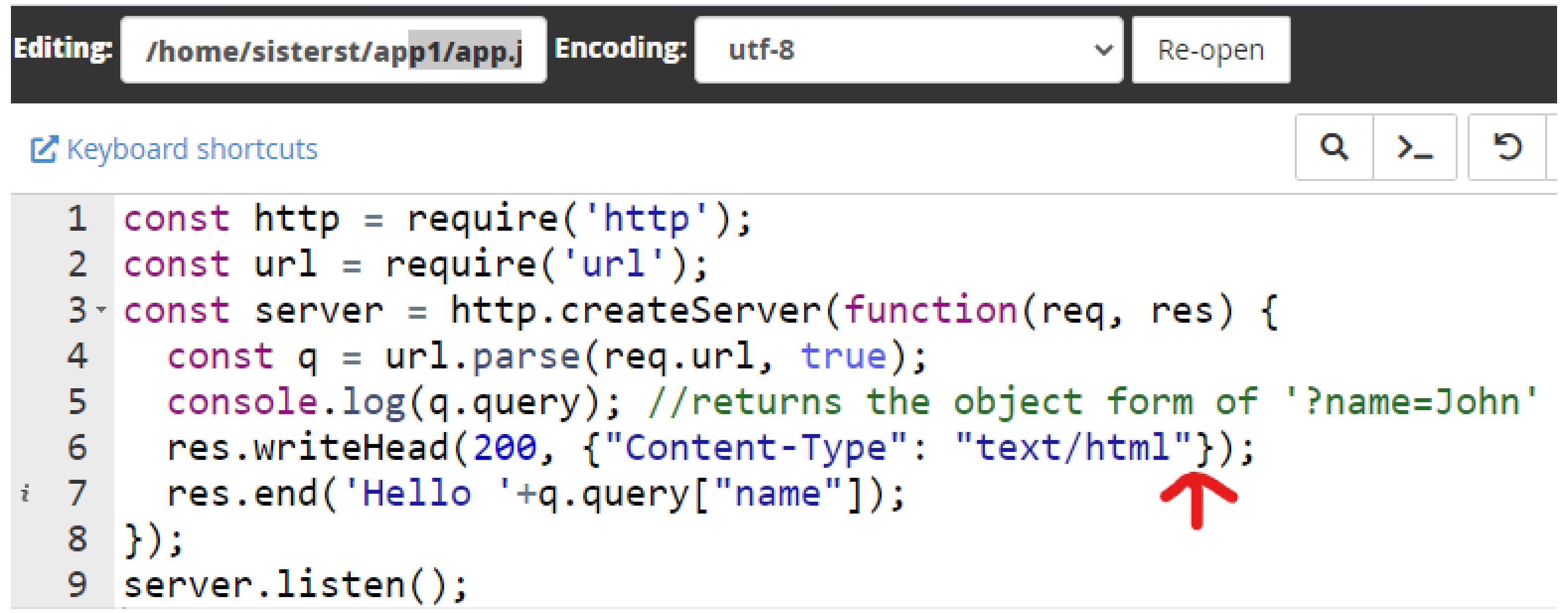
- 谁见过类似的消息?



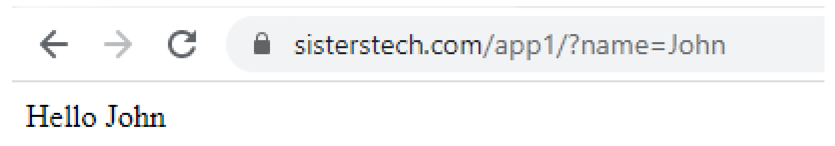
```
Access to XMLHttpRequest at 'http://localhost:8888/?name=John' from origin
'null' has been blocked by CORS policy: The 'Access-Control-Allow-Origin' he
the invalid value
```

## Demo 3

- Last time we tested this:



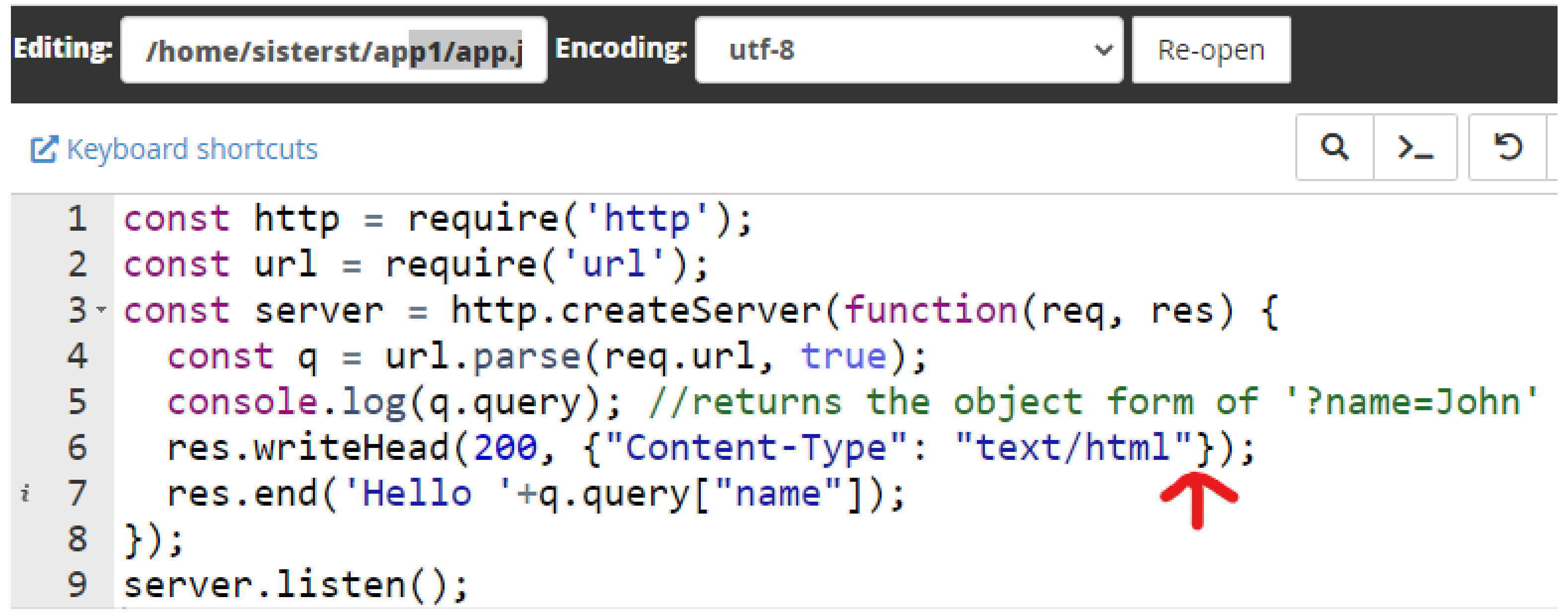
```
const http = require('http');
const url = require('url');
const server = http.createServer(function(req, res) {
  const q = url.parse(req.url, true);
  console.log(q.query); //returns the object form of '?name=John'
  res.writeHead(200, {"Content-Type": "text/html"});
  res.end('Hello '+q.query["name"]);
});
server.listen();
```



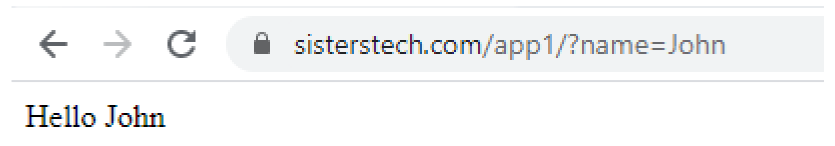
```
Hello John
```

## 演示 3

- 上次我们测试了此功能：



```
const http = require('http');
const url = require('url');
const server = http.createServer(function(req, res) {
  const q = url.parse(req.url, true);
  console.log(q.query); //returns the object form of '?name=John'
  res.writeHead(200, {"Content-Type": "text/html"});
  res.end('Hello '+q.query["name"]);
});
server.listen();
```



```
Hello John
```

## CORS – when a browser sends a req to a server

- CORS is a security feature **implemented in web browsers** and is **configured by the contacted server** in the response's header to the AJAX requests made via the browser[1]
- CORS (Cross-Origin Resource Sharing)
- By default[2] , web browsers enforce a security policy called "Same-Origin Policy " (SOP) means the html files containing the AJAX requests have to be hosted in the same origin[3] that also hosts the server side scripts, responding to those requests.
- browsers will block[2] the **Cross-Origin Requests** unless **the server** being contacted **explicitly allows** it

1: requests made mainly via AJAX, ( also in most browsers via forms the url address bar).
2: expected behaviour for most browsers
3: An origin is ( supposed to be) defined by the combination of domain, protocols and ports ( 80 is the default for HTTP, 443 for HTTPS)
4: When a web page hosted in originA needs to cross and send requests to a server in another origin, originB

Let's see how a server script configures/dictates CORS for browsers making the requests

## CORS——当浏览器向服务器发送请求时

q 向服务器发送请求时

- CORS 是一种安全机制，**由 Web 浏览器实现**，并由所访问的服务器**在响应浏览器发起的 AJAX 请求时，通过响应头进行配置**[1]
- CORS（跨源资源共享）
- 默认情况下[2]，Web 浏览器强制执行一种名为“同源策略”（SOP）的安全策略，这意味着包含 AJAX 请求的 HTML 文件必须与响应这些请求的服务端脚本托管在同一源[3]上。
- 浏览器将阻止[2]跨源请求，除非**被访问的服务器**明确允许该行为

1：请求主要通过 AJAX 发起（在大多数浏览器中，也可通过表单或地址栏中的 URL 发起）；2：这是大多数浏览器的预期行为；3：源（origin）应由域名、协议和端口三者共同定义（HTTP 默认端口为 80，HTTPS 默认端口为 443）

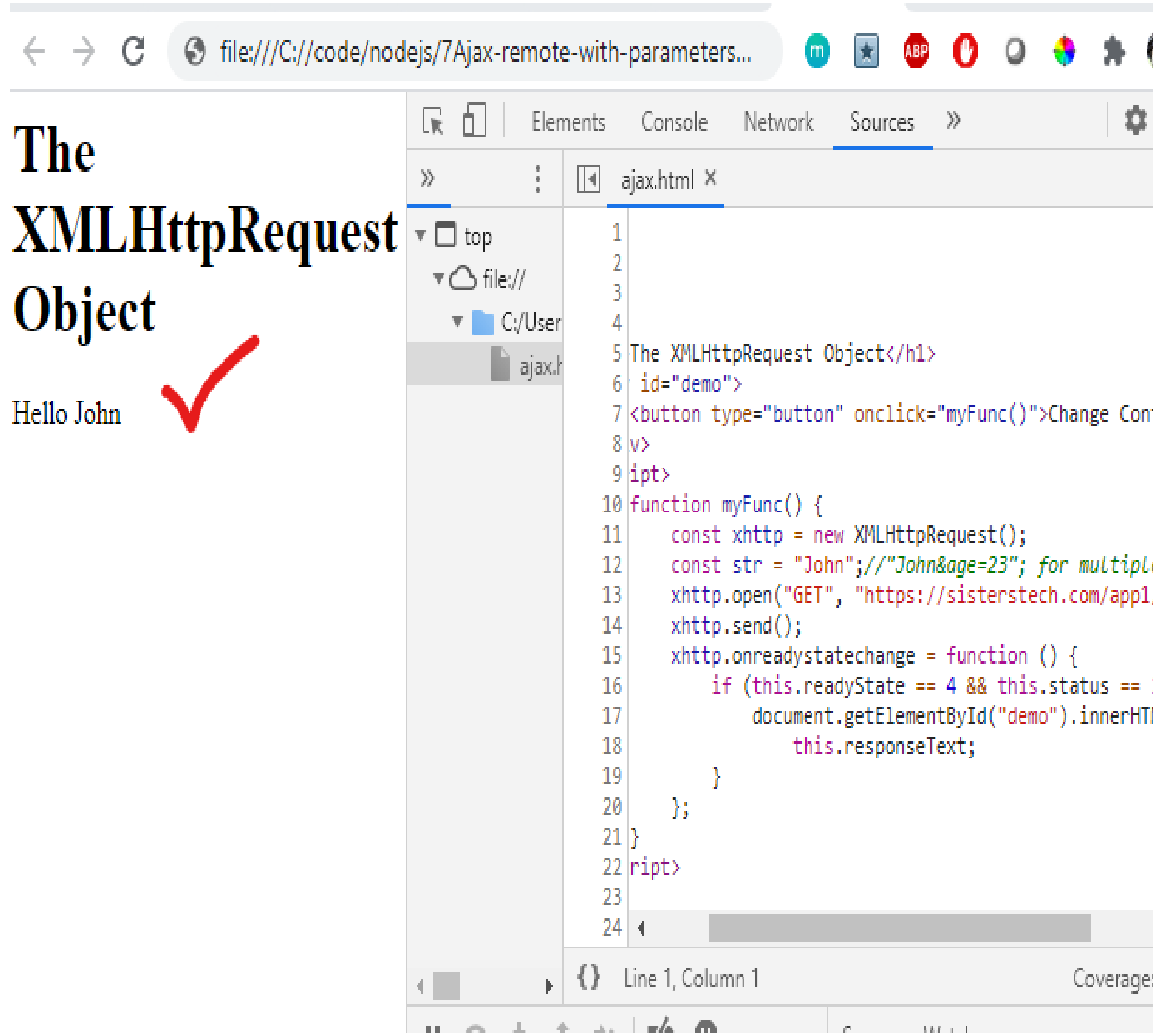
4：当托管在源 A（originA）中的网页需要跨源向另一源（源 B，originB）中的服务器发送请求时

让我们来看看服务器脚本如何为发起请求的浏览器配置或控制 CORS（跨域资源共享）

## Configure the browser CORS in your server side script!

- We now pass one more argument to the writehead () method which creates connection back to the client.
- The "Access-Control-Allow-Origin" **tells the browser** what origins can send requests to this server
- In our case we chose "*" which means **everyone!** all origins around the world Can Send request to this server
- Since we said "Access-Control-Allow-Origin": "*"

Client side (HTML Javascript)

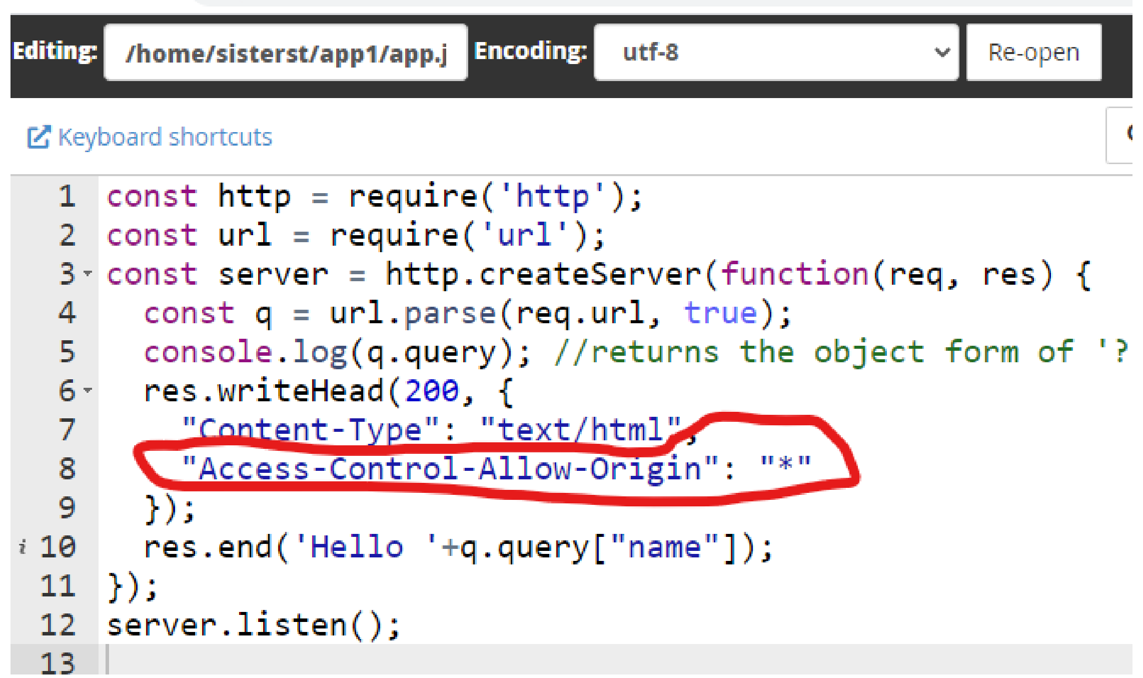


# The XMLHttpRequest Object

Hello John ✓

```
The XMLHttpRequest Object</h1>
 id="demo">
<button type="button" onclick="myFunc()">Change Con
v>
ipt>
function myFunc() {
    const xhttp = new XMLHttpRequest();
    const str = "John";//"John&age=23"; for multipl
    xhttp.open("GET", "https://sisterstech.com/app1
    xhttp.send();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status ==
            document.getElementById("demo").innerHT
                this.responseText;
        }
    };
}
ript>
```

server side (node js)

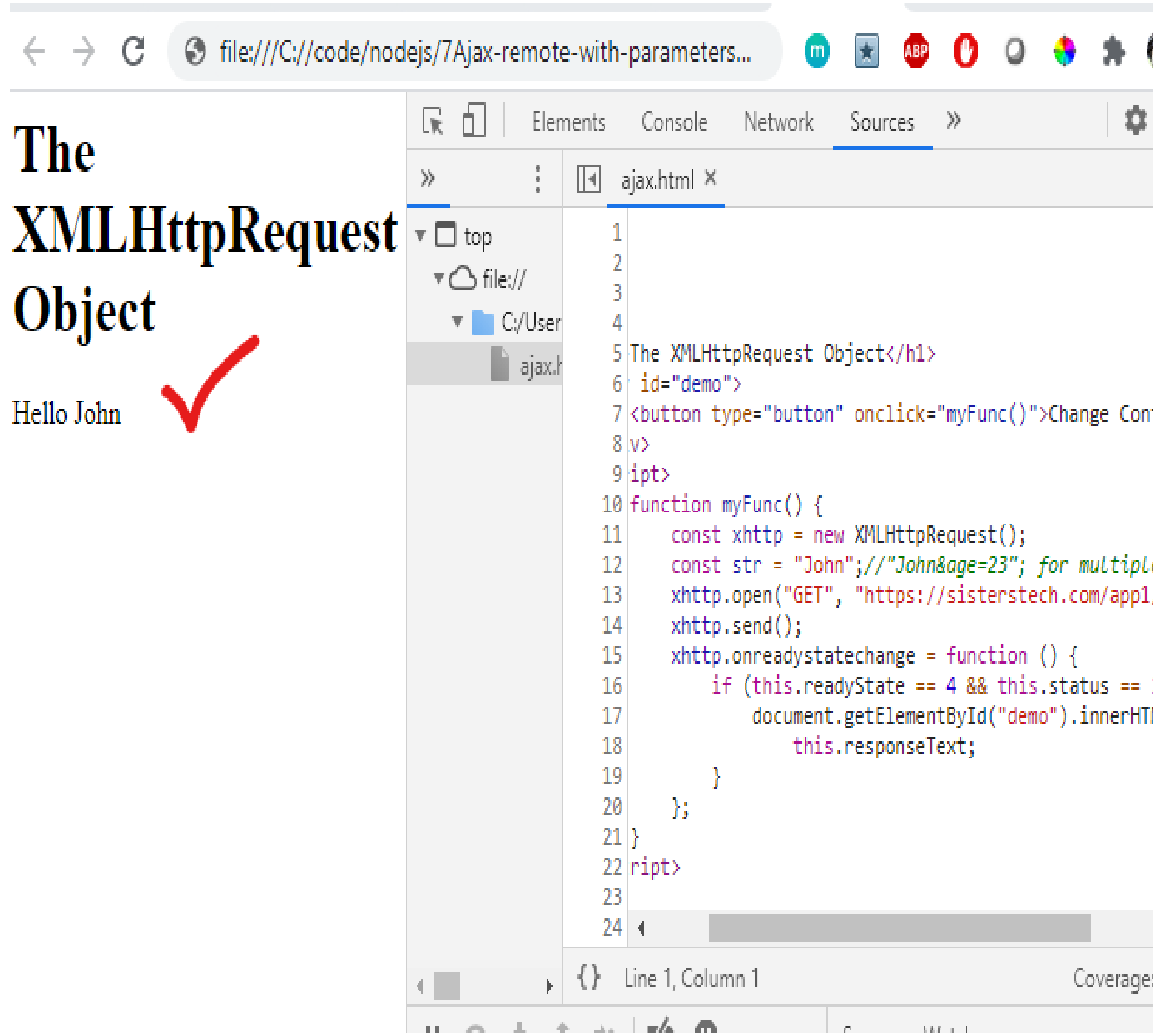


```
const http = require('http');
const url = require('url');
const server = http.createServer(function(req, res) {
  const q = url.parse(req.url, true);
  console.log(q.query); //returns the object form of '?
  res.writeHead(200, {
    "Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"
  });
  res.end('Hello '+q.query["name"]);
});
server.listen();
```

## 请在服务器端脚本中配置浏览器的跨域资源共享（CORS）！

- 现在，我们向 writehead () 方法额外传递一个参数，该参数用于创建与客户端的连接。
- 其中，"Access-Control-Allow-Origin" **告诉浏览器**哪些源（origin）可以向该服务器发送请求。
- 在本例中，我们选择了 "*"，表示**允许所有源！**全球所有源均可向该服务器发送请求
服务器
- 既然我们已声明 "Access-Control-Allow-Origin": "*"

客户端（HTML、JavaScript）

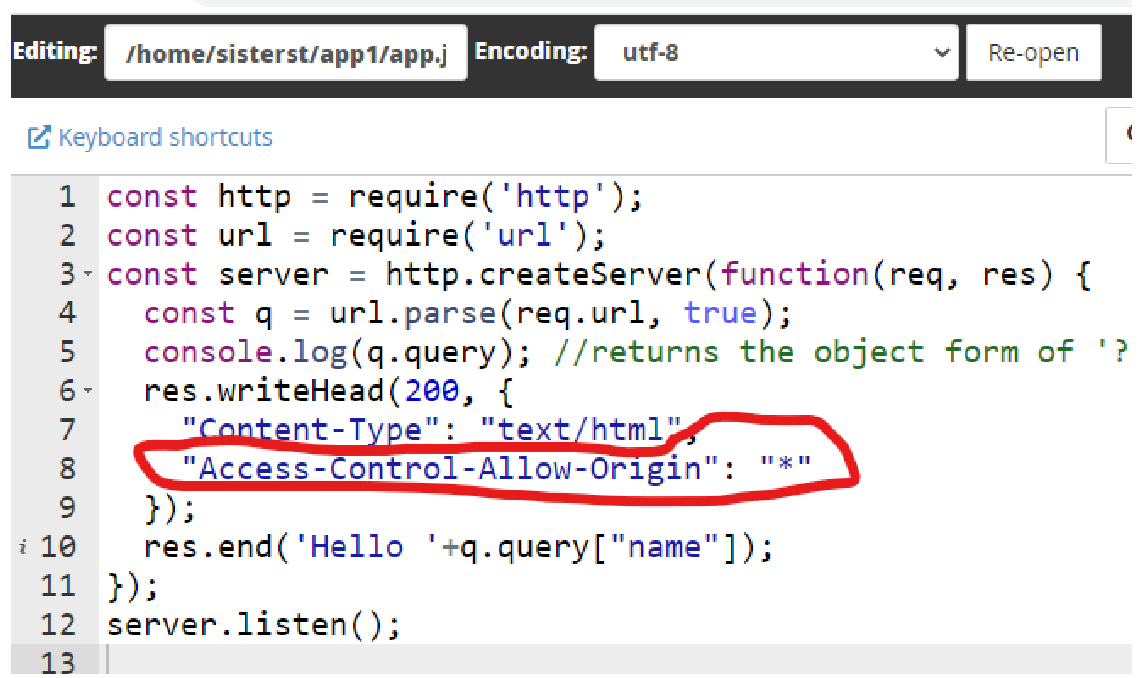


# The XMLHttpRequest Object

Hello John ✓

```
The XMLHttpRequest Object</h1>
 id="demo">
<button type="button" onclick="myFunc()">Change Con
v>
ipt>
function myFunc() {
    const xhttp = new XMLHttpRequest();
    const str = "John";//"John&age=23"; for multipl
    xhttp.open("GET", "https://sisterstech.com/appl
    xhttp.send();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status ==
            document.getElementById("demo").innerHT
                this.responseText;
        }
    };
}
ript>
```

服务器端（Node.js）



```
const http = require('http');
const url = require('url');
const server = http.createServer(function(req, res) {
  const q = url.parse(req.url, true);
  console.log(q.query); //returns the object form of '?
  res.writeHead(200, {
    "Content-Type": "text/html",
    "Access-Control-Allow-Origin": "*"
  });
  res.end('Hello '+q.query["name"]);
});
server.listen();
```

Q: What do these two do?

Access-Control-Allow-Methods

Access-Control-Allow-Headers

Q: What is a **Preflight Request** and how is it relevant to the http method OPTIONS

Q: How the http method OPTIONS should be implemented at the server side ( what should the server return upon an OPTIONS request?

问：这两项分别有何作用?

Access-Control-Allow-Methods（允许的跨域请求方法）

访问控制允许的请求头

问：什么是**预检请求（Preflight Request）**？它与 HTTP 方法 OPTIONS 有何关联?

问：服务器端应如何实现 HTTP 方法 OPTIONS（即服务器在收到 OPTIONS 请求时应返回哪些内容）？

- Q: Can we send requests to any url and expect respond using Postman regardless of the CORST setting ?

- This statement has been generated by chatGPT 3.5.
  “CORS allows for secure communication between web applications hosted on different domains while still maintaining control over which domains have access. ”
- Q: what seems to be wrong with the above statement?

- 问：我们能否向任意 URL 发送请求，并期望 Postman 返回响应，而完全不受 CORS 设置的影响?

- 该陈述由 ChatGPT 3.5 生成：“CORS 允许托管在不同域名上的 Web 应用程序之间进行安全通信，同时仍能控制哪些域名有权访问。”
- 问：上述陈述存在什么问题?

3

Demo 4

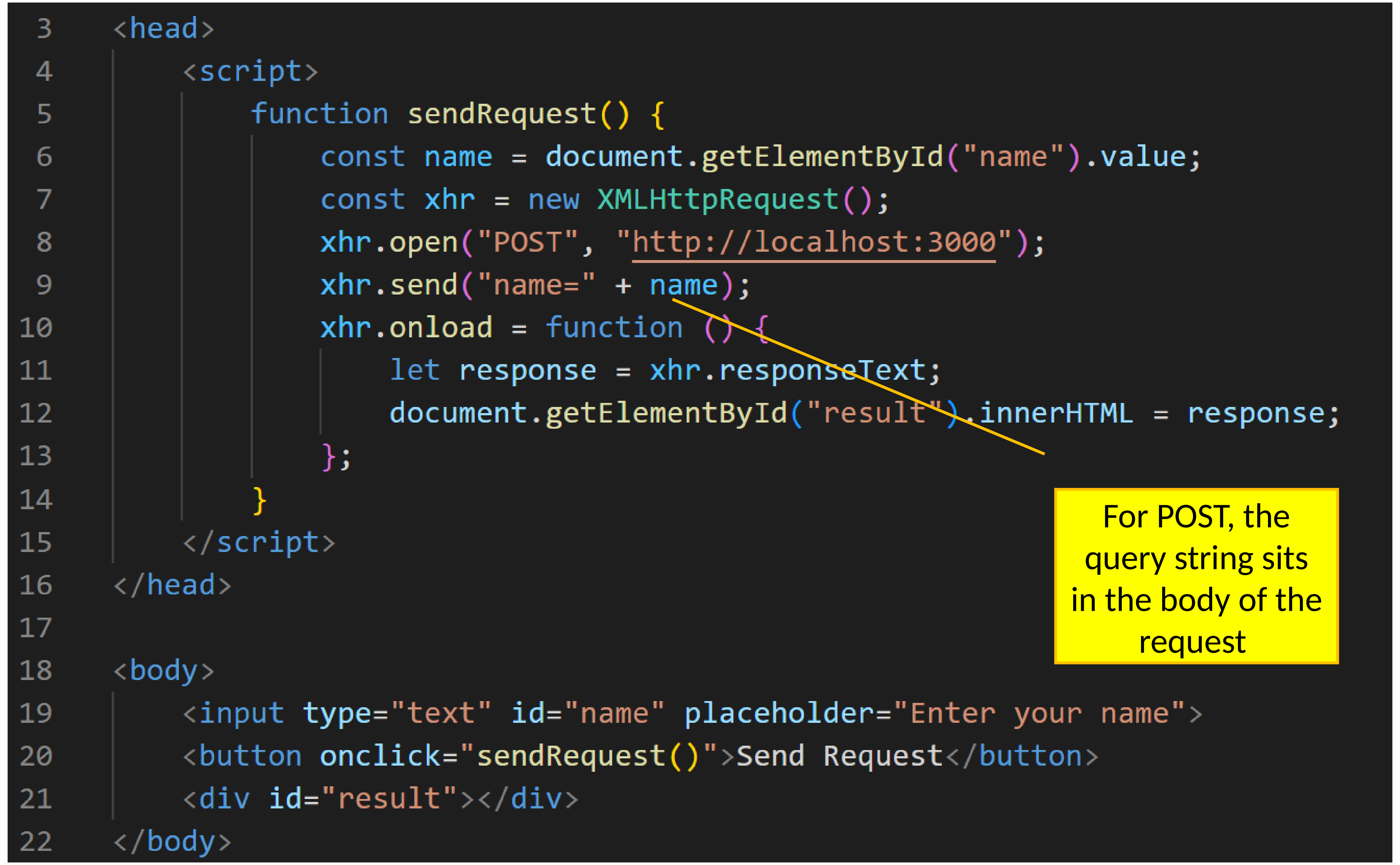
Ajax call using
POST

3

演示 4

使用 POST 方法
发起 Ajax 请求

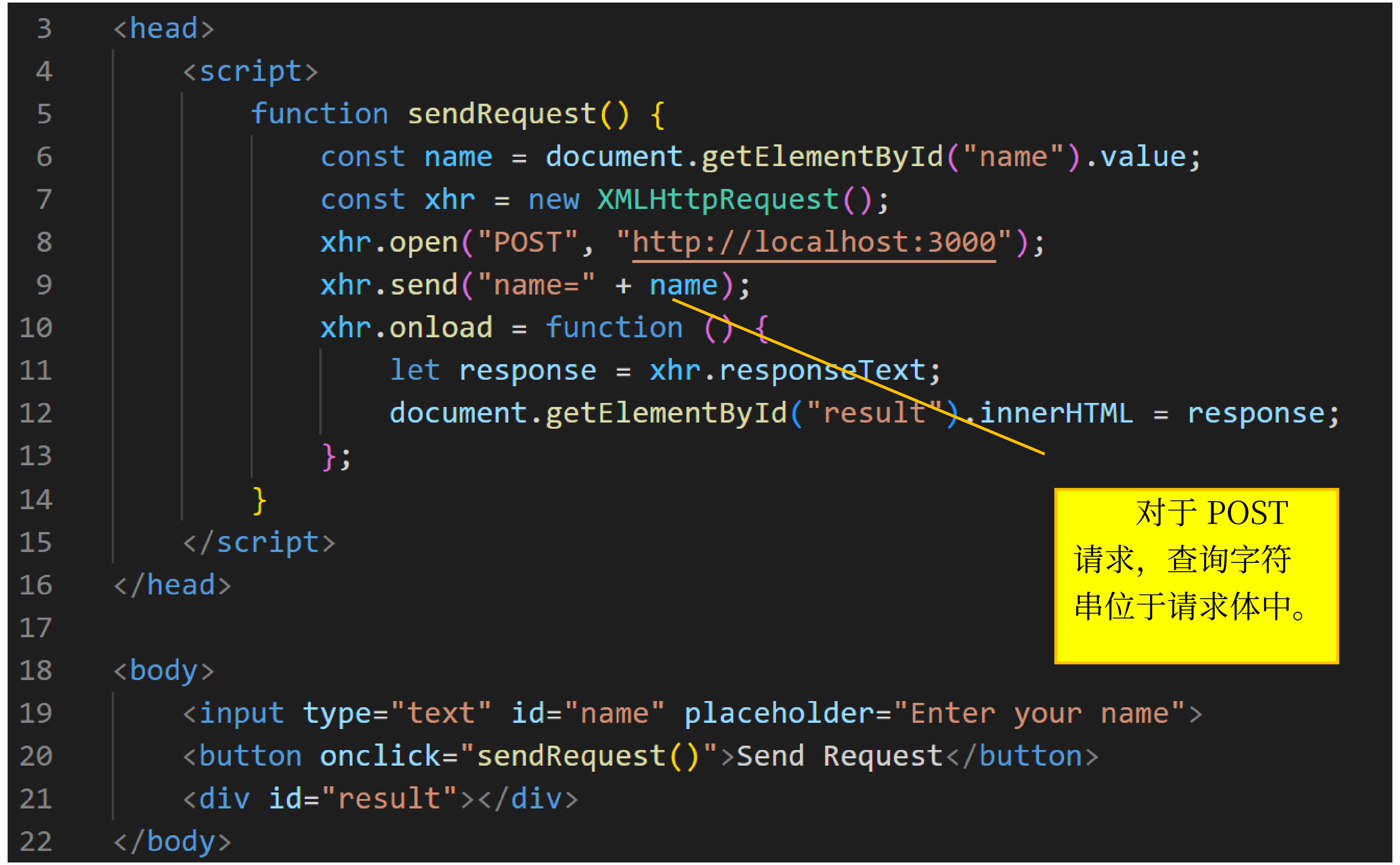
## POST client side



```
<head>
    <script>
        function sendRequest() {
            const name = document.getElementById("name").value;
            const xhr = new XMLHttpRequest();
            xhr.open("POST", "http://localhost:3000");
            xhr.send("name=" + name);
            xhr.onload = function () {
                let response = xhr.responseText;
                document.getElementById("result").innerHTML = response;
            };
        }
    </script>
</head>

<body>
    <input type="text" id="name" placeholder="Enter your name">
    <button onclick="sendRequest()">Send Request</button>
    <div id="result"></div>
</body>
```

For POST, the query string sits in the body of the request

## 客户端 POST

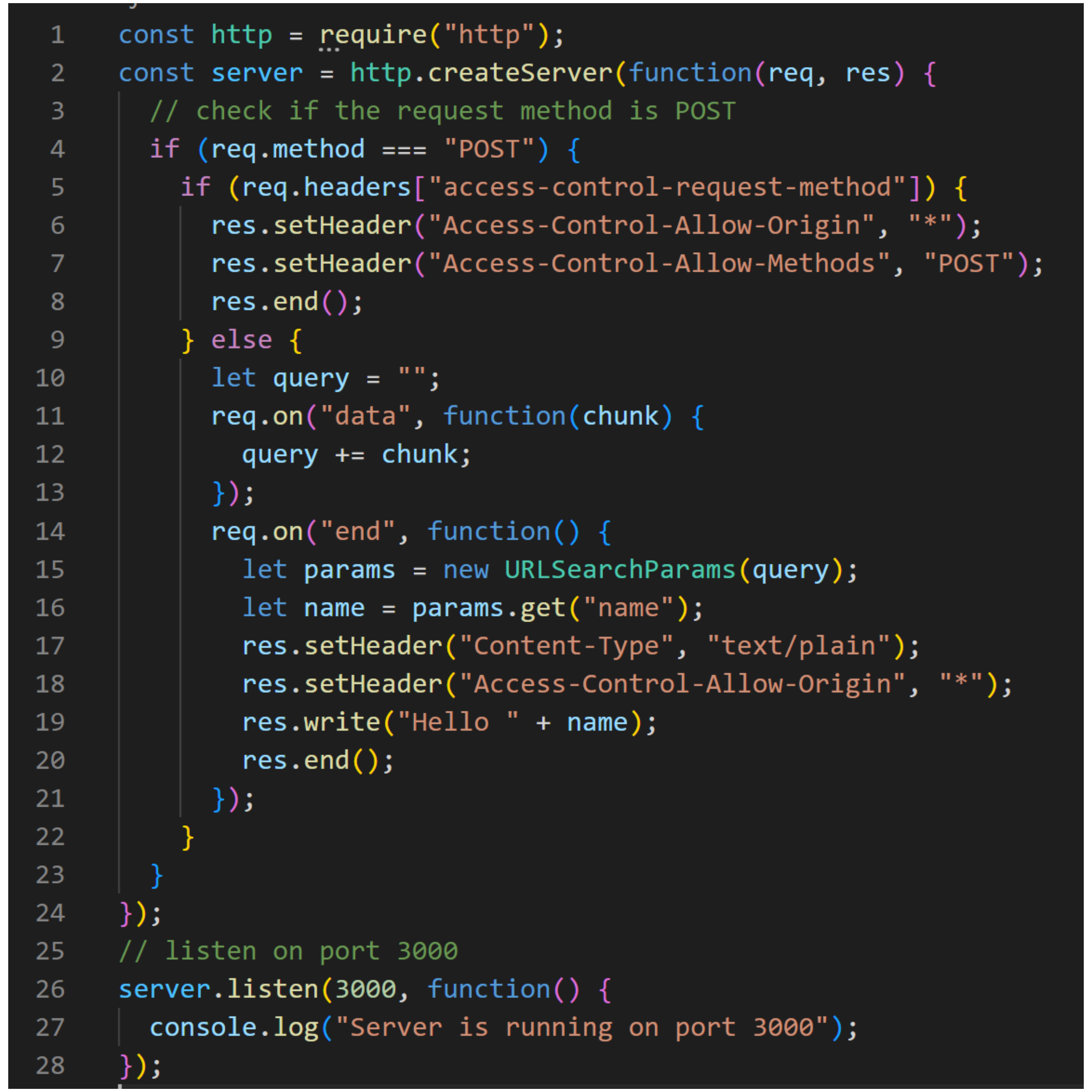


```
<head>
    <script>
        function sendRequest() {
            const name = document.getElementById("name").value;
            const xhr = new XMLHttpRequest();
            xhr.open("POST", "http://localhost:3000");
            xhr.send("name=" + name);
            xhr.onload = function () {
                let response = xhr.responseText;
                document.getElementById("result").innerHTML = response;
            };
        }
    </script>
</head>

<body>
    <input type="text" id="name" placeholder="Enter your name">
    <button onclick="sendRequest()">Send Request</button>
    <div id="result"></div>
</body>
```

对于 POST 请求，查询字符串位于请求体中。

## POST server side

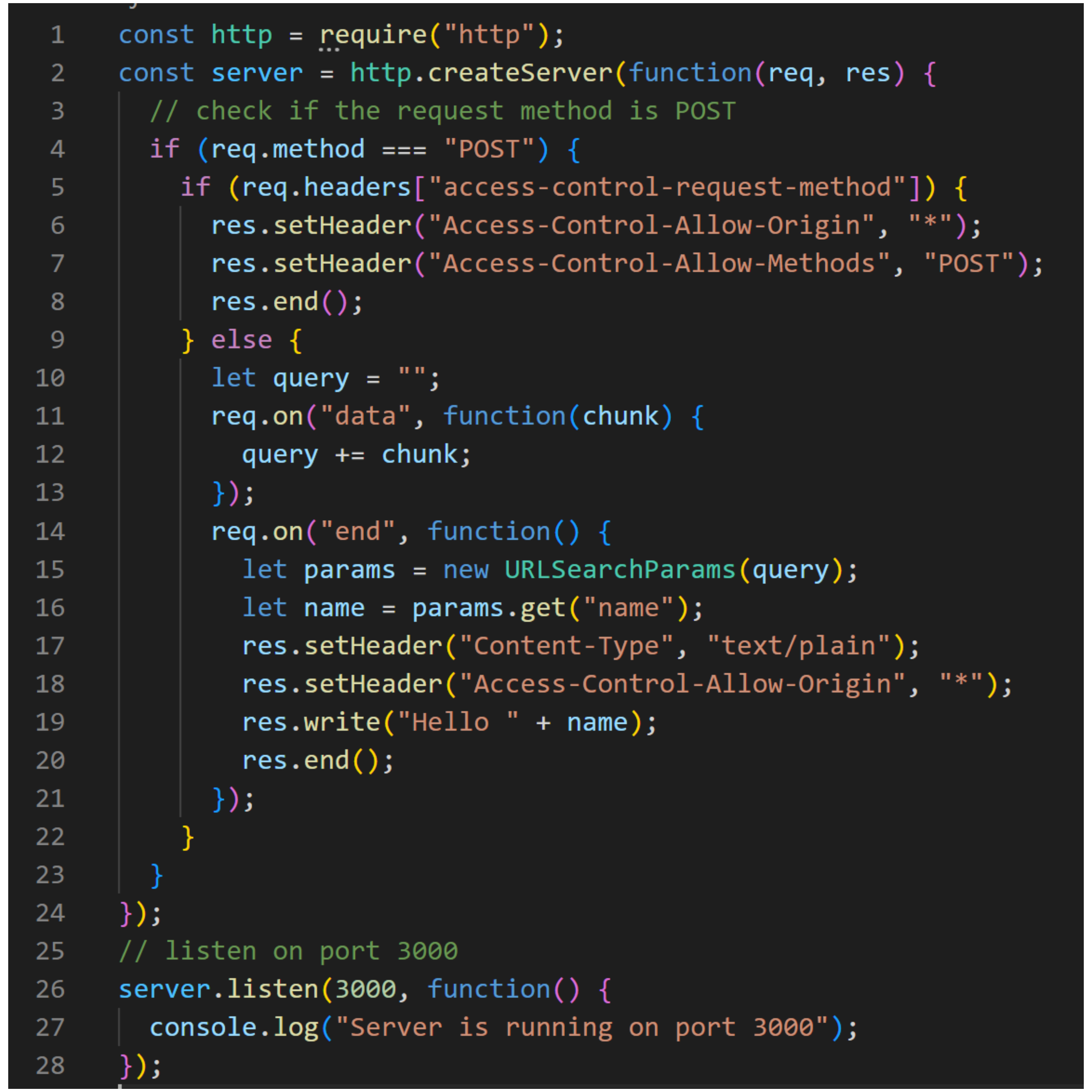
- See how data is
- extracted from the
- POST request ***in chunks***



```
const http = require("http");
const server = http.createServer(function(req, res) {
  // check if the request method is POST
  if (req.method === "POST") {
    if (req.headers["access-control-request-method"]) {
      res.setHeader("Access-Control-Allow-Origin", "*");
      res.setHeader("Access-Control-Allow-Methods", "POST");
      res.end();
    } else {
      let query = "";
      req.on("data", function(chunk) {
        query += chunk;
      });
      req.on("end", function() {
        let params = new URLSearchParams(query);
        let name = params.get("name");
        res.setHeader("Content-Type", "text/plain");
        res.setHeader("Access-Control-Allow-Origin", "*");
        res.write("Hello " + name);
        res.end();
      });
    }
  }
});
// listen on port 3000
server.listen(3000, function() {
  console.log("Server is running on port 3000");
});
```

## POST 服务器端

- 了解数据如何
- 从 POST 请求中
- 分块提取 （***in chunks***）



```
const http = require("http");
const server = http.createServer(function(req, res) {
  // check if the request method is POST
  if (req.method === "POST") {
    if (req.headers["access-control-request-method"]) {
      res.setHeader("Access-Control-Allow-Origin", "*");
      res.setHeader("Access-Control-Allow-Methods", "POST");
      res.end();
    } else {
      let query = "";
      req.on("data", function(chunk) {
        query += chunk;
      });
      req.on("end", function() {
        let params = new URLSearchParams(query);
        let name = params.get("name");
        res.setHeader("Content-Type", "text/plain");
        res.setHeader("Access-Control-Allow-Origin", "*");
        res.write("Hello " + name);
        res.end();
      });
    }
  }
});
// listen on port 3000
server.listen(3000, function() {
  console.log("Server is running on port 3000");
});
```

## Demo 4

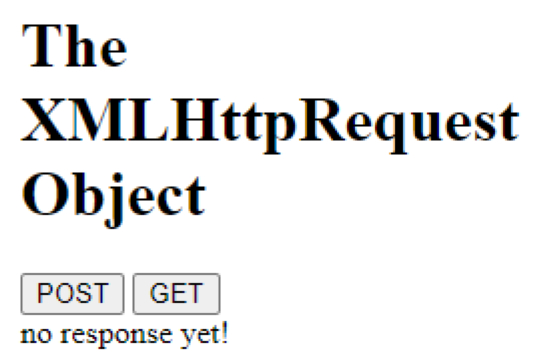
Ajax call using both
POST and GET
methods

## 演示 4

使用 POST 和 GET
方法发起 Ajax 请求

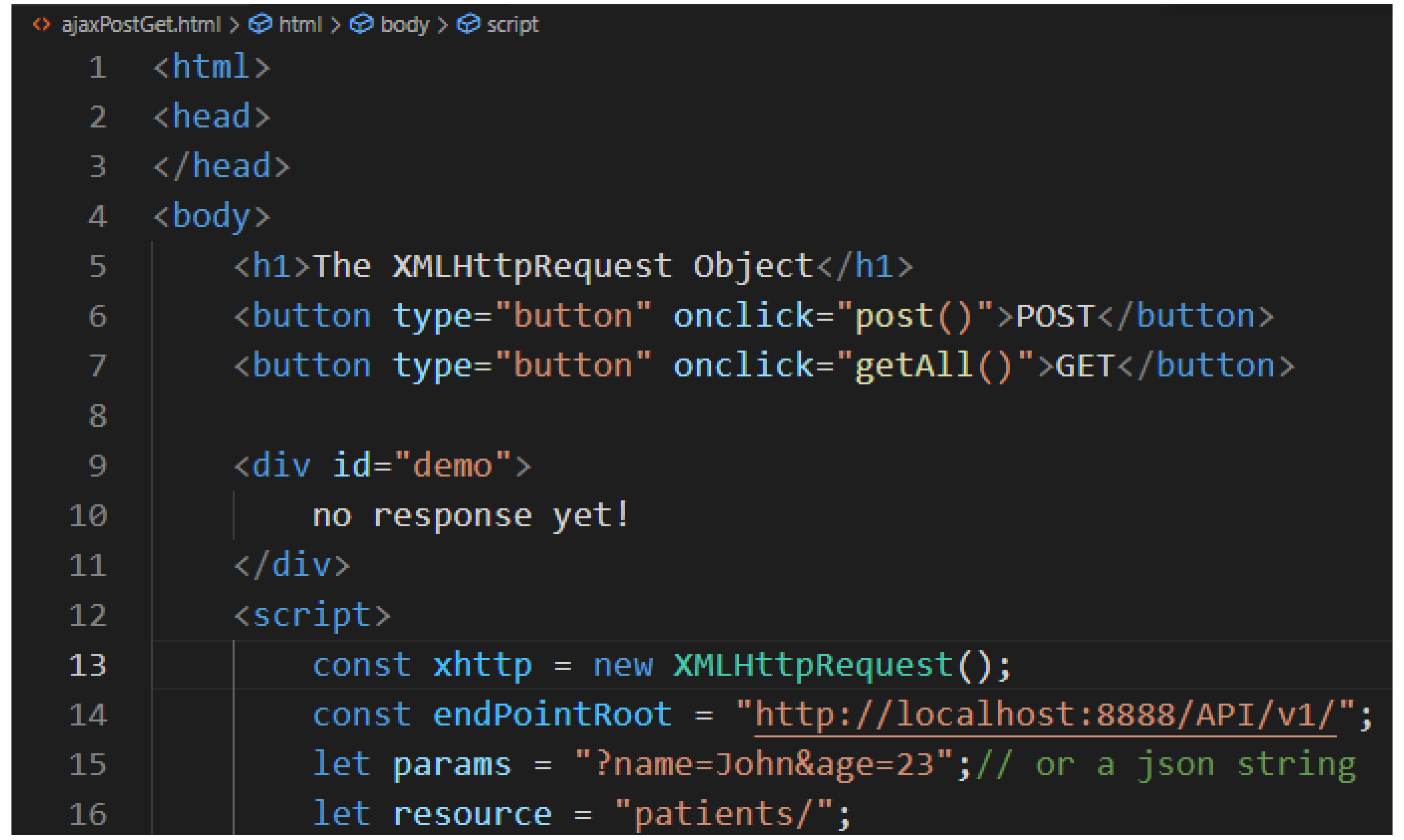
## Client side: POST and GET (ajaxPostGet.html)

- Some variable initializations


**The XMLHttpRequest Object**

POST GET

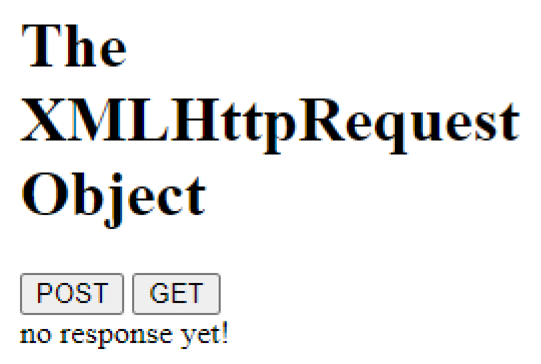
no response yet!


ajaxPostGet.html > html > body > script

```
<html>
<head>
</head>
<body>
    <h1>The XMLHttpRequest Object</h1>
    <button type="button" onclick="post()">POST</button>
    <button type="button" onclick="getAll()">GET</button>

    <div id="demo">
        no response yet!
    </div>
    <script>
        const xhttp = new XMLHttpRequest();
        const endPointRoot = "http://localhost:8888/API/v1/";
        let params = "?name=John&age=23";// or a json string
        let resource = "patients/";
```

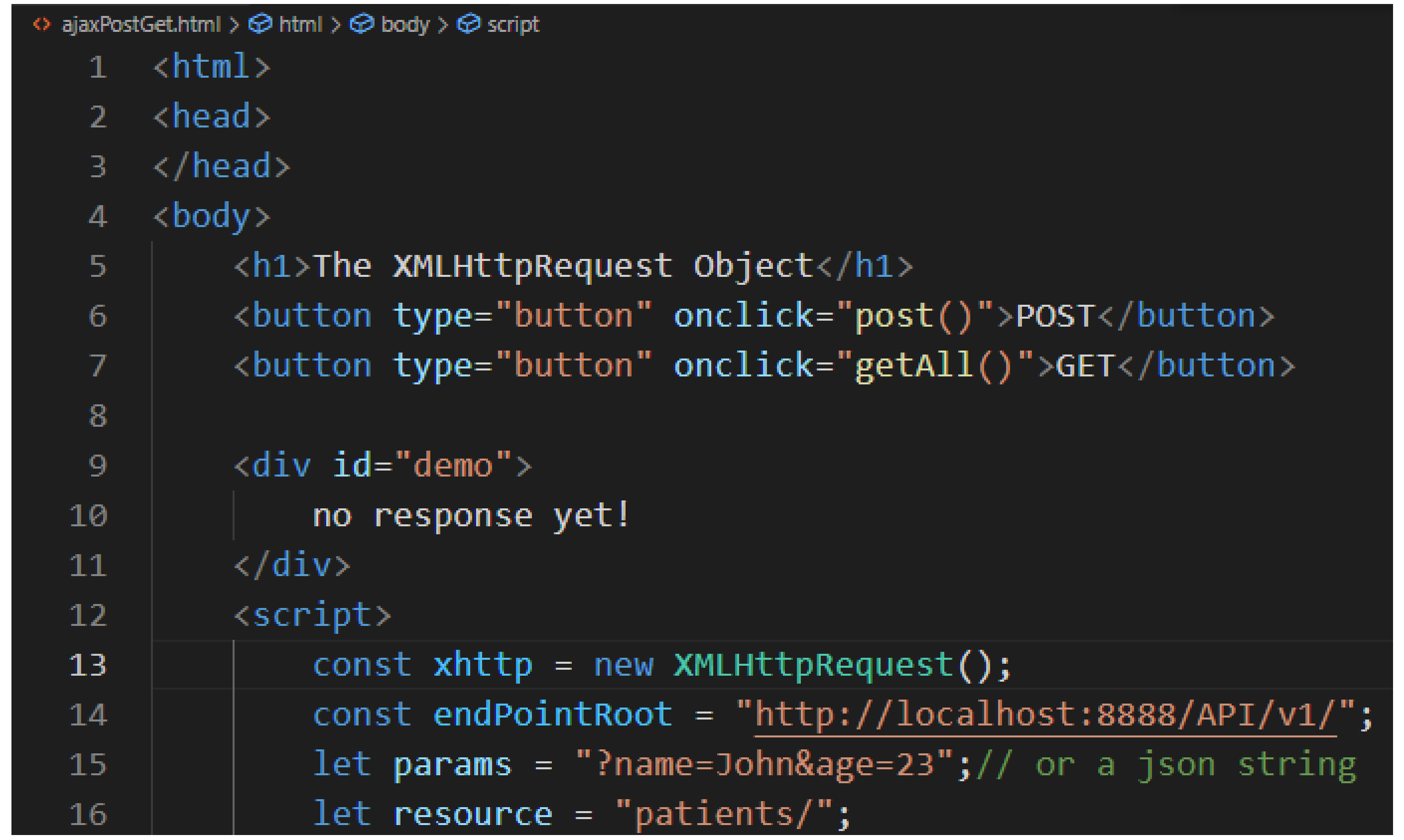
## 客户端：POST 和 GET（ajaxPostGet.html）

- 部分变量初始化


**The XMLHttpRequest Object**

POST GET

no response yet!

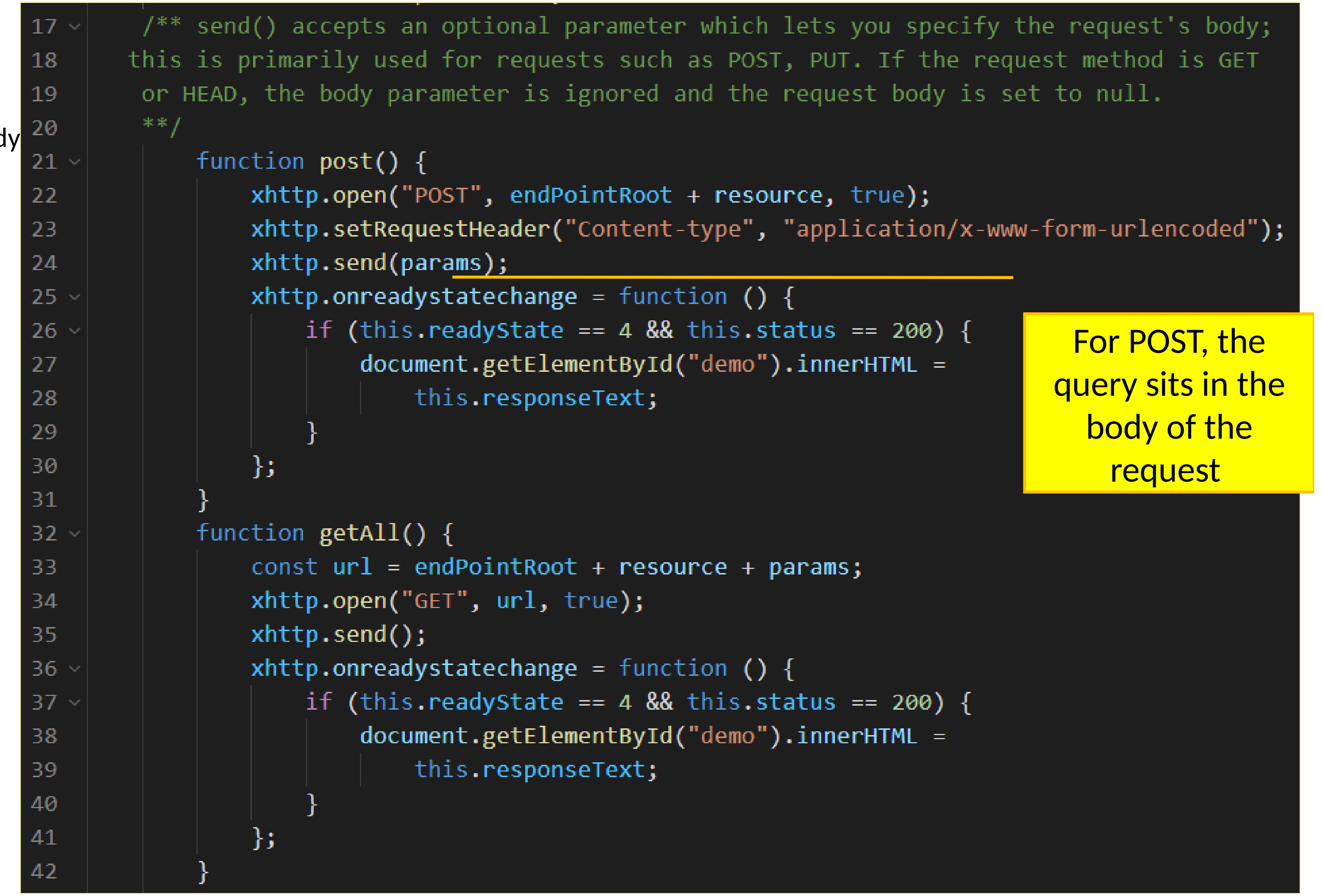

ajaxPostGet.html > html > body > script

```
<html>
<head>
</head>
<body>
    <h1>The XMLHttpRequest Object</h1>
    <button type="button" onclick="post()">POST</button>
    <button type="button" onclick="getAll()">GET</button>

    <div id="demo">
        no response yet!
    </div>
    <script>
        const xhttp = new XMLHttpRequest();
        const endPointRoot = "http://localhost:8888/API/v1/";
        let params = "?name=John&age=23";// or a json string
        let resource = "patients/";
```

## Client side: POST and GET (ajaxPostGet.html)

- The 'application/x-www-form-urlencoded'
- content type describes form data that is
- sent in a single block in the HTTP message body

- A header is a piece of information about the data sent via HTTP request. It tells the server receiving the request what type of data is enclosed, its formatting, the language used. (If you remember, The server also puts something similar in the head part of the response)
- More than one header can be put on a HTTP request; each in a 'name' and a 'value' pair



```
 /** send() accepts an optional parameter which lets you specify the request's body;
this is primarily used for requests such as POST, PUT. If the request method is GET
 or HEAD, the body parameter is ignored and the request body is set to null.
 **/
    function post() {
        xhttp.open("POST", endPointRoot + resource, true);
        xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhttp.send(params);
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
    function getAll() {
        const url = endPointRoot + resource + params;
        xhttp.open("GET", url, true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
```

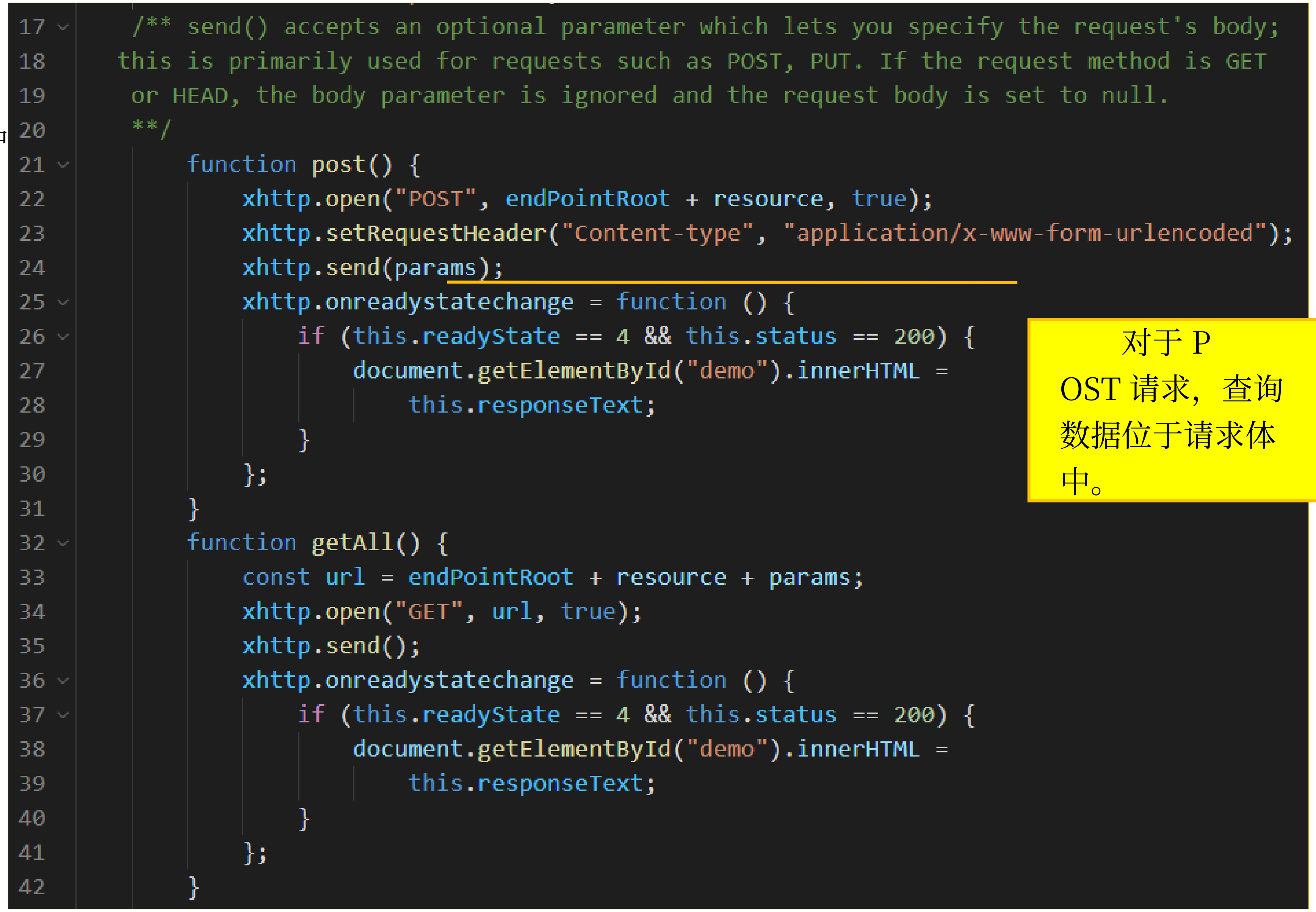
For POST, the query sits in the body of the request

## 客户端：POST 和 GET（ajaxPostGet.html）

- “application/x-www-form-urlencoded”
- 内容类型用于描述表单数据，该数据
- 以单一数据块的形式发送至 HTTP 消息体中

- HTTP 头部是关于通过 HTTP 请求所发送数据的一段信息，它向接收该请求的服务器说明：

  所封装数据的类型、格式及所用语言。（如果您还记得的话，服务器也会在响应消息的头部中添加类似的信息。）
- 一个 HTTP 请求中可包含多个请求头，每个请求头均由一个“名称”和一个“值”组成。

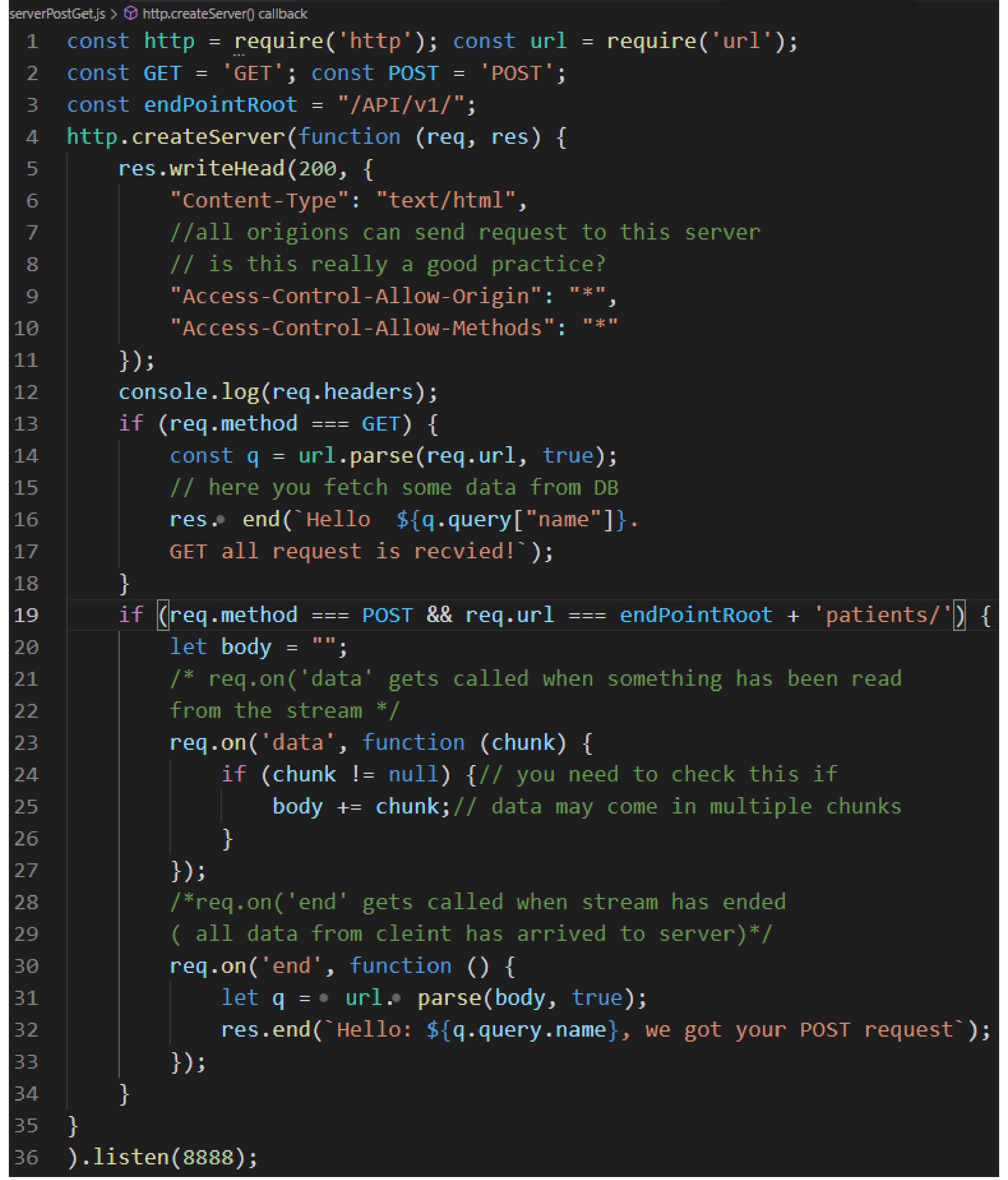


```
 /** send() accepts an optional parameter which lets you specify the request's body;
this is primarily used for requests such as POST, PUT. If the request method is GET
 or HEAD, the body parameter is ignored and the request body is set to null.
 **/
    function post() {
        xhttp.open("POST", endPointRoot + resource, true);
        xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
        xhttp.send(params);
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
    function getAll() {
        const url = endPointRoot + resource + params;
        xhttp.open("GET", url, true);
        xhttp.send();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                document.getElementById("demo").innerHTML =
                    this.responseText;
            }
        };
    }
```

对于 P OST 请求，查询数据位于请求体中。

## **Server** side: POST and GET (serverPostGet.js)

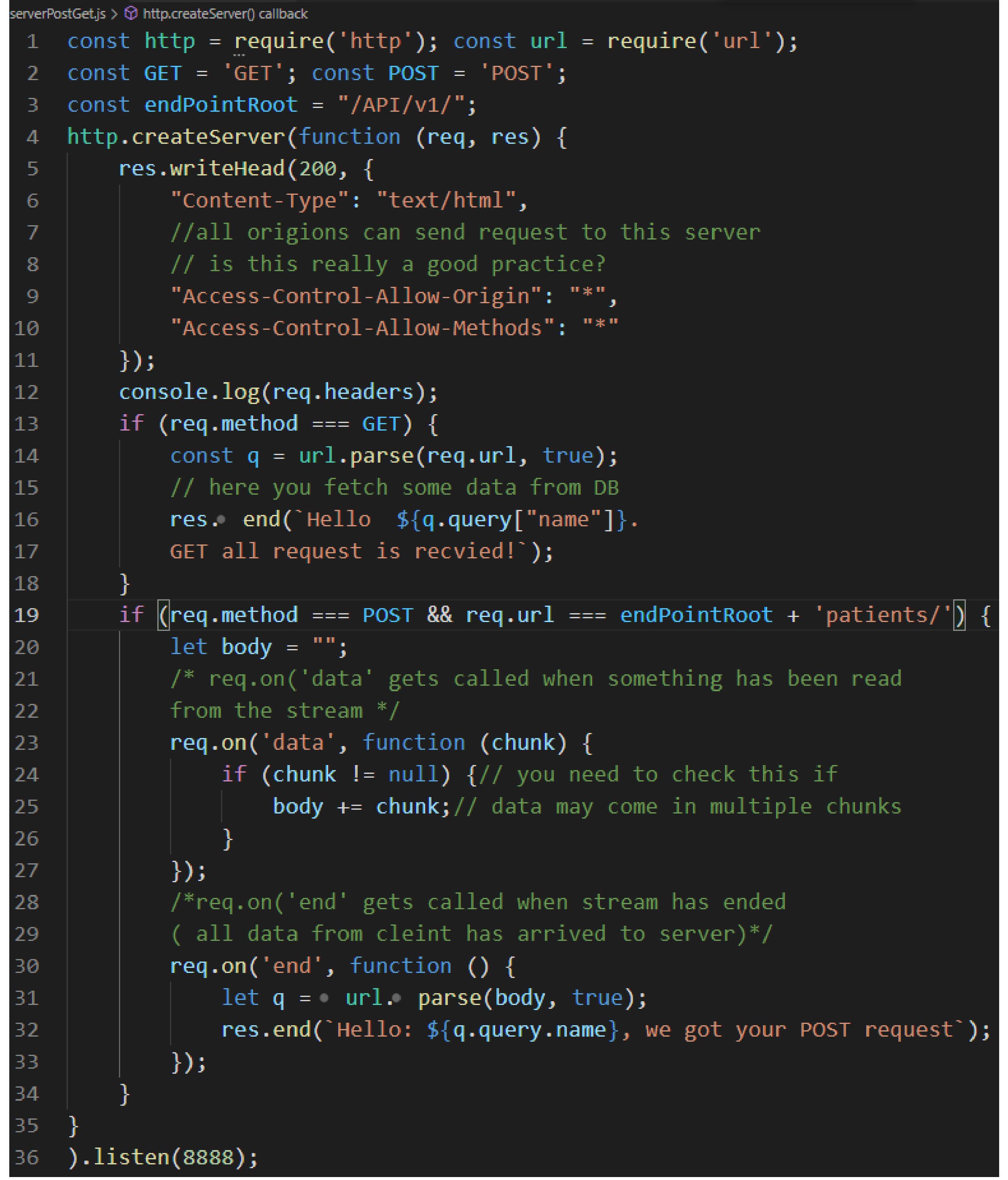
- Note the differences between handling a GET request and a POST request
- See how data is extracted from the POST request ***in chunks***



```javascript
const http = require('http'); const url = require('url');
const GET = 'GET'; const POST = 'POST';
const endPointRoot = "/API/v1/";
http.createServer(function (req, res) {
    res.writeHead(200, {
        "Content-Type": "text/html",
        //all origions can send request to this server
        // is this really a good practice?
        "Access-Control-Allow-Origin": "*",
        "Access-Control-Allow-Methods": "*"
    });
    console.log(req.headers);
    if (req.method === GET) {
        const q = url.parse(req.url, true);
        // here you fetch some data from DB
        res. end(`Hello  ${q.query["name"]}.
        GET all request is recvied!`);
    }
    if (req.method === POST && req.url === endPointRoot + 'patients/') {
        let body = "";
        /* req.on('data' gets called when something has been read
        from the stream */
        req.on('data', function (chunk) {
            if (chunk != null) {// you need to check this if
                body += chunk;// data may come in multiple chunks
            }
        });
        /*req.on('end' gets called when stream has ended
        ( all data from cleint has arrived to server)*/
        req.on('end', function () {
            let q = url. parse(body, true);
            res.end(`Hello: ${q.query.name}, we got your POST request`);
        });
    }
}
).listen(8888);
```

## 服务器端：POST 和 GET（serverPostGet.js）

- 注意处理 GET 请求与 POST 请求之间的区别
- 查看如何从 POST 请求中以 分块方式 提取数据



```javascript
const http = require('http'); const url = require('url');
const GET = 'GET'; const POST = 'POST';
const endPointRoot = "/API/v1/";
http.createServer(function (req, res) {
    res.writeHead(200, {
        "Content-Type": "text/html",
        //all origions can send request to this server
        // is this really a good practice?
        "Access-Control-Allow-Origin": "*",
        "Access-Control-Allow-Methods": "*"
    });
    console.log(req.headers);
    if (req.method === GET) {
        const q = url.parse(req.url, true);
        // here you fetch some data from DB
        res. end(`Hello  ${q.query["name"]}.
        GET all request is recvied!`);
    }
    if (req.method === POST && req.url === endPointRoot + 'patients/') {
        let body = "";
        /* req.on('data' gets called when something has been read
        from the stream */
        req.on('data', function (chunk) {
            if (chunk != null) {// you need to check this if
                body += chunk;// data may come in multiple chunks
            }
        });
        /*req.on('end' gets called when stream has ended
        ( all data from cleint has arrived to server)*/
        req.on('end', function () {
            let q = url. parse(body, true);
            res.end(`Hello: ${q.query.name}, we got your POST request`);
        });
    }
}
).listen(8888);
```

Question!

In the context of HTTP methods and Cross-Origin Resource Sharing (CORS), what is a "preflight request,“?

问题!

在 HTTP 方法和跨域资源共享（CORS）的背景下,“预检请求”（preflight request）指的是什么?

- Disclaimer: It’s the students' responsibility to write their own code.
- Codes provided in lecture notes are just samples to get students a head start .

- 免责声明：编写自身代码是学生的责任。
- 讲义中提供的代码仅作为示例，旨在帮助学生快速入门。

## Working with a Relational Database

4

Content from this point onward are to be self studied for the next week . . .

## 使用关系型数据库

4

从本处开始的内容需在接下来的一周内自主学习……

## CRUD

- This week we will discuss how to connect node js to a relational DB such as MySQL server ( the free open source version is called MarinaDB)
- We will learn how to run SQL queries which can be used to
- **C**reate a table  or DB
- **R**ead from a DB
- **U**pdate records on a DB
- **D**elete a record or Drop a table
- All of this operation with DB all referred as CRUD in brief
- In this lecture we focus on the node js part mostly assuming writing SQL statement is covered in your term 2 DB course

## CRUD

- 本周我们将讨论如何将 Node.js 连接到关系型数据库（例如 MySQL 服务器，其免费开源版本称为 MariaDB）
- 我们将学习如何执行 SQL 查询，这些查询可用于
- **C**创建表或数据库
- **R**从数据库中读取数据
- **U**更新数据库中的记录
- **D**删除某条记录或删除某个表
- 所有这些针对数据库的操作，简称为 CRUD。
- 本讲主要聚焦于 Node.js 部分，假设 SQL 语句的编写已在你们第二学期的数据库课程中涵盖。

Introduction

## to Connecting to DB using node js on your **PC**

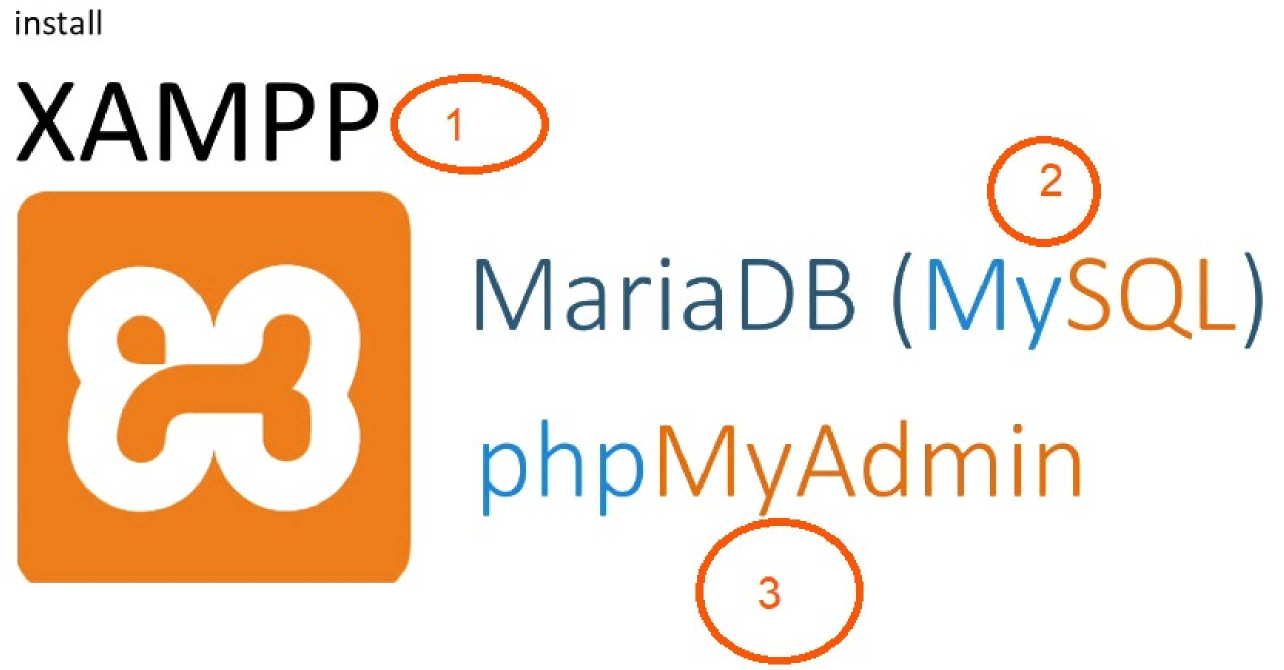
简介

## to 连接数据库 使用 Node.js 在您的 **PC** 上

## Using XAMPP package installer

- You need a DB engine that runs SQL statements for you
- And a DB administrative tool like an editor on which you could write your SQL statements and see result visually
- 1- XAMPP installs both
- 2- The DB engine (mySQL)
- 3- And DB admin tool (phpMyAdmin)
-

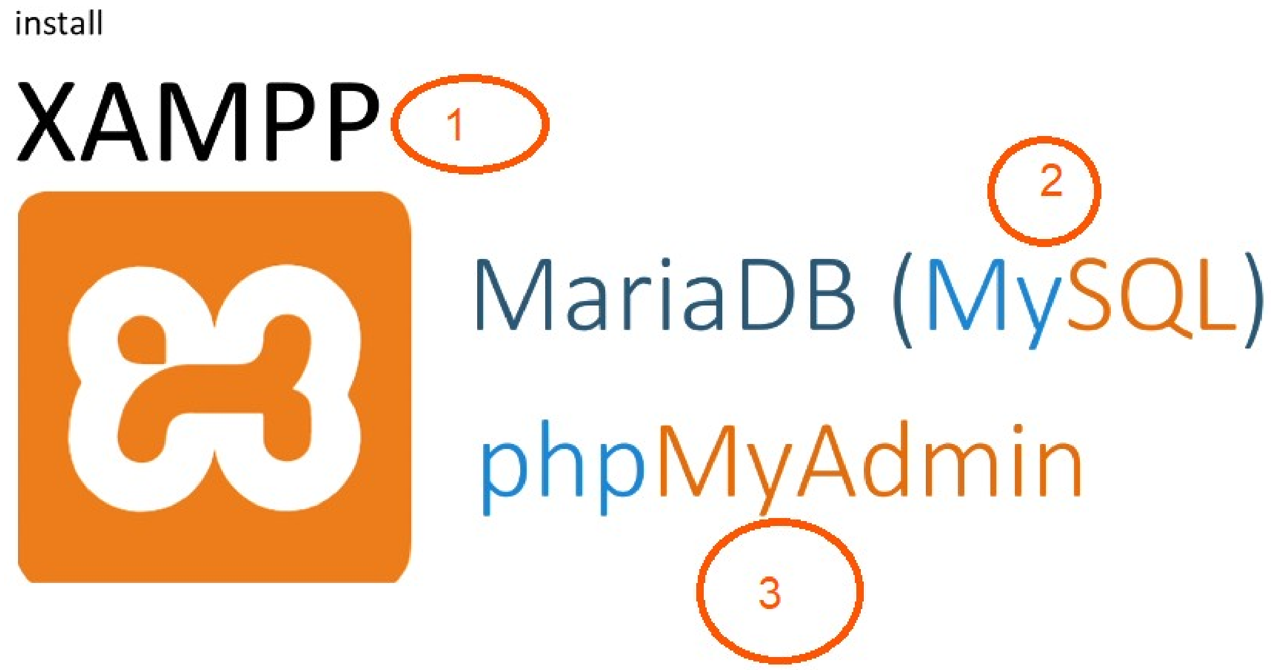
5



## 使用 XAMPP 软件包安装程序

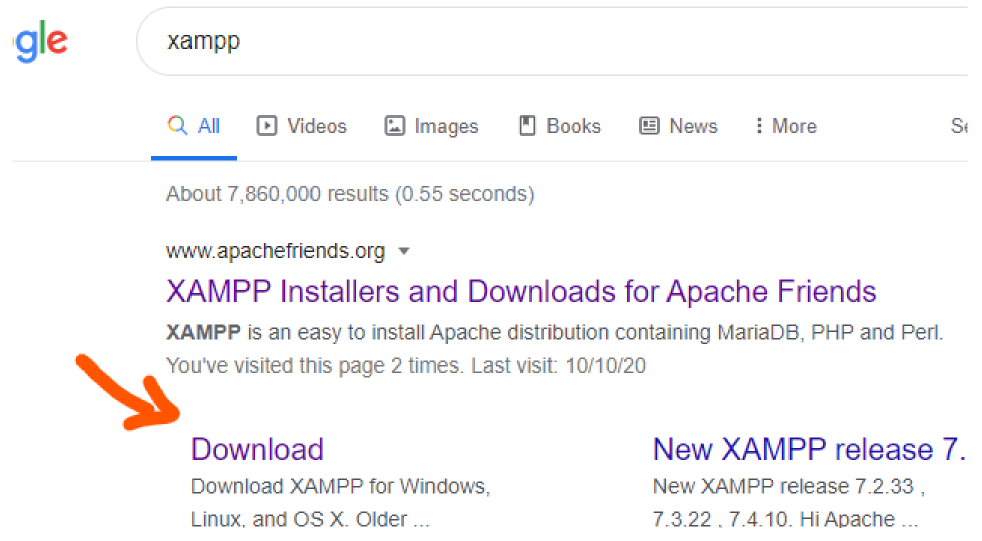
- 您需要一个能够为您执行 SQL 语句的数据库引擎
- 以及一个数据库管理工具（例如类似编辑器的应用程序）， 您可借此编写 SQL 语句并直观地查看执行结果
- 1- XAMPP 同时安装这两者
- 2- 数据库引擎（MySQL）
- 3- 以及数据库管理工具（phpMyAdmin）
-

5



## Setting up DB server on your PC

- We need to install mySQL DB server. It’s a relational DB engine which runs our SQL queries
- As well we need phpMyAdmin which is a web based database administrative tool. It is web based therefore you can use it remotely or share a datable with your partner
- phpMyAdmin needs PHP engine to run
- XAMPP is an application package installer that installs all of those for us



xampp

All Videos Images Books News More

About 7,860,000 results (0.55 seconds)

www.apachefriends.org

XAMPP Installers and Downloads for Apache Friends

XAMPP is an easy to install Apache distribution containing MariaDB, PHP and Perl.

You've visited this page 2 times. Last visit: 10/10/20

Download

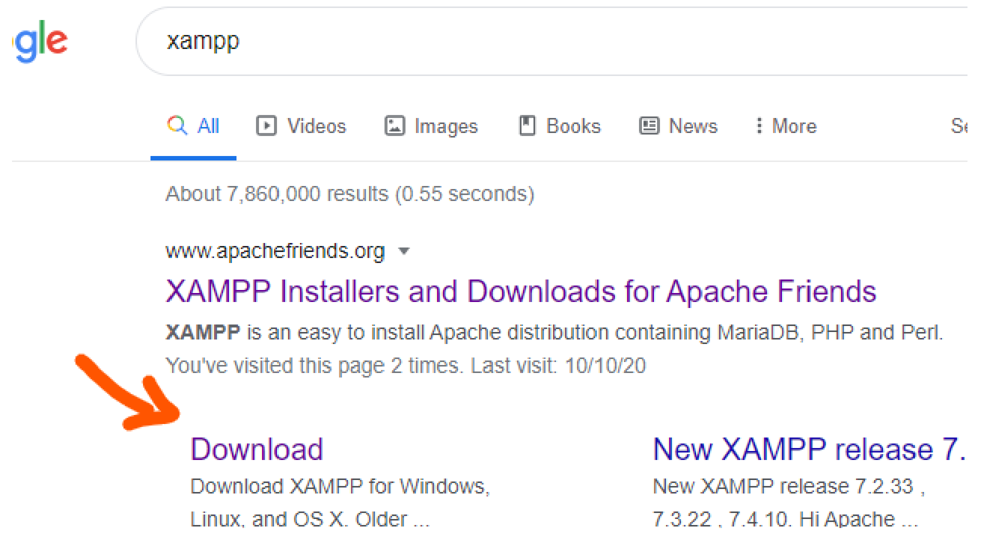
Download XAMPP for Windows, Linux, and OS X. Older ...

New XAMPP release 7.

New XAMPP release 7.2.33 , 7.3.22 , 7.4.10. Hi Apache ...

## 在您的电脑上设置数据库服务器

- 我们需要安装 MySQL 数据库服务器。它是一种关系型数据库引擎，用于执行我们的 SQL 查询。
- 此外，我们还需要 phpMyAdmin——一款基于 Web 的数据库管理工具。由于它是基于 Web 的，因此您可远程使用，或与合作伙伴共享数据库。
- phpMyAdmin 的运行依赖 PHP 引擎。
- XAMPP 是一个应用软件包安装程序，可为我们自动安装上述所有组件。



xampp

All Videos Images Books News More

About 7,860,000 results (0.55 seconds)

www.apachefriends.org

XAMPP Installers and Downloads for Apache Friends

XAMPP is an easy to install Apache distribution containing MariaDB, PHP and Perl.

You've visited this page 2 times. Last visit: 10/10/20

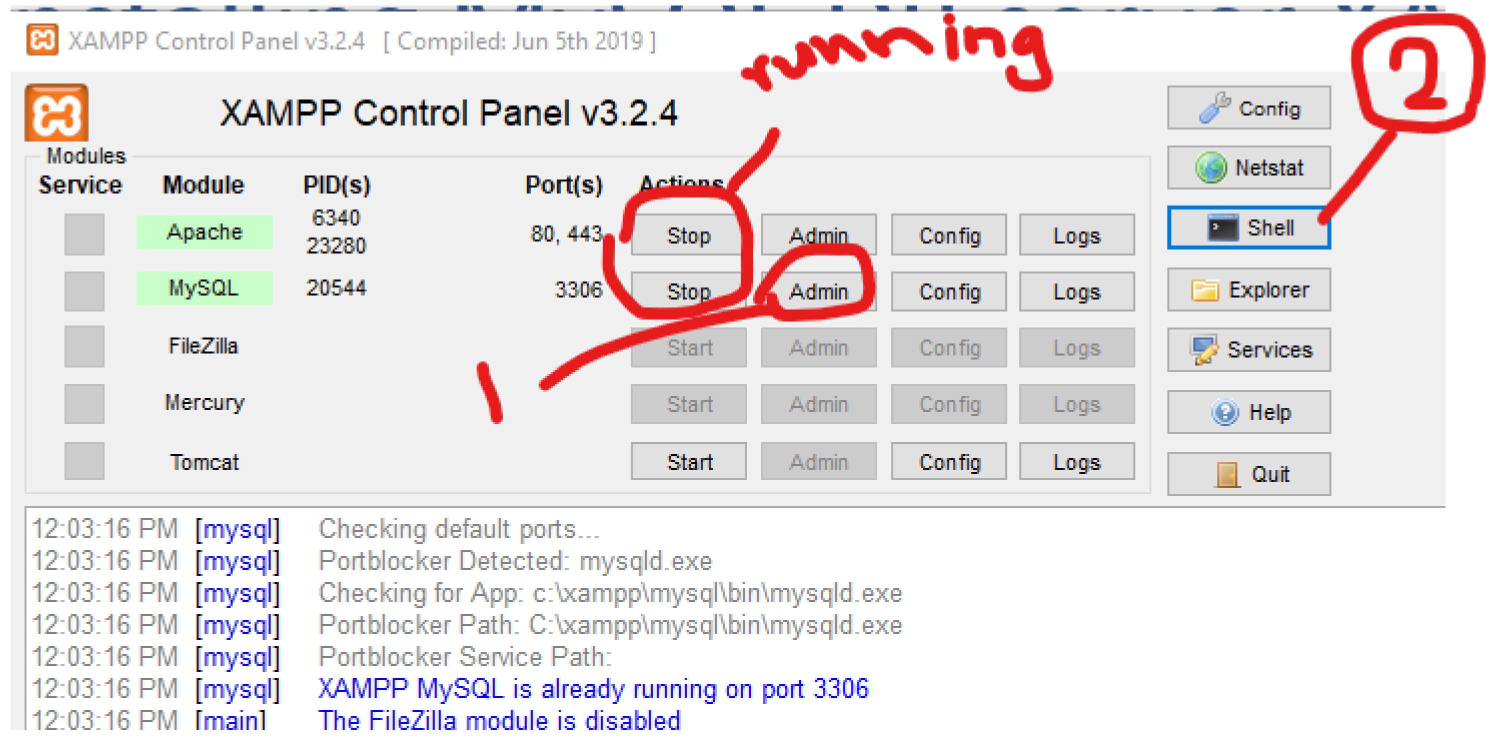
Download

Download XAMPP for Windows, Linux, and OS X. Older ...

New XAMPP release 7.

New XAMPP release 7.2.33 , 7.3.22 , 7.4.10. Hi Apache ...

- 1: clicking on admin will take you to phpMyAdmin
- 2- Clicking on shell will take you to the mySQL command line



| Module | PID(s) | Port(s) |
|---|---|---|
| Apache | 6340<br>23280 | 80, 443 |
| MySQL | 20544 | 3306 |
| FileZilla | | |
| Mercury | | |
| Tomcat | | |

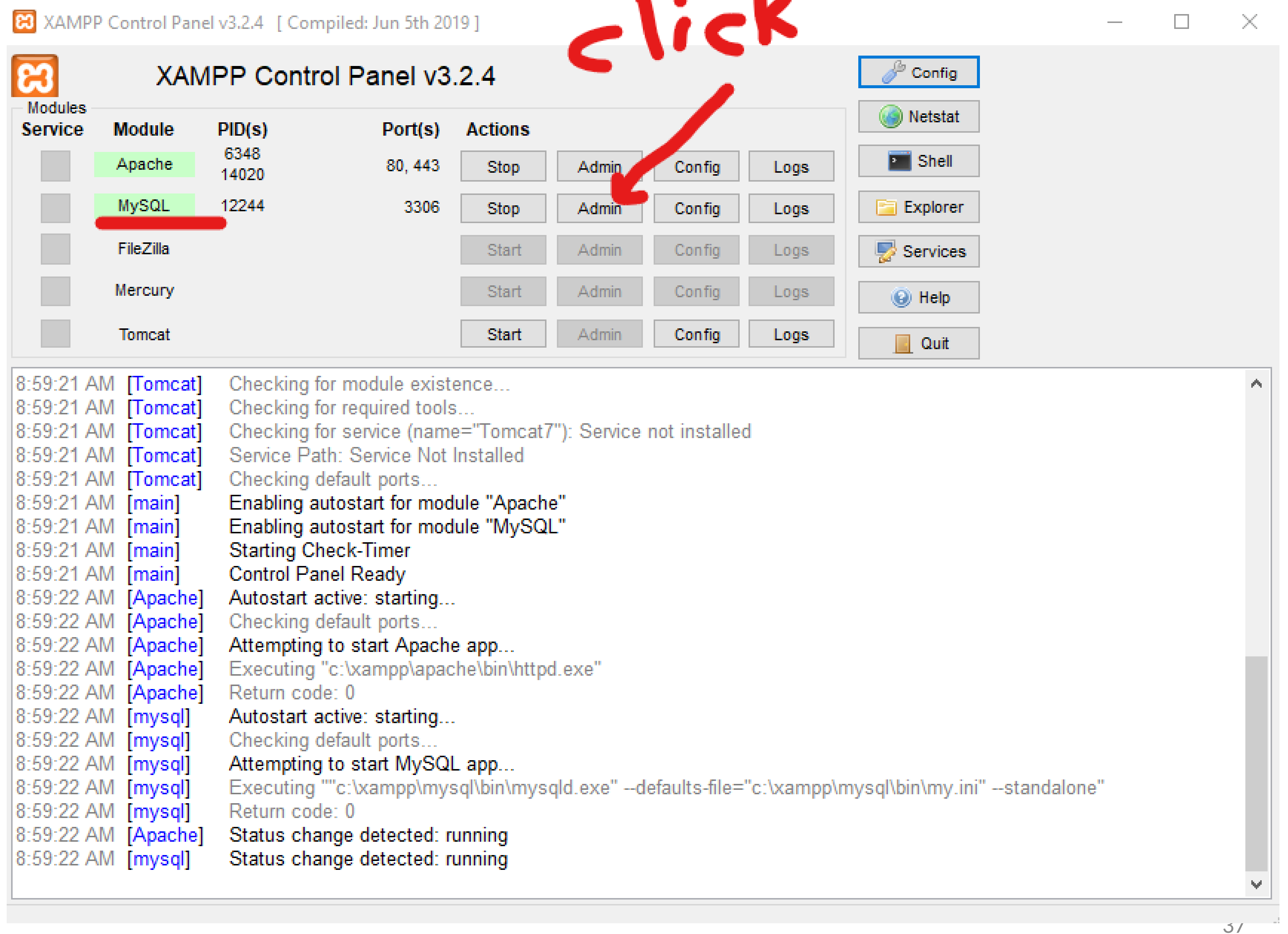
```
12:03:16 PM  [mysql]   Checking default ports...
12:03:16 PM  [mysql]   Portblocker Detected: mysqld.exe
12:03:16 PM  [mysql]   Checking for App: c:\xampp\mysql\bin\mysqld.exe
12:03:16 PM  [mysql]   Portblocker Path: C:\xampp\mysql\bin\mysqld.exe
12:03:16 PM  [mysql]   Portblocker Service Path:
12:03:16 PM  [mysql]   XAMPP MySQL is already running on port 3306
12:03:16 PM  [main]    The FileZilla module is disabled
```

- 1: 单击 “admin” 将跳转至 phpMyAdmin
- 2— 单击 “shell” 将跳转至 MySQL 命令行界面

```
12:03:16 PM  [mysql]   Checking default ports...
12:03:16 PM  [mysql]   Portblocker Detected: mysqld.exe
12:03:16 PM  [mysql]   Checking for App: c:\xampp\mysql\bin\mysqld.exe
12:03:16 PM  [mysql]   Portblocker Path: C:\xampp\mysql\bin\mysqld.exe
12:03:16 PM  [mysql]   Portblocker Service Path:
12:03:16 PM  [mysql]   XAMPP MySQL is already running on port 3306
12:03:16 PM  [main]    The FileZilla module is disabled
```

## 2- goto phpMyAdmin and to create a DB and a table

- Clicking on that link Loads
- [http://localhost/phpmyadmin/](http://localhost/phpmyadmin/)
- ( Make sure MySQL is running on XAMPP)



XAMPP Control Panel v3.2.4 [ Compiled: Jun 5th 2019 ]

XAMPP Control Panel v3.2.4

Modules

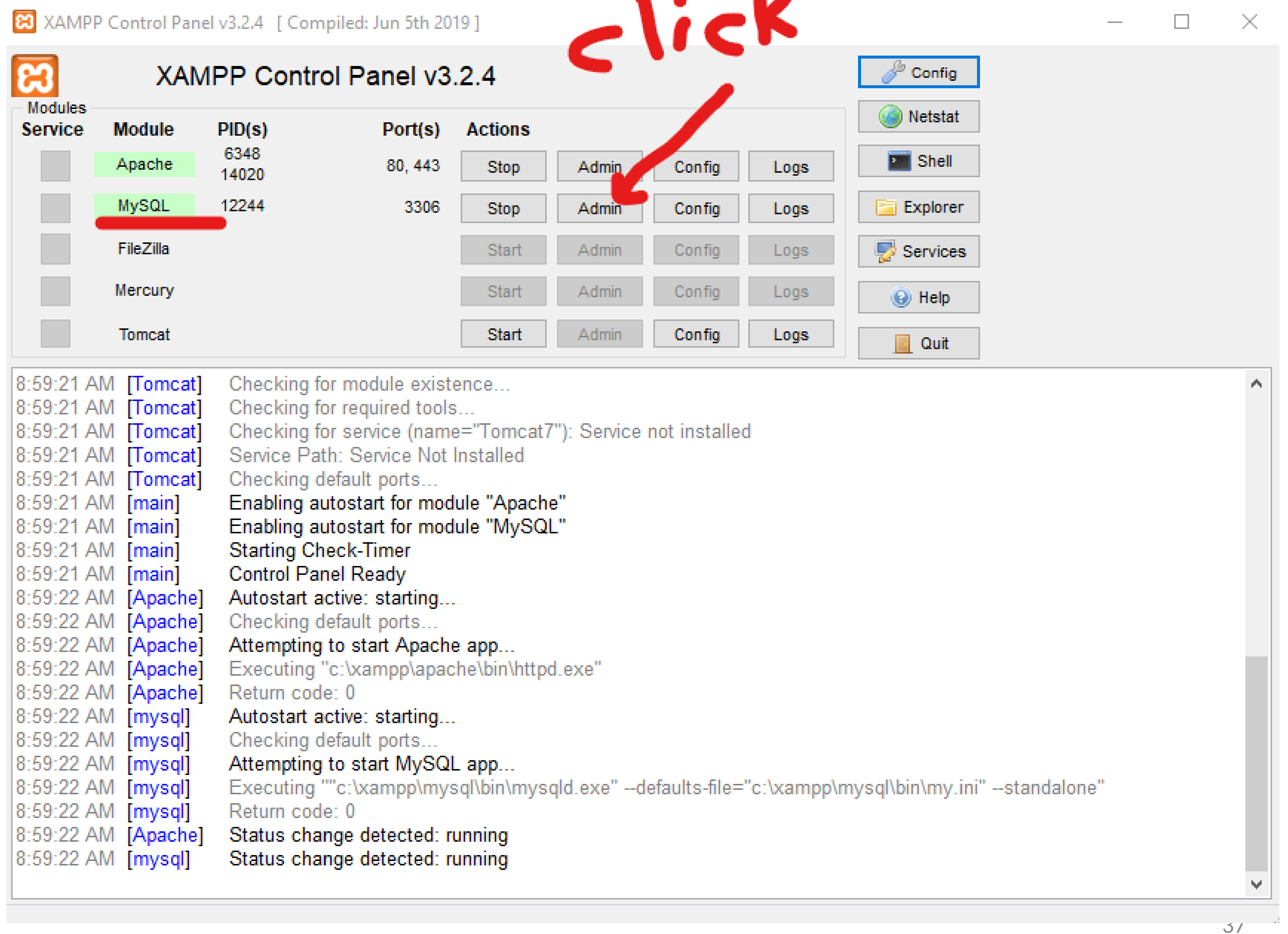
| Service | Module | PID(s) | Port(s) | Actions | | | |
|---|---|---|---|---|---|---|---|
| | Apache | 6348 14020 | 80, 443 | Stop | Admin | Config | Logs |
| | MySQL | 12244 | 3306 | Stop | Admin | Config | Logs |
| | FileZilla | | | Start | Admin | Config | Logs |
| | Mercury | | | Start | Admin | Config | Logs |
| | Tomcat | | | Start | Admin | Config | Logs |

Config, Netstat, Shell, Explorer, Services, Help, Quit

```
8:59:21 AM  [Tomcat]   Checking for module existence...
8:59:21 AM  [Tomcat]   Checking for required tools...
8:59:21 AM  [Tomcat]   Checking for service (name="Tomcat7"): Service not installed
8:59:21 AM  [Tomcat]   Service Path: Service Not Installed
8:59:21 AM  [Tomcat]   Checking default ports...
8:59:21 AM  [main]     Enabling autostart for module "Apache"
8:59:21 AM  [main]     Enabling autostart for module "MySQL"
8:59:21 AM  [main]     Starting Check-Timer
8:59:21 AM  [main]     Control Panel Ready
8:59:22 AM  [Apache]   Autostart active: starting...
8:59:22 AM  [Apache]   Checking default ports...
8:59:22 AM  [Apache]   Attempting to start Apache app...
8:59:22 AM  [Apache]   Executing "c:\xampp\apache\bin\httpd.exe"
8:59:22 AM  [Apache]   Return code: 0
8:59:22 AM  [mysql]    Autostart active: starting...
8:59:22 AM  [mysql]    Checking default ports...
8:59:22 AM  [mysql]    Attempting to start MySQL app...
8:59:22 AM  [mysql]    Executing ""c:\xampp\mysql\bin\mysqld.exe" --defaults-file="c:\xampp\mysql\bin\my.ini" --standalone"
8:59:22 AM  [mysql]    Return code: 0
8:59:22 AM  [Apache]   Status change detected: running
8:59:22 AM  [mysql]    Status change detected: running
```

## 2. 进入 phpMyAdmin 并创建数据库及数据表

- 单击该链接即可加载
- http://localhost/phpmyadmin/
- （请确保 MySQL 正在运行，XAMPP）



XAMPP Control Panel v3.2.4 [ Compiled: Jun 5th 2019 ]

XAMPP Control Panel v3.2.4

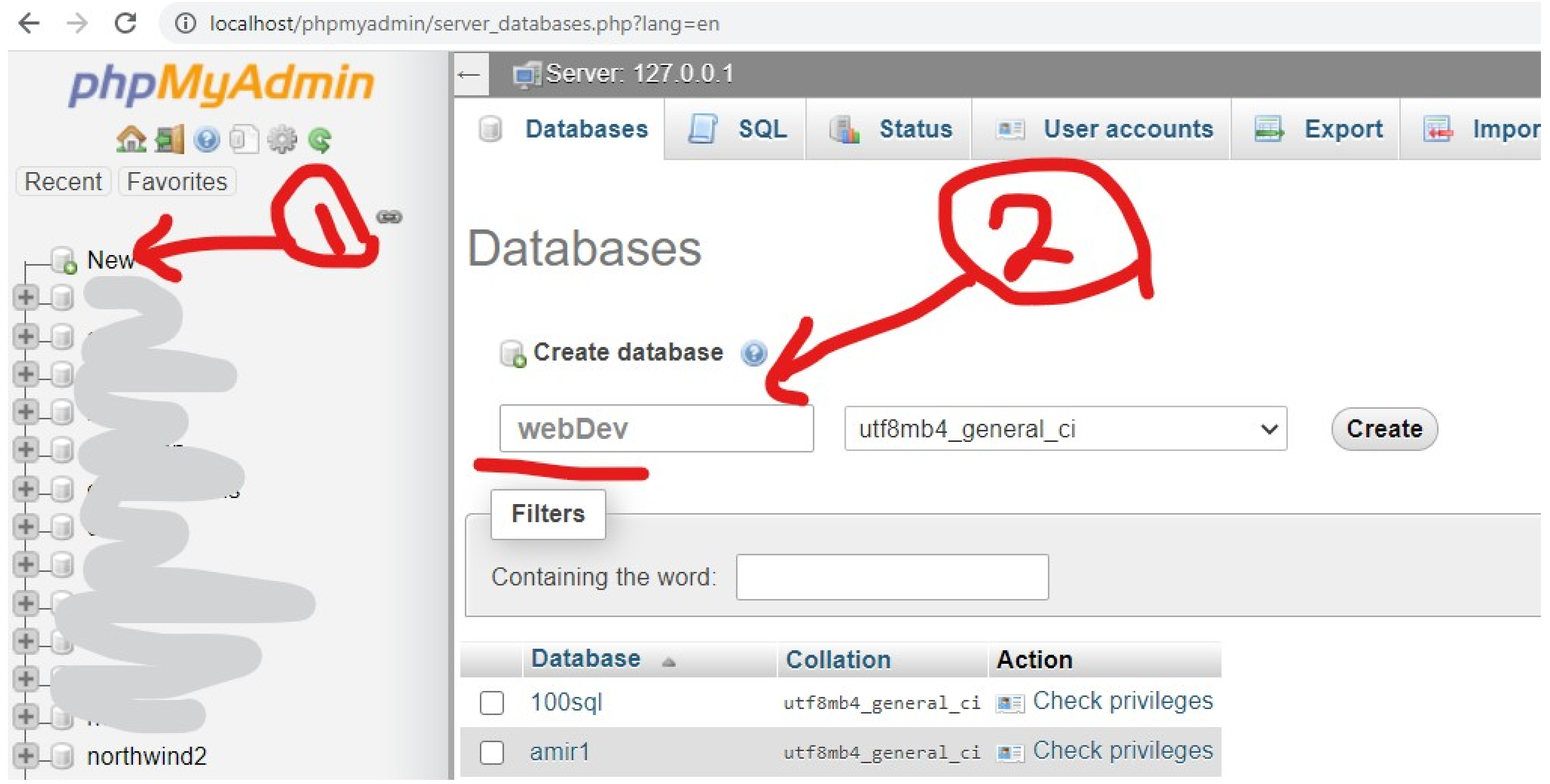
Modules

| Service | Module | PID(s) | Port(s) | Actions | | | |
|---|---|---|---|---|---|---|---|
| | Apache | 6348 14020 | 80, 443 | Stop | Admin | Config | Logs |
| | MySQL | 12244 | 3306 | Stop | Admin | Config | Logs |
| | FileZilla | | | Start | Admin | Config | Logs |
| | Mercury | | | Start | Admin | Config | Logs |
| | Tomcat | | | Start | Admin | Config | Logs |

Config, Netstat, Shell, Explorer, Services, Help, Quit

```
8:59:21 AM  [Tomcat]   Checking for module existence...
8:59:21 AM  [Tomcat]   Checking for required tools...
8:59:21 AM  [Tomcat]   Checking for service (name="Tomcat7"): Service not installed
8:59:21 AM  [Tomcat]   Service Path: Service Not Installed
8:59:21 AM  [Tomcat]   Checking default ports...
8:59:21 AM  [main]     Enabling autostart for module "Apache"
8:59:21 AM  [main]     Enabling autostart for module "MySQL"
8:59:21 AM  [main]     Starting Check-Timer
8:59:21 AM  [main]     Control Panel Ready
8:59:22 AM  [Apache]   Autostart active: starting...
8:59:22 AM  [Apache]   Checking default ports...
8:59:22 AM  [Apache]   Attempting to start Apache app...
8:59:22 AM  [Apache]   Executing "c:\xampp\apache\bin\httpd.exe"
8:59:22 AM  [Apache]   Return code: 0
8:59:22 AM  [mysql]    Autostart active: starting...
8:59:22 AM  [mysql]    Checking default ports...
8:59:22 AM  [mysql]    Attempting to start MySQL app...
8:59:22 AM  [mysql]    Executing ""c:\xampp\mysql\bin\mysqld.exe" --defaults-file="c:\xampp\mysql\bin\my.ini" --standalone"
8:59:22 AM  [mysql]    Return code: 0
8:59:22 AM  [Apache]   Status change detected: running
8:59:22 AM  [mysql]    Status change detected: running
```

## 3- create a DB in phpmyadmin



localhost/phpmyadmin/server_databases.php?lang=en

phpMyAdmin

Recent Favorites

New

[redacted]

northwind2

Server: 127.0.0.1

Databases SQL Status User accounts Export Impor

Databases

Create database

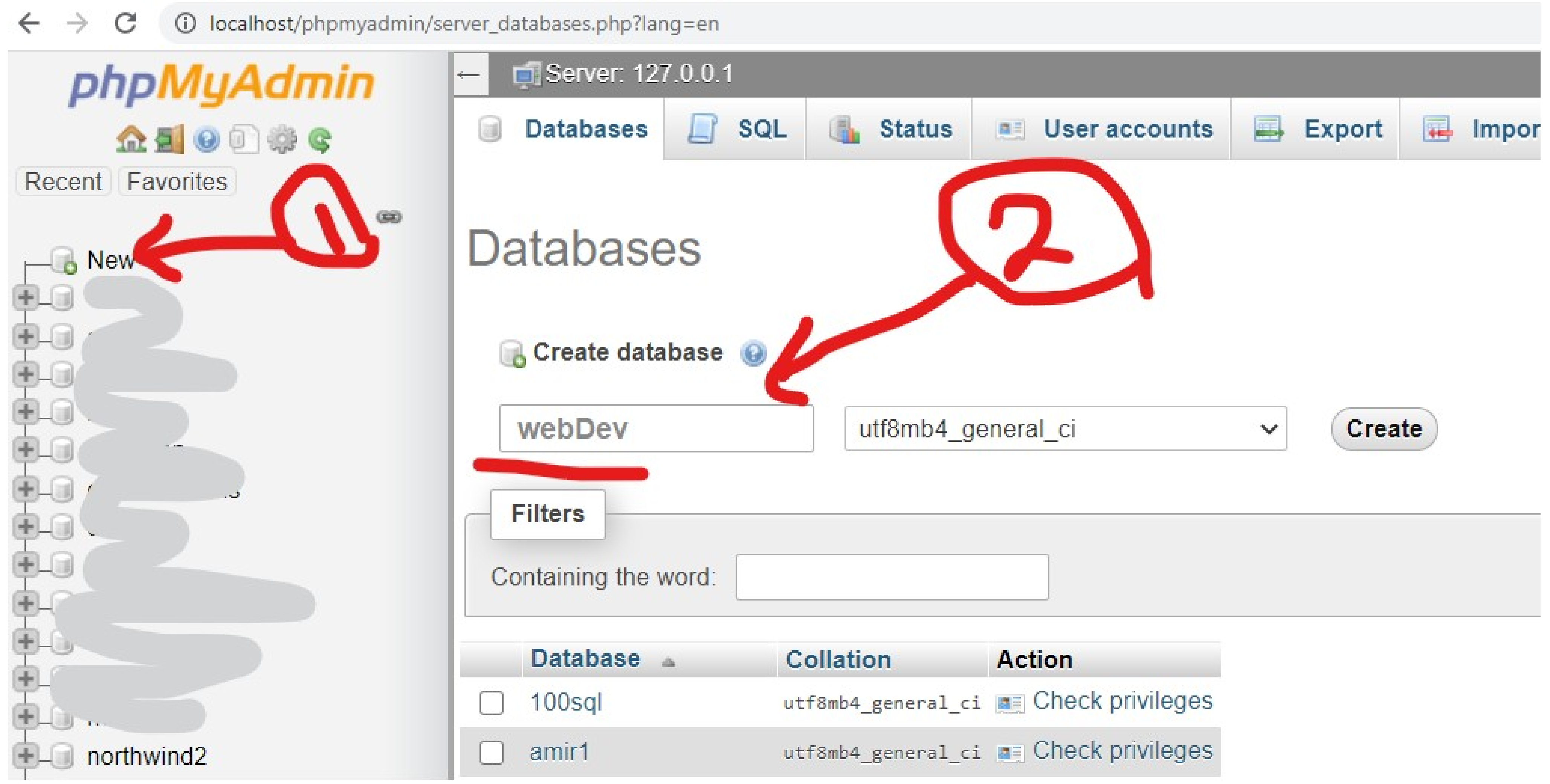
webDev | utf8mb4_general_ci | Create

Filters

Containing the word:

| Database | Collation | Action |
|---|---|---|
| 100sql | utf8mb4_general_ci | Check privileges |
| amir1 | utf8mb4_general_ci | Check privileges |

## 3. 在 phpMyAdmin 中创建数据库



localhost/phpmyadmin/server_databases.php?lang=en

phpMyAdmin

Recent Favorites

New

[redacted]

northwind2

Server: 127.0.0.1

Databases SQL Status User accounts Export Impor

Databases

Create database

webDev | utf8mb4_general_ci | Create

Filters

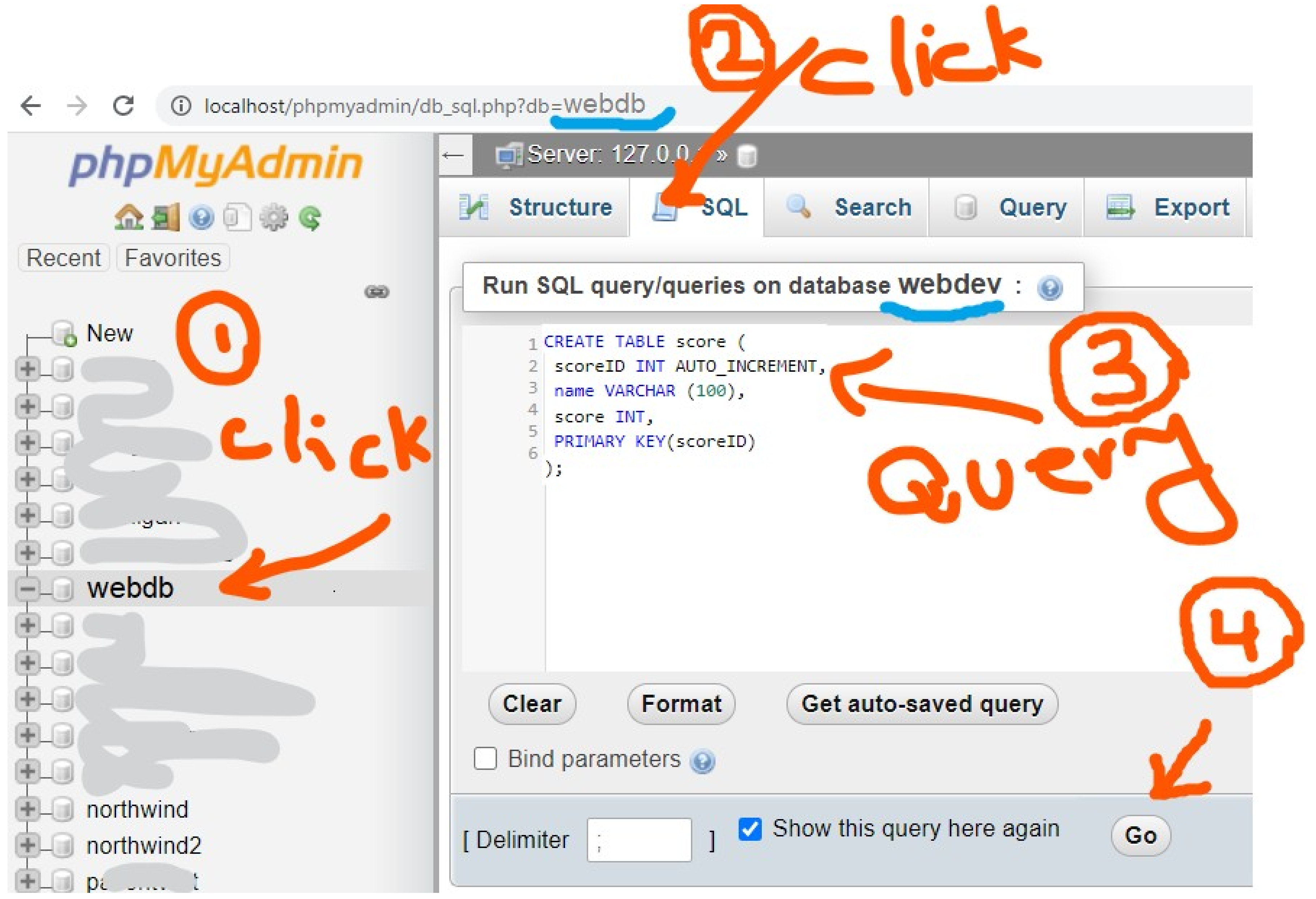
Containing the word:

| Database | Collation | Action |
|---|---|---|
| 100sql | utf8mb4_general_ci | Check privileges |
| amir1 | utf8mb4_general_ci | Check privileges |

## 4- create a table in that DB to store scores

```
CREATE TABLE score (
 scoreID INT AUTO_INCREMENT,
 name VARCHAR (100),
 score INT,
 PRIMARY KEY(scoreID)
);
```

- Note: It is ok to have column names with the same name as the table name, both are score
- the Now our DB is ready!



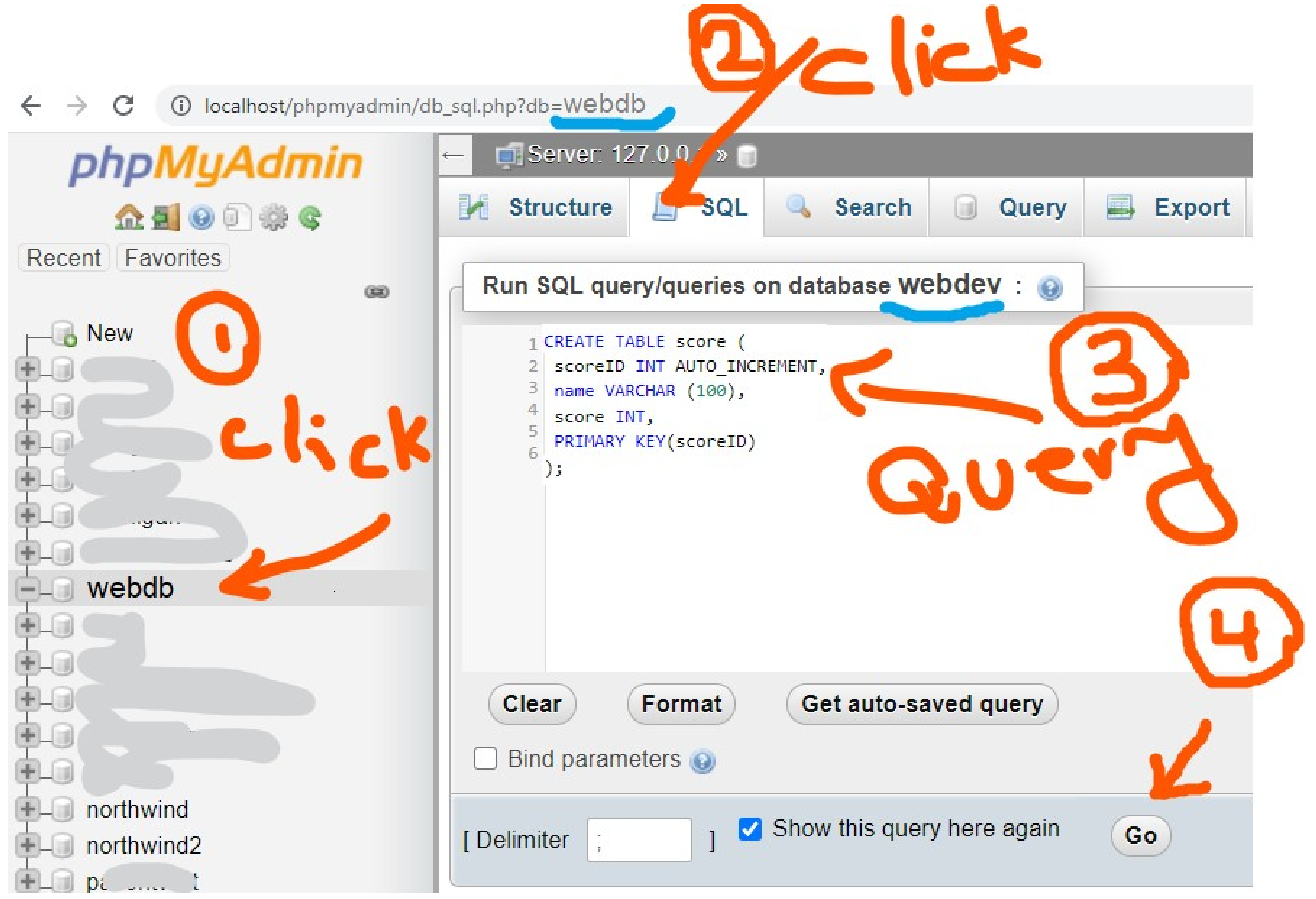
```
CREATE TABLE score (
 scoreID INT AUTO_INCREMENT,
 name VARCHAR (100),
 score INT,
 PRIMARY KEY(scoreID)
);
```

## 4. 在该数据库中创建一张表，用于存储分数。

```
CREATE TABLE score (
 scoreID INT AUTO_INCREMENT,
 name VARCHAR (100),
 score INT,
 PRIMARY KEY(scoreID)
);
```

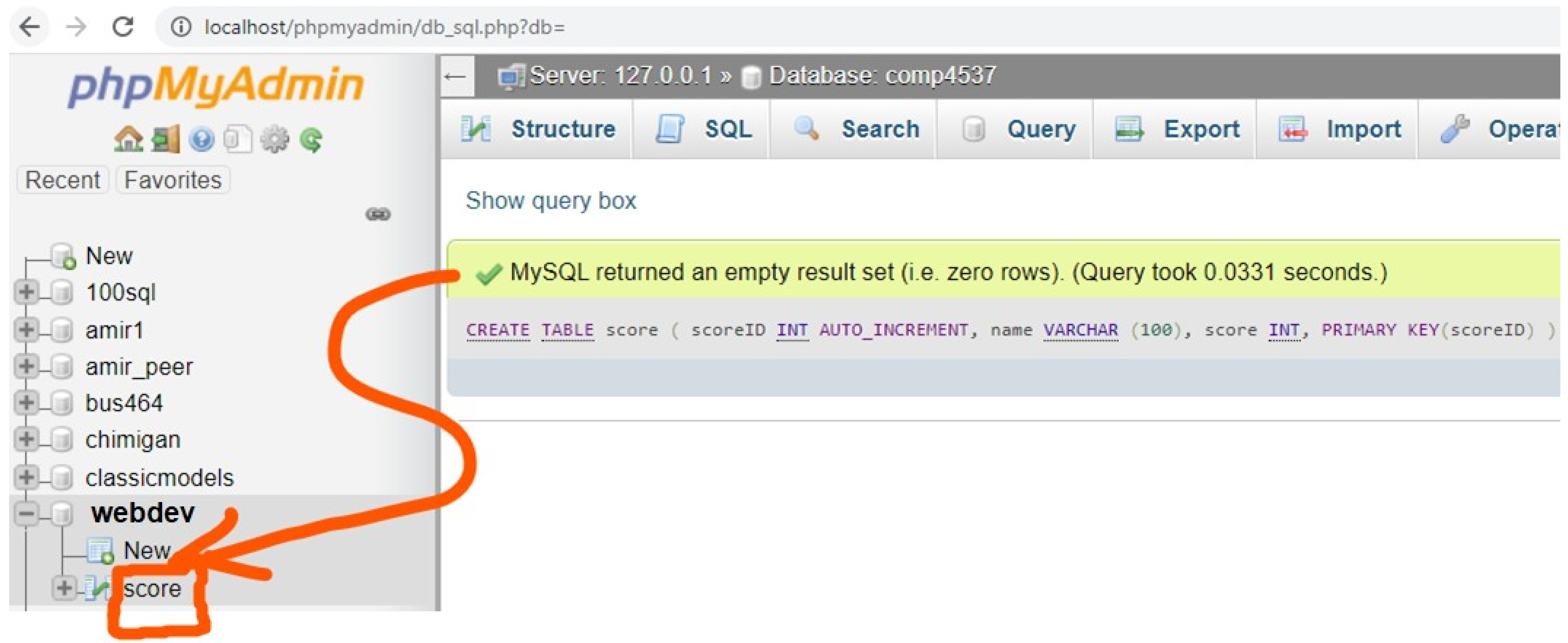
- 注意：列名与表名相同是允许的，两者均可命名为

  score
- 现在，我们的数据库已准备就绪！



```
CREATE TABLE score (
 scoreID INT AUTO_INCREMENT,
 name VARCHAR (100),
 score INT,
 PRIMARY KEY(scoreID)
);
```

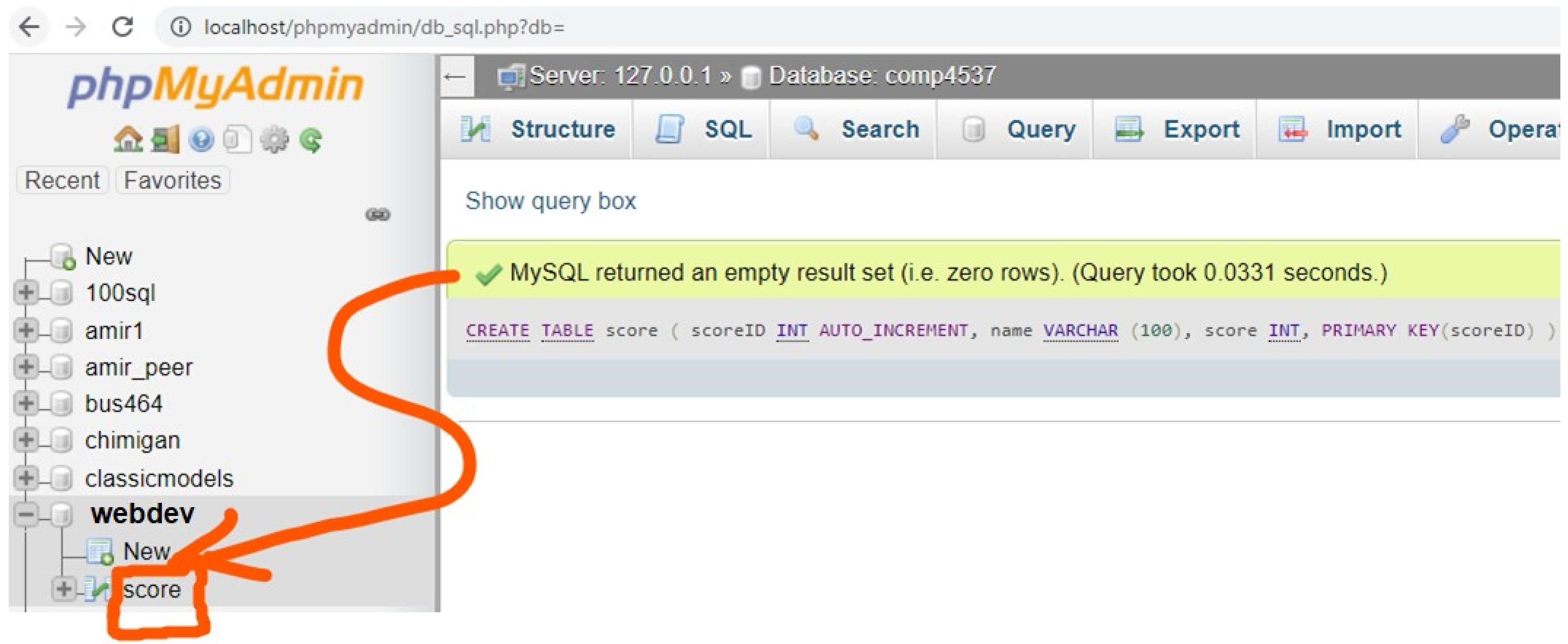
## 5- notice the confirmation message on creation of table



✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0331 seconds.)

```
CREATE TABLE score ( scoreID INT AUTO_INCREMENT, name VARCHAR (100), score INT, PRIMARY KEY(scoreID) )
```

## 5 — 注意创建表时显示的确认消息


✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0331 seconds.)

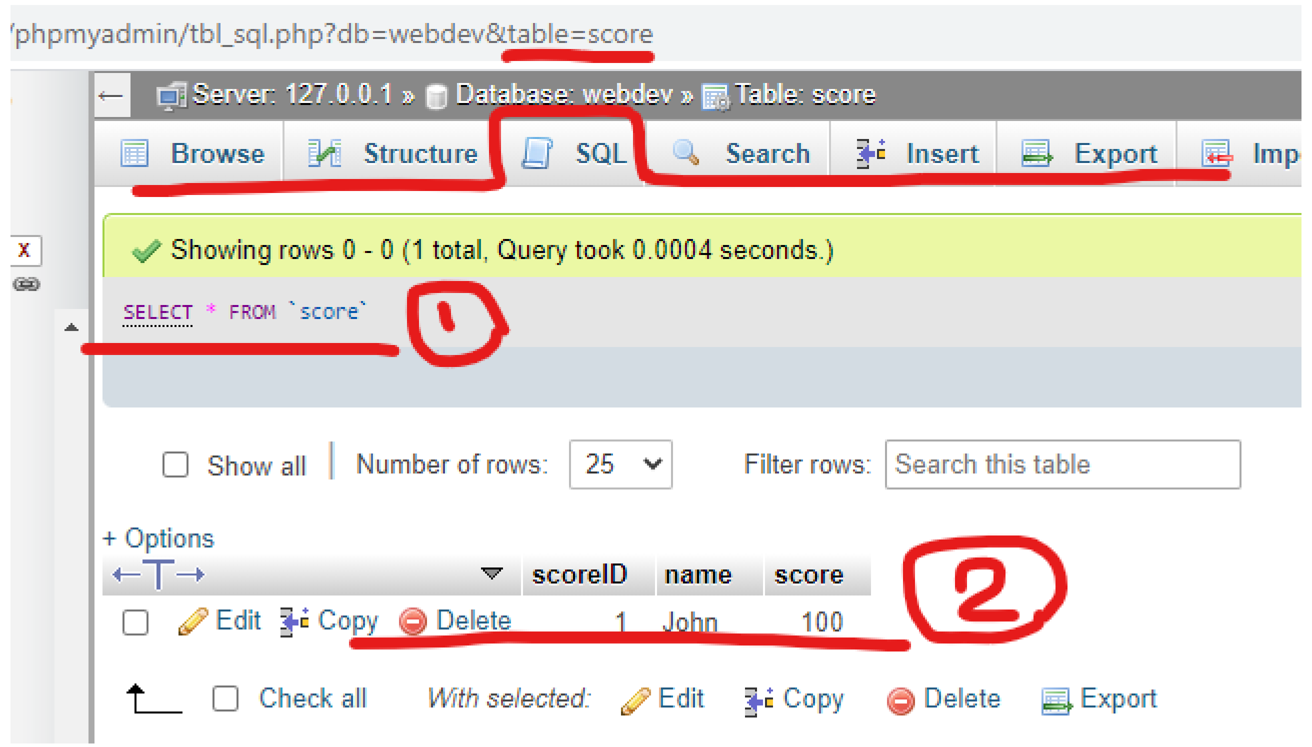
CREATE TABLE score ( scoreID INT AUTO_INCREMENT, name VARCHAR (100), score INT, PRIMARY KEY(scoreID) )

## 6- run few SQL statements on table you have created

0- make sure you have selected the DB you have created to run the SQL query on that

1- type down your query at SQL tab

2- press GO button and the result of your query appears!



phpmyadmin/tbl_sql.php?db=webdev&table=score

Server: 127.0.0.1 » Database: webdev » Table: score

Browse Structure SQL Search Insert Export Imp

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT * FROM `score`
```

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

| | scoreID | name | score |
|---|---|---|---|
| Edit Copy Delete | 1 | John | 100 |

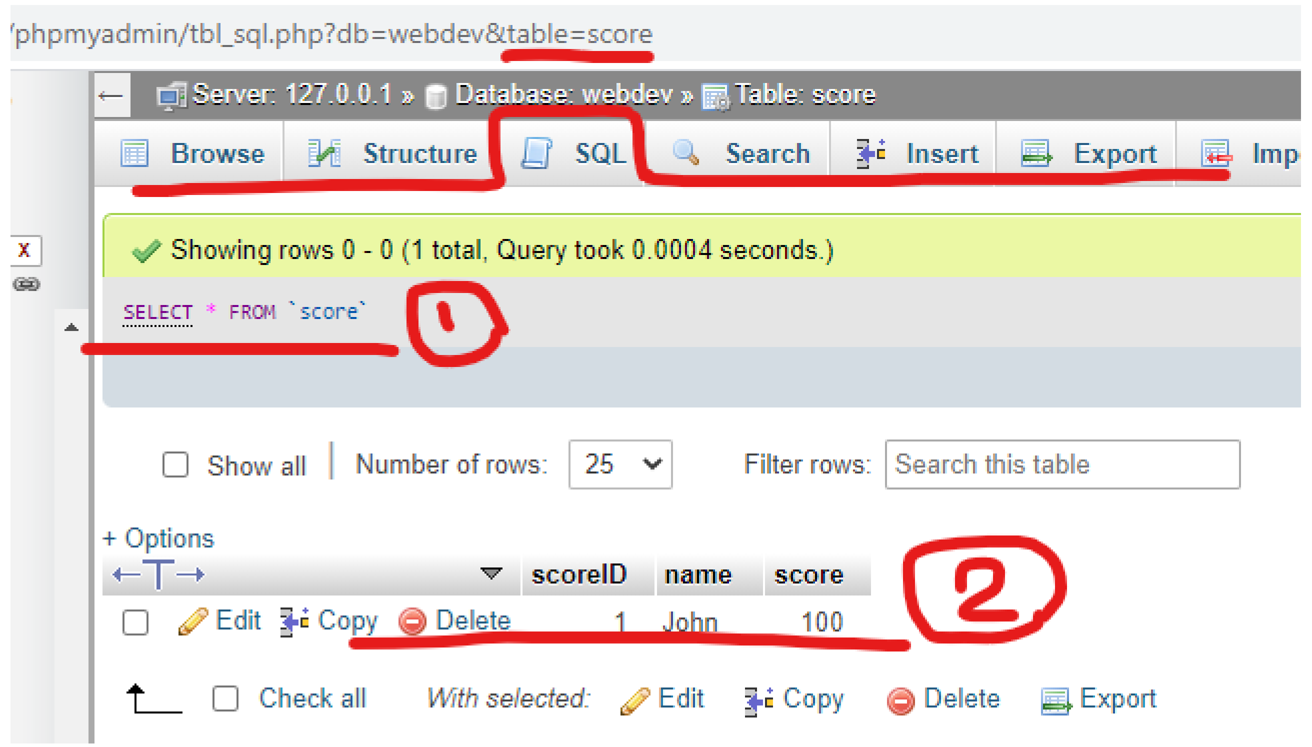
Check all With selected: Edit Copy Delete Export

## 6- 在您创建的表上执行若干 SQL 语句

0- 确保已选中您所创建的数据库，以便在该数据库上执行 SQL 查询

1- 输入您的查询语句（SQL）tab

2- 点击“执行”按钮，查询结果即会显示！



phpmyadmin/tbl_sql.php?db=webdev&table=score

Server: 127.0.0.1 » Database: webdev » Table: score

Browse Structure SQL Search Insert Export Imp

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

SELECT * FROM `score`

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

scoreID name score

Edit Copy Delete 1 John 100

Check all With selected: Edit Copy Delete Export

6

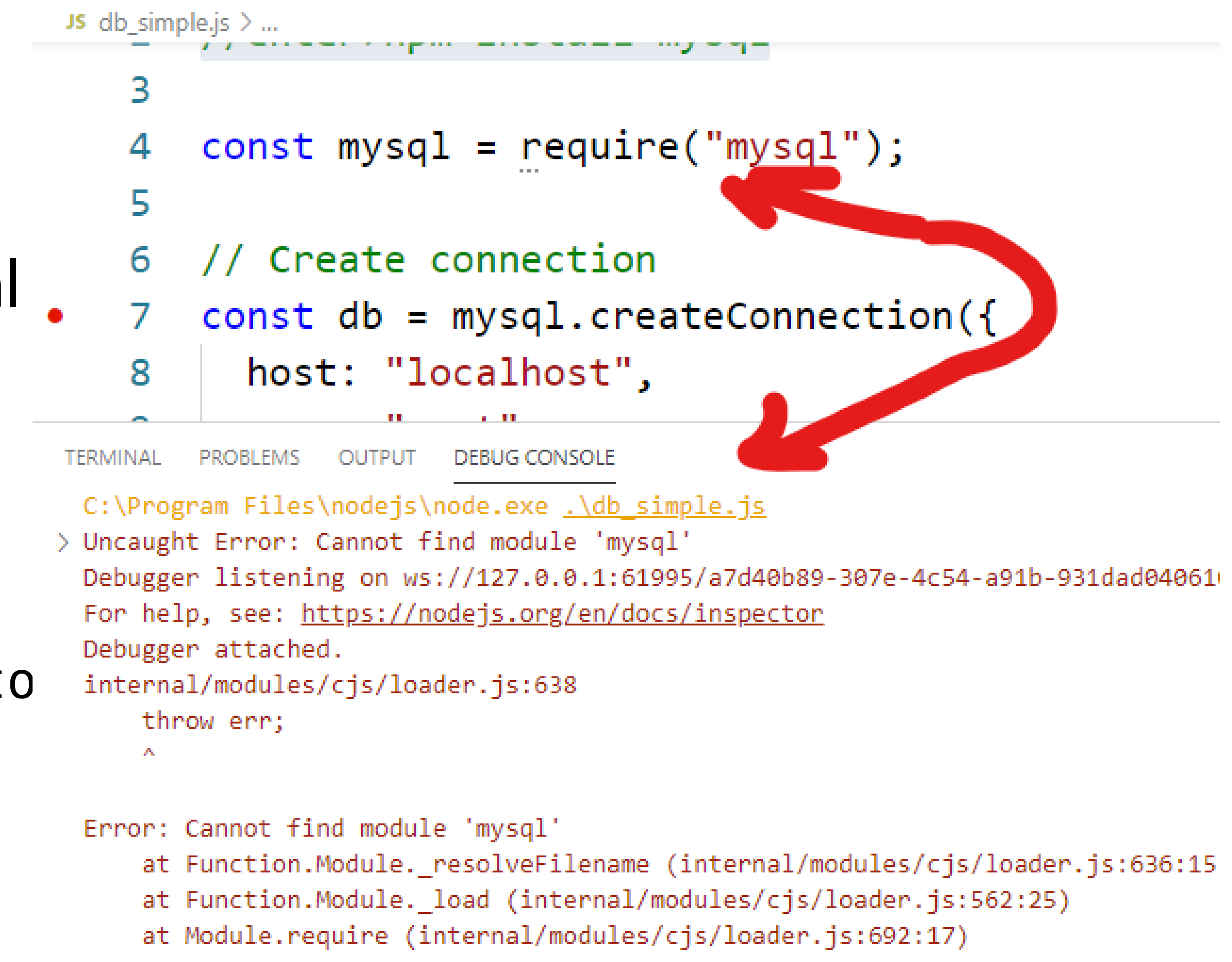
## Connecting to mySQL DB using nodejs

6

## 使用 Node.js 连接到 MySQL 数据库

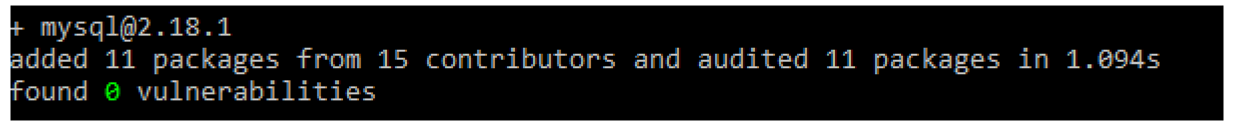
## 7. Connect with DB using node js

- Turns out, mysql is an external module and needs to be installed
- Here is how to
- 1 at command prompt switch to the current directory of this file
- 2 enter
  >npm install mysql



```
db_simple.js > ...

const mysql = require("mysql");

// Create connection
const db = mysql.createConnection({
  host: "localhost",
```

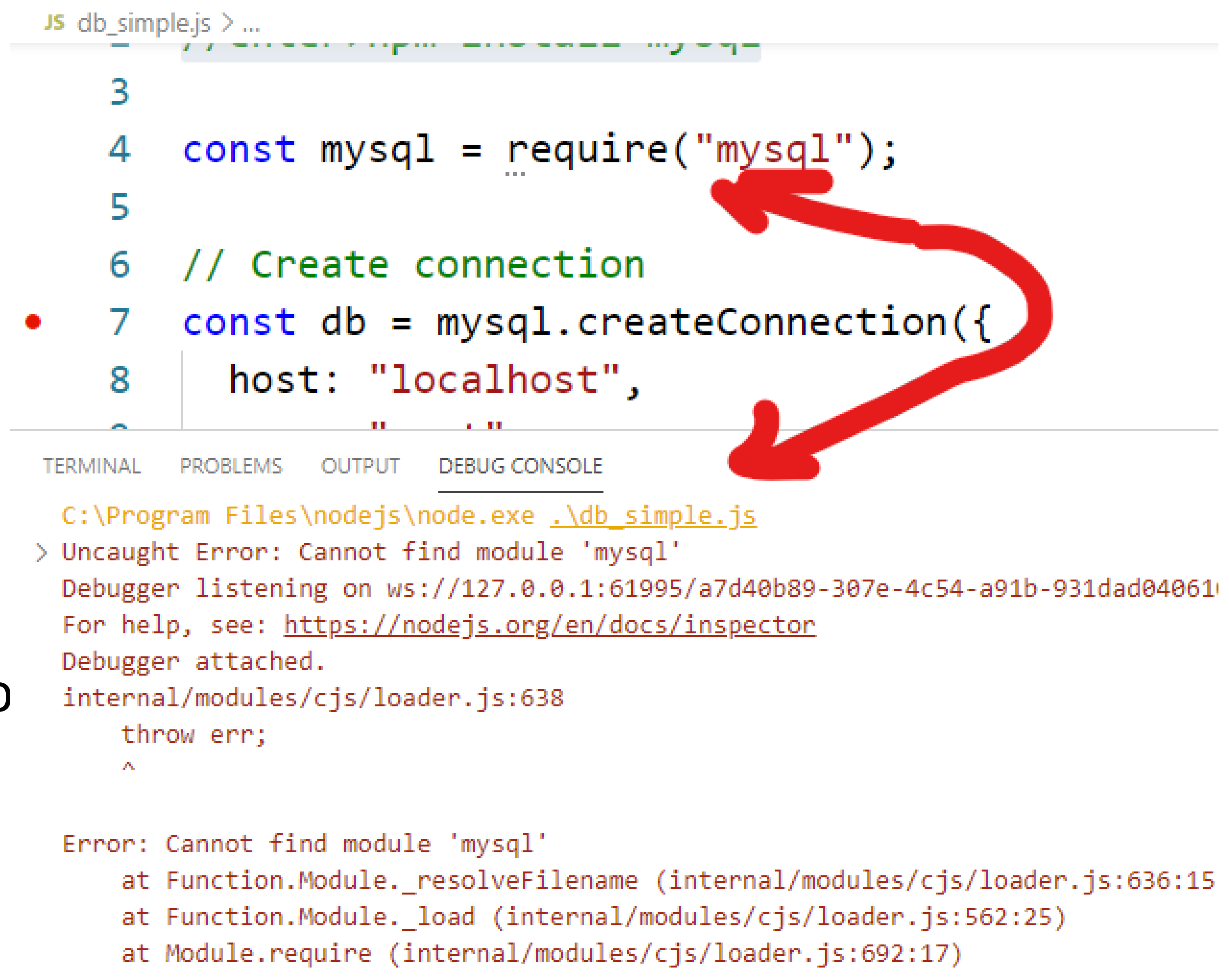
```
C:\Program Files\nodejs\node.exe .\db_simple.js
Uncaught Error: Cannot find module 'mysql'
Debugger listening on ws://127.0.0.1:61995/a7d40b89-307e-4c54-a91b-931dad04061
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
internal/modules/cjs/loader.js:638
    throw err;
    ^

Error: Cannot find module 'mysql'
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:636:15
    at Function.Module._load (internal/modules/cjs/loader.js:562:25)
    at Module.require (internal/modules/cjs/loader.js:692:17)
```



```
+ mysql@2.18.1
added 11 packages from 15 contributors and audited 11 packages in 1.094s
found 0 vulnerabilities
```

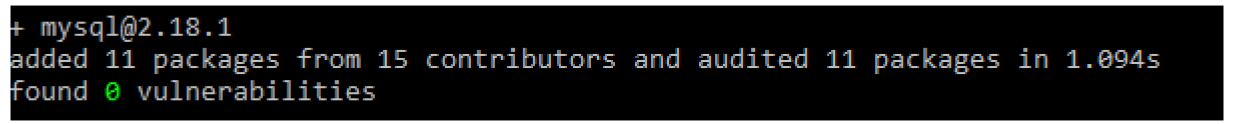
## 7. 使用 Node.js 连接数据库

- 事实证明，mysql 是一个外部模块，需要单独安装
- 具体操作方法如下
- 1 at command prompt switch to the current directory of this file
- 2 enter
  >npm install mysql



```
db_simple.js > ...

const mysql = require("mysql");

// Create connection
const db = mysql.createConnection({
  host: "localhost",
```

```
C:\Program Files\nodejs\node.exe .\db_simple.js
Uncaught Error: Cannot find module 'mysql'
Debugger listening on ws://127.0.0.1:61995/a7d40b89-307e-4c54-a91b-931dad04061
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
internal/modules/cjs/loader.js:638
    throw err;
    ^

Error: Cannot find module 'mysql'
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:636:15
    at Function.Module._load (internal/modules/cjs/loader.js:562:25)
    at Module.require (internal/modules/cjs/loader.js:692:17)
```



```
+ mysql@2.18.1
added 11 packages from 15 contributors and audited 11 packages in 1.094s
found 0 vulnerabilities
```

- First create a myslq object

- Then connect
- And run a SQL statement

```
// at command prompt switch to the current directory of this file
//enter>npm install mysql
const mysql = require("mysql");
// Create connection
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "webdev"
});
// Connect to MySQL to run SQL query
con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    let sql = "INSERT INTO score(name, score) values ('elon musk', 2900)
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
    });
  });
```

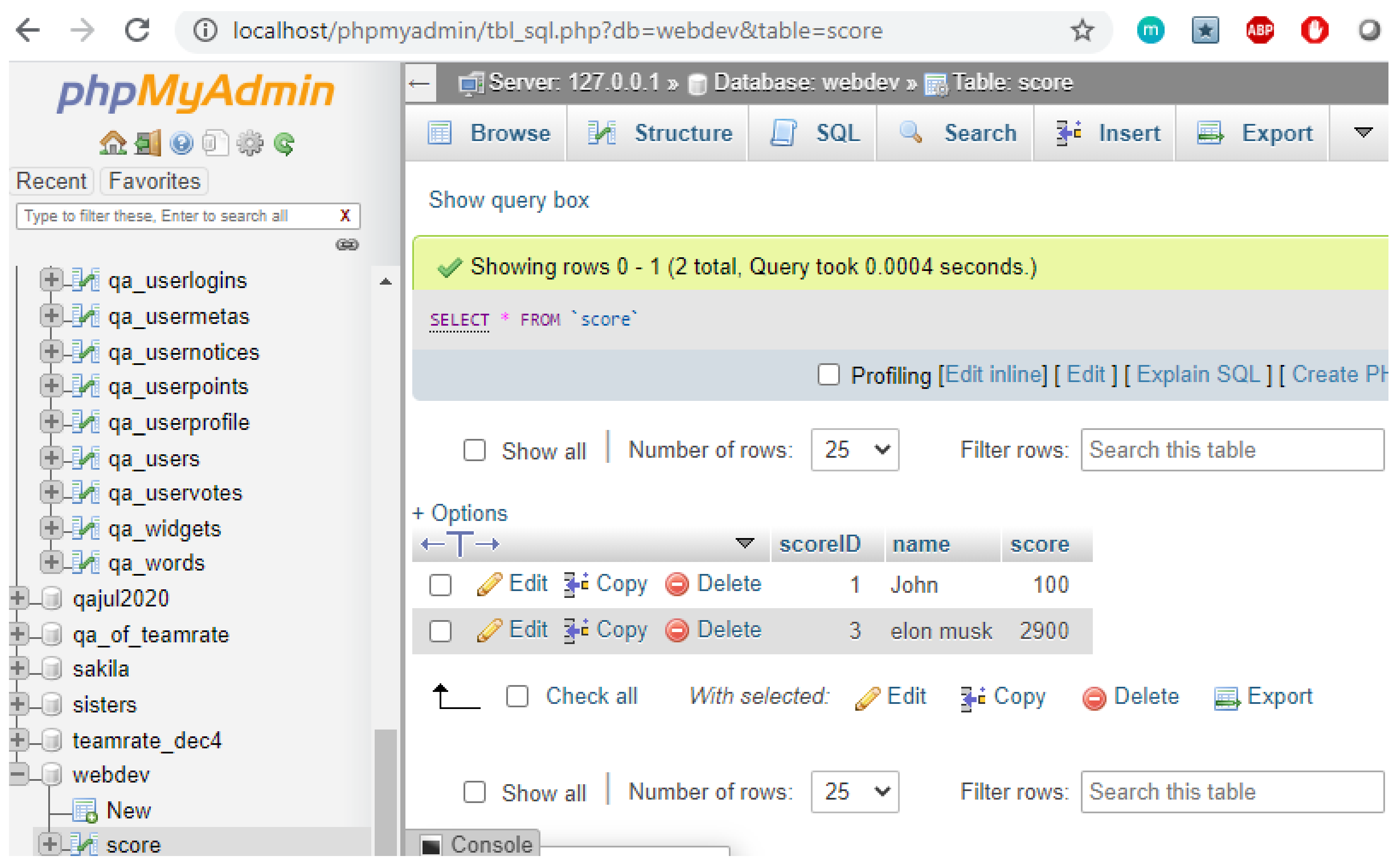
- 首先创建一个 MySQL 对象

- 然后建立连接
- 并执行一条 SQL 语句

```
// at command prompt switch to the current directory of this file
//enter>npm install mysql
const mysql = require("mysql");
// Create connection
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "webdev"
});
// Connect to MySQL to run SQL query
con.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    let sql = "INSERT INTO score(name, score) values ('elon musk', 2900)
    con.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
    });
  });
```

## Check your table to verify the SQL ran successfully on node

- As you can see
- The score of
- elon musk was entered in our table
- Successfully!



localhost/phpmyadmin/tbl_sql.php?db=webdev&table=score

phpMyAdmin

Recent Favorites

Type to filter these, Enter to search all

qa_userlogins
qa_usermetas
qa_usernotices
qa_userpoints
qa_userprofile
qa_users
qa_uservotes
qa_widgets
qa_words
qajul2020
qa_of_teamrate
sakila
sisters
teamrate_dec4
webdev
New
score

Server: 127.0.0.1 » Database: webdev » Table: score

Browse Structure SQL Search Insert Export

Show query box

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT * FROM `score`
```

Profiling [Edit inline] [ Edit ] [ Explain SQL ] [ Create PH

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

| | scoreID | name | score |
|---|---|---|---|
| Edit Copy Delete | 1 | John | 100 |
| Edit Copy Delete | 3 | elon musk | 2900 |

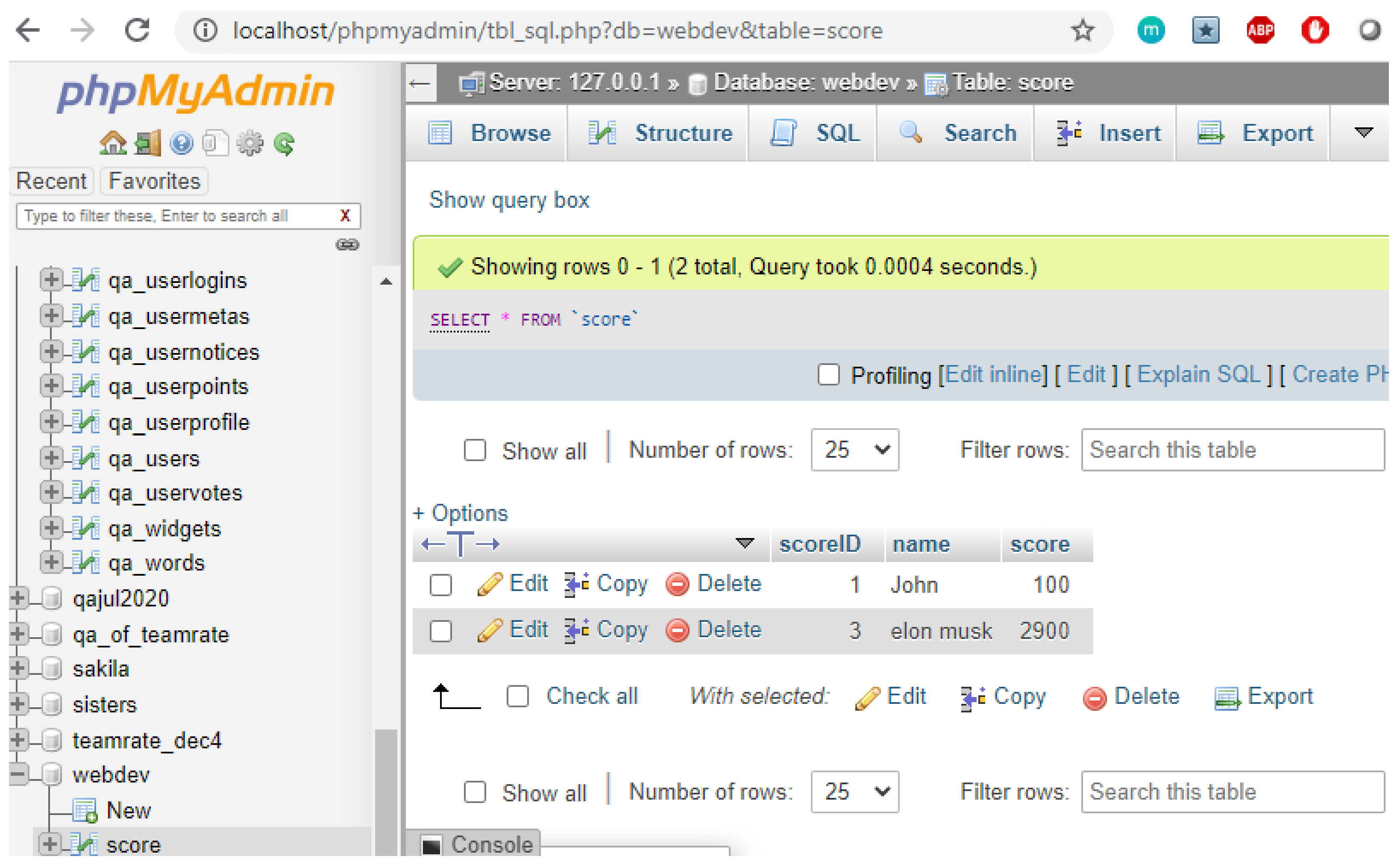
Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Console

## 请检查您的数据表，以确认 SQL 已在节点上成功执行。

- 如您所见
- 分数为
- 埃隆·马斯克为已录入我们的数据表中
- 成功!



localhost/phpmyadmin/tbl_sql.php?db=webdev&table=score

phpMyAdmin

Recent Favorites

Type to filter these, Enter to search all

qa_userlogins
qa_usermetas
qa_usernotices
qa_userpoints
qa_userprofile
qa_users
qa_uservotes
qa_widgets
qa_words
qajul2020
qa_of_teamrate
sakila
sisters
teamrate_dec4
webdev
New
score

Server: 127.0.0.1 » Database: webdev » Table: score

Browse Structure SQL Search Insert Export

Show query box

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

SELECT * FROM `score`

Profiling [Edit inline] [ Edit ] [ Explain SQL ] [ Create PH

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

scoreID name score
Edit Copy Delete 1 John 100
Edit Copy Delete 3 elon musk 2900

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Console

## Hosting Node js, MySQL on remote server

7

Please watch this video
**Hosting nodejs MySql remotely CPanel Shared Hosting.mp4**

## 在远程服务器上托管 Node.js 和 MySQL

7

请观看此视频
**远程托管 Node.js 与 MySQL（cPanel 共享主机）.mp4**

Beginner 'sguid

# Node js + MySQL

cPanel

remote

server

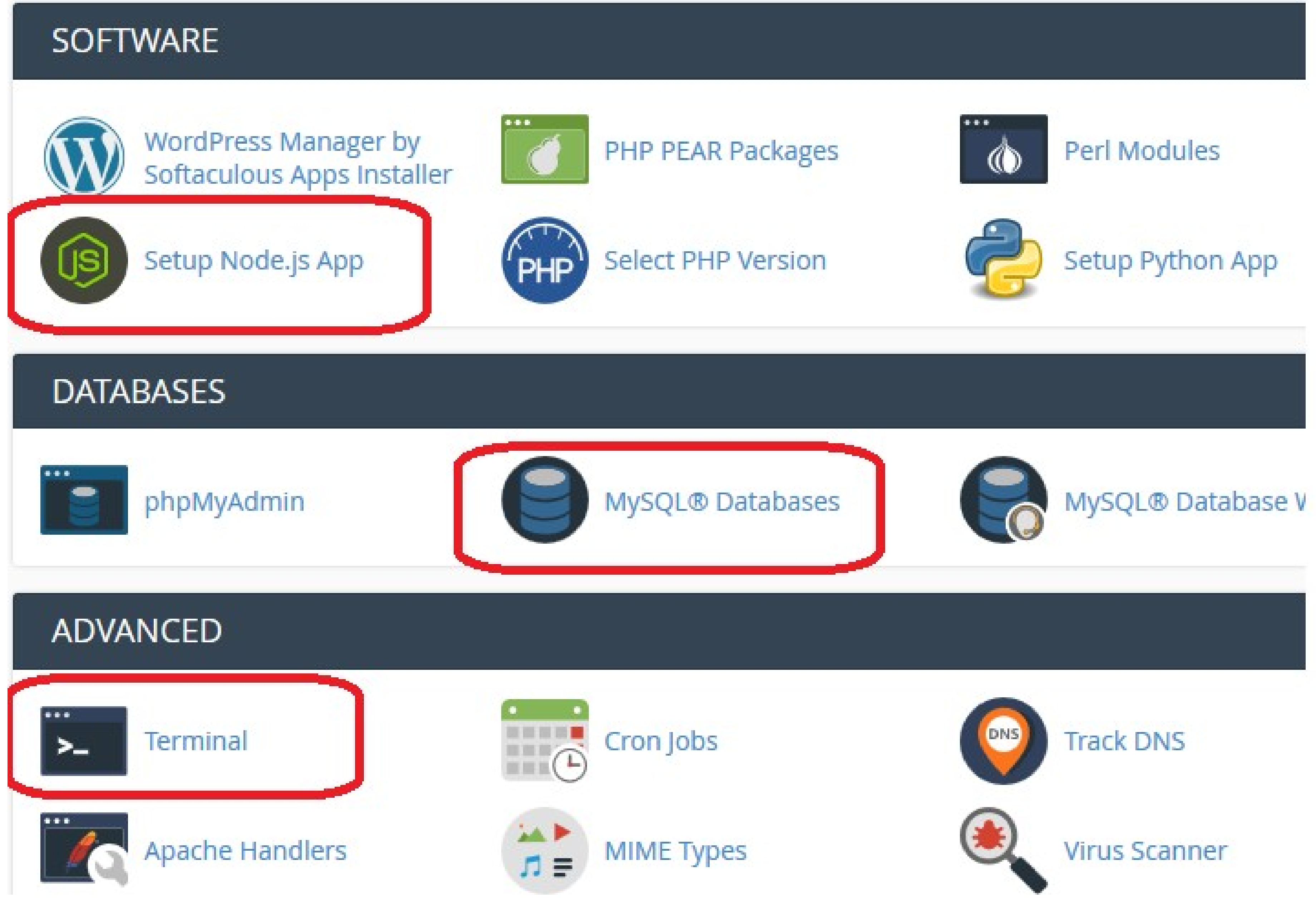
# Node.js + MySQL

cPanel

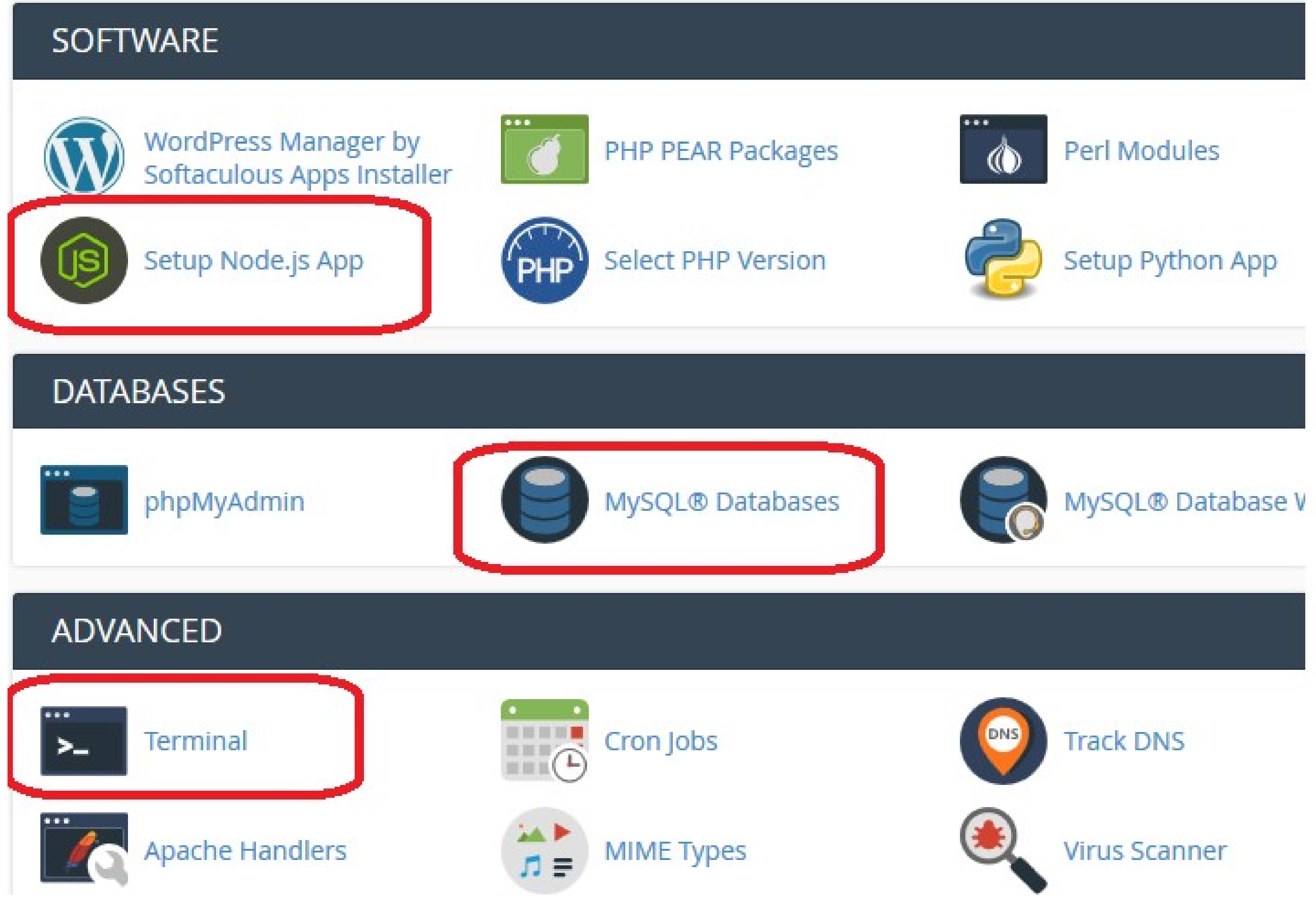
远程

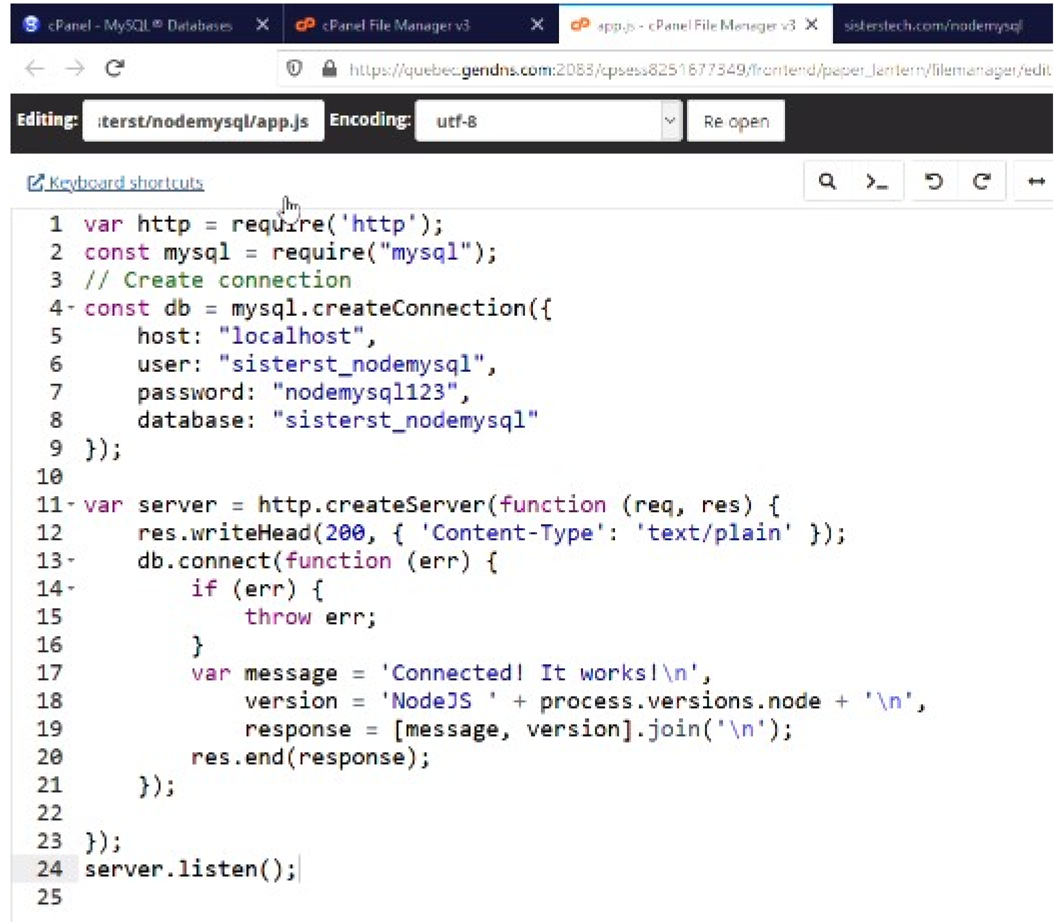
服务器

## How to host nodejs MySQL in share hosting accounts

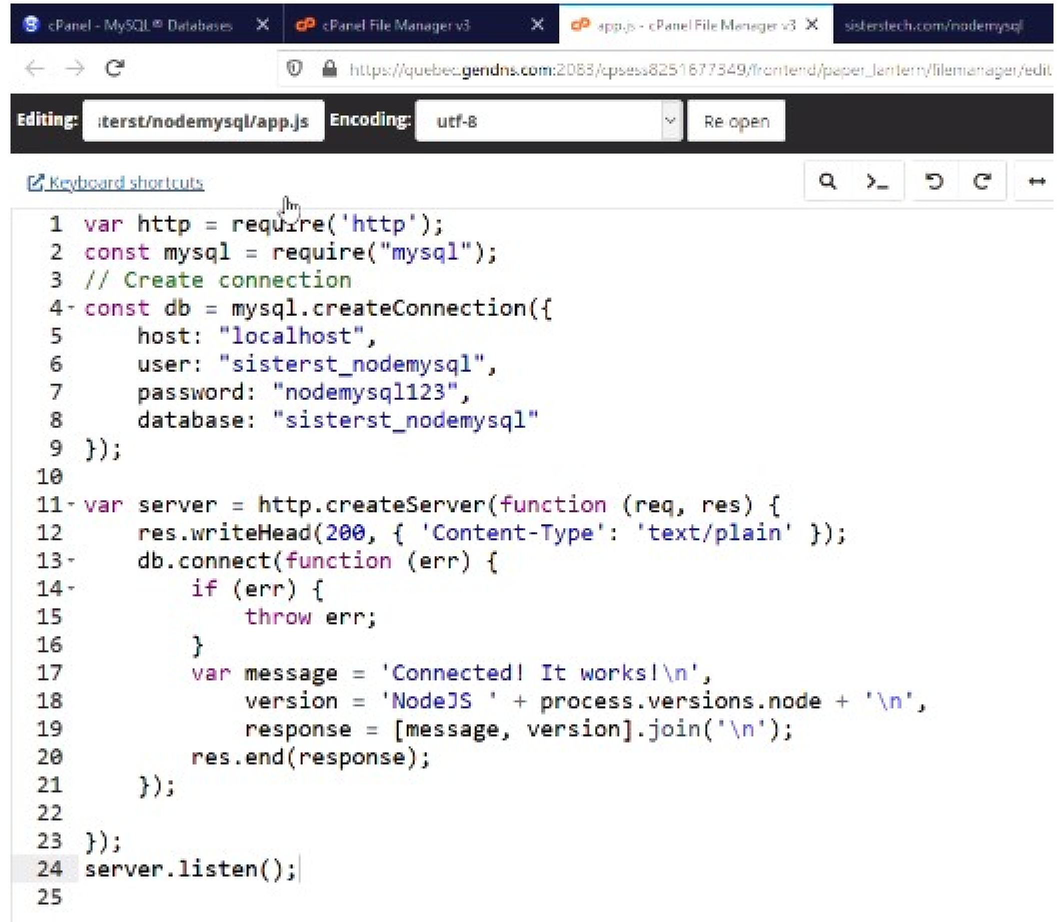


## 如何在共享主机账户中托管 Node.js 和 MySQL





```
var http = require('http');
const mysql = require("mysql");
// Create connection
const db = mysql.createConnection({
    host: "localhost",
    user: "sisterst_nodemysql",
    password: "nodemysql123",
    database: "sisterst_nodemysql"
});

var server = http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    db.connect(function (err) {
        if (err) {
            throw err;
        }
        var message = 'Connected! It works!\n',
            version = 'NodeJS ' + process.versions.node + '\n',
            response = [message, version].join('\n');
        res.end(response);
    });

});
server.listen();
```



```
var http = require('http');
const mysql = require("mysql");
// Create connection
const db = mysql.createConnection({
    host: "localhost",
    user: "sisterst_nodemysql",
    password: "nodemysql123",
    database: "sisterst_nodemysql"
});

var server = http.createServer(function (req, res) {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    db.connect(function (err) {
        if (err) {
            throw err;
        }
        var message = 'Connected! It works!\n',
            version = 'NodeJS ' + process.versions.node + '\n',
            response = [message, version].join('\n');
        res.end(response);
    });

});
server.listen();
```

49

## source

- chatGPT 3.5 used to generate partial code for this set

## 源代码

- ChatGPT 3.5 曾用于为此套代码生成部分代码