Node.js 入门

JavaScript 运行时环境

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

目录

Contents

什么是 Node.js?

定义：

> **Node.js**
>
> Node.js 是一个跨平台 JavaScript 运行环境，使开发者可以搭建服务器端的 JavaScript 应用程序。

概念：使用 Node.js 编写后端程序 / 支持前端工程化

- ✓ 后端程序：提供接口和数据，网页资源等
- ✓ 前端工程化：对代码压缩，转译，整合（使用各种工具，提升效率）

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>时钟案例</title>
  <style>…
  </style>
</head>
<body>
  <div class="box">
    <div id="HH">00</div>
    <div>:</div>
    <div id="mm">00</div>
    <div>:</div>
    <div id="ss">00</div>
  </div>
</body>
</html>
```

```
1  <!DOCTYPE html><html lang="en"><head><meta charset="UTF-8"><meta name="viewport"
   content="width=device-width, initial-scale=1.0"><meta http-equiv="X-UA-Compatible"
   content="ie=edge"><title>时钟案例</title><style>html,body{margin:0;padding:0;height:100%;
   background-image:linear-gradient(to bottom right,red,gold);}.box{width: 400px;height:250px;
   background-color:rgba(255,255,255,0.6);border-radius:6px;position:absolute;left:50%;top:40%;
   transform:translate(-50%,-50%);box-shadow:1px 1px 10px #fff;text-shadow:0px 1px 30px white;
   display:flex;justify-content:space-around;align-items:center;font-size:70px;user-select:none;
   padding:0 20px;-webkit-box-reflect:below 0px -webkit-gradient(linear,left top,left bottom,from
   (transparent),color-stop(0%,transparent),to(rgba(250,250,250,.2)));}</style></head><body><div
   class="box"><div id="HH">00</div><div>:</div><div id="mm">00</div><div>:</div><div id="ss">00</
   div></div></body></html><script>window.addEventListener('load',function(){clock();setInterval
   (clock,1000);});function clock(){let dt=new Date();let HH=dt.getHours();let mm=dt.getMinutes();
   let ss=dt.getSeconds();document.querySelector('#HH').innerHTML=padZero(HH);document.querySelector
   ('#mm').innerHTML=padZero(mm);document.querySelector('#ss').innerHTML=padZero(ss);};function
   padZero(n){return n>9?n:'0'+n;}</script>   5.3333rem, 3.3333rem, 0.08rem, 0.0133rem, 0.0133rem, 0.
```

Node.js 为何能执行 JS?

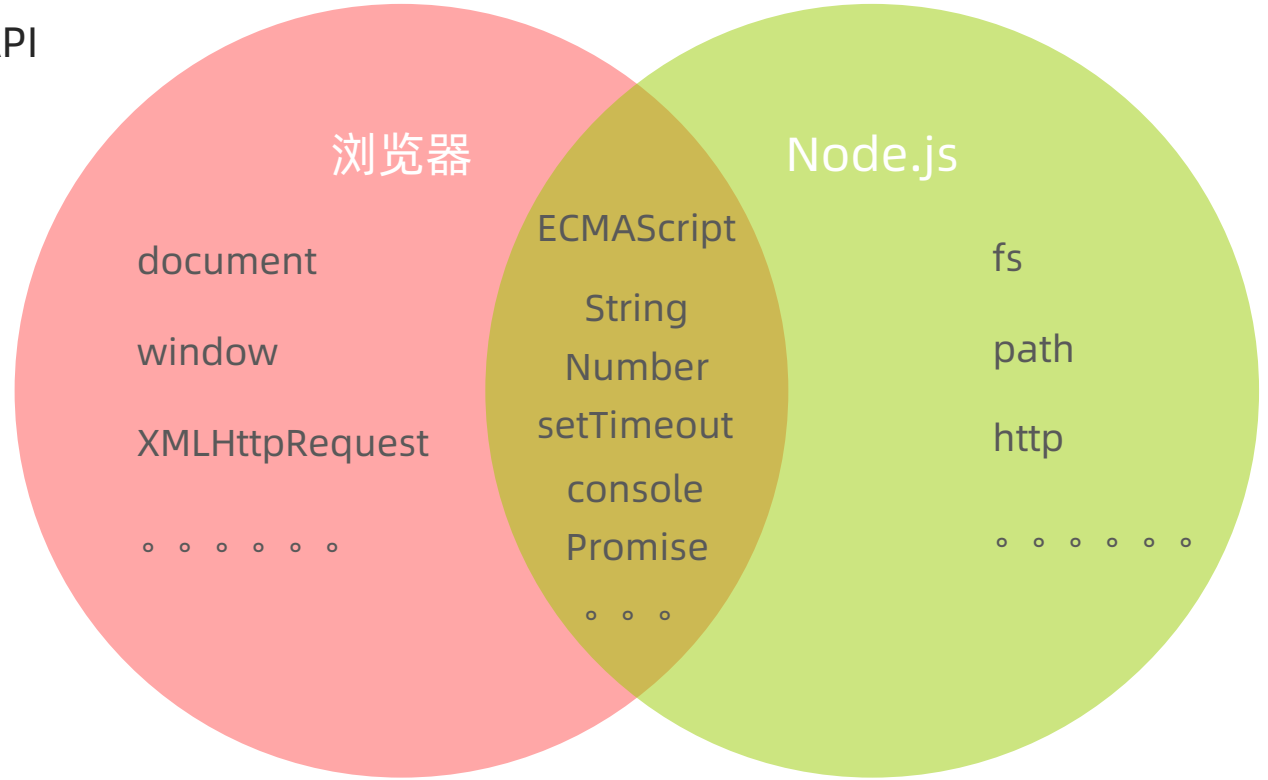
首先：浏览器能执行 JS 代码，依靠的是内核中的 V8 引擎（C++ 程序）

其次：Node.js 是基于 Chrome V8 引擎进行封装（运行环境）

区别：都支持 ECMAScript 标准语法，Node.js 有独立的 API

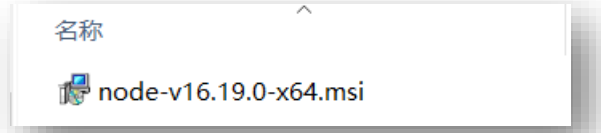
浏览器

Node.js



注意：Node.js 环境没有 DOM 和 BOM 等

Node.js 安装

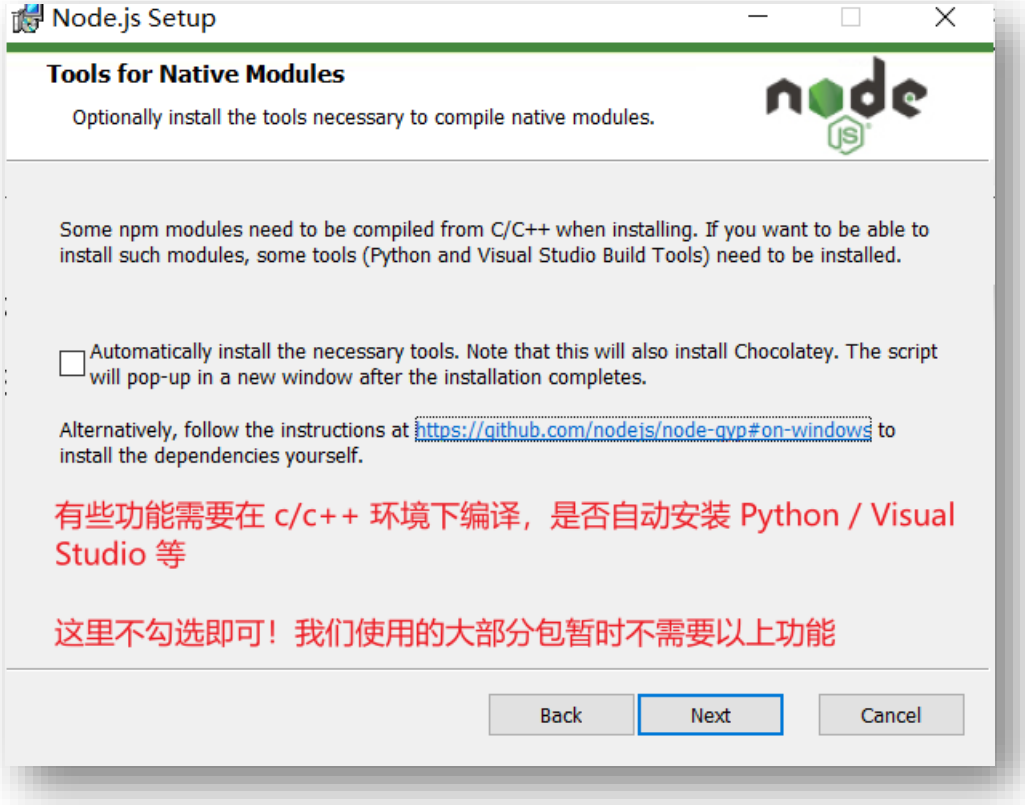


要求：下载 node-v16.19.0.msi 安装程序（指定版本：为了兼容后期学习的项目）

安装过程：默认下一步即可

注释事项：

1. 安装在非中文路径下
2. 无需勾选自动安装其他配套软件



成功验证：

1. 打开 cmd 终端，输入 node -v 命令查看版本号
2. 如果有显示，则代表安装成功

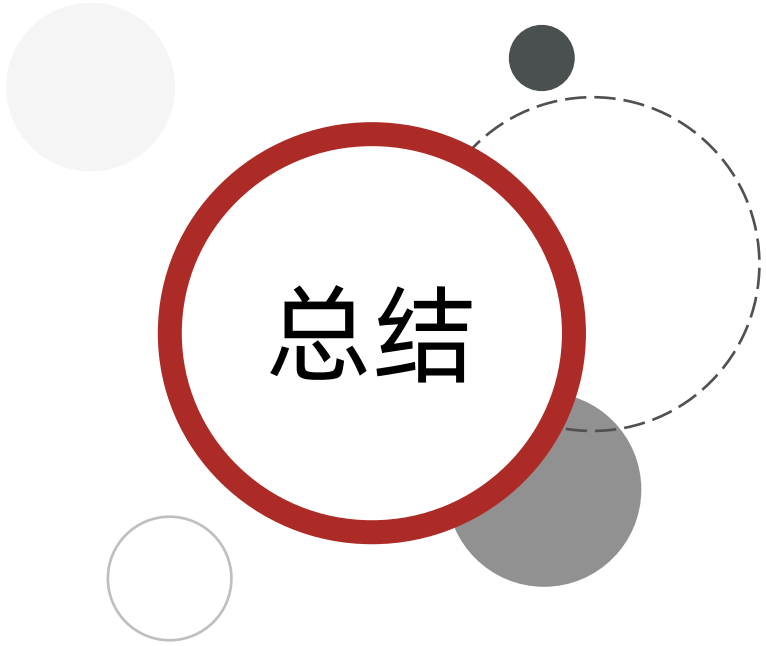
使用 Node.js

需求：新建 JS 文件，并编写代码后，在 node 环境下执行

命令：在 VSCode 集成终端中，输入 node xxx.js，回车即可执行

```
console.log('Hello, World')
for (let i = 0; i < 3; i++) {
  console.log(6)
}
```

```
问题  输出  终端  调试控制台

D:\备课代码\2_node_3天\Node_代码\Day01_Node.js入门\代码>node 01.js
Hello, World
6
6
6

D:\备课代码\2_node_3天\Node_代码\Day01_Node.js入门\代码>
```



1. Node.js 有什么用？
   - 编写后端程序：提供数据和网页资源等
   - 前端工程化：翻译压缩整合代码等，提高开发效率
2. Node.js 为何能执行 JS 代码？
   - 基于 Chrome 的 V8 引擎封装
3. Node.js 与浏览器环境的 JS 最大区别？
   - Node.js 环境中没有 BOM 和 DOM，但也是用 JS 语法
4. Node.js 如何执行代码？
   - 在 VSCode 终端中输入：node xxx.js 回车即可执行（注意路径）

fs 模块 - 读写文件

模块：类似插件，封装了方法/属性

fs 模块：封装了与本机文件系统进行交互的，方法/属性

语法：

1. 加载 fs 模块
2. 写入文件内容
3. 读取文件内容

```
const fs = require('fs') // fs 是模块标识符：模块的名字
```

```
fs.writeFile('文件路径', '写入内容', err => {
  // 写入后的回调函数
})
```

```
fs.readFile('文件路径', (err, data) => {
  // 读取后的回调函数
  // data 是文件内容的 Buffer 数据流
})
```

path 模块 - 路径处理

问题：Node.js 代码中，相对路径是根据终端所在路径来查找的，可能无法找到你想要的文件

path 模块 - 路径处理

建议：在 Node.js 代码中，使用绝对路径

补充：__dirname 模块内置变量（获取当前模块目录名）

- ✓ windows：D:\备课代码\2_node_3天\Node_代码\Day01_Node.js入门\代码\03
- ✓ mac：/Users/xxx/Desktop/备课代码/2_node_3天/Node代码/Day01_Node.js入门/代码/03

注意：path.join() 会使用特定于平台的分隔符，作为定界符，将所有给定的路径片段连接在一起

语法：

1. 加载 path 模块
2. 使用 path.join 方法，拼接路径

```
const path = require('path')
```

```
path.join('路径1', '路径2', ...)
```

案例 - 压缩前端 html

目标：压缩前端代码，让浏览器加载网页更快（体验前端工程化）

前端工程化：对代码压缩，转译，整合，测试，自动部署等等

需求：把 回车符（\r）和换行符（\n）去掉进行压缩，写入到新 html 中

步骤：

1. 读取源 html 文件内容
2. 正则替换字符串
3. 写入到新的 html 文件中

04 > public > index.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>时钟案例</title>
  <style>
    html,
    body {…
    }
    .box {…
    }
  </style>
</head>
<body>
  <div class="box">
    <div id="HH">00</div>
    <div>:</div>
    <div id="mm">00</div>
    <div>:</div>
    <div id="ss">00</div>
  </div>
</body>
</html>
```

04 > dist > index.html > html

```
<!DOCTYPE html><html lang="en"><head>  <meta charset="UTF-8">  <meta name="viewport"
content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible"
content="ie=edge">  <title>时钟案例</title>  <style>    html,    body {      margin: 0;
padding: 0;      height: 100%;      background-image: linear-gradient(to bottom right, red,
gold);    }    .box {      width: 400px;      height: 250px;      background-color: rgba(255,
255, 255, 0.6);      border-radius: 6px;      position: absolute;      left: 50%;      top:
40%;      transform: translate(-50%, -50%);      box-shadow: 1px 1px 10px #fff;      text-shadow:
0px 1px 30px white;      display: flex;      justify-content: space-around;      align-items:
center;      font-size: 70px;      user-select: none;      padding: 0 20px;
-webkit-box-reflect: below 0px -webkit-gradient(linear, left top, left bottom, from(transparent),
color-stop(0%, transparent), to(rgba(250, 250, 250, .2)));    }  </style></head><body>  <div
class="box">    <div id="HH">00</div>    <div>:</div>    <div id="mm">00</div>    <div>:</div>
<div id="ss">00</div>  </div></body></html>  5.3333rem, 3.3333rem, 0.08rem, 0.0133rem, 0.0133rem, 0.
```

案例 - 压缩前端 JS

目标：压缩和整合前端 JS 代码，写入新到 html 中

步骤：

1. 读取 js 文件内容
2. 正则替换换行，回车字符串和打印语句
3. 拼接 html 内容和 js 代码，写入到新的 html 文件中

04 > dist > index.html > script

```
<!DOCTYPE html><html lang="en"><head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <meta http-equiv="X-UA-Compatible" content="ie=edge"> <title>时钟案例</title> <style> html, body { margin: 0; padding: 0; height: 100%; background-image: linear-gradient(to bottom right, red, gold); } .box { width: 400px; height: 250px; background-color: rgba(255, 255, 255, 0.6); border-radius: 6px; position: absolute; left: 50%; top: 40%; transform: translate(-50%, -50%); box-shadow: 1px 1px 10px #fff; text-shadow: 0px 1px 30px white; display: flex; justify-content: space-around; align-items: center; font-size: 70px; user-select: none; padding: 0 20px; -webkit-box-reflect: below 0px -webkit-gradient(linear, left top, left bottom, from(transparent), color-stop(0%, transparent), to(rgba(250, 250, 250, .2))); } </style></head><body> <div class="box"> <div id="HH">00</div> <div>:</div> <div id="mm">00</div> <div>:</div> <div id="ss">00</div> </div></body></html><script>window.addEventListener('load', function () { clock(); setInterval(clock, 1000);});function clock() { let dt = new Date(); let HH = dt.getHours(); let mm = dt.getMinutes(); let ss = dt.getSeconds(); document.querySelector('#HH').innerHTML = padZero(HH); document.querySelector('#mm').innerHTML = padZero(mm); document.querySelector('#ss').innerHTML = padZero(ss);};function padZero(n) { return n > 9 ? n : '0' + n;}
</script>  5.3333rem, 3.3333rem, 0.08rem, 0.0133rem, 0.0133rem, 0.1333rem, 0rem, 0.0133rem, 0.4rem,
```

04 > public > index.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>时钟案例</title>
  <style>
    html,
    body {…
    }
    .box {…
    }
  </style>
</head>
<body>
  <div class="box">
    <div id="HH">00</div>
    <div>:</div>
    <div id="mm">00</div>
    <div>:</div>
    <div id="ss">00</div>
  </div>
</body>
</html>
```

04 > public > JS index.js > ...

```
window.addEventListener('load', function () {
  console.log('测试-网页加载完成了');
  clock();
  setInterval(clock, 1000);
});
function clock() {
  console.log('测试-时钟函数调用了');
  let dt = new Date();
  let HH = dt.getHours();
  let mm = dt.getMinutes();
  let ss = dt.getSeconds();
  console.log(dt, HH, mm, ss);
  document.querySelector('#HH').innerHTML = padZero(HH);
  document.querySelector('#mm').innerHTML = padZero(mm);
  document.querySelector('#ss').innerHTML = padZero(ss);
};
function padZero(n) {
  console.log(n);
  return n > 9 ? n : '0' + n;
}

```

URL 中的端口号

URL：统一资源定位符，简称网址，用于访问网络上的资源

端口号：标记服务器里对应服务程序（0-65535 的整数）


钥匙


地址


房间号


物品位置

http://hmajax.itheima.net:80/api/province

协议　域名　端口号　资源路径

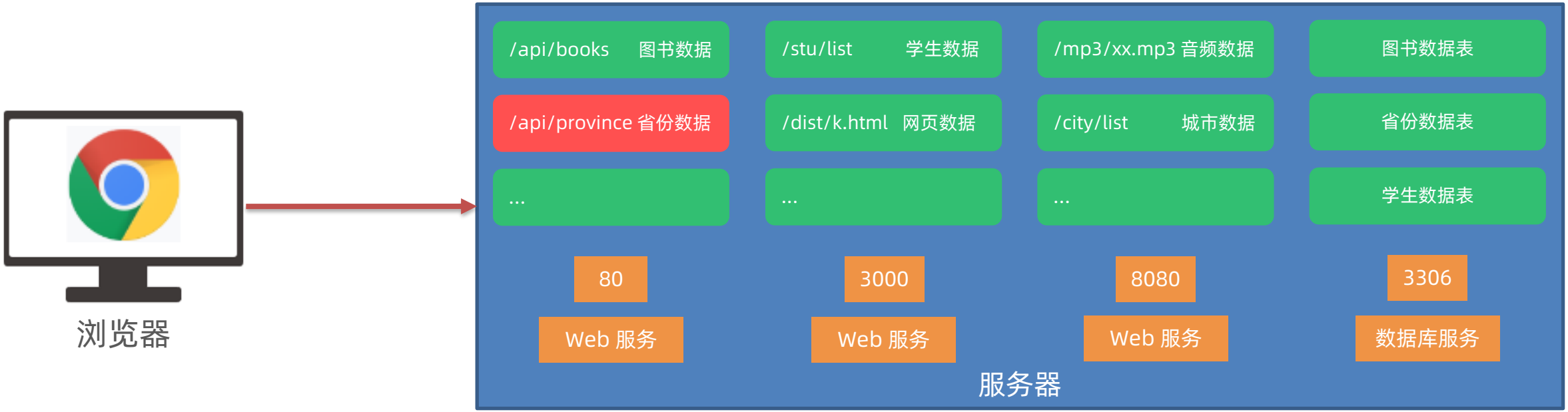
注意：http 协议，默认访问 80 端口

URL 中的端口号

端口号：标记服务器里对应服务程序

Web 服务：一个程序，用于提供网上信息浏览功能

注意：0-1023 和一些特定端口号被占用，我们自己编写服务程序请避开使用



总结

1. 端口号的作用？
   - 标记区分服务器里不同的服务程序
2. 什么是 Web 服务？
   - 提供网上信息浏览的程序

http 模块-创建 Web 服务

需求：基于 http 模块编写程序，返回给请求方 'hello，world'

步骤：

1. 引入 http 模块，创建 Web 服务对象
2. 监听 request 请求事件，对本次请求，做一些响应处理
3. 启动 Web 服务监听对应端口号
4. 运行本服务在终端，用浏览器发起请求

http://localhost:3000/

```
const http = require('http')
const server = http.createServer()

server.on('request', (req, res) => {
  res.end('hello, world')
})

server.listen(3000, () => {
  console.log('Web 服务已启动')
})
```

Web 服务-支持中文字符

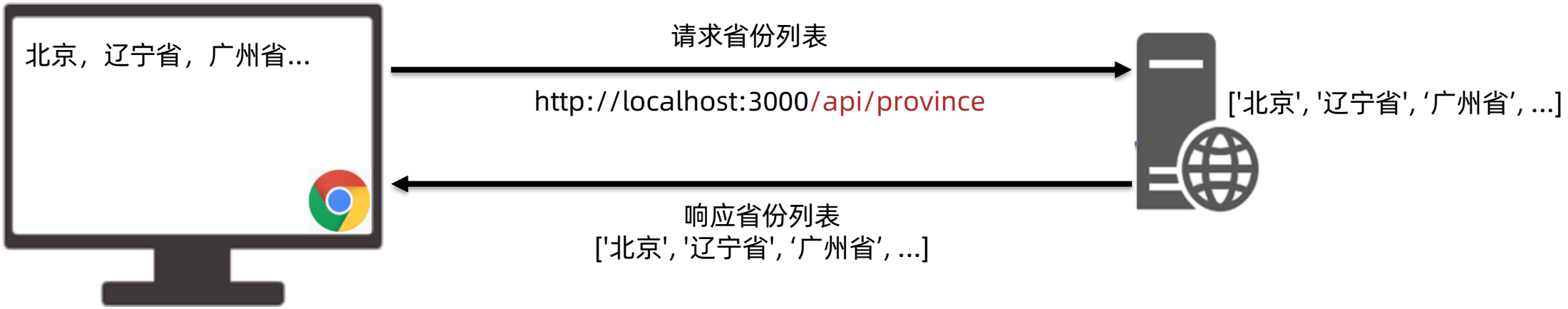
需求：让 Web 服务返回的中文字符，浏览器正常解析

步骤：给响应头添加内容类型，如截图：

```
const http = require('http')
const server = http.createServer()

server.on('request', (req, res) => {
  // 设置响应头，内容类型，普通 html 文本；编码格式为 utf-8
  res.setHeader('Content-Type', 'text/html;charset=utf-8')

  res.end('你好，亲爱的世界')
})

server.listen(3000, () => {
  console.log('Web 服务已启动')
})
```

案例 获取省份列表-接口开发

需求：基于 Web 服务，开发提供省份列表数据的接口，了解下后端的代码工作过程



案例 获取省份列表-接口开发

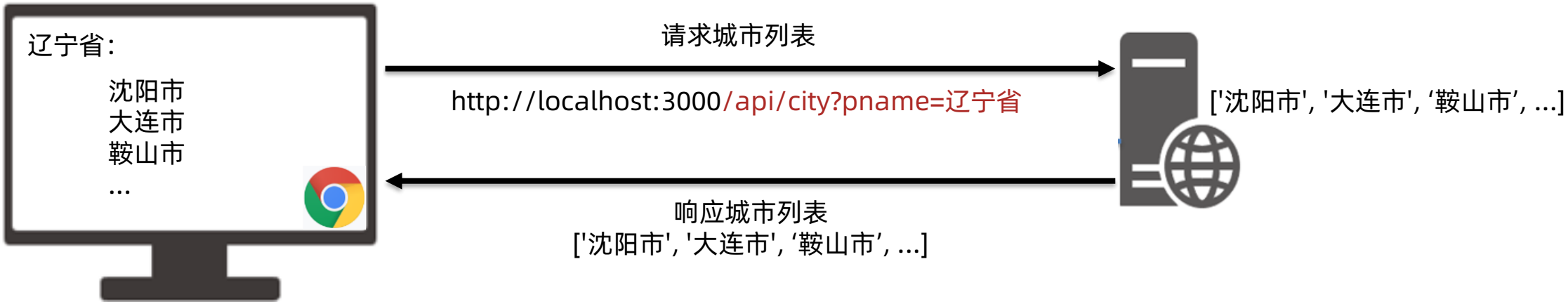
步骤：

1. 基于 http 模块，创建 Web 服务
2. 使用 req.url 获取请求资源路径，并读取 province.json 里省份数据返回给请求方
3. 其他路径，暂时返回不存在的提示
4. 运行 Web 服务，用浏览器发起请求

```
server.on('request', (req, res) => {
  if (req.url === '/api/province') {
    fs.readFile(path.join(__dirname, 'data/province.json'), (err, data) => {
      res.setHeader('Content-Type', 'application/json;charset=utf-8')
      res.end(data.toString())
    })
  } else {
    res.setHeader('Content-Type', 'text/html;charset=utf-8')
    res.end('你要访问的路径不存在')
  }
})
```

案例 获取城市列表-接口开发

需求：基于 Web 服务，开发提供城市列表数据的接口，了解下后端的代码工作过程



案例 获取城市列表-接口开发

前端请求：http://localhost:3000/api/city?pname=辽宁省

步骤：

1. 判断 req.url 资源路径+查询字符串，路径前缀匹配 /api/city
2. 借助 querystring 模块的方法，格式化查询字符串
3. 读取 city.json 城市数据，匹配省份名字下属城市列表
4. 返回城市列表，启动 Web 服务测试

```
const qs = require('querystring')
const query = qs.parse('参数名1=值1&参数名2=值2')
/*
  query的结果：
   {
    参数名1: 值1,
    参数名2: 值2
   }
*/
```

```
server.on('request', (req, res) => {
  if (req.url === '/api/province') {
    // 折叠了...
    // 1. 路径以 /api/city 开头
  } else if (req.url.startsWith('/api/city')) {
    // 2. 获取查询字符串中，省份名字
    const queryStr = req.url.split('?')[1]
    const query = qs.parse(queryStr)
    const pname = query.pname
    // 3. 查找匹配省份下的城市列表
    fs.readFile(path.join(__dirname, 'city.json'), (err, data) => {
      const dataObj = JSON.parse(data.toString())
      const cityList = dataObj[pname]
      const cityJSON = JSON.stringify(cityList)
      // 4. 返回匹配的城市列表
      res.setHeader('Content-Type', 'application/json; charset=utf-8')
      res.end(cityJSON)
    })
  } else {
    res.setHeader('Content-Type', 'text/html; charset=utf-8')
    res.end('你要访问的路径不存在')
  }
})
```

案例 浏览时钟

需求：基于 Web 服务，开发提供网页资源的功能，了解下后端的代码工作过程

案例 浏览时钟

步骤：

1. 基于 http 模块，创建 Web 服务
2. 使用 req.url 获取请求资源路径，并读取 index.html 里字符串内容返回给请求方
3. 其他路径，暂时返回不存在的提示
4. 运行 Web 服务，用浏览器发起请求

```
server.on('request', (req, res) => {
  if (req.url === '/index.html') {
    fs.readFile(path.join(__dirname, 'dist/index.html'), (err, data) => {
      res.setHeader('Content-Type', 'text/html;charset=utf-8')
      res.end(data.toString())
    })
  } else {
    res.setHeader('Content-Type', 'text/html;charset=utf-8')
    res.end('你要访问的路径不存在')
  }
})
```

传智教育旗下高端IT教育品牌