Node.js基础

课程目标

- 了解nodejs特点和应用场景
- 掌握node模块系统使用
- 掌握核心api使用
- 实战一个简版Express服务器

NodeJS是什么

node.js是一个异步的事件驱动的JavaScript运行时

node.js特性：

- 非阻塞I/O
- 事件驱动

准备工作

- 运行node程序 `node 01-runnode.js`
- 调试node程序：Debug - Start Debugging
- 使用模块(module)
  - node内建模块

    ```
    require('os')
    ```

  - 第三方模块

    ```
    require("cpu-stat");
    ```

  - 自定义模块：

```
// 导出
module.exports = {}

// 导入
require('./conf')
```

核心API

- fs

```
const fs = require('fs');

fs.readFile('./笔记.md', (err, data) => {
    if (err) throw err;
    console.log(data);
});
```

- Buffer

```
// 创建
const buf1 = Buffer.alloc(10);
const buf2 = Buffer.from([1, 2, 3]);
const buf3 = Buffer.from('Buffer创建方法');

// 写入
buf1.write('hello');

// 读取
console.log(buf3.toString());

// 合并
const buf4 = Buffer.concat([buf1, buf3]);
```

- http

```
const http = require('http');
const server = http.createServer((request, response) => {
    console.log('there is a request');
    response.end('a response from server');
});
server.listen(3000);
```

- stream

```
const rs = fs.createReadStream('./conf.js')
const ws = fs.createWriteStream('./conf2.js')
rs.pipe(ws);
```

仿写一个简版Express

- 体验express

```
// npm i express
const express = require("express");
const app = express();
app.get("/", (req, res) => {
  res.end("Hello World");
});
app.get("/users", (req, res) => {
  res.end(JSON.stringify([{ name: "tom", age: 20 }]));
});
app.listen(3000, () => {
  console.log("Example app listen at 3000");
});
```

- 实现kexpress

```
const http = require("http");
const url = require("url");

let router = [];

class Application {
  get(path, handler) {
    console.log("get...", path);
    if (typeof path === "string") {
      router.push({
        path,
        method: "get",
        handler
      });
    } else {
      router.push({
        path: "*",
        method: "get",
        handler: path
      });
    }
  }

  listen() {
    //在Application原型上添加listen方法匹配路径，执行对应的handler方法
    const server = http.createServer(function(req, res) {
      console.log(url.parse(req.url, true));

      let { pathname } = url.parse(req.url, true);
      router.forEach(route => {
        let { path, method, handler } = route;
```

```
        if (pathname == path && req.method.toLowerCase() == method) {
          return handler(req, res);
        }
        if (path == "*") {
          return handler(req, res);
        }
      });
    });
    server.listen(...arguments);
  }
}
module.exports = function createApplication() {
  return new Application();
};
```