1. 什么是浅拷贝与深拷贝：
2. 什么是浅拷贝 ？

只复制某个对象的内存地址，并不复制对象本身，共用一个内存空间，改变其中一个对象会影响另外一个对象。

1. 什么是深拷贝？

复制并创建对象本身，在内存中重新开辟内存空间，对象之间互不影响。改变其中一个对象不会影响另外一个对象

//深浅拷贝是针对的复杂数据类型：对象、数组

let obj = {};

// let obj2 = obj;

obj2 = JSON.parse(JSON.stringify(obj));

obj2.x = 22;

console.log(obj, obj2);

let obj3 = {

    "age": 21,

    "usr": 'demo',

    "eat": function() {

        console.log('eat...');

    }

};

//注意：这种深拷贝不能拷贝对象中的方法，但可以使用浅拷贝+递归来解决

let obj4 = JSON.parse(JSON.stringify(obj3));

console.log(obj3, obj4);

let obj5 = deepCopy(obj3);

console.log(obj3, obj5);

//浅拷贝+递归

function deepCopy(obj) {

    if (Object.prototype.toString.call(obj).slice(8, -1) == 'Object') {

        var result = {}

    } else if (Object.prototype.toString.call(obj).slice(8, -1) == 'Array') {

        var result = []

    } //判断数据类型类型  如果是数组则初始一个  []  如果是一个Object则初始一个 {}

    //浅拷贝，但是+ 递归思想，就实现了深拷贝

    for (var attr in obj) {

        if (typeof obj[attr] == 'object') {

            result[attr] = deepCopy(obj[attr])

        } else {

            result[attr] = obj[attr]

        }

    }

    return result

}

1. Buffer缓冲器:

node操作文件、图片等，而这些操作都与二进制数据有关，再比如node在处理像TCP流或文件流时，必须使用到二进制数据，因此在Node.js中，定义了一个 Buffer 类，该类用来创建一个专门存放二进制数据的缓存区。Buffer与数组类似，可以直接操作内存，但数组不能放二进制数据。

//创建Buffer

let buf1 = Buffer.alloc(10);

console.log(buf1);

buf1[0] = 6;

buf1[1] = 88;

buf1[2] = 0xaa; //十六进制

buf1[3] = 256;

buf1[4] = 355;

// buf1[10] = 200; //Buffer大小一旦确定不能修改

buf1[9] = 255;

01100100

console.log(buf1);

//allocUnsafe()在创建buffer时不会初始化

//alloc()在创建buffer时会初始化

let buf2 = Buffer.allocUnsafe(10);

console.log(buf2);

let str = 'this is 测试';

let buf3 = Buffer.from(str);

console.log(buf3.length); //在buffer中所占空间长度

console.log(str.length);

console.log(buf3.toString(), buf3);

let obj = { "x": 1, "y": 'tom' };

let buf4 = Buffer.from(JSON.stringify(obj));

console.log(buf4.toString());

// let buf5 = Buffer.from([1, 2, 8]);

let buf5 = Buffer.from(["1", "2", "c"]);

// console.log(buf5);

1. Buffer缓冲器应用：

//需求：将buffer中的内容写入文件

const fs = require('fs');

const path = require('path');

let buf = Buffer.from('hello web...');

// console.log(buf);

//将buffer写文件

fs.writeFileSync(path.resolve(\_\_dirname, 'test.txt'), buf);

//读取文件内容

let data = fs.readFileSync(path.resolve(\_\_dirname, 'test.txt'));

console.log(data.toJSON());

console.log(data, data.toString());

1. 使用淘宝的cnpm命令：
2. 到npm.taoba.org镜像服务器上找到下面这句，在dos小黑窗中执行下面的命令安装cnpm:

npm install -g cnpm --registry=https://registry.npm.taobao.org

1. cnpm与npm可以同时存在，而且npm命令的参数在cnpm命令中都支持

1. Express简介：
2. Epxress简介： 基于 Node.js平台，快速、开放、极简的Web开发框架[后端]
3. 安装express**，在当前项目根目录中**执行下面的命令：

npm i express

1. 使用express搭建web服务器

//引入express

const express = require('express');

// console.log(express);

//创建应用

const app = express();

//监听端口

app.listen(4000, () => {

    console.log('server start at port 4000');

});

1. Express的app级别路由语法： app.请求方式[get/post](‘路径’,回调方法)

**注意**： **所有路径都是以/开头**

//引入express

const express = require('express');

// console.log(express);

//创建应用

const app = express();

//监听端口

app.listen(4000, () => {

    console.log('server start at port 4000');

});

//设计路由：

//express中的路由分为两种：app级别路由、router级别路由

//app级别路由的语法：app.请求方式[get/post]('路径',回调方法)

//注意：所有的路径都是以斜杠开(/)头

app.get('/', (req, res) => {

    res.send('这是首页面 ');

});

// app.get('/ab?cd', function(req, res) {

//     res.send('ab?cd')

// })

// app.get('/ab(cd)?e', function(req, res) {

//     res.send('ab(cd)?e')

// })

app.get(/a/, function(req, res) {

    res.send('/a/')

})

//返回老师相关信息

app.get('/teacher', (req, res) => {

    res.send('这是老师相关信息'); //experss提供的方法

});

//返回学生相关信息

app.get('/student', (req, res) => {

    res.writeHead(200, 'ok', { "content-type": "text/html;charset=utf-8" });

    res.end('这是学生相关信息');

});

//用户登录

app.post('/login', (req, res) => {

    //  res.send('用户登录');

    let obj = { x: 1, y: 33 };

    res.send(obj);

});

//app.all()：能匹配所有请求方式(get/post)

//app.all('/teacher', (req, res) => {

app.all('\*', (req, res) => {

    res.send('404错误');

});

**总结**：app.get(‘路径’,回调方法) app.post(‘路径’,回调方法) app.all(‘\*’,回调方法)

app.all()通常放在所有路由的最后面，通常用来处理错误。

1. Express接收get方式发送的参数：
2. **get方式发送参数的写法有两种**：查询字符串、path路径

(1)查询字符串(标准的url传参写法)，如：

    http://localhost:3000/student?uid=21&usr=tom

  在express中接收get方式中的查询字符串格式的参数：

   req.query;

(2)path路径，如：

   http://localhost:3000/student/21/tom

在express中接收get方式中的path路径格式的参数：

  req.params

**注意**：**path路径写法**：app.get('/student/:x/:y');

const express = require('express');

const app = express();

app.listen(3000, () => {

    console.log('server start at port 3000');

});

//设计路由// app.get('/student', (req, res) => {

app.get('/student/:uid/:usr/:email', (req, res) => {

    //接收get方式的查询字符串格式发送的参数

    // let obj = req.query;

    // console.log(obj);

    // let { uid, usr, email } = req.query;

    //接收get方式的path路径格式发送的参数

    // let obj = req.params;

    // console.log(obj);

    let { uid, usr, email } = req.params;

    res.send(`这是学生信息${uid} ${usr} ${email}`);

    // res.send(`这是学生信息`);

});

1. Express接收get方式发送参数的应用：用户登录

1、login.html代码如下：

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <form action="/login" method="get">

        帐号：<input id="usr" type="text" name="usr"><br> 密码：

        <input id="pwd" type="password" name="pwd"><br>

        <input type="submit" value="登录"><br>

    </form>

</body>

</html>

2、”用户登录.js”文件代码如下：

const express = require('express');

const path = require('path');

const app = express();

app.listen(4000, () => {

    console.log('server start at port 4000');

});

//设计路由

//显示用户登录界面

app.get('/', (req, res) => {

    res.sendFile(path.resolve(\_\_dirname, 'login.html'));

});

//处理用户登录

app.get('/login', (req, res) => {

    //接收get方式中的查询字符串格式发送的参数

    //接收用户输入的帐号及密码

    let { usr, pwd } = req.query;

    // if(usr == 'lisi' && pwd == 1234){

    // }else{

    // }

    //先判断帐号是否正确

    if (usr == 'lisi') { //帐号正确

        //再判断密码是否正确

        if (pwd == 1234) { //密码正确

            res.send('登录成功');

        } else { //密码错误

            res.send("<script>alert('密码错误');location.href='/';</script>");

        }

    } else { //帐号错误

        res.send("<script>alert('帐号错误');location.href='/';</script>");

    }

});