ES6 编程

服务端

客户端

基础语法

入门

交互

1、scanf

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

2、console.log()
3、const 常量
4、let 变量

运算

1、加：+
2、减：-
3、乘：*
4、除：/
5、求余:%　（重点理解一下）

案例1参考代码：

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

let a=20,b=10;
let c=a+b;
```

```
let d=a*b;
let e=a/b;
let f=a%b;
console.log('a='+a,'b='+b);
console.log('c=a+b='+c);
console.log('d=a*b='+d);
console.log('e=a/b='+e);
console.log('f=a%b='+f);
```

案例2参考代码：

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

console.log('请输入两个数，计算两个数的和：请按回车继续');
scanf('%d');
console.log('输入a:');
let a=scanf('%d');
console.log('输入b:');
let b=scanf('%d');
console.log('a+b='+a+b);
```

分支

1、if...else

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

console.log('输入两个数，判断两个数，并输出最大值');
console.log('输入a:');
let a=scanf('%d');
console.log('输入b:');
let b=scanf('%d');
if(a>b){
    console.log('最大值是a: '+a);
}else if(b>a){
    console.log('最大值是b: '+b);
}else{
```

```
    console.log('a=b='+a);
}
```

数组

1、数组的定义

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

let arr=[1,2,3];
console.log('数组arr:长度为'+arr.length+'\n分别是:'+arr);
```

2、数组的应用

循环

1、while循环
案例参考代码：欢迎进入XXX系统

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

console.log('欢迎进入XXX系统');
while(1){
    console.log('1---添加');
    console.log('2---显示全部');
    console.log('3---删除');
    console.log('4---退出');
    let con=scanf('%d');
    if(con===1){
        console.log('1---添加');
        console.log('点击回车继续');
        scanf('%d');
    }
    if(con===2){
        console.log('2---显示全部');
        console.log('点击回车继续');
```

```
            scanf('%d');
        }
        if(con===3){
            console.log('3---删除');
            console.log('点击回车继续');
            scanf('%d');
        }
        if(con===4){
            console.log('退出');
            break;
        }
    }
```

2、for循环

案例参考代码：循环遍历数组

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

let arr=[1,2,3,4,5,6];
// 循环遍历数组
for(let i=0;i<arr.length;i++){
    let xinlong = arr[i];
    console.log(xinlong);
}
```

对象

1、对象的定义

2、对象的使用

案例参考代码：

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

let xinlong={
    a:1,   //横坐标
    b:2    //纵坐标
}
```

```
console.log('横坐标：'+xinlong.a);
console.log('纵坐标：'+xinlong.b);
console.log('坐标：('+xinlong.a+','+xinlong.b+')');
```

函数

1、函数使用

```
const scanf=require('scanf');
// 计算根号2
let a=MAth.sqrt(2);
console.log(a);
```

2、自定义函数

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

//自定义函数例如求和
// 定义add函数
function add(x,y) {
    // 在此可直接使用x,y
    return x+y;
}
// 调用add函数
let b=add(1,2);
console.log(b);
```

3、函数执行顺序

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

//举例：理解函数执行顺序
function xxx() {
    console.log(1);
    console.log(2);
```

```
}

// 案例1
console.log(3);
xxx();
console.log(4);

// 案例2：运行次序已知,如下，会执行两次xxx()函数
xxx();
xxx();
```

4、异步函数（烧水函数）

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

// 调用异步函数
console.log('异步函数案例：烧水函数');
const {plog}=require('tuomaxu-sdk');
plog(1);
plog(2);
plog(3);
```

```
const scanf=require('scanf');
const {plog}=require('tuomaxu-sdk');
console.log('异步函数案例：烧水函数');
// 为了解决异步函数执行结果不定问题，引入async-await 修饰符，把异步变为同步函数,顺序打印plog
async function pp() {
    await plog(1);
    await plog(2);
    await plog(3);
}
// pp();

// 使用plog 打印1，输出之后，在使用console.log()输出'完成'
async function pp2(){
    await plog(1);
    console.log('完成');
}
// pp2();
```

```
// 案例：以此打印1231完成
async function pp3(){
    await pp();
    await pp2();
}
pp3();
```

第一阶段任务

幸运数字管理

所需讲解知识点为数组对象

```
let arr = [];//数组定义

arr.push(1);//向数组添加元素
arr.pop();//删除数组中最后一个元素

//使用for循环遍历数组中所有元素
for (let index = 0; index < arr.length; index++) {
    const element = arr[index];
    console.log(element)
}
```

案例参考代码

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

console.log('欢迎进入幸运数字管理系统');//启动logo

let arr = [];

while (1) {
```

```
    console.log('1---添加幸运数字');
    console.log('2---删除最后一个幸运数字');//删除指定的幸运数字
    console.log('3---显示全部幸运数字');
    console.log('4---退出');

    console.log('请输入相应功能编号');

    let code = scanf('%d');

    if(code === 1){
        console.log('请输入一个要添加的幸运数字');
        let x = scanf('%d');
        arr.push(x);
        console.log('添加成功');

        console.log('点击回车继续');
        scanf('%d');
    }

    if(code === 2){
        console.log('开始删除');
        arr.pop();
        console.log('删除成功');

        console.log('点击回车继续');
        scanf('%d');
    }

    if(code === 3){
        console.log('全部幸运数字如下：');

        //此for循环作用为：打印数组中的每一个幸运数字
        for(let i=0;i<arr.length;i++){
            let q = arr[i];
            console.log(q);
        }

        console.log('点击回车继续');
        scanf('%d');
    }

    if(code === 4){
        console.log('程序退出');
        break;
```

```
    }
}
```

敏感词管理

第二阶段任务

位置坐标管理

案例参考代码：

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

console.log('欢迎进入坐标管理系统');
// 定义数组
let arr=[];
while(1){
    console.log('1---添加一个坐标');
    console.log('2---显示全部坐标');
    console.log('3---删除最后一个坐标');
    console.log('4---退出');
    let con=scanf('%d');  //输入一个数字
    if(con===1){
        console.log('1---添加一个坐标');
        console.log('请输入横坐标：');
        let x=scanf('%d');
        console.log('请输入纵坐标：');
        let y=scanf('%d');
        let zuobiao={
            'x':x,
            'y':y
        };
        arr.push(zuobiao);  //往数组中添加一个数字
        console.log('添加坐标成功');
        scanf('%d');
    }
```

```
    if(con===2){
        console.log('2---显示全部坐标');
        for(let i=0;i<arr.length;i++){
            let syk=arr[i];
            // console.log(syk);
            console.log(`第${i+1}坐标为(${syk.x},${syk.y})`);
        }
        console.log('点击回车继续');
        scanf('%d');
    }
    if(con===3){
        console.log('3---删除最后一个坐标');
        arr.pop();  //删除数组中最后一个数字
        console.log('删除坐标成功');
        scanf('%d');
    }
    if(con===4){
        console.log('退出');
        break;
    }
}
```

第三阶段

学生成绩管理

案例参考代码：

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

console.log('欢迎进学生成绩管理系统');
// 定义数组
let arr=[];
while(1){
    console.log('1---添加一个学生学科成绩');
    console.log('2---显示全部学生学科成绩');
    console.log('3---删除最后一个学生学科成绩');
    console.log('4---退出');
```

```
    let con=scanf('%d');   //输入一个数字
    if(con===1){
        console.log('1---添加一个学生学科成绩');
        console.log('请输入学生姓名：');
        let xingming=scanf('%s');
        console.log('请输入语文成绩：');
        let yuwen=scanf('%d');
        console.log('请输入数学成绩：');
        let shuxue=scanf('%d');
        let student={
            'xingming':xingming,
            'yuwen':yuwen,
            'shuxue':shuxue
        };
        arr.push(student);   //往数组中添加一个数字
        console.log('添加成绩成功');
        scanf('%d');
    }
    if(con===2){
        console.log('2---显示全部学生学科成绩');
        for(let i=0;i<arr.length;i++){
            let syk=arr[i];
            // console.log(syk);
            console.log(`学生：${syk.xingming}\n语文：${syk.yuwen}\n数学：
${syk.shuxue}`);
        }
        console.log('点击回车继续');
        scanf('%d');
    }
    if(con===3){
        console.log('3---删除最后一个学生学科成绩');
        arr.pop();   //删除数组中最后一个数字
        console.log('删除最后一名学生成绩成功');
        scanf('%d');
    }
    if(con===4){
        console.log('退出');
        break;
    }
}
```

ORM操作

数据库

搭建

1、准备
npm init　//初始化操作
npm i scanf　//安装scanf输入插件
npm i mysql2　//连接数据库驱动2
npm i sequelize　//简化数据库操作,可以使数据库操作与数组操作类似

```
const scanf=require('scanf');
const Sequelize=require('sequelize');  //首字母必须大写

//需要使用数据库的准备工作代码

// 1、创建sequelize 对象，通过sequelize对象操作数据库
const seq=new Sequelize('es6','root','123456',{dialect:'mysql'});// new的对象首字母大写
//数据库名，用户名，密码，方言（指定数据库类型）

//2、定义数据库中的数据格式，数据名字和数据类型
//sequelize.INTEGER===整数
//point:定义数据模型
const Point=seq.define('point',{//定义数据模型，point数据库表名称，再数据库生成时会自动加s
    x:Sequelize.INTEGER,
    y:Sequelize.INTEGER,
},{
    //如果设置此字段，数据库中表名不会增加s,数据表名则与此处设置的数据模型point一致
    freezeTableName:true,
});

// 3、同步数据库
seq.sync();
```

操作1（增加）

```
console.log('欢迎进入坐标管理系统');//启动logo
//对函数内代码进行封装
async function main(){
      console.log('请输入横坐标：');
      let x=scanf('%d');
      console.log('请输入纵坐标：');
      let y=scanf('%d');
      let zuobiao={
          'x':x,
          'y':y
          //向数据库中保存数据
          await Point.create(zuobiao);
          console.log('添加坐标成功');
          scanf('%d');
      }
}
main(); //执行main()函数
```

操作2（查询）

```
console.log('欢迎进入坐标管理系统');//启动logo
//对函数内代码进行封装
async function main(){
      //findALL获取数据库中所有数据
      let arr=await Point.findAll();
      for(let i=0;i<arr.length;i++){
          let syk=arr[i];
          // console.log(syk);
          console.log(`坐标id:${syk.id}坐标为(${syk.x},${syk.y})`);
      }
      console.log('点击回车继续');
      scanf('%d');
}
main(); //执行main()函数
```

操作3（增删改查）

```
console.log('欢迎进入坐标管理系统');//启动logo
//对函数内代码进行封装
async function main(){
    while(1){
            console.log('1---添加一个坐标');
            console.log('2---显示全部坐标');
            console.log('3---通过id删除指定坐标');
            console.log('4---退出');
            let con=scanf('%d');   //输入一个数字
            if(con===1){
                console.log('1---添加一个坐标');
                console.log('请输入横坐标：');
                let x=scanf('%d');
                console.log('请输入纵坐标：');
                let y=scanf('%d');
                let zuobiao={
                    'x':x,
                    'y':y
                };

                //向数据库中保存数据
                await Point.create(zuobiao);

                console.log('添加坐标成功');
                scanf('%d');
            }
            if(con===2){
                console.log('2---显示全部坐标');
                //findALL获取数据库中所有数据
                let arr=await Point.findAll();

                for(let i=0;i<arr.length;i++){
                    let syk=arr[i];
                    // console.log(syk);
                    console.log(`坐标id:${syk.id}坐标为(${syk.x},${syk.y})`);
                }
                console.log('点击回车继续');
                scanf('%d');
            }
            if(con===3){
```

```
                console.log('3---通过id删除指定坐标');
                let id=scanf('%d');
                // arr.pop();  //删除数组中最后一个数字
                await Point.destroy({
                    where:{
                        'id':id
                    }
                });
                console.log('删除坐标成功');
                scanf('%d');
            }
            if(con===4){
                console.log('退出');
                break;
            }
        }
}
main(); //执行main()函数
```

第四阶段任务

坐标管理系统（数据库）

```
const scanf=require('scanf');
const Sequelize=require('sequelize');  //首字母必须大写

//需要使用数据库的准备工作代码

// 1、创建sequelize 对象，通过sequelize对象操作数据库
const seq=new Sequelize('es6','root','123456',{dialect:'mysql'});// new的对象首字母大写
//数据库名，用户名，密码，方言（指定数据库类型）

//2、定义数据库中的数据格式，数据名字和数据类型
//sequelize.INTEGER===整数
//point:定义数据模型
const Point=seq.define('point',{//定义数据模型，point数据库表名称，再数据库生成时会自动加s
```

```
    x:Sequelize.INTEGER,
    y:Sequelize.INTEGER,
},{
    //如果设置此字段，数据库中表名不会增加s,数据表名则与此处设置的数据模型point一致
    freezeTableName:true,
});

// 3、同步数据库
seq.sync();

console.log('欢迎进入坐标管理系统');//启动logo
async function main(){ //对函数内代码进行封装
    while(1){
            console.log('1---添加一个坐标');
            console.log('2---显示全部坐标');
            console.log('3---通过id删除指定坐标');
            console.log('4---退出');
            let con=scanf('%d');  //输入一个数字
            if(con===1){
                console.log('1---添加一个坐标');
                console.log('请输入横坐标：');
                let x=scanf('%d');
                console.log('请输入纵坐标：');
                let y=scanf('%d');
                let zuobiao={
                    'x':x,
                    'y':y
                };
                // arr.push(zuobiao);  //往数组中添加一个数字

                //向数据库中保存数据
                await Point.create(zuobiao);

                console.log('添加坐标成功');
                scanf('%d');
            }
            if(con===2){
                console.log('2---显示全部坐标');
                //findALL获取书库中所有数据
                let arr=await Point.findAll();

                for(let i=0;i<arr.length;i++){
                    let syk=arr[i];
                    // console.log(syk);
                    console.log(`坐标id:${syk.id}坐标为(${syk.x},${syk.y})`);
```

```
                }
                console.log('点击回车继续');
                scanf('%d');
            }
            if(con===3){
                console.log('3---通过id删除指定坐标');
                let id=scanf('%d');
                // arr.pop();  //删除数组中最后一个数字
                await Point.destroy({
                    where:{
                        'id':id
                    }
                });
                console.log('删除坐标成功');
                scanf('%d');
            }
            if(con===4){
                console.log('退出');
                break;
            }
        }
}
main(); //执行main()函数
```

HTTP通信

服务端（API）

1、准备

npm init //初始化操作

npm i mysql2 //连接数据库驱动2

npm i sequelize //简化数据库操作,可以使数据库操作与数组操作类似

npm i express //安装express框架

npm i body-parser //客户端数据功能

```
// api 主要实现增删改查功能，但是自己不执行，由客户端发送执行命令
// 将增删改查操作功能封装到一个函数中
```

```
// 连接数据库
const Sequelize = require('sequelize');

//使用express框架，将服务段工程中的函数，发布为可以通过HTTP(网络)进行访问的接口
const express=require('express');

//引用处理客户端数据功能
const bodyParser=require('body-parser');

// 1、创建express对象
const app=express();

// 2、使用body-parser
app.use(bodyParser.json());


// 1、新建数据库，api
const seq = new Sequelize('api', 'root', '123456', {
    dialect: 'mysql'
});
const Point = seq.define('point', {
    x: Sequelize.INTEGER,
    y: Sequelize.INTEGER,
}, {
    freezeTableName: true,
});
seq.sync();

// requset:保存客户端发送过来信息
// response: 保存发送给客户端信息
// 1、添加
async function add(requset,response) {
    let x = requset.body.x;
    let y = requset.body.y;
    let p = {
        'x': x,
        'y': y
    };
    await Point.create(p); //写入到数据库

    // 写入成功，发送个客户端
    // response.send({
    //     success: true,
    // });
```

```
    let q={
        success:true,
    };
    response.send(q);
}

//发布请求函数，有这个才能被访问，post有两个请求参数决定，第一个为请求地址、第二个是请求地址的请求函数
app.post('/add',add);

// 2、删除
async function del(requset,response) {
    let id = requset.body.id;
    await Point.destroy({
        where: {
            id: id
        }
    });
    // response.send({
    //     success: true,
    // });
    let q={
        success:true,
    };
    response.send(q);
}
app.post('/del',del);

// 3、查询全部
async function all(requset,response) {
    let arr = await Point.findAll();
    let q={
        'arr':arr,
        success:true,
        // xxx:ddd, // 传到客户端时候只有q和q里的xxx 。 ddd只是容器名称，不会影响内部东西，因为需要显示查询结果，要传到客户端显示
    };
    response.send(q);
}
app.post('/all',all);

//2.监听一个端口，
//IP地址为计算机地址，但一个计算机中包含很多应用
//比如一个消息，要发送到QQ而不是微信，计算机内的应用可以监听计算机端口
//端口为一个整数，取值范围为1---65535，一般使用10000之后的任意端口
```

```
app.listen(10001);
```

客户端（APP）

```
// 1、装axios（请求服务器）、scanf
const scanf=require('scanf');
const axios=require('axios');

async function main() {
    while(1){
        console.log('1---请求name');
        console.log('2---请求login');
        console.log('请选择');

        let code=scanf('%d');
        // 请求name
        if (code===1) {
            console.log('请求结果为:');

            let response=await axios.get('http://127.0.0.1:80/name');
            console.log(response.data);
            console.log('点击回车继续');
            scanf('%d');
        }
        if (code===2) {
            console.log('请输入用户名:');
            let username=scanf('%s');
            console.log('请输入密码:');
            let password=scanf('%s');

            let body={
                'username':username,
                'password':password,
            }
            let response= await axios.post('http://127.0.0.1:80/login',body);
            console.log(response.data);
        }
    }
}
main();
```

第五阶段任务

1、坐标管理系统（服务端+客户端）

服务端

1、准备
npm init　//初始化操作
npm i mysql2　//连接数据库驱动2
npm i sequelize　//简化数据库操作,可以使数据库操作与数组操作类似
npm i express　//安装express框架
npm i body-parser　//客户端数据功能

```
// api 主要实现增删改查功能，但是自己不执行，由客户端发送执行命令
// 将增删改查操作功能封装到一个函数中
// 连接数据库
const Sequelize = require('sequelize');

//使用express框架，将服务段工程中的函数，发布为可以通过HTTP(网络)进行访问的接口
const express=require('express');

//引用处理客户端数据功能
const bodyParser=require('body-parser');

// 1、创建express对象
const app=express();

// 2、使用body-parser
app.use(bodyParser.json());


// 1、新建数据库，api
const seq = new Sequelize('api', 'root', '123456', {
    dialect: 'mysql'
});
```

```
const Point = seq.define('point', {
    x: Sequelize.INTEGER,
    y: Sequelize.INTEGER,
}, {
    freezeTableName: true,
});
seq.sync();

// requset:保存客户端发送过来信息
// response: 保存发送给客户端信息
// 1、添加
async function add(requset,response) {
    let x = requset.body.x;
    let y = requset.body.y;
    let p = {
        'x': x,
        'y': y,
    };
    await Point.create(p); //写入到数据库

    // 写入成功，发送个客户端
    // response.send({
    //     success: true,
    // });

    let q={
        success:true,
    };
    response.send(q);
}

//发布请求函数，有这个才能被访问，post有两个请求参数决定，第一个为请求地址、第二个是请求地址的请求函数
app.post('/add',add);

// 2、删除
async function del(requset,response) {
    let id = requset.body.id;
    await Point.destroy({
        where: {
            id: id
        }
    });
    // response.send({
    //     success: true,
```

```
    // });
    let q={
        success:true,
    };
    response.send(q);
}
app.post('/del',del);

// 3、查询全部
async function all(requset,response) {
    let arr = await Point.findAll();
    let q={
        'arr':arr,
        success:true,
        // xxx:ddd, // 传到客户端时候只有q和q里的xxx 。 ddd只是容器名称，不会影响内部东西，因为需要显示查询结果，要传到客户端显示
    };
    response.send(q);
}
app.post('/all',all);

//2.监听一个端口，
//IP地址为计算机地址，但一个计算机中包含很多应用
//比如一个消息，要发送到QQ而不是微信，计算机内的应用可以监听计算机端口
//端口为一个整数，取值范围为1---65535，一般使用10000之后的任意端口

app.listen(10001);
```

客户端

```
// 1、装axios（请求服务器）、scanf
const scanf=require('scanf');
const axios=require('axios');

async function main() {
    while(1){
        console.log('1---请求point,添加一个坐标');
        console.log('2---请求point,删除一个坐标');
        console.log('3---请求point,显示所有');
        console.log('请选择');
```

```
let code=scanf('%d');
// 请求point,添加一个坐标
if (code===1) {
    console.log('请输入x坐标:');
    let x=scanf('%d');
    console.log('请输入y坐标:');
    let y=scanf('%d');
    let body={
        'x':x,
        'y':y,
    }
    let response=await axios.post('http://127.0.0.1:10001/add',body);
    console.log(response.data);
    console.log('点击回车继续');
    scanf('%d');
}
// 请求point,删除一个坐标
if(code===2){
    console.log('请输入要删除坐标的id:');
    let id=scanf('%d');
    let body={
        id:id
    }
    let response=await axios.post('http://127.0.0.1:10001/del',body);
    console.log(response.data);
    console.log('点击回车继续');
    scanf('%d');
}
// 请求point,显示所有
if (code===3) {
    console.log('显示所有坐标:');
    // for(let i=0;i<arr.length;i++){
    //     let syk=arr[i];
    //     console.log(`坐标id:${syk.id}坐标为(${syk.x},${syk.y})`);
    // }
    let response=await axios.post('http://127.0.0.1:10001/all');

    console.log(response.data.arr);

    let arr=response.data.arr;
    for (let index = 0; index < arr.length; index++) {
        let syk=arr[index];
        console.log(`坐标id:${syk.id}坐标为(${syk.x},${syk.y})`);
    }
    console.log('点击回车继续');
```

```
            scanf('%d');
        }
    }
}
main();
```