1 解构赋值和参数默认值结合 {host="127.0.0.1", username,password, port}

        function connect({host="127.0.0.1", username,password, port}){

            console.log(host)

            console.log(username)

            console.log(password)

            console.log(port)

        }

        connect({

            host: 'atguigu.com',

            username: 'root',

            password: 'root',

            port: 3306

        })

解构赋值

const obj={

            name:"马涛",

            age:18,

            say:'12222'

        }

let {name,age,say}=obj

2 迭代器 Symbol.iterator

        const xiyou = ['唐僧','孙悟空','猪八戒','沙僧'];

        //使用 for...of 遍历数组

        // for(let v of xiyou){ }

        let iterator = xiyou[Symbol.iterator]();

        //调用对象的next方法

        console.log(iterator.next());

        console.log(iterator.next());

        console.log(iterator.next());

        console.log(iterator.next());

        console.log(iterator.next());

3 迭代器自定义遍历对象 Symbol.iterator

        const banji = {

            name: "终极一班",

            stus: [

                'xiaoming',

                'xiaoning',

                'xiaotian',

                'knight'

            ],

            [Symbol.iterator]() {

                //索引变量

                let index = 0;

                //

                let \_this = this;

                return {

                    next: function () {

                        if (index < \_this.stus.length) {

                 const result = { value: \_this.stus[index], done: false };

                            //下标自增

                            index++;

                            //返回结果

                            return result;

                        }else{

                            return {value: undefined, done: true};

                        }

                    }

                };

            }

        }

        //遍历这个对象

        for (let v of banji) {

            console.log(v);

        }

4 对象方法扩展

Object.is 判断两个值是否完全相等

Object.assign 对象的合并

Object.setPrototypeOf 设置原型对象  Object.getPrototypeof

//1. Object.is 判断两个值是否完全相等

        // console.log(Object.is(120, 120));// ===

        // console.log(Object.is(NaN, NaN));// ===

        // console.log(NaN === NaN);// ===

        //2. Object.assign 对象的合并

        // const config1 = {

        //     host: 'localhost',

        //     port: 3306,

        //     name: 'root',

        //     pass: 'root',

        //     test: 'test'

        // };

        // const config2 = {

        //     host: 'http://atguigu.com',

        //     port: 33060,

        //     name: 'atguigu.com',

        //     pass: 'iloveyou',

        //     test2: 'test2'

        // }

        // console.log(Object.assign(config1, config2));

        //3. Object.setPrototypeOf 设置原型对象  Object.getPrototypeof

        const school = {

            name: '尚硅谷'

        }

        const cities = {

            xiaoqu: ['北京','上海','深圳']

        }

        Object.setPrototypeOf(school, cities);

5 简化对象写法

const school = {

            name,

            change,

            improve(){

                console.log("我们可以提高你的技能");

            }

        }

6箭头函数

函数里面本身没有this,也没有arguments

let pow = n => n \* n;

7 数组的解构

        // const F4 = ['小沈阳','刘能','赵四','宋小宝'];

        // let [xiao, liu, zhao, song] = F4;

        //2. 对象的解构

        // const zhao = {

        //     name: '赵本山',

        //     age: '不详',

        //     xiaopin: function(){

        //         console.log("我可以演小品");

        //     }

        // };

        // let {name, age, xiaopin} = zhao;

8 扩展运算符应用

  //1. 数组的合并

        // const kuaizi = ['王太利','肖央'];

        // const fenghuang = ['曾毅','玲花'];

        // const zuixuanxiaopingguo = [...kuaizi, ...fenghuang];

        //2. 数组的克隆

        // const sanzhihua = ['E','G','M'];

        // const sanyecao = [...sanzhihua];//  ['E','G','M']

         //3. 将伪数组转为真正的数组

        const divs = document.querySelectorAll('div');

        const divArr = [...divs];

9 类的静态成员,属性和方法在构造函数上

class Phone{

            //静态属性

            static name = '手机';

            static change(){

                console.log("我可以改变世界");

            }

        }

10 类继承- extends super(brand, price)

 class Phone{

            //构造方法

            constructor(brand, price){

                this.brand = brand;

                this.price = price;

            }

            //父类的成员属性

            call(){

                console.log("我可以打电话!!");

            }

        }

        class SmartPhone extends Phone {

            //构造方法

            constructor(brand, price, color, size){

                super(brand, price);// Phone.call(this, brand, price)

                this.color = color;

                this.size = size;

            }

            photo(){

                console.log("拍照");

            }

            playGame(){

                console.log("玩游戏");

            }

            call(){

                console.log('我可以进行视频通话');

            }

        }

11 模板字符串

 //2. 内容中可以直接出现换行符

        let str = `<ul>

                    <li>沈腾</li>

                    <li>玛丽</li>

                    <li>魏翔</li>

                    <li>艾伦</li>

                    </ul>`;

        //3. 变量拼接

        let lovest = '魏翔';

        let out = `${lovest}是我心目中最搞笑的演员!!`;

12 生成器函数实例

  //模拟获取  用户数据  订单数据  商品数据

        function getUsers(){

            setTimeout(()=>{

                let data = '用户数据';

                //调用 next 方法, 并且将数据传入

                iterator.next(data);

            }, 1000);

        }

        function getOrders(){

            setTimeout(()=>{

                let data = '订单数据';

                iterator.next(data);

            }, 1000)

        }

        function getGoods(){

            setTimeout(()=>{

                let data = '商品数据';

                iterator.next(data);

            }, 1000)

        }

        function \* gen(){

            let users = yield getUsers();

            let orders = yield getOrders();

            let goods = yield getGoods();

        }

        //调用生成器函数

        let iterator = gen();

        iterator.next();

13 数值扩展

Number.EPSILON 是 JavaScript 表示的最小精度

        //EPSILON 属性的值接近于 2.2204460492503130808472633361816E-16

        // function equal(a, b){

        //     if(Math.abs(a-b) < Number.EPSILON){

        //         return true;

        //     }else{

        //         return false;

        //     }

        // }

        // console.log(0.1 + 0.2 === 0.3);

        // console.log(equal(0.1 + 0.2, 0.3))

二进制和八进制

        // let b = 0b1010;

        // let o = 0o777;

        // let d = 100;

        // let x = 0xff;

        // console.log(x);

Number.isFinite  检测一个数值是否为有限数

        // console.log(Number.isFinite(100/0));

        // console.log(Number.isFinite(Infinity));

Number.isNaN 检测一个数值是否为 NaN

        // console.log(Number.isNaN(123));

Number.parseInt Number.parseFloat字符串转整数

        // console.log(Number.parseInt('5211314love'));

        // console.log(Number.parseFloat('3.1415926神奇'));

Number.isInteger 判断一个数是否为整数

        // console.log(Number.isInteger(5));

        // console.log(Number.isInteger(2.5));

Math.trunc 将数字的小数部分抹掉

        // console.log(Math.trunc(3.5));

Math.sign 判断一个数到底为正数 负数 还是零

        console.log(Math.sign(100));

        console.log(Math.sign(0));

        console.log(Math.sign(-20000));

14 class的set-get

     // get 和 set

        class Phone{

            get price(){

                console.log("价格属性被读取了");

                return 'iloveyou';

            }

            set price(newVal){

                console.log('价格属性被修改了');

            }

        }

15 class

        class Shouji{

            //构造方法 名字不能修改

            constructor(brand, price){

                this.brand = brand;

                this.price = price;

            }

            //方法必须使用该语法, 不能使用 ES5 的对象完整形式

            call(){

                console.log("我可以打电话!!");

            }

        }

16 const

        const SCHOOL = '尚硅谷';

        //1. 一定要赋初始值

        //2. 一般常量使用大写(潜规则)

        //3. 常量的值不能修改

        //4. 块儿级作用域

        // {

        //     const PLAYER = 'UZI';

        // }

        //5. 对于数组和对象的元素修改, 不算做对常量的修改, 不会报错

        const TEAM = ['UZI','MXLG','Ming','Letme'];

        // TEAM.push('Meiko');

17 let

//1. 变量不能重复声明

        //2. 块儿级作用域

        //3. 不存在变量提升

        //4. 不影响作用域链

        {

            let school = '尚硅谷';

            function fn(){

                console.log(school);

            }

            fn();

        }

18 Map

        let m = new Map();

        //添加元素

        let key = {

            school : 'ATGUIGU'

        };

//set

        m.set(key, ['北京','上海','深圳']);

//Map(1) {{…} => Array(3)}

        size

        // console.log(m.size);

       delete

        // m.delete('name');

        get

        // console.log(m.get(key));

        clear

        // m.clear();

        //遍历

        for(let v of m){

            console.log(v);

        }

19 rest 参数 ...

        // function date(...args){

        //     console.log(args);// filter some every map

        // }

        // date('阿娇','柏芝','思慧');

        // rest 参数必须要放到参数最后

        // function fn(a,b,...args){

        //     console.log(a);

        //     console.log(b);

        //     console.log(args);

        // }

        // fn(1,2,3,4,5,6);

20 new Set()

        let s = new Set();

        let s2 = new Set(['大事儿','小事儿','好事儿','坏事儿','小事儿']);

//Set(4) {'大事儿', '小事儿', '好事儿', '坏事儿'}

1没有重复的元素

        2 s2.size

        3 s2.add('喜事儿');

        4 s2.delete('坏事儿');

      5 s2.has('糟心事'));

        6 s2.clear();

        7 for(let v of s2){ }

21 globalThis //指向全局this

22 new Set(arr)集合实践

let arr = [1,2,3,4,5,4,3,2,1];

        //1. 数组去重

        // let result = [...new Set(arr)];

        //2. 交集

        let arr2 = [4,5,6,5,6];

        // let result = [...new Set(arr)].filter(

item => new Set(arr2).has(item));

        //3. 并集

        // let union = [...new Set([...arr, ...arr2])];

        //4. 差集

        let diff = [...new Set(arr)].filter(

item => !(new Set(arr2).has(item)));

23 spread扩展运算符

        //声明一个数组 ...

        const tfboys = ['易烊千玺','王源','王俊凯'];

       ...tfboys '易烊千玺','王源','王俊凯'

24 创建Symbol('尚硅谷')

        let s = Symbol();

        let s2 = Symbol('尚硅谷');

        let s3 = Symbol('尚硅谷');

        //Symbol.for 创建

        let s4 = Symbol.for('尚硅谷');

        let s5 = Symbol.for('尚硅谷');

        //向对象中添加方法 up down

        let game = {

            name:'俄罗斯方块',

            up: function(){},

            down: function(){}

        };

        //声明一个对象

        // let methods = {

        //     up: Symbol(),

        //     down: Symbol()

        // };

        // game[methods.up] = function(){ }

        let youxi = {

            name:"狼人杀",

            [Symbol('say')]: function(){}

        }

25 Promise实践-读取多个文件

const p = new Promise((resolve, reject) => {

    fs.readFile("./resources/为学.md", (err, data) => {

        resolve(data);

    });

});

p.then(value => {

    return new Promise((resolve, reject) => {

        fs.readFile("./resources/插秧诗.md", (err, data) => {

            resolve([value, data]);

        });

    });

}).then(value => {

    return new Promise((resolve, reject) => {

        fs.readFile("./resources/观书有感.md", (err, data) => {

            //压入

            value.push(data);

            resolve(value);

        });

    })

}).then(value => {

    console.log(value.join('\r\n'));

});

26 import

        //1. 通用的导入方式

        //引入 m1.js 模块内容

//分别暴露

export let school = '尚硅谷';

export function teach() {}

     import \* as m1 from "./src/js/m1.js";

    引入 m2.js 模块内容

//统一暴露

let school = '尚硅谷';

function findJob(){}

export {school, findJob};

  import \* as m2 from "./src/js/m2.js";

      引入 m3.js

export default {

    school: 'ATGUIGU',

    change: function(){

        console.log("我们可以改变你!!");

    }

}

        import \* as m3 from "./src/js/m3.js";

        //2. 解构赋值形式

        import {school, teach} from "./src/js/m1.js";

        import {school as guigu, findJob} from "./src/js/m2.js";

        import {default as m3} from "./src/js/m3.js";

        //3. 简便形式  针对默认暴露

        import m3 from "./src/js/m3.js";

27 es6转化为es5

<!--

0.npm init --yes

        1. 安装工具 npm i babel-cli babel-preset-env browserify -D

        2. 编译 npx babel src/js -d dist/js --presets=babel-preset-env

        3. 打包 npx browserify dist/js/app.js -o dist/bundle.js

     -->

28 includes \*\*

   // const mingzhu = ['西游记','红楼梦','三国演义','水浒传'];

       mingzhu.includes('西游记')

       2 \*\* 10===Math.pow(2, 10)

29 async function fn(){

            // 返回的结果不是一个 Promise 类型的对象, 就是成功 Promise 对象

            //抛出错误, 返回的结果是一个失败的 Promise

            // throw new Error('出错啦!');

            //返回的结果如果是一个 Promise 对象

            return new Promise((resolve, reject)=>{

                resolve('成功的数据');

                // reject("失败的错误");

            });

        }

        const result = fn();

        //调用 then 方法

        result.then(value => {

            console.log(value);

        }, reason => {

            console.warn(reason);

        })

30 await

//1. 引入 fs 模块

const fs = require("fs");

//读取『为学』

function readWeiXue() {

    return new Promise((resolve, reject) => {

        fs.readFile("./resources/为学.md", (err, data) => {

            //如果失败

            if (err) reject(err);

            //如果成功

            resolve(data);

        })

    })

}

function readChaYangShi() {

    return new Promise((resolve, reject) => {

        fs.readFile("./resources/插秧诗.md", (err, data) => {

            //如果失败

            if (err) reject(err);

            //如果成功

            resolve(data);

        })

    })

}

//声明一个 async 函数

async function main(){

    //获取为学内容

    let weixue = await readWeiXue();

    //获取插秧诗内容

    let chayang = await readChaYangShi();

}

main();

31 Object.keys Object.values Object.entries

        const school = {

            name:"尚硅谷",

            cities:['北京','上海','深圳'],

            xueke: ['前端','Java','大数据','运维']

        };

        //获取对象所有的键

        console.log(Object.keys(school))

// ['name', 'cities', 'xueke']

        //获取对象所有的值

        console.log(Object.values(school));

// ['尚硅谷', Array(3), Array(4)]

        //entries

        console.log(Object.entries(school))

  [Array(2), Array(2), Array(2)]

0: (2) ['name', '尚硅谷']

1: (2) ['cities', Array(3)]

2: (2) ['xueke', Array(4)]

        //创建 Map

        // const m = new Map(Object.entries(school));

      //对象属性的描述对象

        Object.getOwnPropertyDescriptors(school)

      // const obj = Object.create(null, {

        //     name: {

        //         //设置值

        //         value: '尚硅谷',

        //         //属性特性

        //         writable: true,

        //         configurable: true,

        //         enumerable: true

        //     }

        // });

32 rest 参数 添加了对象的运用

        function connect({host, port, ...user}){

            console.log(host);

            console.log(port);

            console.log(user);

        }

        connect({

            host: '127.0.0.1',

            port: 3306,

            username: 'root',

            password: 'root',

            type: 'master'

        });

  扩展运算符

        const skillOne = {

            q: '天音波'

        }

        const skillTwo = {

            w: '金钟罩',

        }

        const skillThree = {

            e: '天雷破'

        }

        const skillFour = {

            r: '猛龙摆尾'

        }

        const mangseng = {...skillOne, ...skillTwo, ...skillThree, ...skillFour};

33 正则命名分组

let str = '<a href="http://www.atguigu.com">尚硅谷</a>';

        //分组命名

        const reg = /<a href="(?<url>.\*)">(?<text>.\*)<\/a>/;

        const result = reg.exec(str);

        console.log(result.groups.url);

        console.log(result.groups.text);

34 正则断言

        let str = 'JS5211314你知道么555啦啦啦';

        //正向断言

        const reg = /\d+(?=啦)/;

        const result = reg.exec(str);

        //反向断言

        const reg = /(?<=么)\d+/;

        const result = reg.exec(str);

35 正则dotAll模式  /s 可以是.匹配到换行符\s

//dot  .  元字符  除换行符以外的任意单个字符

        let str = `

            <li>

                <a>肖生克的救赎</a>

                <p>上映日期: 1994-09-10</p>

            </li>

            <li>

                <a>阿甘正传</a>

                <p>上映日期: 1994-07-06</p>

            </li>`;

        //声明正则

        // const reg = /<li>\s+<a>(.\*?)<\/a>\s+<p>(.\*?)<\/p>/;

        const reg = /<li>.\*?<a>(.\*?)<\/a>.\*?<p>(.\*?)<\/p>/gs;

        let result;

        let data = [];

        while(result = reg.exec(str)){

            data.push({title: result[1], time: result[2]});

        }

36 Object.fromEntries

        const result = Object.fromEntries([

            ['name','尚硅谷'],

            ['xueke', 'Java,大数据,前端,云计算']

        ]);

     //   {name: '尚硅谷', xueke: 'Java,大数据,前端,云计算'}

        Map

        const m = new Map();

        m.set('name','ATGUIGU');

        //Map(1) {'name' => 'ATGUIGU'}

        const result = Object.fromEntries(m);

      //  {name: 'ATGUIGU'}

        //Object.entries ES8

        const arr = Object.entries({

            name: "尚硅谷"

        })

        //[['name', '尚硅谷']]

37 trim trimStart trimEnd

        let str = '   iloveyou   ';

        console.log(str);

        console.log(str.trimStart());

        console.log(str.trimEnd());

38 flat

        //将多维数组转化为低位数组

        // const arr = [1,2,3,4,[5,6,[7,8,9]]];

        //参数为深度 是一个数字

        // console.log(arr.flat(2)); 默认为1

   //flatMap

        const arr = [1,2,3,4];

        const result = arr.flatMap(item => [item \* 10]);

        console.log(result);

39 私有属性

  class Person{

            //公有属性

            name;

            //私有属性

            #age;

            #weight;

            //构造方法

            constructor(name, age, weight){

                this.name = name;

                this.#age = age;

                this.#weight = weight;

            }

            intro(){

                console.log(this.name);

                console.log(this.#age);

                console.log(this.#weight);

            }

        }

40 正则matchAll

 let str = `<ul>

            <li>

                <a>肖生克的救赎</a>

                <p>上映日期: 1994-09-10</p>

            </li>

            <li>

                <a>阿甘正传</a>

                <p>上映日期: 1994-07-06</p>

            </li>

        </ul>`;

        //声明正则

        const reg = /<li>.\*?<a>(.\*?)<\/a>.\*?<p>(.\*?)<\/p>/sg

        //调用方法

        const result = str.matchAll(reg);

        for(let v of result){

            console.log([...v]);

        }

41 Promise.allSettled

    //声明两个promise对象

        const p1 = new Promise((resolve, reject)=>{

            setTimeout(()=>{

                resolve('商品数据 - 1');

            },1000)

        });

        const p2 = new Promise((resolve, reject)=>{

            setTimeout(()=>{

                resolve('商品数据 - 2');

                // reject('出错啦!');

            },1000)

        });

        //调用 allsettled 方法

         const result = Promise.allSettled([p1, p2]);

        // const res = Promise.all([p1, p2]);

42 ?. 对象读取可以不会报错

        function main(config){

            // const dbHost = config && config.db && config.db.host;

            const dbHost = config?.db?.host;

                }

43 动态import加载

export function hello(){

    alert('Hello');

}

const btn = document.getElementById('btn');

btn.onclick = function(){

    import('./hello.js').then(module => {

        module.hello();

    });

}

44 BigInt

        // let n = 521n;

        // let n = 123;

        // BigInt(n)

        //大数值运算

        let max = Number.MAX\_SAFE\_INTEGER;

        console.log(max + 2); //不能运算

        console.log(BigInt(max) + BigInt(2))