## let和const：

1. let有哪三点需要注意的？
   1. 无变量提升 ②需要先声明再使用（存在一个暂时死去）③相同作用域内不能重复声明 ④如果外层声明了变量，内层用let重新声明，那么变量属于内层。所以变量使用仍然应在声明之后，否则报错。
2. const和let有哪些异同？如何让const变量的属性也不让变？Object.freeze怎么用？
   1. 同：都有作用域、相同作用域内不能重复声明、无变量提升
   2. 异：1，const~~声明的数值变量不能变~~本质是变量指向的内存地址保存的数据不变（数值，对象地址、数组地址）,2，const的声明和赋值必须一起
   3. 变量的属性也不让变？const声明时，将对象冻结，用Object.freeze const foo = Object.freeze({ }); 这种冻结在严格模式下生效，一般模式下不生效
   4. Object.freeze，用于
3. es6有几种变量声明方法，分别是什么？

function var let const import export 一共6种方法

## 字符串扩展：

1. ES5中，如何用unicode码表示一个字符？在es6中做了什么改进？为什么要做这个改进？
   1. es5中，unicode码表示一个字符的方法：\uxxxx 的方式表示一个字符，可以表示\u0000到\uFFFF之间的字符，超过FFFF的只能用双字节来表示
   2. es6做的改进，\u{20BB7}
   3. 这样做可读取码点超过\uFFFF的字符
2. ES5是以什么格式存储字符的，每个字符多少字节？对于4字节的字符，在ES6中是用哪一个新增的方法处理的？
   1. ：UTF-16 的方式，存储字符，一个字符2字节。
   2. 4字节的字符，es6中codePointAt方法处理
3. codePointAt返回的是码点是多少进制的？如何转化为16进制？
   1. codePointAt返回的码点是十进制
   2. 转化为16进制，toString一下
4. 如何解决codePointAt在使用时如果有4字节字符，码点不对的问题？
   1. 用for…of循环

**let** s **=** '𠮷a'**;**

**for** **(let** ch **of** s**)** **{**

console**.**log**(**ch**.**codePointAt**(**0**).**toString**(**16**));**

**}**

1. 如何检测一个字符是2字节还是4字节？请写函数is32Bit(c).

**function** is32bit**(**ch**){**

**return(**ch**.**codePointAt**(**0**)>** 0xFFFF**);//测试有效**

**}**

1. es5如何从码点返回一个对应的字符串？这个方法有什么局限？es6又是如何实现的？
   1. es5中，码点返回字符串，formCharCode
   2. 局限：不能返回32位的utf-16字符
   3. es6实现：fromCodePoint实现
2. 简述一下startsWith（param1，param2），endsWith（param1，param2）和includes（param1，param2）用法？
   1. s.startsWith（str,index） 从字符串s的index位置开始，向后找str，能马上找到返回true，否则false
   2. s.endsWit(str,index) 从字符串s的index-1位置开始，向前找str，能马上找到返回true，否则false
   3. s.includes（str,index）从是的index位置开始找，能否找到str，能找到返回true，不能，返回false
3. ES6将字符转化为为码点的方法 codePointAt ，将码点转化为字符 fromCodePoint
4. 如何将字符串重复n次，用的是什么函数？此函数的param取正整数，非正整数，字符串，布尔值时，分别表示重复了多少次？
   1. 重复n次：repeat函数。使用方法：str.repeat(count) str重复count次。
   2. 浮点数时，count会取整；非负数字符串时，count=0；布尔值true时，count=1；布尔false时，count=0 ；负数或负数字符串是时，报错
5. 头部补全和尾部补全，用的是什么方法？用补全的方法，写一个例子，取当天日期，格式是‘yyyy-mm-dd’
   1. 头部补零：s.padStart(len,str)：将字符串s头部用str补充，直到长度是len，返回新字符串
   2. 尾部补零：s.padEnd(len,str)：将字符串s尾部用str补充，直到长度是len，返回新字符串

**function** getToday**(){**

**var** now **=** **new** Date**();**

**let** year **=** now**.**getFullYear**();**

**let** month **=** **(**now**.**getMonth**()+**1**+**''**).**padStart**(**2**,**'0'**);**

**let** day **=** **(**now**.**getDate**()+**''**).**padStart**(**2**,**'0'**);**

console**.**log**(**year**+**'-'**+**month**+**'-'**+**day**);**

**}**

1. 反引号是怎么用的？用在什么场合？（两个基本使用方法）
   1. ` `
   2. 场合：换行和赋值 换行时，可在两个反引号中写html代码。赋值时使用美元符+花括号的形式，如：${ param }
2. 数字转化为字符串有哪几种方法？（三种）

toString()方法： num.toString() num为null或者undefined时，会报错

String()方法: String(num) num为null时，转化为“null”，为undefined时，转化为“undefined”

num+” ” : 结果同String( )方法

## 正则的扩展：

1. ES5中正则表达式的两种声明方式是什么？分别用两种方式，声明正则表达式 /xyz/ig
   1. 两种声明方式：直接写表达式var re = /xyz/ig; 等价于var re = new RegExp(/xyz/ig) ;使用new方法，传入一个字符串和修饰符。var re = new RegExp(‘xyz’,’ig’)；
2. ES6对RegExp函数做了哪些修正，这给正则的声明带来了什么好处？

当RegExp的第一个参数是正则时，可以有第二个参数，且第二个参数会覆盖正则中的修饰符。new RegExp(/xyz/ig,’i’); 的修饰符是u const r1 = /hello/u r1.unicode

1. ES6用什么修饰符处理码点大于\uFFFF的unicode字符？什么属性可以判断正则是否用了此修饰符？这个修饰符最基本用法是哪些地方（5个），分别给个例子。写一个codePointLength函数返回字符串的真实长度。
   1. es6用u修饰符处理码点大于\uFFFF的Unicode字符。
   2. unicode属性能判断，是否使用了u字符。用法如下：const r1 = /xy/u; r1.unicode;结果为true表示使用了u修饰符，为false表示没有。
   3. unicode使用的地方：一：点字符（. 识别码点大于0xFFFF时会失效），二：{ }花括号表示法 ，三：量词，四：预定义模式（\S能匹配所有非空字符，码点大于0xFFFF的除外） 五：i修饰符
2. 什么是粘性匹配？粘性匹配的修饰符是什么？它和g修饰符的区别在哪里？哪个属性可以判断是否使用了粘性修饰符？
   1. 粘性匹配：（sticky）必须从剩余的第一个位置开始匹配，修饰符是y。
   2. g修饰符是全局匹配，只要剩余位置存在匹配即可。
   3. sticky属性可判断粘性修饰符。为true时，存在y修饰符，为false时，不存在y修饰符。
3. lastIndex是什么？在粘性修饰符和g修饰符中有什么用？
   1. lastIndex是正则下一次匹配的位置。在y和g修饰符中，都能通过修改lastIndex的值，指定下一次匹配的位置。在粘性模式下，lastIndex开始值是0
4. 用粘性修饰符匹配“a1a2a3”，得到[a1,a2,a3]。写出计算过程。

**let** s1 **=** "a1a2a3"**;**

**let** arr **=** s1**.**match**(/a\d/gy);//[a1,a2,a3]**

1. 如何返回正则/Abc/gi的正文和修饰符。正文返回的是什么？修饰符返回的又是什么？
   1. 正文：source属性. **/Abc/gi.**source //”Abc”
   2. 修饰符：flags属性 **let** flags **=** **/Abc/gi.**flags //’gi’
2. 修饰符中，点号有什么局限，如果点号要匹配所有字符，该使用什么修饰符解决？如果正则中用了此修饰符，有什么属性能判断。
   1. 点号不能匹配码点超过0xFFFF的字符, 用u修饰符解决。点号无法匹配行终止符（断行、回车、行分隔、段分隔符），可用s修饰符解决。
   2. 如果点号代表一切是，可用dotall属性判断是否在这种模式，返回true或false

/foo.bar/s.dotAll 结果就是true

1. 什么是先行断言和后行断言，它们分别的正则表达式是怎样的？分别写正则，识别“100% percent”和“$99 is not cheap”
   1. 先行断言：x在y前面才匹配，基本js语法就支持。语法格式/x(?=y)/
   2. 后行断言：x在后面，才匹配,es2018引入。写法是：/(?<=y)x/
2. 正则表达式如何匹配符合Unicode某种属性的所有字符？如何匹配所用空格和十进制字符？
   1. 匹配符合unicode属性的所有字符：/\p{propertyName =paropertValue}/
   2. 匹配所有十进制：**const** regexDecimal **=** **/\p{Decimal\_Number}/u;**
   3. 匹配所有空格：**const** regexSpace **=** **/\p{White\_Space}/u;**
3. 什么是具名组匹配？这种匹配的写法是怎样的？匹配‘2018-11-02’，返回y，m和d，请尝试用解构赋值的方式获取y，m，d的值。用replace将日期改成2018/11/02
   1. 具名组匹配：该怎么分就怎么分，分完后，左括号加上？+尖括号+name。例子：const regDate = /(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/
   2. 解构赋值，获取y,m,d的值：

**const** regDate **=** **/(?<year>\d{4})-(?<month>\d{2})-(?<day>\d{2})/;**

**let** str **=** '2018-08-20'

**const** matchObj **=** regDate**.**exec**(**str**);**

**let** **{**groups**:{**year**,**day**,**month**}}** **=** matchObj**;**

console**.**log**(**year**,**month**,**day**);** //2018 08 20

* 1. replace，将日期转化为2018/08/20模式：需要在字符串替换中引用具名组，方式是$<组名>

**let** transDay **=** str**.**replace**(**regDate**,**'$<year>/$<month>/$<day>'**);**//"2018/08/20"

1. 正则表达式内，如何使用组名？结合ES5，使用组名的两种方法是什么。匹配’abc!abc!’，进行实例分析。
   1. 正则表达式内，引用组名，使用\k<组名>
   2. 使用具名组时，\k<组名>和\1,\2的方式，都行
   3. 匹配‘abc!abc’的方法：

**let** str\_t **=** 'abc!abc!ab'**;**

**let** reg\_twice **=** **/(?<tw>[a-z]{3})!\k<tw>/;**

console**.**log**(**reg\_twice**.**test**(**str\_t**));** //true

1. 遍历器转化为数组的两种方法？
   1. Array.from(arrLike,mapFn,thisArg)
   2. ...运算符

## 数值的扩展：

1. 二进制和八进制分别如何表示？使用二进制和八进制表示10。如何将二进制和八进制转化为十进制？
   1. 二进制：0b1010 八进制：0o14
   2. Number，二进制或八进制转化为十进制 Number(0b1010)
2. Number.isFinite( )和Number.isNaN( )如何用，他们和isFinite()和isNaN的差别?
   1. Number.isFinite( )是否无限，Number.isNaN( )是否为NaN
   2. Number.isFinite(para )直接判断para满足条件否，isFinite(para)会将para先Number一下，再看是否满足条件。
3. Number,parseInt()、Number.parseFloat()与parseInt()和parseFloat()的区别是什么？
   1. ：Number.parseInt()和parseInt（）等价 同理，Number.parseFloat()和parseFloat（）也是等价
4. 如何判断一个数是否是正整数？Number.EPSILON是什么？它的应用场景又是什么? 写一下函数acceptError(left,right) 返回值是2的-50次方
   1. 判断一个数是否正整数：Number.isInteger(para)
   2. Number.EPSILON 是一个常量。2的-52次方。
   3. 应用场景：浮点数计算的误差范围

function **acceptError**(left,right){

return Math.**abs**(left - right) < **Number**.EPSILON\*Math.**pow**(2,2);

}*//精度是2的-50次方*

let resError = **acceptError**(0.1+0.2,0.3);

1. js能表示的数的上限和下限是多少？最大上限整数和最小下限整数分别是多少？在上下限之间的安全整数又如何表示？如何验证某个计算结果是安全整数内？
   1. JS能表示的下限和上限：2^-53到2^53之间的数
   2. 最大上限整数：Number.MAX\_SAFE\_INTEGER 最小下限整数： Number.MIN\_SAFE\_INTEGER
   3. 上限和下限质检的数：Number.isSafeInteger()
   4. 验证是否在安全数内：

function **trusty**(left,right,result){

if(**Number**.**isSafeInteger**(left)&&

**Number**.**isSafeInteger**(right)&&

**Number**.**isSafeInteger**(result)

){

return result;

}

throw new **RangeError**('Operation cannot be trusted!');

}

1. Math.trunc(val)是什么意思？如何判断某数的正、负、0？如何计算一个数的立方根？如何算一个数有多少个先导0？如何计算两数的乘积？如何将64位双精度浮点数转化为32为单精度浮点数？如何求几个数的平方和的平方根？
   1. Math.trunc(val)的意思是：去除val的小数部分。运算前，会先把val Number一下
   2. 判断正负0的方法：Math.sign(val) 运算之前，也会把val Number一下，返回+-1,0
   3. 计算立方根方法：Math.cbrt(val) 返回val的立方根
   4. 计算某个数的先导0：Math.clz32(val) 返回先导0的数目
   5. 计算两个数的乘积：Math.imul(num1,num2) 返回乘积的值
   6. 将64位双精度浮点，转化为32位单精度浮点Math.fround
   7. 几个数的平方和的平方根：Math.hypot(val1,val2,val3) Math.hypot(1,2,2,4)， 返回5
2. 如何计算ex-1?如何计算1+x的自然对数？如何返回10为底x的对数?如何返回2为底，x的对数？如何进行指数计算，如2的3次方？
   1. 计算 ex-1 ： Math.expm1（x）
   2. 计算1+x自然对数 ： Math.log1p(x)
   3. 计算10位底x对数 ： Math.log10(x)
   4. 返回2为底x对数 ： Math.log2(x)
   5. 指数运算符 \*\* ：console.log(4 \*\* 3);//64

## 函数的扩展：

1. 说一下，函数参数默认值声明时，几点注意事项？（3点）

函数声明注意事项：

* 1. 函数体内，不能用let或const再次声明
  2. 使用了函数默认值时，不能用重名参数
  3. 参数的默认值是惰性求值，每调用一次，参数重新计算一次

1. 分析一下function m1({x=0,y=0} = {}) { return [x,y]; }和function m2({x,y} = {x:0,y:0}) { return [x,y]; }，传值不同时，分别输出什么?
   1. m1和m2都对参数设了默认值。m1的参数默认是{},m2是{x:0,y:0}
   2. m1设置了对象的解构赋值，m2没有。
   3. 当不传值，传{x:0,y:0}或者传其他参数时，效果一样。但是传一个参数时，不一样
2. 在函数调用中，传值时，什么样的参数能省略，什么样的参数不能省略？在定义函数时，如何表明函数的某个参数是能省略的？
   1. 尾部的默认参数能省略
   2. 非尾部的默认参数不能省略
   3. 显示传入undefined，触发此参数等于默认值 f(1,undefined,2)
3. 函数的length属性时如何计算的？能否包括默认参数和rest属性？
   1. length:预期传入的参数个数。要用参数的总个数-默认参数和它后面参数一起，总参数个数
   2. 不包括默认参数和rest属性
4. 函数声明初始化时，什么是作用域？
   1. 当有默认参数时。函数声明初始化时，会形成单独的作用域。初始化结束，作用域消失。
5. 具名函数和赋值的匿名函数，name属性的值是多少？
   1. 具名函数的name属性是function后的名称
   2. 赋值的匿名函数，name属性是变量名
   3. 构造函数，返回的是anonymous
6. 箭头函数和解构赋值一起用时，const full =（{first,last}）=>(first + ‘-‘ +last)；full({first：‘chen’,last:’ke’}) 调用结果是啥？
   1. 调用结果：chen-ke 参数传的是个对象，并且用了最基本的对象解构赋值
7. 箭头函数的this指向的是什么？箭头函数自身有没有this？arguments,super,new.target，这些在箭头函数内部存在吗？什么是函数的管道机制？
   1. 箭头函数this指向：箭头函数定义生效时，所在的对象。箭头函数无this，this指向的是外层变量。
   2. arguments，super，new.target在箭头函数内部不存在的。
   3. 管道机制：前一个函数的输出是后一个函数的输入
8. 什么是尾用调用？尾用调用优化的原则是什么？
   1. 尾用调用：一个函数的最后一步调用了另一个函数。
   2. 尾用调用的原则：内层函数不再用到外层函数的内部变量时，才能进行尾调用优化
9. 用尾递归实现阶乘和斐波拉切的累加和。
   1. 尾递归实现阶乘：

function **factorial**(n,total=1){

if(n == 1) return total;

return **factorial**(n-1,n\*total);

}

* 1. 尾递归实现斐波拉契数列：

function **fb**(n,ac1=1,ac2=1){

if(n <= 1){

return ac2;

}

return **fb**(n-1,ac2,ac1+ac2);

}

将普通函数改成尾递归，只需要将中间变量改写成函数的参数。

## 数组的扩展：

1. 什么是扩展运算符？数组运算符通常的应用场景是？
   1. 扩展运算符：三个点，将数组转化为参数序列，rest参数的逆运算
   2. 数组运算符应用场景：主要用在函数的调用（参数部分）
2. 用扩展运算符求一个数组的最大值。将数组arr1[1,2,3]，加到数组[‘a’,’b’,’c’]的尾部
   1. 扩展运算符，求数组最大值：

let maxVal = Math.**max**(...[12,45,8,]);

* 1. 将数组arr[1,2,3]，加到数组arr1[‘a’,’b’,’c’]

let newArr = arr1.**push**(...arr);

1. 用ES5的方法实现数组复制。3种方法，用ES6实现数组复制。以上方法是浅拷贝还是深拷贝？如何实现深拷贝？
   1. 用ES5方法实现数组复制（一维数组）：

let arrTwo = [2,3,4,5];

let arrTwo2 = arrTwo.**concat**();

一维数组深拷贝。当arrTwo改变时，arrTwo2没变

* 1. 用ES6实现数组复制：

方法一：

let arrThree = [2,3,4,5];

let arrThree1 = [];

arrThree1.**push**(...arrThree);

一维数组深拷贝。

方法二：

let [...arrThree2] = arrThree;

一维数组深拷贝。

方法三：

let arrThree3 = [...arrThree]

一维数组深拷贝。

* 1. 实现深拷贝方法：
     1. 多维数组深拷贝

function **deepCloneArr**(arr){

let out = [],len = arr.length;

for(let i = 0; i < len; i ++){

if(arr[i] instanceof **Array**){

out[i] = **deepCloneArr**(arr[i]);

}else{

out[i] = arr[i];

}

}

return out;

}

* + 1. 包含对象数组深拷贝

function **deepCloneArrObj**(arr){

let out = [],len = arr.length;

for(let i=0; i < len; i ++){

if(arr[i].constructor === **Array**){

out[i] = **deepCloneArr**(arr[i]);

}else if(arr[i].constructor === **Object**){

let str = JSON.**stringify**(arr[i]);

out[i] = JSON.**parse**(str);

}else{

out[i] = arr[i];

}

}

return out;

}

测试有效

* + 1. 对象深拷贝方法

function **deepCloneObj**(obj){

var newobj;

if(typeof obj != "object"){

newobj = obj;

}

else{

newobj = obj.constructor == **Array** ? [] : {};

var str = JSON.**stringify**(obj);

newobj = JSON.**parse**(str);

}

return newobj;}

1. 用ES5,ES6的方法，对三个数组arr1，arr2，arr3进行数组合并。
   1. ES5实现三个数组合并

let arr = [].**concat**(arr1,arr2,arr3)

* 1. ES6实现三个数组合并

方法一：

let [...arr] = [...arr1,...arr2,...arr3];

方法二：

let arrOne = [...arr1,...arr2,...arr3];

1. 扩展运算符可以和解构赋值相结合，生成数组。这种写法，得到的first，middle，last，分别是多少？const [first, ...middle, last] = [1, 2, 3, 4, 5]，这种写法呢？const [first, ...rest] = [1, 2, 3, 4, 5]
   1. 扩展运算符和结构赋值相结合

let [first,...last] = [1,2,3,4];*//first: 1 last:[2,3,4]*

* 1. const [first, ...middle, last] = [1,2,3,4,5] 这种写法会报错。因为middle在中间。用结构赋值，给数组进行赋值时，只能放在最后一个参数
  2. const [first, ...rest] = [1, 2, 3, 4, 5] *//first: 1 last:[2,3,4,5]*

1. 字符串如何转化为数组，并求其长度。
   1. 字符串转化为数组

方法一：ES5的for … of写法

let strFour = 'hello,world';

let arrFour = [];

for(let i of strFour){

arrFour.**push**(i);

}

*//["h", "e", "l", "l", "o", ",", "w", "o", "r", "l", "d"]*

方法二：ES6的扩展运算符写法

let strSix = 'telev';

let arrSix = [...strSix]; //["t", "e", "l", "e", "v"]

* 1. 求字符串长度： 字符串部署了Iterator接口

strSix.length 或者 [...strSix].length

1. 字符串如何翻转？用es5和es6的方法实现。两种方法求出来的值都是准确的吗？
   1. 字符串如何翻转？

方法一：ES5使用了数组的reverse方法，字符串-数组-字符串

let strSeven = 'myString';

let strSeven2 = strSeven.**split**('').**reverse**().**join**('');

方法二：ES6使用了扩展运算符，一步到位

let strSeven3 = [...strSeven].**reverse**().**join**('');

1. 扩展运算符和Iterator的接口有什么关系?请问，如何实现页面上div的数组？
   1. 扩展运算符和Iterator接口关系：部署了Iterator接口，就能用扩展运算符
   2. 页面div的数组：

let divs = document.**getElementsByTagName**('div');*//HTMLCollection(6) [div, div, div, div, div, div]*

let divArr = [...divs];

1. 请定义一个map对象，并遍历它的key和value，用数组形式返回。
   1. 用map创建一个对象：

let mapItera1 = new **Map**([

[5,'one'],

[9,'two'],

[13,'three']

]) //  {5 => "one", 9 => "two", 13 => "three"}

* 1. 数组形式返回，map对象的key

let mapKeys = [...mapItera1.**keys**()];

* 1. 数组形式返回，map对象的value

let mapValues = [...mapItera1.**values**()];

1. Array.from( )是用来干什么用？它的第二个参数是干嘛用的？写个例子，将类数组[1,,3,,4]中,空出来的数，转化为0
   1. Array.from将类似数组或者可遍历对象（有Iterator接口）转化为数组。

看下面几个例子：

例子一：类似数组转化为数组，并进行过滤（数组的filter方法）

let div2 = document.**querySelectorAll**('div');

let arr2 = **Array**.**from**(div2).**filter**(divEl => {

return divEl.innerHTML.length > 2

})

例子二：类似数组转化为数组，同时对数组进行处理（Array.from接受第2个参数）

let arrLike = {

0:'a',

1:'b',

2:'c',

length:2*//必不可少*

}

let arr3 = **Array**.**from**(arrLike,(item,index) => {

return item+item+index

}) // ["aa0", "bb1"]

部署了Iterator接口的数据结构都能遍历。目前发现的有这几个：字符串，数组，类似数组的对象，set和map结构

* 1. Array.from的第二个参数，用来处理转化之后数组中的单个数据
  2. 将类数组中的空出来的位置，用0补上

let arrFour = [1,,2,,3];

let arrFour1 = **Array**.**from**(arrFour,(item)=>{

*// return item === undefined ? 0:item; //写法一*

return item || 0;*//写法二*

})

1. 如何将一组值转化为数组（两种方法，ES5和ES6分别一种）？

ES5中有new Array(para)方法，ES6中有array.of(para)方法

无参数，参数个数大于1或参数为空时，两种方法效果相同。参数个数为1时，效果不同。array.of(para)正是为了规避这个，才出现的。

参数个数大于1时：

let arrFive = new **Array**(3,12);

let arrFive1 = **Array**.**of**(3,12);

*//都是 [3, 12]*

let arrFive2 = new **Array**();

let arrFive3 = **Array**.**of**();

*//都是 []*

参数个数为1时：

let arrFive4 = new **Array**(3);*//[,,]*

let arrFive5 = **Array**.**of**(5);*//[5]*

1. copyWithin( ) 方法如何用？详细解释下，它的几个参数。
   1. copyWithin( ) 的用法：数组中将指定位置的元素覆盖其他位置的元素。会改变被操作的数组
   2. 三个参数的含义：

target（必须）：从此处开始替换

start（可选）：被复制的起始位置

end（可选）：被复制的结束位置(复制时，不包括此位置)。复制的字符串长度是end-start

let arrSix = ['a','b','c','d','e','f','g'];

arrSix.**copyWithin**(2,0);*//["a", "b", "a", "b", "c", "d", "e"]*

*//分析：从arrSix[2]开始，到后面的元素，被arrSix从第一个元素开始覆盖*

let arrSix1 = ['a','b','c','d','e','f','g'];

arrSix1.**copyWithin**(2,3,5);*//["a", "b", "d", "e", "e", "f", "g"]*

1. 数组的find和findIndex如何用？它们的返回值是多少？可以接收几个参数？
   1. find：返回数组中满足条件的第一个数，没有就返回undefined

let arrSeven = [10,1,4,12,-6,-20];

let itemSeven = arrSeven.**find**(item => {

return item < 0;

});*//-6*

find可接收的参数是函数。函数有三个参数.function(currentVal,currentIndex,sourceArr)

* 1. findIndex:用法和find是一样，传的参数也类似。但是，findIndex返回的是index值。 都不符合，就返回-1
  2. find和findIndex都可接收第二个参数。绑定回调函数的this

例子：

function **fnV**(v){

return v>this.age;

}

let person = {name:'John',age:21};

[10,13,34,16].**find**(fnV,person);*//34*

1. fill函数如何用?可以接收几个参数，每个参数是什么意思？
   1. fill(val,start,end)的用法：按给定的值，将数组指定位置的值进行覆盖

let arrFill = new **Array**(3);

arrFill.**fill**('a');*//['a','a','a']*

arrFill.**fill**('b',1,2);*//['a','b','a']*

* 1. val：必填，填充的值

start：非必填，起始位置，默认是0

end：非必填，结束位置的后一位，默认是数组最后一位的下一位

1. 如何遍历数组的key，value，以及key:value键值对。用什么属性，使用这个属性，用2种方式来实现数组的遍历。
   1. 遍历数组的key：arr.keys() 遍历数组的arr.values() 遍历key：value 值对，arr.entries()
   2. 这三个属性得到的结果都是一个Iterator对象，要用for…of循环，进行遍历。看例子：

let keysArrEight = arrEight.**keys**();

for(let item of keysArrEight){

**console**.**log**(item);

}

1. includes表示什么意思？返回值是什么？可以接收几个参数？
   1. includes表示是否包含某个值
   2. 如果包含，返回true，否则false
   3. 接受2个参数，第一个是值，第二个是起始搜索的位置，默认是0

## 解构赋值：

1. 数组解构赋值的匹配模式是怎样的？
   1. 数组匹配模式：左边和变量和右边数组的数据，按顺序匹配
2. 如何完全解构，获取数组的第一个和最后一个元素 [1,2,3,4]?
   1. 完全解构获取第一个和最后一个元素。 let [x21,,,x22] = [1,2,3,4]

(中间不需要的参数，全部空出来)

1. 如何用不完全解构，获取数组的第一个和第二个元素[1,2,3,4]？
   1. 不完全解构。let [x31,x32] = [1,2,3,4]
2. 数组解构时，默认值是表达式时有什么特点？
   1. 默认值是表达式时，惰性赋值。只有取到默认值时，表达式才会执行。不取默认值时不执行。
3. 如何触发解构赋值的默认值（两种方式）？
   1. 左边的变量，设了默认值，同时，变量在右边没找到
   2. 左边的变量，设了默认值，同时，变量在右边找到了，但值为undefined
4. 对象的解构赋值的内部机制是什么？
   1. 左边的变量，在右边找相应的key值。如果找到了，将此key的value值，赋给左边对应的变量
5. 对象解构赋值，默认值生效的条件是什么？
   1. 左边的变量在右边找不到 ②在右边找到了，但是是undefined
6. 对象解构赋值时，先声明再赋值怎么做？
   1. 用圆括号，将赋值的语句括起来。如图：

({x4,y4} = {x4:1,y4:21})

1. 如何将Math.floor和Math.abs解构到floor和abs上？
   1. 解构赋值的方式，解构到floor和abs上，如图：

let {floor,abs} = Math;

1. 如何用对象解构的方式，获取数组【1,2,3,4】的第一位和最后一位数值？
   1. 数组也可以用对象解构的方式，来进行赋值。如图：

let arr1 =[12,1,23,3,45,2,122,56,54];

let {0:firstE,[arr1.length-1]:lastE} = arr1;

1. 字符串为什么可以进行解构赋值？解构一个数组，试着解构下数组的length属性
   1. 字符串有length属性，能转化为类似数组。
   2. let [strA,strB,strC]='helcabc'
   3. 解构数组的length属性： let {length:len} = arr1 数组是对象，并且数组对象有length属性
2. 试着分析下下面两个函数，当传值不同时，他们分别输出什么？

function **moveFn**({x =0,y=0}={}){

return [x,y];

}

function **moveFn2** ({x,y} = {x:0,y:0}){

return [x,y];

}

* 1. 两个函数都运用了函数参数赋值和参数的解构赋值。
  2. moveFn函数的默认参数是{ }，moveFn2的默认参数是{x:0,y:0}
  3. moveFn参数的解构赋值中，x和y的默认值是0，moveFn2中x和y无默认值

1. 列举下变量解构赋值的用途。（7种）
   1. 数据的交换

let x1 = 12,y1 = 14;

[x1,y1] = [y1,x1];

* 1. 函数参数的解构

function **fnPara**({x,y,z}){

return x\*y+z;

}

let resPara = **fnPara**({x:2,y:3,z:10})

* 1. 函数结果返回，返回多个值

function **returnPara**(pa){

let x = pa+1;

let y = pa\*2;

let z = pa\*3;

return[x,y,z]

}

let [first,second,third] = **returnPara**(3);

* 1. 提取json值

let datas = {

url:'www.baidu',

time:'2018-8-8 17:59:58',

data:[12,23,34] }

let {url:target,time,data:number} = datas;

target，time和number的值，对应datas的url，time和data

* 1. 函数参数的默认值

在参数定义里，定义参数默认值。

* 1. 遍历Map结构

for(let [key] of map){

}

for(let [,value] of map){

}

* 1. 输入模块的指定方法：

const { SourceMapConsumer, SourceNode } = **require**("source-map");