变量的声明：不存在变量提升

一：let命令

1、是声明变量的一种方式跟var差不多

2、是一个块级作用域声明的变量只在let块内有用

3、不允许重复声明变量

If(true){ 判断

Let a = 10;

Console.log(a) //10

}

Console.log(a) //a is not defined

1. 用于循环

For(let i=0;i<10;i++){ 如果i是var 声明的在全局作用域都有效

如果是let声明的只在本轮循环有效，每循环一次i 都是一个新的

}

设置循环变量的部分是父作用域，而循环体是一个子作用域

二：const命令

1. const声明一个只读变量，一旦声明，常量值就不可以改变
2. 一旦声明就要立即赋值，不能到以后赋值。
3. 变量声明，也是不会提升。一样不会重复声明
4. 只在声明所在的块级作用域内有效

三：变量的解构

1. 数组的结构：
2. 只要等号左右两边的模式相同，右边的变量就会被赋值对应的值
3. 只要结构不成功，变量的值就是undefined；
4. 如果等号右边的不是可便利的结构，那么就会报错。

Es5：let a=1；let b=2；let c = 3；

Es6：let[a,b,c] = [1,2,3]

1. 对象的结构：
2. 数组的元素是按照次序排列的，变量的取值由它位置决定
3. 对象没有次序，但是 变量必须要与属性同名，才可以取到正确的值
4. 也就是先找到同名属性，然后再进行赋值，真正被赋值的是后者，不是前者

Es6：let{foo,baz} = {foo:”aaa”,baz:”bbb”}

四：字符串的拓展

1. includes(n) //表示是否找到了参数n这个字符串，返回boolen
2. Startswith() //表示参数字符串是否在原字符串头部，返回boolen
3. Endswith() //表示参数字符串是否在原字符串尾部，返回boolen
4. Repeat() //返回一个新字符串，表示原字符串循环多少次 ，如果是小数会取整，0-1之间的小数等于0；如果字符串会先转换成数字，如果是负数会报错
5. Padstart() //头部补全
6. 如果某个字符串长度不够，会在头部补全
7. 如果用来补全的字符串与原字符串长度之和超过了指定的最小长度，会截取超出位数的补全字符串
8. 如果省略第二个参数，默认使用空格补全长度

Padend() //尾部补全

1. Boolen
2. boolen(“”); //false
3. Boolen(NaN); //false
4. Boolen(undefined); //false
5. Boolen(null); //false
6. Boolen(0); //false
7. Boolen(-1); //true
8. Boolen(!true) //false

五：数值的拓展

1. Number.isFinite() //检查一个数值是否为有限的
2. Number.isNaN() //检查一个数值是否为NaN
3. 将全局方法parseInt(),parseFloat()移植到Nuber上边,但是行为保持不变
4. Number.parseint() === parseint //true
5. Number.parseFloat() === parseFloat//true
6. Number.isInteger() //判断一个数值是否为整数

六：Math对象的拓展

1. Math.trunc() //去除一个数的小数部分，返回一个整数
2. Math.sign() //判断一个数到底是正数还是负数还是0，非数值会先转换为数值
3. 参数为正，返回1
4. 参数为负返回-1
5. 参数为0返回0
6. 参数为-0返回-0
7. 其他值，返回NaN
8. Math.cbrt() //计算一个数的立方根
9. Math.hypot() //计算所有参数的平方和的平方根
10. Math.log10() //以10为低的x的对数，如果x小于0，返回NaN
11. Math.log2() //以2为低的x的对数，如果x小于0，返回NaN

七：运算符

1. \*\*； 2\*\*3 //8 n\*\*m 也就是n的m次方

八：数组的拓展

1. Array.from() //将两类对象转为真正的数组
2. Array.of() //将一组数值转换为数组
3. Includes() //反回一个boolen，表示某个数组是否包含给定的值

第二个参数表示搜索的起始位置，默认为0，如果为负值。则表示为倒数，如果大于数组长度，则会重置从0开始

1. Copywithin() //在数组内部，将制定位置的成员复制到其他位置，然后返回当前数组 接受三个参数
   1. Target(必选) //从开始位置替换数据
   2. Start(可选) //从当前位置读取数据，默认为0，如果为负数，表示倒数
   3. End(可选) //到该位置前停止读取数据，默认等于数组长度，如果为负，表示倒数
2. find //找到第一个就停止
3. Find index //返回找到的第一个符合条件的数组成员，如果都不符合返回-1
4. Fill() //使用给定值，填充一个数组，用于空数组的初始化，数组中已有元素都会被抹去 第二个和第三个参数，用于制定填充的起始位置和结束位置。
5. Entries() //是对键值对的遍历

For(let index of[‘a’,’b’].keys()){

Console.log(index)

}

1. Keys() //对建名的遍历

For(let index of[‘a’,’b’].keys()){

Console.log(index)

}

1. Values() //对键值的遍历

For(let index of[‘a’,’b’].keys()){

Console.log(index)

}

1. for...of... //循环
2. 可以代替数组实例的for Each方法
3. 原生js的for..in循环，只可以获取对象的键名，不能获取键值 for..of可以

九：函数的拓展

1. 箭头函数

如果箭头函数不需要传参或需要多个参数，就使用一个圆括号代替参数部分

如果箭头函数的代码块部分多余一条语句，就要使用大括号将他们包括起来，并且使用return返回

由于大括号被解释为代码块，所以如果箭头函数直接返回一个对象，必须在对象外边加上括号，否则报错。

Var f = v=>v;

等同于：var f = function(v){return v}

注意：

1. 函数内的this对象，就是定义的对象，而不是使用时的对象
2. 不可以当作构造函数，也就是不可以使用new，否则会报错。
3. 不可以使用arguments对象，该对象在函数体内不存在，如果要用，可以用rest参数代替
4. 不可以使用yield命令，一次箭头函数你能用作generator函数。

十：数据结构

1. set //类似于数组，但是成员的值都是唯一的，没有重复的值。

//本身也是一个构造函数，用来生成set数据结构。

//也可以接受一个数组作为参数，用来初始化。

1. set.prototype.constructor //构造函数，默认就是set函数。

set.prototype.size //返回set实例的成员总数

1. 分为操作方法和遍历方法
2. 四个操作方法

Add(value) //添加某个值，返回set结构本身

Delete(value)//删除某个值，返回一个boolen，表示删除成功

Has(value) //返回一个boolen，表示该值是否是set的成员

Clear() //清除所有成员，没有返回值

2、四个