1. let、const

let：块级作用域；声明不会被提升;不同于var可以多次声明一个变量，let只能声明一次。

const:“常量”，声明是必须被赋值；类似与指针，可以被改变；常量，大写变量名

示例：

var name = "name";//和var name = "name"用法一致

1. 箭头函数

var fun = () => {return true;}

1. 函数默认值

let fun = (a,b=7) = > {return a+b;}

1. 扩展运算符，三个点：...
   1. 个人理解为对象/变量操作符：可以把参数对象中的可遍历对象取出，复制到当前对象中去
   2. 等价于Object.assign() 方法： Object.assign方法用于对象的合并，将源对象（source）的所有可枚举属性，复制到目标对象（target）。Object.assign方法的第一个参数是目标对象，后面的参数都是源对象。(如果目标对象与源对象有同名属性，或多个源对象有同名属性，则后面的属性会覆盖前面的属性)。
2. 结构赋值

针对数据或对象进行模式匹配，对其中的变量进行赋值。解构的源一般是数组或对象。

示例1：let [a, b, c] = [1, 2, 3]; //a=1;b=2;c=3;

示例2：let {a, b, ...rest} = {a: 10, b: 20, c: 30, d: 40}; //a=10;b=20;rest={c:3,d=40}

1. Symbol

ES6是一种新的原始数据类型Symbol，表示唯一的值，最大的用法是定义对象的唯一属性名。

ES6数据类型除外Number，String，Boolean，Objec t，null和undefined，还添加了Symbol。

用法：由于每一个 Symbol 的值都是不相等的，所以 Symbol 作为对象的属性名，可以保证属性不重名。 --> 定义常量；唯一对象属性名。 --> 是原始数据类型不是对象，所以该对象的对应属性不能用“.”方法获取值，只能使用中括号[]获取。

1. map

Map 对象保存键值对。任何值(对象或者原始值) 都可以作为一个键或一个值。

示例：

var myMap = new Map();

var keyObj = {}; //对象

var keyFunc = function () {}; // 函数

myMap.set(0, "zero");

myMap.set(1, keyObj);

myMap.set(keyFunc, "和键 keyFunc 关联的值");

// 将会显示三个 logs。 一个是 "0 = zero" 另一个是 "1 = [object Object]"，还有一个是 "function () {} = 和键 keyFunc 关联的值"。

myMap.forEach(function(value, key) { console.log(key + " = " + value); }, myMap)

1. Set

Set 对象允许你存储任何类型的唯一值，无论是原始值或者是对象引用。 --> 数组去重。

示例：

var mySet = new Set([1, 2, 3, 4, 4]);

[...mySet]; // [1, 2, 3, 4]

并集、交集、差集 示例：

var a = new Set([1, 2, 3]);

var b = new Set([4, 3, 2]);

var union = new Set([...a, ...b]); // {1, 2, 3, 4}

var intersect = new Set([...a].filter(x => b.has(x))); // {2, 3}

var difference = new Set([...a].filter(x => !b.has(x))); // {1}

原理：set存储的唯一性；扩展运算符... ；

1. filter

filter:对数组进行过滤，返回符合要求的值组成的新数组。

语法：Array.filter(function(currentValue, indedx, arr), thisValue)

示例：

let nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

let res = nums.filter((num) => {

return num > 5;

});

console.log(res); // [6, 7, 8, 9, 10]

1. 字符判断相关方法
   1. **模板字符串 ``**

相当于加强版的字符串，用反引号 **`**,除了作为普通字符串，还可以用来定义多行字符串，还可以在字符串中加入变量和表达式。

示例：

let string = "apple";

let string1 = `Hey, can you stop angry now? I like ${string}.`;

console.log(string1);

// Hey,

// can you stop angry now? I like apple.

* 1. 判断字符是否存在

includes()：返回布尔值，判断是否找到参数字符串。第一个是需要搜索的字符串，第二个是搜索的起始位置。

startsWith()：返回布尔值，判断参数字符串是否在原字符串的头部。

endsWith()：返回布尔值，判断参数字符串是否在原字符串的尾部。

示例：

let string = "apple,banana,orange";

string.includes("banana"); // true

string.startsWith("apple"); // true

string.endsWith("apple"); // false

string.startsWith("banana",6) // true

* 1. 字符串补全

padStart：返回新的字符串，表示用参数字符串从头部（左侧）补全原字符串。

padEnd：返回新的字符串，表示用参数字符串从尾部（右侧）补全原字符串。

示例：

如果指定的长度小于或者等于原字符串的长度，则返回原字符串:

console.log("hello".padStart(5,"A")); // "hello"

如果原字符串加上补全字符串长度大于指定长度，则截去超出位数的补全字符串:

console.log("hello".padEnd(10,",world!")); // "hello,worl"

常用于补全位数：

console.log("123".padStart(10,"0")); // "0000000123"

* 1. 字符串重复

repeat()：返回新的字符串，表示将字符串重复指定次数返回。次数向下取整（正），0到-1间为0次，小于-1报错，Infinity 、NaN报错。

示例：

console.log("Hello,".repeat(2)); // "Hello,Hello,"

1. 数值

Math() 对象新增17个属性相关的静态方法。

数字取整：Math.trunc();

指数运算符: \*\*

2 \*\* 2 \*\* 3; // 256 =2^2^3=2^8

// \*\*=

let exam = 2;

exam \*\* = 2; // 4

1. 对象

ES6允许对象的属性直接写变量，这时候属性名是变量名，属性值是变量值。方法名也可以简写。

示例：

const age = 12;

const name = "Amy";

const person = {age, name}; //{age: 12, name: "Amy"}

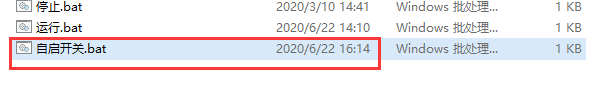
函数：

const person = { sayHi(){ console.log("Hi"); } } //es6

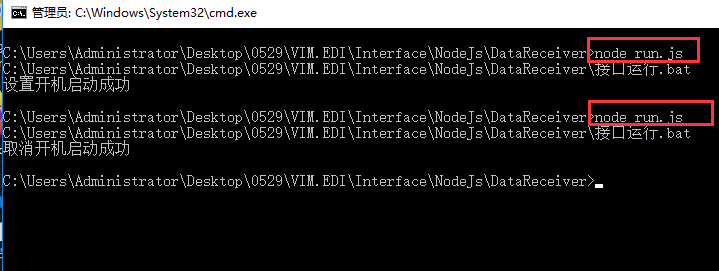
//等同于

const person = { sayHi:function(){ console.log("Hi"); } }

直接双击自启开关，或者使用下面命令行的方式。



在命令行窗口使用输入：node run.js ，出现提示：“设置开机启动成功”



在“任务管理器”--> “启动” --> 可以看到“运行.bat”，即是设置开机自启成功。若不想开机自启，在执行一次“node run.js”即可取消开机自启。

