1. 变量不能重复声明。如let a=0;let a=90;
2. let有块级作用域，var没有块级作用域。let出了本作用域后如果还被调用会报错，但是var不会（var默认为全局）。如{let a=90;}consolo.log(a);会报错。但{var a=90;}consolo.log(a)能输出a=90；
3. let不存在变量提升,var存在变量提升。如console.log(a);var a=10;是对的，因为var是全局，相当于var a=10；console.log(a); 但是console.log(a);let a=10;是错误的
4. const用于声明常量，声明后的常量值不可改（let、var可改）。1const一定要赋初始值，不然会报错，2常量一般使用大写，3const有块级作用域，如{const a=90;}consolo.log(a);会报错。4const对于数组和对象的修改，不算对常量的修改。如const TEAM={‘1”,”2”};TEAM.push=[“3”]；添加了一个字符串，是正确的
5. arguments对象。函数的形参，可以传也可以不传，若方法里没有写参数却传入了参数，该如何拿到参数呢，答案就是通过arguments来获得参数。每一个函数都有一个arguments对象，它包括了函数所要调的参数，通常我们把它当作数组使用，用它的length得到参数数量，但它却不是数组。如：function chunwan(){console.log(arguments);};chunwan(‘1’,’2’,);结果输出1,2
6. 构造函数名称首字母一般为大写
7. 解构赋值：按照一定模式从数组和对象中提取值从而对变量赋值。数组如const f3={“小”，”大”，”中”};let[xiao,da,zhong]=f3;则输出结果为xiao=”小”，da=”大”，zhong=”中”。对象如：const func={name:’zhaopi’,age:’18’,xiaopin:function(){“-------”}};let{bu,hao,de}=func;输出结果为bu=’zhaopi’;hao=’18’;de=function(){“--------”}; 注意：解构赋值的形参要一样，否则undefined
8. 模板字符串：``（1左边的按键）。优点：1内容中可以出现换行符，如html的表单，’和”单引号和双引号中不能出现换行符，如输出表单，会报错，得用拼串的方法才不会报错。
9. 模板字符串之变量拼接（${}）：如let lovest=”gaga”;let out=`${lovest}是我的偶像`;
10. 大括号里可以直接写入变量和函数作为对象的属性结方法。如let name=”爱我”;let change=function(){“-----”};const school={name,change,};
11. 箭头函数：let func=function(a,b){} === let func=(a,b)=>{} 。1注意箭头函数的this是静态的，this始终指向函数声明时所在作用域下的this的值，不会改变。2不能作为构造函数实例化对象。如let person=()=>{------}; let me=new person(-----);会报错，因为person是个箭头函数（this是静态的）。3箭头函数里不能使用arguments变量。4箭头函数可以简写。
12. 箭头函数的简写：1省略小括号。如：let add=(n)=>{return n+n};console.log(add(9));输出结果为18。可以改成：let add=n=>{return n+n};console.log(add(9)); 2省略大括号，当代码体只有一条语句时，可以省略大括号，而且return也必须省略。 如 let add=n=> n+n;console.log(add(9));输出结果为18
13. 箭头函数适合与this无关的回调、定时器、数组等方法的调用
14. 技巧：this永远都指向写有function或者带箭头的那个函数。如func=()=>{settimeout(-----),100-----};this指向func函数不是settimeout
15. 允许给函数参数赋初始值。1形参赋初始值，一般位置靠后。如function add(a,b,c=10){return a+b+c}
16. yiled：使生成器函数执行暂停。yield关键字后面的表达式的值返回给生成器的调用者(相当于return)。yield只执行右边的语句，不执行左边的语句。1、yield关键字实际返回一个IteratorResult（迭代器）对象，它有两个属性，value和done，分别代表返回值和是否完成。2、yield无法单独工作，需要配合generator(生成器)的其他函数，如next()，next()方法是可以带参数的，其中的参数就替换了上一次yield执行的结果。

function\* gen(x, y) {

        let mynum1 = yield x + y;

let mynum2 = yield mynum1 + x;

      }

      var g = gen(5, 6);

      console.log(g.next());//value:11,done:false

      console.log(g.next());//value:undefined,done:true，因为yield 左边的mynum1 没有被赋值，所以mynum1 +x为undefined

Console.log(g.next(11));//vaule：55，done：false，因为next传入的11会赋值给mynum1

1. Generator函数是ES6提供的一种异步编程解决方案。通过yield标识位和next()方法调用，实现函数的分段执行。Generator要和迭代器一起使用，这里的next充当迭代器的接口
2. rest参数
3. 拓展运算符：... 如：const tfboys=[‘王俊凯’,’王源’,’易烊千玺’]; function chunwan(){console.log(arguments);};chunwan(...tfboys);输出结果为‘王俊凯’,’王源’,’易烊千玺’。注意：用于取出参数对象中的所有可遍历属性，拷贝到当前对象之中
4. 数组的合并：es5：用concat()。es6：扩展运算符（...数组1,...数组2）。如const arr1=[...];const arr2=[...];const arr3=[...arr1,...arr2];
5. Symbol：前端的的第六种基本数据类型。特点：1值是唯一的，用来解决命名冲突的问题。let s1=Symbol(”11”);let s2=Symbol(”11”);console.log(s1===s2);输出结果为false,console.log(s1),输出结果为Symbol(”11”)而不是11。2symbol值不能与其他值进行数据计算。3symbol定义的对象属性不能使用for...in循环遍历（因此这个symbol的作用是防止某个属性被暴露出去）.4使用symbol作为其他对象的属性名时要加中括号：如[s1]，let obj1={[s1]:’我是哈哈哈’},console.log(obj1),输出结果为Symbol(”11”)：我是哈哈哈，console.log(obj.[s1]),则输出哈哈哈
6. 迭代器：迭代器是一种接口，为各种不同的数据结构提供统一的访问机制。任何数据结构只要部署iterator接口，就可以完成遍历操作了。Es6遍历命令：for..of..为循环键值item ，for...in..为循环键名index下标
7. for...in...:

let se = Symbol("我是symbol");

      let obj1 = { name: "mike", age: 15, [se]: "我是哈哈哈" };

      for (index in obj1) {

        console.log(obj1[index]);

      }//输出结果为mike，15，不会输出symbol定义的属性

1. Promise：用于解决异步编程，解决因为层层嵌套而导致的回调地狱。

1、Promise要两个参数函数，一个resolve，一个reject。

2、Promise还有一个then属性，then函数需要两个参数函数，分别是传进来的第一个参数为成功（resolve）时传进来的函数，第二个参数为失败（reject）时传进来的函数。

3、promise的all属性：promise.all包含n个promise的数组，然后重新返回一个新的promise对象。如let p1=new promise... let p2=new promise... let p3=new promise... const result=promise.all([p1])

1. Symbol、生成器函数、promise、call、map没看懂！！
2. Set（集合）。Es6提供的一个新的数据结构，类似于数组，但是成员的值是唯一的（不重复），集合实现了iterator接口，所以可以使用扩展运算符、for...of来遍历（set）集合的属性和方法
3. 声明集合： let s=new Set(); 查看集合元素个数：s.size; 集合添加新元素：s.add(“helloworld”); 集合删除元素s.delete(“helloworld”); 检测集合是否存在当前元素：s.has(“helloworld”);has()的返回值为Boolean
4. 集合作用：1数组去重，利用成员的值是唯一的特点。2两数组交集，用到filter、has函数。3两数组并集，用到扩展运算符（...数组1,...数组2）再去重（设置成集合）。4差集（互相没有的）:思路和求并集类似
5. 集合和数组可以相互转换，但是要注意数组转换成集合会自动去重。arr2为数组，变成集合： let s2=new Set(arr2); 将s2变成数组：[...s2]或者[...new Set(arr2)];
6. class 静态成员：如function Phone(){Phone.name=’手机’;Phone.change=function(){....}};let nokia=new Phone(); 注意：这里的实例化对象nokia没有Phone里的属性name、change。因为name、change属于Phone对象的属性function Phone(){Phone.name=’手机’;Phone.change=function(){....}};相当于function Phone(){static name=’手机’;satic change=function(){....}};。因此name和change属于静态Phone的静态成员
7. prototype：返回对象类型原型的引用。每一个构造函数都有一个属性叫做原型，这个原型为一个特定类声明通用的变量或者函数。如function test(){console.log(test.prototype)};输出结果为object
8. 数值的扩展：1Number.EPSILON是JavaScript表示的最小精度，2Number.isFinite检测一个数值是否为有限数，如console.log(Number.isFinite(100))；结果为true，3Number.isNaN检测一个数是否为NaN。4Math.trunc将数字的小数部分去掉5Math.sign检测一个数是整数或负数或零（整数返回1，负数返回-1，零返回0）如console.log(Math.trunc(-900));输出结果为-1
9. 对象方法扩展：1object.is:判断两个值是否相等，如console.log(object.is(1,2));。2object.assign：合并对象。如const con1={host:”localhost”};const con2={name:”root”};console.log(object.assign(con1,con2));输出结果为host:”localhost”；name:”root”

ES7新增

1、includes方法用来检测数组中是否包含某个元素，返回值为boolrea 。如const arr1=[“11”,”22”];console.log(arr1.includes(“11”));输出结果为true