**1、let 声明变量**

只在块级作用域起作用（适合在for循环中使用）、无变量提升；   
不可重复声明相同变量，不可重复声明函数内的参数。

**2、const 声明常量**

声明必须赋值、声明后不可修改、只在块级作用域起作用（适合在for循环中使用）、无变量提升、传址赋值

**3、浏览器兼容ES6——babel**

安装babel script引入babel的浏览器版本

**4、解构赋值**

**数组：**   
let [a,b,c] = [1,2,3]; => a=1;b=2;c=3;   
let[ a,b,[ c1,c2 ] ] = [ 1,2,[ 3.1,3.2 ] ]; //可嵌套   
let [a,b,c] = [1,2]; => c为undefined 不完全解构   
let [a,b,c=3] =[1,2,4]; //允许设定默认值，且默认值可覆盖   
**对象：**   
var { a,b,c } = {“a”:1,”c”:3,”b”:2}; //a:1 b:2 c:3 找不到相同属性 则undefined   
var { b:a,} = {“b”:2}; //a：2   
**字符串：**   
var [a,b,c,d,e] = “我是前端君”; //a:我 b:是 c:前 d:端 e:君

**用途：**   
交换变量值、提取函数返回值中的值、定义函数参数、设定函数参数默认值

**5、String 新特性**

1）模板字符串：   
let str = He is ${name},he is a ${occupation}; //${} 中可以放任意表达式：运算表达式、对象的属性、函数的调用   
let str = write once , //直接换行   
run anywhere;   
2）标签模板： 标签函数 + 模板字符串   
3）repeat： str.repeat(num) => str 重复 n 遍   
4）includes： str.includes(‘aaa’,num) => str中是否包含字符串 aaa， num为开始搜索的位置，可省   
5）startsWith： str.startsWith(‘aaa’, num) => 字符串aaa是否出现在str的num位置，num选填，默认为零   
6）endsWith： str.endsWith(‘aaa’, num) => 字符串aaa是否出现在str的num-1位置，num选填，默认为最后一个   
7）raw： 转义字符不转义，直接生成z

**6、数值扩展**

1）isNaN isFinite parseInt parseFloat 移植到Number对象上

Number.isInteger函数 判断是否为整数

Number.isFinite 非数字或无穷大时返回false 有穷数字返回true   
极小常量 安全整数

Math扩展了17个新函数

**7、数组扩展**

**Array.from** 将类似数组的对象或者可遍历的对象转换成真正的数组   
**Array.of** 将一组值，转换成数组   
**find** 找出数组中符合条件的第一个元素   
**findIndex** 找出数组中符合条件的第一个元素的index   
**fill** let arr = [1,2,3]; arr.fill(4,1,3); //结果：[1,4,4] 将位置1开始，3之前填充为4   
**entries** 对数组的键值对进行遍历，返回一个遍历器，可以用for..of对其进行遍历   
**keys** 对数组的索引键进行遍历，返回一个遍历器   
**values** 对数组的元素进行遍历，返回一个遍历器   
数组推导 [for(i of arr1) i \* 2]; [for(i of array1) if(i>3) i];

**8、对象扩展**

1）Object.assign(target,origin); 将源对象的属性赋值到目标对象上，最后出现的属性覆盖前面的同名属性。   
2）Object.getPrototypeOf( )函数 获取一个对象的prototype属性   
3）Object.setPrototypeOf()函数 设置一个对象的prototype属性   
4）模拟面向对象编程：

关键步骤：1、构造函数；2、给prototype对象添加属性和方法；3、实例化；4、通过实例化后的对象调用类的方法或者属性。   
new（） => 构造函数实例化，创建一个新对象，将构造函数的作用域赋给新对象（this指向新对象），执行构造函数中的代码，返回新对象。

**9、函数扩展**

1）**给参数设默认值**，若部分设定，设默认值的放在最后。只有当传入的参数为undefined，才会触发默认值赋值   
2）**rest参数：**在实参中，除了第一个参数以外，剩余的参数都会被…values获取到   
rest参数必须是函数的最后一个参数，后面不能再跟其他参数。   
3）\*\*箭头函数：\*\*var sum = (a,b) => {return a+b}

**扩展运算符：** 它一般结合数组使用，把数组的元素用逗号分隔开来，组成一个序列 => 将一个数组转成一个对应的参数数列

**箭头函数的this指向的是定义时的this对象，而不是执行时的this对象**

**10、模块化 module**

1）批量导入导出： export {name,age,say} import { name,age,say } from “./module-B.js”;   
2）重命名导入的变量：import { name as myname } from “./module-B.js”;   
3）整体导入：import \* as obj from “./module-B.js”;   
4）每个模块支持到处默认导出：

**11、类 class**

基于原型prototype的实现方式做了进一步的封装，让我们使用起来更简单明了。也就是说它实际上也是函数function和原型prototype实现。

ES5 通过构造函数生成实例对象，而 class 实际上也是这样一种实现，相当于是构造函数的另一种写法。在这个类里面可以直接定义方法，而这些方法其实就相当于是定义在 prototype 属性上。

类必须使用 new 调用，否则报错。

类的方法内部如果含有 this，它默认指向类的实例。因此，使用时要小心，可以在 constructor 中绑定 this

**静态方法：**类内的方法如果前面加上 static 关键字，name这个方法不会被实例继承，而是通过类来直接调用。父类的静态方法可以被子类继承。

**super：**父类的构造函数，用来新建父类的this对象。子类必须在constructor方法中调用super方法，否则子类就得不到this对象，新建实例时会报错。

\*\*Extends：\*\*ES5 通过原型链进行继承很麻烦，extends 相当于对继承进行了封装。

**12、Set 、Map**

**1）Set & WeakSet**   
两者均不会有重复项；   
均有 add、delete、has 方法；   
Set 还有 size 属性，clear、entries、forEach 方法，WeakSet 没有（不可遍历）；   
WeakSet 的成员必须是对象类型的值。

**2）Map & WeakMap**   
Map 与对象类似，但它的 key 键名不再局限于字符串，可以是各种类型的值。   
Map 包含 set、get、delete、clear 方法。   
遍历（value,key）

WeakMap 键名只支持引用类型的数据（数组、对象、函数）；且不支持 clear、遍历。

**14、装饰器**

**1）类的装饰器**   
类的装饰器是修饰类的，可以修饰类本身，比如添加静态变量，也可以修饰类实例，比如添加一个属性。

@testDecorator

class ClassA {

}

function testDecorator(target) {

target.addedParam = "I am decorator";

}

console.log(ClassA.addedParam);

**2）类方法的装饰器**

class Person {

@readonly

name() { return `${this.first} ${this.last}` }

}

function readonly(target, name, descriptor){

descriptor.writable = false;

return descriptor;

}

修饰器第一个参数是类的原型对象，第二个参数是所要修饰的属性名，第三个参数是该属性的描述对象。

参考：<https://www.cnblogs.com/sghy/p/8027311.html>

Promise：

Promise 对象用于一个异步操作的最终完成（或失败）及其结果值的表示。简单点说，它就是用于处理异步操作的，异步处理成功了就执行成功的操作，异步处理失败了就捕获错误或者停止后续操作。

then和settimeout执行顺序，即**setTimeout(fn, 0)在下一轮“事件循环”开始时执行，Promise.then()在本轮“事件循环”结束时执行**。因此then 函数先输出，settimeout后输出。

# [理解 async/await](https://segmentfault.com/a/1190000010244279)：

Async 返回 promise 对象，await 等待 promise 对象。

https://segmentfault.com/a/1190000010244279