ES标准的最终确立要经历几个阶段，包括初稿、草案、候选、完成。由于有些特性在ES6提及或者在ES7中经过以上几个阶段或者在完成时纳入了ES8中，这里不进行明确区分，看一下新特性

方法

Array.prototype.includes

```
> ['a', 'b', 'c'].includes('a')
true
> ['a', 'b', 'c'].includes('d')
false
```

Object.values(obj) Object.values 函数会返回指定对象的可枚举的属性值数组，数组中值顺序与 for-in 循环保持一致，函数的声明为

```
const obj = { x: 'xxx', y: 1 };
Object.values(obj); // ['xxx', 1]

const obj = ['e', 's', '8']; // same as { 0: 'e', 1: 's', 2: '8' };
Object.values(obj); // ['e', 's', '8']

// when we use numeric keys, the values returned in a numerical
// order according to the keys
const obj = { 10: 'xxx', 1: 'yyy', 3: 'zzz' };
Object.values(obj); // ['yyy', 'zzz', 'xxx']
Object.values('es8'); // ['e', 's', '8']
```

str.padStart(targetLength [, padString])

str.padEnd(targetLength [, padString]) ES8 中添加了内置的字符串填充函数，分别为 padStart 与 padEnd，该函数能够通过填充字符串的首部或者尾部来保证字符串达到固定的长度；开发者可以指定填充的字符串或者使用默认的空格，

```
'es8'.padStart(2);          // 'es8'
'es8'.padStart(5);          // '  es8'
'es8'.padStart(6, 'woof');  // 'wooes8'
'es8'.padStart(14, 'wow');  // 'wowwowwowwoes8'
'es8'.padStart(7, '0');     // '0000es8'
'es8'.padEnd(2);          // 'es8'
```

```
'es8'.padEnd(5);          // 'es8  '
'es8'.padEnd(6, 'woof');  // 'es8woo'
'es8'.padEnd(14, 'wow');  // 'es8wowwowwowwo'
'es8'.padEnd(7, '6');     // 'es86666'
```

getOwnPropertyDescriptors 函数会返回指定对象的某个指定属性的描述符；该属性必须是对象自己定义而不是继承自原型链，函数的声明为：

```
const obj = { get es8() { return 888; } };
Object.getOwnPropertyDescriptor(obj, 'es8');
// {
//   configurable: true,
//   enumerable: true,
//   get: function es8(){}, //the getter function
//   set: undefined
// }
```

异步

async/await 语法来定义与执行异步函数

```
function fetchTextByPromise() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve("es8");
    }, 2000);
  });
}
async function sayHello() {
  const externalFetchedText = await fetchTextByPromise();
  console.log(`Hello, ${externalFetchedText}`); // Hello, es8
}
sayHello();

console.log(1);
sayHello();
console.log(2);

// 调用结果
1 // immediately
2 // immediately
Hello, es8 // after 2 seconds
```

promise和generator co 是一个使用了 Promises 和 generators 让编程风格看起来更像异步的库，但也需要使用类似于前一个例子的风格去编码：

```
const fetchJson = co(function* () {
    try {
        let request = yield fetch(url);
        let text = yield request.text();
        return JSON.parse(text);
    }
    catch (error) {
        console.log(`ERROR: ${error.stack}`);
    }
});
```

每次回调函数（一个 generator 函数！）产生一个 Promise 给 co，回调函数就会被挂起。一旦这个 Promise 完成，co 就会恢复该回调函数：如果 Promise 被实现，yield 会返回这个实现的值，如果被拒绝，则 yield 会抛出异常。另外，co 能处理回调函数返回的结果（与 then() 相似）

扩展

新提出来的特性是将**作为指数操作的中缀运算符：等同Math.pow(x,y)

```
let squared = 3 ** 2; // 9
```

函数参数列表与调用中的尾部逗号

该特性允许我们在定义或者调用函数时添加尾部逗号而不报错：

```
function es8(var1, var2, var3,) {
  // ...
}
es8(10, 20, 30,);
```

共享内存与原子操作

共享内存允许多个线程并发读写数据，而原子操作则能够进行并发控制，确保多个存在竞争关系的线程顺序执行。

add /sub - 增加或者减去某个位置的某个值 and / or /xor - 进行位操作 load - 获取值