



谢成





ECMAScript **VS** JavaScript

ECMAScript 6 **VS** ECMAScript 2015

浏览器的支持:

<http://kangax.github.io/compat-table/es6/>



<http://es6.ruanyifeng.com/>

let:

- ①不存在变量提升
- ②不允许重复声明
- ③暂时性死区
- ④块级作用域

const:声明一个只读的常量。一旦声明，常量的值就不能改变。

const实际上保证的，并不是变量的值不得改动，而是变量指向的那个内存地址不得改动。

变量的解构赋值

从数组和对象中提取值，对变量进行赋值，这被称为解构

①数组解构

②对象解构

③字符串解构

④应用:函数参数的解构赋值\函数返回值\变量互换
\Json应用

...

- ① 不确定参数
- ② 作为数组一部分
- ③ 数组复制
- ④ 合并数组
- ⑤ 合并对象
- ⑥ 解构

剩余的

- ①模板字符串(反引号): ``${}`` 支持标签、换行 数学运算
- ②`includes()` VS `indexOf()`
- ③`startsWith()` `endsWith()`
- ④`padStart(5, 'abc')` `padEnd()`
- ⑤`repeat(5)`

数值扩展

- ①二进制0B 八进制00
- ②是否为数字: ***Number.isFinite(5);***
- ③***Number.isNaN(NaN)***
- ④***Number.isInteger(5)***
- ⑤***Number.parseFloat(5)***
- ⑥***Number.parseInt(5.5)***
- ⑦***Number.MAX_SAFE_INTEGER***
MIN_SAFE_INTEGER

一些方法移植到**Number**对象上面，行为完全保持不变。这样做的目的，是逐步减少全局性方法，使得语言逐步模块化。

数组扩展1

- ① **Array.from()** json数组格式、类数组
- ② **Array.of()** 将一组值，转换为数组。
- ③ **arr.copyWithIn(target, start = 0, end = this.length)**
 - target**（必需）：从该位置开始替换数据。
 - start**（可选）：从该位置开始读取数据，默认为0。如果为负值，表示倒数。
 - end**（可选）：到该位置前停止读取数据，默认等于数组长度。如果为负值，表示倒数。
- ④ **arr.find(function(val, key, arr){});**用于找出第一个符合条件的数组成员 **arr.findIndex()**

数组扩展2

- ① **arr.fill(); new Array(3).fill(7)**
- ② **arr.includes()**数组是否包含给定的值，与字符串的 **includes**方法类似
- ③ **entries(), keys()**和**values()**——用于遍历数组
- ④ **for ...of**循环
- ⑤ 数组遍历：
 - forEach()**:没有返回值，只是针对每个元素调用**func**
 - map()**:返回一个新的**Array**，每个元素为调用**func**的结果
 - filter()**:返回一个符合**func**条件的元素数组
 - some()**:返回一个**boolean**，判断是否有元素是否符合**func**条件
 - every()**:返回一个**boolean**，判断每个元素是否符合**func**条件

①参数默认值 与解构赋值结合

②方法名.length 返回没有指定默认值的参数个数

③方法名.name

④rest参数

⑤'use strict'

⑥箭头函数 =>

(1)函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。

(2)不可以当作构造函数，也就是说，不可以使用new命令，否则会抛出一个错误。

(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。

严格模式 'use strict'

- ①全局变量必须显式声明
- ②禁止**this**关键字指向全局对象 构造函数必须**new**
- ③对象不能有重名的属性
- ④函数不能有重名的参数
- ⑤禁止使用**with**语句
- ⑥**arguments.callee()**
- ⑦...

①**ES6**:规定只要函数参数使用了默认值、解构赋值、或者扩展运算符，那么函数内部就不能显式设定为严格模式，否则会报错。

严格模式好处：

- 消除**Javascript**语法的一些不合理、不严谨之处，减少一些怪异行为；
- 消除代码运行的一些不安全之处，保证代码运行的安全；
- 提高编译器效率，增加运行速度；
- 为未来新版本的**Javascript**做好铺垫。



Thank you

谢

谢

观

看