

ECMAScript



谢成

Accessories
Canvas
& Suede
Backpack
\$165.00

Wear
Weiss F
Watch
(White Dial)
\$950.00

ing Camping
Hatchet
\$42.00
-30%

Soft Leather
Weekender P

2 ITEMS IN YOUR CART

Marshall Pack
Qty: 1
\$359.00

EDC Kit 2
Qty: 1
\$229.00

DISCOUNT

TOTAL

CONDITION

New
Manufacturer Refurbished
Refurbished
Not Working

Mue
Shaving
\$45.00

SHOP NOW

Home Decor
Handmade Picking Baskets
\$129.00



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。

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.isSafeInteger()***

Number.MAX_SAFE_INTEGER

Number.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。

数值扩展

- ①Math.trunc() 去除一个数的小数部分，返回整数部分。
- ②Math.sign() 判断一个数到底是正数、负数、还是零。

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函数不能有重名的参数
- ④禁止使用**with**语句
- ⑤**arguments.callee()**
- ⑥...

严格模式 'use strict'

①函数内部

②**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。

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('xx', 1, 3);` `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

对象扩展

- ①属性简洁表示法 包括属性和方法 (**this**)
- ②属性名表达式
- ③**Object.is**('foo', 'foo'); 比较两个值是否严格相等, 与严格比较运算符 (**===**) 的行为基本一致 **+0===-0 NaN===NaN**
- ④**Object.assign()**; 对象合并, 第一个参数是目标对象, 后面的参数都是源对象。浅拷贝
- ⑤**'name' in obj** 判断对象是否包含某个属性
0 in arr 判断数组位置是否有值

Symbol

新的原始数据类型Symbol，表示独一无二的值。

Symbol 值通过Symbol函数生成。这就是说，对象的属性名现在可以有两种类型，一种是原来就有的字符串，另一种就是新增的Symbol 类型。凡是属性名属于 Symbol 类型，就都是独一无二的，可以保证不会与其他属性名产生冲突。

```
let name = Symbol();  
obj[name] = 'XX';
```

该属性不会出现在for...in、for...of循环中

Set

- ①Set类似于数组，但是成员的值都是唯一的，没有重复的值。函数接受数组或类数组作为参数。`new Set();`
- ②`add(value)`: 添加某个值，返回Set结构本身。
`delete(value)`: 删除某个值，返回一个布尔值，表示删除是否成功。
`has(value)`: 返回一个布尔值，表示该值是否为Set的成员。
`clear()`: 清除所有成员，没有返回值
- ③遍历: `keys()` `values()` `entires()` `forEach()` `for...of`
- ④WeakSet:成员只能是对象，而不能是其他类型的值。`add()`增加值

Map

- ①类似于对象，也是键值对的集合，但是“键”的范围不限于字符串，各种类型的值（包括对象）都可以当作键。`new Map()`
- ②`.size`属性返回 Map 结构的成员总数
- ③`.set(key, value)`
- ④`.get(key)`
- ⑤`.has(key)`
- ⑥`.delete(key)`
- ⑦`.clear()`
- ⑧遍历
- ⑨**WeakMap**: 只接受对象作为键名（`null`除外），不接受其他类型的值作为键名。



Thank you

谢

谢

观

看