

W

WEICHUANG

ES6





ECMAScript **VS** **JavaScript**

ECMAScript 6 **VS** **ECMAScript 2015**



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

声明方 式

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

扩展运算符...

- ①作为数组一部分
- ②数组复制
- ③合并数组
- ④合并对象
- ⑤类数组转化成数组、解构

rest

字符串扩展

- 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)**

- ①是否为数字: ***Number.isFinite(5);***
- ②***Number.isNaN(NaN)***
- ③***Number.isInteger(5)***
- ④***Number.parseFloat(5)***
- ⑤***Number.parseInt(5.5)***

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

- ① `Math.trunc()` 去除一个数的小数部分，返回整数部分。
- ② `Math.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。

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**for ...of**循环
- ④**entries()**, **keys()**和**values()**——用于遍历数组
- ⑤数组遍历：
 - forEach(val, key, arr):**没有返回值，只是针对每个元素调用**func**
 - filter():**过滤返回一个符合**func**条件的元素数组
 - map();**

forEach(val, key, arr):没有返回值，只是针对每个元素调用**func**

filter():过滤返回一个符合**func**条件的元素数组

map();

对象扩展

- ①属性简洁表示法 包括属性和方法
- 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**Object.keys()** **Object.Values()** **Object.entries();**
- ⑥‘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。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**循环中

- ①**Set**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**
- ④长度: **.size**
- ⑤**WeakSet**:成员只能是对象，而不能是其他类型的值。**add()**增加值

- 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。

Proxy

```
var pro = new Proxy({  
    name: 'lisi'  
}, {  
    //get是获得值之前做的事  
    get: (target, key, property) => {  
        console.log('get');  
        return target[key];  
    },  
    //set是改变值的时候做事，一定要有return 否则值不会被改变  
    set: (target, key, value, receiver) => {  
        console.log(`setting ${key} = ${value}`);  
        return target[key] = value;  
    }  
});
```

Proxy

```
//apply 调用方式时
let target = function () {
    return 'I am js';
};

var handler = {
    apply(target, ctx, args) {
        console.log('do apply');
        return Reflect.apply(...arguments);
    }
};

var pro = new Proxy(target, handler);

console.log(pro());
```

- ① Ajax: Asynchronous Javascript And XML (异步JavaScript和XML)
- ② Jquery: \$.get(). \$.post()
- ③ Ajax原理
- ④ Promise 是异步编程的一种解决方案，比传统的解决方案——更合理和更强大。回调函数和事件

Promise

① 异步编程的一种解决方案，比传统的解决方案——回调函数和事件——更合理和更强大。

Promise对象有以下两个特点。

(1) 对象的状态不受外界影响。Promise对象代表一个异步操作，有三种状态：**pending**（进行中）、**fulfilled**（已成功）和**rejected**（已失败）。只有异步操作的结果，可以决定当前是哪一种状态，任何其他操作都无法改变这个状态。这也是Promise这个名字的由来，它的英语意思就是“承诺”，表示其他手段无法改变。

(2) 一旦状态改变，就不会再变，任何时候都可以得到这个结果。Promise对象的状态改变，只有两种可能：从**pending**变为**fulfilled**和从**pending**变为**rejected**。只要这两种情况发生，状态就凝固了，不会再变了，会一直保持这个结果，这时就称为**resolved**（已定型）。如果改变已经发生了，你再对Promise对象添加回调函数，也会立即得到这个结果。这与事件（Event）完全不同，事件的特点是，如果你错过了它，再去监听，是得不到结果的。

- ①类 --- 对象
- ②**constructor()**
- ③类和模块的内部， 默认就是严格模式， 所以不需要使用**use strict**指定运行模式。只要你的代码写在类或模块之中， 就只有严格模式可用。考虑到未来所有的代码， 其实都是运行在模块之中， 所以**ES6** 实际上把整个语言升级到了严格模式。
- ④**extends**

babel: ES6=>ES5

- ① 安装nodejs(一路下一步)
- ② **npm init -y**
- ③ **npm install -g babel-cli**
- ④ **npm install --save-dev babel-preset-es2015 babel-cli**

```
"devDependencies": {  
    "babel-cli": "^6.26.0",  
    "babel-preset-es2015": "^6.24.1"  
}
```

- ⑤ 创建文件: **.babelrc**

```
{  
  "presets": [  
    "es2015"  
  ],  
  "plugins": []  
}
```

- ⑥ 文件转化:

babel src/index.js -o dist/index.js

文件夹转化: **babel src -d dist**

实时: **babel src/index.js -w -o dist/index.js**

babel src -w -d dist

live server

- ① **https://www.npmjs.com/package/live-server**
- ② **npm init -y**
- ③ **npm install -g live-server**
- ④ **运行: live-server**

模块

- ① **export** :负责进行模块化，也是模块的输出。
- ② **import** : 负责把模块引，也是模块的引入操作。
- ③ **as**
- ④ **export default**

babel-node index.js

注意： **ES6**的模块化不能直接在浏览器中预览，必须要使用**Babel**进行编译之后正常看到结果。



A wide-angle aerial photograph of the Shanghai skyline at dusk. The city is illuminated by numerous skyscrapers and streetlights, with the distinctive spherical structure of the Oriental Pearl Tower standing out on the right. The Huangpu River flows through the center, reflecting the lights of the buildings. The sky is filled with scattered clouds, transitioning from blue to orange and yellow near the horizon.

Thank you

谢

谢

观

看