（一）

从 var 到 let/const

ES5只有全局作用域和函数作用域，没有块级作用域，这带来很多不合理的场景。而在ES5 中申明使用 var，这些变量都是函数级作用域的，它们的作用域是包含它们的最内层的函数。var 的行为偶尔会使人混淆。

在 ES6 中还可以使用 let 和 const 来申明变量。这类变量是块级作用域的，它们的作用域是包含它们最近的块。let 可以理解为块级作用域中的 var，const 与 let 类似，只是用 const 申明的变量其值是不可修改的。

（二）

从字符串拼接，到模板字面量（template string 模版字符串）

ES5 中你想把在字符串中引用一些值，你需要将那些值和一些零碎的字符串连接起来，要用一堆的'+'号来连接文本与变量。ES6 中可以在模板字面量中使用字符串插值：用反引号**（`）**来标识起始，用**${}**来引用变量。

ES6模板字面量也带来了多行文本的表现形式。

（三）

从函数表达式到箭头函数（Arrow Function）

当前 ES5 代码中，在使用了函数表达式的时候，你必须小心处理 this。

要在 B 行访问到指向 say对象的 this。在\_this(A 行)创建一个作为辅助变量。

在ES6中当我们使用箭头函数时，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。

ES5中回调相对繁琐：ES6使用箭头函数就简洁得多。

（四）

要通过数组或对象返回多个值。在 ES5 中，你需要创建一个临时变量来访问那些值。但在 ES6 中你可以使用解构。

数组的解构赋值；

对象的解构赋值：对象的解构与数组有一个重要的不同-数组元素按次序排列，变量的取值由他的位置决定。对象的属性没有次序，变量必须与属性同名才能取到正确的值。

（五）

默认参数值

在 ES5 中指定参数的默认值需要 这样：

ES6 有更漂亮的语法：

（六）

**Rest（…扩展运算符）**

如果你想在 ES5 中让函数(或方法)接受任意数量的参数，必须使用特殊变量 arguments；

ES6 中则可以通过扩展运算符运算符定义一个剩余参数。

（七）

类与继承（class, extends, super）

ES5 中直接实现一个构造函数；

ES6 的类语法提供了比构造函数稍微方便一些的语法；注意简化的方法定义语法 —— 不再需要 function 关键字。

派生类

ES5 中实现子类是件麻烦的事情，尤其是引用父类构造函数和父类属性的时候。

ES5实现继承的方式：利用原型链的特性来实现继承，将子类构造函数的原型作为父类构造函数的实例，结合借用构造函数和原型链的方法。

ES6 内置支持子类，只需要使用 extends 语句。

（八）

（九）

### 新的字符串方法

ECMAScript 6 标准库为字符串提供了一些新的方法。

从indexOf 到 startsWith:

if (str.indexOf('x') === 0) {} // ES5

if (str.startsWith('x')) {} // ES6

从indexOf 到 endsWith:

function endsWith(str, suffix) { // ES5

var index = str.indexOf(suffix);

return index >= 0

&& index === str.length-suffix.length;

}

str.endsWith(suffix); // ES6

从indexOf 到 includes:

if (str.indexOf('x') >= 0) {} // ES5

if (str.includes('x')) {} // ES6

从join 到 repeat (ES5 中重复字符串的方法更需要技巧)：

new Array(3+1).join('#') // ES5

'#'.repeat(3) // ES6