# **ECMAScript 6简介**

ECMAScript 6.0（以下简称 ES6）是 JavaScript 语言的下一代标准，已经在2015年6月正式发布了。它的目标，是使得 JavaScript 语言可以用来编写复杂的大型应用程序，成为企业级开发语言。

## **ECMAScript 和 JavaScript 的关系**

一个常见的问题是，ECMAScript 和 JavaScript 到底是什么关系？

要讲清楚这个问题，需要回顾历史。1996年11月，JavaScript 的创造者 Netscape 公司，决定将 JavaScript 提交给国际标准化组织ECMA，希望这种语言能够成为国际标准。次年，ECMA 发布262号标准文件（ECMA-262）的第一版，规定了浏览器脚本语言的标准，并将这种语言称为 ECMAScript，这个版本就是1.0版。

该标准从一开始就是针对 JavaScript 语言制定的，但是之所以不叫 JavaScript，有两个原因。一是商标，Java 是 Sun 公司的商标，根据授权协议，只有 Netscape 公司可以合法地使用 JavaScript 这个名字，且 JavaScript 本身也已经被 Netscape 公司注册为商标。二是想体现这门语言的制定者是 ECMA，不是 Netscape，这样有利于保证这门语言的开放性和中立性。

因此，ECMAScript 和 JavaScript 的关系是，前者是后者的规格，后者是前者的一种实现（另外的 ECMAScript 方言还有 Jscript 和 ActionScript）。日常场合，这两个词是可以互换的。

## **ES6 与 ECMAScript 2015 的关系**

ECMAScript 2015（简称 ES2015）这个词，也是经常可以看到的。它与 ES6 是什么关系呢？

2011年，ECMAScript 5.1版发布后，就开始制定6.0版了。因此，ES6 这个词的原意，就是指 JavaScript 语言的下一个版本。

但是，因为这个版本引入的语法功能太多，而且制定过程当中，还有很多组织和个人不断提交新功能。事情很快就变得清楚了，不可能在一个版本里面包括所有将要引入的功能。常规的做法是先发布6.0版，过一段时间再发6.1版，然后是6.2版、6.3版等等。

但是，标准的制定者不想这样做。他们想让标准的升级成为常规流程：任何人在任何时候，都可以向标准委员会提交新语法的提案，然后标准委员会每个月开一次会，评估这些提案是否可以接受，需要哪些改进。如果经过多次会议以后，一个提案足够成熟了，就可以正式进入标准了。这就是说，标准的版本升级成为了一个不断滚动的流程，每个月都会有变动。

标准委员会最终决定，标准在每年的6月份正式发布一次，作为当年的正式版本。接下来的时间，就在这个版本的基础上做改动，直到下一年的6月份，草案就自然变成了新一年的版本。这样一来，就不需要以前的版本号了，只要用年份标记就可以了。

ES6 的第一个版本，就这样在2015年6月发布了，正式名称就是《ECMAScript 2015标准》（简称 ES2015）。2016年6月，小幅修订的《ECMAScript 2016标准》（简称 ES2016）如期发布，这个版本可以看作是 ES6.1 版，因为两者的差异非常小（只新增了数组实例的includes方法和指数运算符），基本上是同一个标准。根据计划，2017年6月发布 ES2017 标准。

因此，ES6 既是一个历史名词，也是一个泛指，含义是5.1版以后的 JavaScript 的下一代标准，涵盖了ES2015、ES2016、ES2017等等，而ES2015 则是正式名称，特指该年发布的正式版本的语言标准。本书中提到 ES6 的地方，一般是指 ES2015 标准，但有时也是泛指“下一代 JavaScript 语言”。

## **语法提案的批准流程**

任何人都可以向标准委员会（又称 TC39 委员会）提案，要求修改语言标准。

一种新的语法从提案到变成正式标准，需要经历五个阶段。每个阶段的变动都需要由 TC39 委员会批准。

* Stage 0 - Strawman（展示阶段）
* Stage 1 - Proposal（征求意见阶段）
* Stage 2 - Draft（草案阶段）
* Stage 3 - Candidate（候选人阶段）
* Stage 4 - Finished（定案阶段）

一个提案只要能进入 Stage 2，就差不多肯定会包括在以后的正式标准里面。ECMAScript 当前的所有提案，可以在 TC39 的官方网站[Github.com/tc39/ecma262](https://github.com/tc39/ecma262)查看。

本书的写作目标之一，是跟踪 ECMAScript 语言的最新进展，介绍5.1版本以后所有的新语法。对于那些明确或很有希望，将要列入标准的新语法，都将予以介绍。

## **ECMAScript 的历史**

ES6 从开始制定到最后发布，整整用了15年。

前面提到，ECMAScript 1.0 是1997年发布的，接下来的两年，连续发布了 ECMAScript 2.0（1998年6月）和 ECMAScript 3.0（1999年12月）。3.0版是一个巨大的成功，在业界得到广泛支持，成为通行标准，奠定了 JavaScript 语言的基本语法，以后的版本完全继承。直到今天，初学者一开始学习 JavaScript，其实就是在学3.0版的语法。

2000年，ECMAScript 4.0 开始酝酿。这个版本最后没有通过，但是它的大部分内容被 ES6 继承了。因此，ES6 制定的起点其实是2000年。

为什么 ES4 没有通过呢？因为这个版本太激进了，对 ES3 做了彻底升级，导致标准委员会的一些成员不愿意接受。ECMA 的第39号技术专家委员会（Technical Committee 39，简称TC39）负责制订 ECMAScript 标准，成员包括 Microsoft、Mozilla、Google 等大公司。

2007年10月，ECMAScript 4.0 版草案发布，本来预计次年8月发布正式版本。但是，各方对于是否通过这个标准，发生了严重分歧。以 Yahoo、Microsoft、Google 为首的大公司，反对 JavaScript 的大幅升级，主张小幅改动；以 JavaScript 创造者Brendan Eich为首的Mozilla公司，则坚持当前的草案。

2008年7月，由于对于下一个版本应该包括哪些功能，各方分歧太大，争论过于激烈，ECMA 开会决定，中止 ECMAScript 4.0 的开发，将其中涉及现有功能改善的一小部分，发布为 ECMAScript 3.1，而将其他激进的设想扩大范围，放入以后的版本，由于会议的气氛，该版本的项目代号起名为 Harmony（和谐）。会后不久，ECMAScript 3.1 就改名为 ECMAScript 5。

2009年12月，ECMAScript 5.0 版正式发布。Harmony 项目则一分为二，一些较为可行的设想定名为 JavaScript.next 继续开发，后来演变成 ECMAScript 6；一些不是很成熟的设想，则被视为 JavaScript.next.next，在更远的将来再考虑推出。TC39 委员会的总体考虑是，ES5 与 ES3 基本保持兼容，较大的语法修正和新功能加入，将由 JavaScript.next 完成。当时，JavaScript.next 指的是ES6，第六版发布以后，就指 ES7。TC39 的判断是，ES5 会在2013年的年中成为 JavaScript 开发的主流标准，并在此后五年中一直保持这个位置。

2011年6月，ECMAscript 5.1 版发布，并且成为 ISO 国际标准（ISO/IEC 16262:2011）。

2013年3月，ECMAScript 6 草案冻结，不再添加新功能。新的功能设想将被放到 ECMAScript 7。

2013年12月，ECMAScript 6 草案发布。然后是12个月的讨论期，听取各方反馈。

2015年6月，ECMAScript 6 正式通过，成为国际标准。从2000年算起，这时已经过去了15年。

## **部署进度**

各大浏览器的最新版本，对 ES6 的支持可以查看[kangax.github.io/es5-compat-table/es6/](http://kangax.github.io/es5-compat-table/es6/)。随着时间的推移，支持度已经越来越高了，超过90%的 ES6 语法特性都实现了。

Node 是 JavaScript 的服务器运行环境（runtime）。它对 ES6 的支持度更高。除了那些默认打开的功能，还有一些语法功能已经实现了，但是默认没有打开。使用下面的命令，可以查看 Node 已经实现的 ES6 特性。

$ node --v8-options | grep harmony

上面命令的输出结果，会因为版本的不同而有所不同。

我写了一个工具 [ES-Checker](https://github.com/ruanyf/es-checker)，用来检查各种运行环境对 ES6 的支持情况。访问[ruanyf.github.io/es-checker](http://ruanyf.github.io/es-checker)，可以看到您的浏览器支持 ES6 的程度。运行下面的命令，可以查看你正在使用的 Node 环境对 ES6 的支持程度。

$ npm install -g es-checker

$ es-checker

=========================================

Passes 24 feature Dectations

Your runtime supports 57% of ECMAScript 6

=========================================