JavaScript 语言的历史

诞生

JavaScript 因为互联网而生，紧跟着浏览器的出现而问世。回顾它的历史，就要从浏览器的历史讲起。

1990年底，欧洲核能研究组织（CERN）科学家 Tim Berners-Lee，在全世界最大的电脑网络——互联网的基础上，发明了万维网（World Wide Web），从此可以在网上浏览网页文件。最早的网页只能在操作系统的终端里浏览，也就是说只能使用命令行操作，网页都是在字符窗口中显示，这当然非常不方便。

1992年底，美国国家超级电脑应用中心（NCSA）开始开发一个独立的浏览器，叫做 Mosaic。这是人类历史上第一个浏览器，从此网页可以在图形界面的窗口浏览。

1994年10月，NCSA 的一个主要程序员 Marc Andreessen 联合风险投资家 Jim Clark，成立了 Mosaic 通信公司（Mosaic Communications），不久后改名为 Netscape。这家公司的方向，就是在 Mosaic 的基础上，开发面向普通用户的新一代的浏览器 Netscape Navigator。

1994年12月，Navigator 发布了1.0版，市场份额一举超过90%。

Netscape 公司很快发现，Navigator 浏览器需要一种可以嵌入网页的脚本语言，用来控制浏览器行为。当时，网速很慢而且上网费很贵，有些操作不宜在服务器端完成。比如，如果用户忘记填写“用户名”，就点了“发送”按钮，到服务器再发现这一点就有点太晚了，最好能在用户发出数据之前，就告诉用户“请填写用户名”。这就需要在网页中嵌入小程序，让浏览器检查每一栏是否都填写了。

管理层对这种浏览器脚本语言的设想是：功能不需要太强，语法较为简单，容易学习和部署。那一年，正逢 Sun 公司的 Java 语言问世，市场推广活动非常成功。Netscape 公司决定与 Sun 公司合作，浏览器支持嵌入 Java 小程序（后来称为 Java applet）。但是，浏览器脚本语言是否就选用 Java，则存在争论。后来，还是决定不使用 Java，因为网页小程序不需要 Java 这么“重”的语法。但是，同时也决定脚本语言的语法要接近 Java，并且可以支持 Java 程序。这些设想直接排除了使用现存语言，比如 Perl、Python 和 TCL。

1995年，Netscape 公司雇佣了程序员 Brendan Eich 开发这种网页脚本语言。Brendan Eich 有很强的函数式编程背景，希望以 Scheme 语言（函数式语言鼻祖 LISP 语言的一种方言）为蓝本，实现这种新语言。

1995年5月，Brendan Eich 只用了10天，就设计完成了这种语言的第一版。它是一个大杂烩，语法有多个来源。

- 基本语法：借鉴 C 语言和 Java 语言。

- 数据结构：借鉴 Java 语言，包括将值分成原始值和对象两大类。
- 函数的用法：借鉴 Scheme 语言和 Awk 语言，将函数当作第一等公民，并引入闭包。
- 原型继承模型：借鉴 Self 语言（Smalltalk 的一种变种）。
- 正则表达式：借鉴 Perl 语言。
- 字符串和数组处理：借鉴 Python 语言。

为了保持简单，这种脚本语言缺少一些关键的功能，比如块级作用域、模块、子类型（subtyping）等等，但是可以利用现有功能找出解决办法。这种功能的不足，直接导致了后来 JavaScript 的一个显著特点：对于其他语言，你需要学习语言的各种功能，而对于 JavaScript，你常常需要学习各种解决问题的模式。而且由于来源多样，从一开始就注定，JavaScript 的编程风格是函数式编程和面向对象编程的一种混合体。

Netscape 公司的这种浏览器脚本语言，最初名字叫做 Mocha，1995年9月改为 LiveScript。12月，Netscape 公司与 Sun 公司（Java 语言的发明者和所有者）达成协议，后者允许将这种语言叫做 JavaScript。这样一来，Netscape 公司可以借助 Java 语言的声势，而 Sun 公司则将自己的影响力扩展到了浏览器。

之所以起这个名字，并不是因为 JavaScript 本身与 Java 语言有多么深的关系（事实上，两者关系并不深，详见下节），而是因为 Netscape 公司已经决定，使用 Java 语言开发网络应用程序，JavaScript 可以像胶水一样，将各个部分连接起来。当然，后来的历史是 Java 语言的浏览器插件失败了，JavaScript 反而发扬光大。

1995年12月4日，Netscape 公司与 Sun 公司联合发布了 JavaScript 语言，对外宣传 JavaScript 是 Java 的补充，属于轻量级的 Java，专门用来操作网页。

1996年3月，Navigator 2.0 浏览器正式内置了 JavaScript 脚本语言。

JavaScript 与 Java 的关系

这里专门说一下 JavaScript 和 Java 的关系。它们是两种不一样的语言，但是彼此存在联系。

JavaScript 的基本语法和对象体系，是模仿 Java 而设计的。但是，JavaScript 没有采用 Java 的静态类型。正是因为 JavaScript 与 Java 有很大的相似性，所以这门语言才从一开始的 LiveScript 改名为 JavaScript。基本上，JavaScript 这个名字的原意是“很像Java的脚本语言”。

JavaScript 语言的函数是一种独立的数据类型，以及采用基于原型对象（prototype）的继承链。这是它与 Java 语法最大的两点区别。JavaScript 语法要比 Java 自由得多。

另外，Java 语言需要编译，而 JavaScript 语言则是运行时由解释器直接执行。

总之，JavaScript 的原始设计目标是一种小型的、简单的动态语言，与 Java 有足够的相似性，使得使用者（尤其是 Java 程序员）可以快速上手。

JavaScript 与 ECMAScript 的关系

1996年8月，微软模仿 JavaScript 开发了一种相近的语言，取名为JScript（JavaScript 是 Netscape 的注册商标，微软不能用），首先内置于IE 3.0。Netscape 公司面临丧失浏览器脚本语言的主导权的局面。

1996年11月，Netscape 公司决定将 JavaScript 提交给国际标准化组织 ECMA（European Computer Manufacturers Association），希望 JavaScript 能够成为国际标准，以此抵抗微软。ECMA 的39号技术委员会（Technical Committee 39）负责制定和审核这个标准，成员由业内的大公司派出的工程师组成，目前共25个人。该委员会定期开会，所有的邮件讨论和会议记录，都是公开的。

1997年7月，ECMA 组织发布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 ECMAScript 1.0 版。之所以不叫 JavaScript，一方面是由于商标的关系，Java 是 Sun 公司的商标，根据一份授权协议，只有 Netscape 公司可以合法地使用 JavaScript 这个名字，且 JavaScript 已经被 Netscape 公司注册为商标，另一方面也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。在日常场合，这两个词是可以互换的。

ECMAScript 只用来标准化 JavaScript 这种语言的基本语法结构，与部署环境相关的标准都由其他标准规定，比如 DOM 的标准就是由 W3C组织（World Wide Web Consortium）制定的。

ECMA-262 标准后来也被另一个国际标准化组织 ISO（International Organization for Standardization）批准，标准号是 ISO-16262。

JavaScript 的版本

1997年7月，ECMAScript 1.0发布。

1998年6月，ECMAScript 2.0版发布。

1999年12月，ECMAScript 3.0版发布，成为 JavaScript 的通行标准，得到了广泛支持。

2007年10月，ECMAScript 4.0版草案发布，对3.0版做了大幅升级，预计次年8月发布正式版本。草案发布后，由于4.0版的目标过于激进，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，争论过于激进，ECMA 开会决定，中止 ECMAScript 4.0 的开发（即废除了这个版本），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 的总体考虑是，ECMAScript 5 与 ECMAScript 3 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ECMAScript 6。第六版发布以后，将指 ECMAScript 7。TC39 预计，ECMAScript 5 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）。到了2012年底，所有主要浏览器都支持 ECMAScript 5.1版的全部功能。

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 正式发布，并且更名为“ECMAScript 2015”。这是因为 TC39 委员会计划，以后每年发布一个 ECMAScript 的版本，下一个版本在2016年发布，称为“ECMAScript 2016”，2017年发布“ECMAScript 2017”，以此类推。

周边大事记

JavaScript 伴随着互联网的发展一起发展。互联网周边技术的快速发展，刺激和推动了 JavaScript 语言的发展。下面，回顾一下 JavaScript 的周边应用发展。

1996年，样式表标准 CSS 第一版发布。

1997年，DHTML（Dynamic HTML，动态 HTML）发布，允许动态改变网页内容。这标志着 DOM 模式（Document Object Model，文档对象模型）正式应用。

1998年，Netscape 公司开源了浏览器，这导致了 Mozilla 项目的诞生。几个月后，美国在线（AOL）宣布并购 Netscape。

1999年，IE 5部署了 XMLHttpRequest 接口，允许 JavaScript 发出 HTTP 请求，为后来大行其道的 Ajax 应用创造了条件。

2000年，KDE 项目重写了浏览器引擎 KHTML，为后来的 WebKit 和 Blink 引擎打下基础。这一年的10月23日，KDE 2.0发布，第一次将 KHTML 浏览器包括其中。

2001年，微软公司时隔5年之后，发布了 IE 浏览器的下一个版本 Internet Explorer 6。这是当时最先进的浏览器，它后来统治了浏览器市场多年。

2001年，Douglas Crockford 提出了 JSON 格式，用于取代 XML 格式，进行服务器和网页之间的数据交换。JavaScript 可以原生支持这种格式，不需要额外部署代码。

2002年，Mozilla 项目发布了它的浏览器的第一版，后来起名为 Firefox。

2003年，苹果公司发布了 Safari 浏览器的第一版。

2004年，Google 公司发布了 Gmail，促成了互联网应用程序（Web Application）这个概念的诞生。由于 Gmail 是在4月1日发布的，很多人起初以为这只是一个玩笑。

2004年，Dojo 框架诞生，为不同浏览器提供了同一接口，并为主要功能提供了便利的调用方法。这标志着 JavaScript 编程框架的时代开始来临。

2004年，WHATWG 组织成立，致力于加速 HTML 语言的标准化进程。

2005年，苹果公司在 KHTML 引擎基础上，建立了 WebKit 引擎。

2005年，Ajax 方法（Asynchronous JavaScript and XML）正式诞生，Jesse James Garrett 发明了这个词汇。它开始流行的标志是，2月份发布的 Google Maps 项目大量采用该方法。它几乎成了新一代网站的标准做法，促成了 Web 2.0时代的来临。

2005年，Apache 基金会发布了 CouchDB 数据库。这是一个基于 JSON 格式的数据库，可以用 JavaScript 函数定义视图和索引。它在本质上有别于传统的关系型数据库，标识着 NoSQL 类型的数据库诞生。

2006年，jQuery 函数库诞生，作者为John Resig。jQuery 为操作网页 DOM 结构提供了非常强大易用的接口，成为了使用最广泛的函数库，并且让 JavaScript 语言的应用难度大大降低，推动了这种语言的流行。

2006年，微软公司发布 IE 7，标志重新开始启动浏览器的开发。

2006年，Google推出 Google Web Toolkit 项目（缩写为 GWT），提供 Java 编译成 JavaScript 的功能，开创了将其他语言转为 JavaScript 的先河。

2007年，Webkit 引擎在 iPhone 手机中得到部署。它最初基于 KDE 项目，2003年苹果公司首先采用，2005年开源。这标志着 JavaScript 语言开始能在手机中使用了，意味着有可能写出在桌面电脑和手机中都能使用的程序。

2007年，Douglas Crockford 发表了名为《JavaScript: The good parts》的演讲，次年由 O'Reilly 出版社出版。这标志着软件行业开始严肃对待 JavaScript 语言，对它的语法开始重新认识，

2008年，V8 编译器诞生。这是 Google 公司为 Chrome 浏览器而开发的，它的特点是让 JavaScript 的运行变得非常快。它提高了 JavaScript 的性能，推动了语法的改进和标准化，改变外界对 JavaScript 的不佳印象。同时，V8 是开源的，任何人想要一种快速的嵌入式脚本语言，都可以采用 V8，这拓展了 JavaScript 的应用领域。

2009年，Node.js 项目诞生，创始人为 Ryan Dahl，它标志着 JavaScript 可以用于服务器端编程，从此网站的前端和后端可以使用同一种语言开发。并且，Node.js 可以承受很大的并发流量，使得开发某些互联网大规模的实时应用变得容易。

2009年，Jeremy Ashkenas 发布了 CoffeeScript 的最初版本。CoffeeScript 可以被转换为 JavaScript 运行，但是语法要比 JavaScript 简洁。这开启了其他语言转为 JavaScript 的风潮。

2009年，PhoneGap 项目诞生，它将 HTML5 和 JavaScript 引入移动设备的应用程序开发，主要针对 iOS 和 Android 平台，使得 JavaScript 可以用于跨平台的应用程序开发。

2009，Google 发布 Chrome OS，号称是以浏览器为基础发展成的操作系统，允许直接使用 JavaScript 编写应用程序。类似的项目还有 Mozilla 的 Firefox OS。

2010年，三个重要的项目诞生，分别是 NPM、BackboneJS 和 RequireJS，标志着 JavaScript 进入模块化开发的时代。

2011年，微软公司发布 Windows 8操作系统，将 JavaScript 作为应用程序的开发语言之一，直接提供系统支持。

2011年，Google 发布了 Dart 语言，目的是为了结束 JavaScript 语言在浏览器中的垄断，提供更合理、更强大的语法和功能。Chromium浏览器有内置的 Dart 虚拟机，可以运行 Dart 程序，但 Dart 程序也可以被编译成 JavaScript 程序运行。

2011年，微软工程师[Scott Hanselman](http://www.hanselman.com/blog/JavaScriptIsAssemblyLanguageForTheWebSematicMarkupIsDeadCleanVsMachinecodedHTML.aspx)提出，JavaScript 将是互联网的汇编语言。因为它无所不在，而且正在变得越来越快。其他语言的程序可以被转成 JavaScript 语言，然后在浏览器中运行。

2012年，单页面应用程序框架（single-page app framework）开始崛起，AngularJS 项目和 Ember 项目都发布了1.0版本。

2012年，微软发布 TypeScript 语言。该语言被设计成 JavaScript 的超集，这意味着所有 JavaScript 程序，都可以不经修改地在 TypeScript 中运行。同时，TypeScript 添加了很多新的语法特性，主要目的是为了开发大型程序，然后还可以被编译成 JavaScript 运行。

2012年，Mozilla 基金会提出 [asm.js](http://asmjs.org/) 规格。asm.js 是 JavaScript 的一个子集，所有符合 asm.js 的程序都可以在浏览器中运行，它的特殊之处在于语法有严格限定，可以被快速编译成性能良好的机器码。这样做的目的，是为了给其他语言提供一个编译规范，使其可以被编译成高效的 JavaScript 代码。同时，Mozilla 基金会还发起了 [Emscripten](https://github.com/kripken/emscripten/wiki) 项目，目标就是提供一个跨语言的编译器，能够将 LLVM 的位代码（bitcode）转为 JavaScript 代码，在浏览器中运行。因为大部分 LLVM 位代码都是从 C / C++ 语言生成的，这意味着 C / C++ 将可以在浏览器中运行。此外，Mozilla 旗下还有 [LLJS](http://mbebenita.github.io/LLJS/) （将 JavaScript 转为 C 代码）项目和 [River Trail](https://github.com/RiverTrail/RiverTrail/wiki) （一个用于多核心处理器的 ECMAScript 扩展）项目。目前，可以被编译成 JavaScript 的[语言列表](https://github.com/jashkenas/coffee-script/wiki/List-of-languages-that-compile-to-JS)，共有将近40种语言。

2013年，Mozilla 基金会发布手机操作系统 Firefox OS，该操作系统的整个用户界面都使用 JavaScript。

2013年，ECMA 正式推出 JSON 的[国际标准](http://www.ecma-international.org/publications/standards/Ecma-404.htm)，这意味着 JSON 格式已经变得与 XML 格式一样重要和正式了。

2013年5月，Facebook 发布 UI 框架库 React，引入了新的 JSX 语法，使得 UI 层可以用组件开发，同时引入了网页应用是状态机的概念。

2014年，微软推出 JavaScript 的 Windows 库 WinJS，标志微软公司全面支持 JavaScript 与 Windows 操作系统的融合。

2014年11月，由于对 Joyent 公司垄断 Node 项目、以及该项目进展缓慢的不满，一部分核心开发者离开了 Node.js，创造了 io.js 项目，这是一个更开放、更新更频繁的 Node.js 版本，很短时间内就发布到了2.0版。三个月后，Joyent 公司宣布放弃对 Node 项目的控制，将其转交给新成立的开放性质的 Node 基金会。随后，io.js 项目宣布回归 Node，两个版本将合并。

2015年3月，Facebook 公司发布了 React Native 项目，将 React 框架移植到了手机端，可以用来开发手机 App。它会将 JavaScript 代码转为 iOS 平台的 Objective-C 代码，或者 Android 平台的 Java 代码，从而为 JavaScript 语言开发高性能的原生 App 打开了一条道路。

2015年4月，Angular 框架宣布，2.0 版将基于微软公司的TypeScript语言开发，这等于为 JavaScript 语言引入了强类型。

2015年5月，Node 模块管理器 NPM 超越 CPAN，标志着 JavaScript 成为世界上软件模块最多的语言。

2015年5月，Google 公司的 Polymer 框架发布1.0版。该项目的目标是生产环境可以使用 WebComponent 组件，如果能够达到目标，Web 开发将进入一个全新的以组件为开发基础的阶段。

2015年6月，ECMA 标准化组织正式批准了 ECMAScript 6 语言标准，定名为《ECMAScript 2015 标准》。JavaScript 语言正式进入了下一个阶段，成为一种企业级的、开发大规模应用的语言。这个标准从提出到批准，历时10年，而 JavaScript 语言从诞生至今也已经20年了。

2015年6月，Mozilla 在 asm.js 的基础上发布 WebAssembly 项目。这是一种 JavaScript 引擎的中间码格式，全部都是二进制，类似于 Java 的字节码，有利于移动设备加载 JavaScript 脚本，执行速度提高了 20+ 倍。这意味着将来的软件，会发布 JavaScript 二进制包。

2016年6月，《ECMAScript 2016 标准》发布。与前一年发布的版本相比，它只增加了两个较小的特性。

2017年6月，《ECMAScript 2017 标准》发布，正式引入了 async 函数，使得异步操作的写法出现了根本的变化。

2017年11月，所有主流浏览器全部支持 WebAssembly，这意味着任何语言都可以编译成 JavaScript，在浏览器运行。

参考链接

- Axel Rauschmayer, [The Past, Present, and Future of JavaScript](http://oreilly.com/javascript/radarreports/past-present-future-javascript.csp)
- John Dalziel, [The race for speed part 4: The future for JavaScript](http://creativejs.com/2013/06/the-race-for-speed-part-4-the-future-for-javascript/)
- Axel Rauschmayer, [Basic JavaScript for the impatient programmer](http://www.2ality.com/2013/06/basic-javascript.html)
- resin.io, [Happy 18th Birthday JavaScript! A look at an unlikely past and bright future](http://resin.io/happy-18th-birthday-javascript/)