

JavaScript とは

JavaScript は動きのある web ページ を実現させるためのプログラミング言語で、HTML や CSS と同じようにブラウザとテキスト・エディタだけを使って完成させることができます。マウスを動かしてカーソルを文字に重ねることでポップ・アップ・メニューを出したり、ボタンをクリックして小さなサブ画面を表示できるような web ページ には 必ず JavaScript が使われています。そして HTML ファイル に jQuery / JSON などのライブラリ (JavaScript を使った拡張機能) を読み込ませたり Ajax というプラグインを導入すると少ない記述で豊かな表現が可能となり、web ページ の操作性が大きく向上します。さらに Angular.js / Vue.js / React.js のような framework(フレームワーク) や、d3.js / chart.js / three.js / tone.js などのライブラリを活用すれば web ページ に特別な機能を持たせることができます。多くのライブラリーやフレームワークが発展した経緯を知るために、JavaScript 誕生から現在までの出来事をまとめてみましょう。



Standard ECMA-262
6th Edition / June 2015

ECMAScript® 2015 **Language Specification**

This is the HTML rendering of ECMA-262 6th Edition, The ECMAScript 2015 Language Specification.

The PDF rendering of this document is located at
<http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf>.

The PDF version is the definitive specification.
Any discrepancies between this HTML version
and the PDF version are unintentional.

JavaScript の歴史

世界初のブラウザが登場したころ、Web 研究者たちはブラウザの中に Java Applet というプログラム言語を組み込んで動きのある web ページを開発できる環境を整えることを目指していました。ところが Java Applet を書くには HTML とは比べ物にならないくらいの専門知識が必要で、いきなり敷居が高くなってしまったことで一般のユーザーが使うことはできませんでした。そこでブラウザに複雑なプログラムを組み込むことは中止となり、もっと手軽に書ける言語の開発を急ぐこととなりました。

そのような状況の中で 1993 年にアメリカ・イリノイ州の大学 University of Illinois の研究者 Brendan Eich(ブレンダン・アイク) 氏が シンプルでわかりやすいプログラム言語 (LiveScript) を開発し、この技術が JavaScript 1.0 として Netscape 2.0 に搭載されました。Netscape 社 が JavaScript のライセンスを取得したことで Netscape Navigator に JavaScript が実装されたわけですが、1996 年にライバルの Microsoft / Internet Explorer 3.0 が出現しました。

もちろんのこと、当時の Microsoft 社 も JavaScript を使用する権利を取得したかったわけですが Netscape 社 は Microsoft 社 に権利を渡しませんでした。そのことから 1994 年から 1996 年にかけて、Netscape 社 と Microsoft 社 が (まるでブラウザの優劣を競う戦争のように) 激しく競争し、その後の JavaScript の歴史に大きな悪影響を与えてしまいました。Internet Explorer(Windows95) に実装されていたプログラムは Microsoft 社 が独自開発した JScript(ジェイスク립ト = JavaScript の類似) であったため、Internet Explorer と Netscape Navigator の動作の違いにユーザーは悩まされていました (デベロッパーからも大不評)。JavaScript の実装法やエラー対処方法などのマニュアルや説明書は一切存在せず、ユーザーは Web ブラウザ のどちらか一方を選択する必要に迫られました。

Netscape Navigator と Internet Explorer が互いに JavaScript / JScript の独自規格を実装し、ブラウザの機能強化のみならずマーケティングにおいてもその競争は激化し続けたが、この技術に互換性がないと Web ページ 制作者に過剰な負担をかけてしまうことが問題となり、Netscape 社 / Microsoft 社 / W3C との話し合い (スイス・ジュネーブの ECMA が国際的な標準化を提案したこと) で、JavaScript と JScript は ECMAScript という名前に変更され、1997 年にやっと標準化されました (正式名は ECMAScript ですが世間一般では JavaScript と呼ぶことに注目)。その後 JavaScript の互換性問題は解決され、すべてのブラウザにおいて共通の JavaScript が実装されることとなり HTML / CSS / JavaScript のソース・コードを HTTP / FTP のプロトコルでエラー無しに転送することが可能となりました。

JavaScript は本来ブラウザ上でアニメーションのように「動きのあるコンテンツ」を実装できることがその特徴でしたが実際は裏腹に、無駄に動く悪質な Web ページが増え、さらにブラウザのセキュリティへの考慮が不十分であったためにウィルスが多く登場し、ほとんどのユーザーはブラウザの Javascript 機能を off にして Web を活用していました。2000 年代に入った頃、安全で閲覧しやすい (適度にスムーズな動きを持つ) Web ページは JavaScript の類似言語である Action Script を使った Flash で作られるようになりました。この時代に一気に Flash の需要が高まり、Flash を使ったサイトが主流となってしまいました。

JavaScript が軽視される時代が続き、2000 年代の後半では「もう JavaScript など必要無い」とまで言われていましたが、Google 社が Ajax(エイジャックス) を発表したことで世間は驚かされました。Ajax とは Asynchronous と JavaScript に XML を組み合わせたことを表す造語で、JavaScript の技術を利用した非同期通信システム (Google Maps JavaScript API) のことです。Ajax 登場のおかげで Gmail や Google Map の動作が可能となり、世間で「JavaScript は悪いもの」という常識が覆りました。そして 2010 年に jQuery が登場しました。jQuery をリンクすると大きな画像が素早く動くようになり、アニメーション付きのスライド・ショーを簡単に実現できます。このころから Web ページ にオシャレでスーパー・リッチな表現を持たせることが可能となり、Flash は逆にセキュリティに問題があることに加えてスマートフォン (タブレット / モバイル) のバッテリーの消費が激しくなるという理由で使われなくなりました。

HTML5 の時代に入って JavaScript はさらに発展し、Angular.js / Vue.js / React.js のような framework(フレームワーク) が誕生しました。さらに JavaScript がサーバー内でも動くプラットフォームとして Node.js が開発され、Web ページ はネットワーク上で情報処理を行う Web アプリケーション に進化しています。そしてこれまでの Web ページ制作者は Front End Engineer (フロント・エンド・エンジニア) と Back End Engineer (バック・エンド・エンジニア) に分類され、開発 / 設計 において異なった業務を担当するようになりました。Node.js が登場したことにより Web ページ / Web アプリケーションの開発はチームで行われるようになり、今ではネイティブ・アプリケーションに負けない優れたツールが次々と登場しています。

JavaScript の仕様は様々な環境で使われることを考慮して毎年書き換えられています。特に重要なのは 2015 年に発表された ECMAScript 2015 という改訂です。現在のブラウザにはこの内容が実装されていますから、ECMAScript 2015 というコーディング方法が主流になることは間違い無いでしょう。今後 JavaScript を使う人は古い仕様と新しい仕様の両方を理解しなければならないため少々混乱が予測されます。過去の資産も重要ですが、これから JavaScript の学習を始める人は ECMAScript 2015 を身につけるべきであると考えられます。

		Compilers/polyfills										Desktop browsers																	
		97%										56% 71% 48% 59% 18% 5% 11% 83% 93% 95% 86% 94% 94% 97% 97% 97% 97% 54% 100% 100% 100% 100%																	
Feature name		Current browser	Traceur	Babel + core-js ^[2]	Closure	TypeScript + core-js	es-shim	KQ 4.14 ^[3]	IE 11	Edge 13 ^[4]	Edge 14 ^[4]	Edge 15 ^[4]	FF 45 ESR	FF 51	FF 52 Beta	FF 53 Aurora	FF 54 Nightly	CH 56, OP 43 ^[1]	CH 57, OP 44 ^[1]	CH 58, OP 45 ^[1]	SF 9	SF 10	SF 10.1	SF TP	WK				
Optimisation																													
proper tail calls (tail call optimisation)		0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2			
Syntax																													
default function parameters		7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	7/7	4/7	6/7	6/7	7/7	7/7	7/7	7/7	7/7	0/7	7/7	7/7	7/7	7/7	7/7			
rest parameters		5/5	4/5	3/5	2/5	4/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	5/5	5/5	5/5	5/5	5/5			
spread (...) operator		15/15	15/15	13/15	12/15	4/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	9/15	15/15	15/15	15/15	15/15	15/15			
object literal extensions		6/6	6/6	6/6	4/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	5/6	6/6	6/6	6/6	6/6	6/6			
for...of loops		9/9	9/9	9/9	6/9	3/9	0/9	0/9	0/9	7/9	7/9	9/9	7/9	7/9	7/9	9/9	9/9	9/9	9/9	9/9	8/9	9/9	9/9	9/9	9/9	9/9			
octal and binary literals		4/4	2/4	4/4	4/4	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4			
template literals		5/5	4/5	4/5	3/5	3/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5			
RegExp "v" and "u" flags		5/5	3/5	3/5	0/5	0/5	0/5	0/5	0/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	5/5	5/5	5/5	5/5	5/5			
destructuring declarations		22/22	20/22	21/22	19/22	15/22	0/22	0/22	0/22	0/22	21/22	22/22	19/22	21/22	21/22	22/22	22/22	22/22	22/22	22/22	19/22	22/22	22/22	22/22	22/22	22/22			
destructuring assignment		24/24	23/24	24/24	17/24	19/24	0/24	0/24	0/24	0/24	23/24	24/24	24/24	21/24	23/24	23/24	24/24	24/24	24/24	24/24	21/24	24/24	24/24	24/24	24/24	24/24			
destructuring parameters		23/23	19/23	20/23	18/23	15/23	0/23	0/23	0/23	0/23	23/23	23/23	18/23	20/23	23/23	23/23	23/23	23/23	23/23	23/23	18/23	23/23	23/23	23/23	23/23	23/23			
Unicode code point escapes		2/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	1/2	1/2	1/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2			
new.target		2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2			
Bindings																													
const		16/16	14/16	14/16	14/16	14/16	0/16	2/16	12/16	12/16	16/16	16/16	12/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	1/16	16/16	16/16	16/16	16/16	16/16			
let		12/12	10/12	10/12	10/12	10/12	0/12	0/12	10/12	10/12	12/12	12/12	10/12	12/12	12/12	12/12	12/12	12/12	12/12	12/12	0/12	12/12	12/12	12/12	12/12	12/12			
block-level function declaration ^[12]		Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes			
Functions																													
arrow functions		13/13	11/13	9/13	10/13	9/13	0/13	0/13	0/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	0/13	13/13	13/13	13/13	13/13	13/13			
class		24/24	17/24	19/24	13/24	19/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	16/24	24/24	24/24	24/24	24/24	24/24			
super		8/8	7/8	4/8	5/8	7/8	0/8	0/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	6/8	8/8	8/8	8/8	8/8	8/8			

Web ページの制作において JavaScript が必要となる場面

マウスの動きに伴うメニューの表示 / 非表示
サブ画面の表示
画像の拡大 / 入れ替え / ポップアップの表示
エフェクトとアニメーションを伴う画像のスライダー
送信フォームと必須項目チェック
Google マップの導入
Google のアクセス分析やサービスの利用
など

Edition	略称	公開日
ECMAScript 1	ES1	1997年6月
ECMAScript 2	ES2	1998年6月
ECMAScript 3	ES3	1999年12月
ECMAScript 4	ES4	破棄
ECMAScript 5	ES5	2009年12月
ECMAScript 5.1	ES5.1	2011年6月
ECMAScript 2015	ES2015 / ES6	2015年6月
ECMAScript 2016	ES2016 / ES7	2016年6月
ECMAScript 2017	ES2017 / ES8	2017年6月
ECMAScript 2018	ES2018 / ES9	2018年6月
ECMAScript 2019	ES2019 / ES10	2019年6月
ECMAScript 2020	ES2020 / ES11	2020年6月