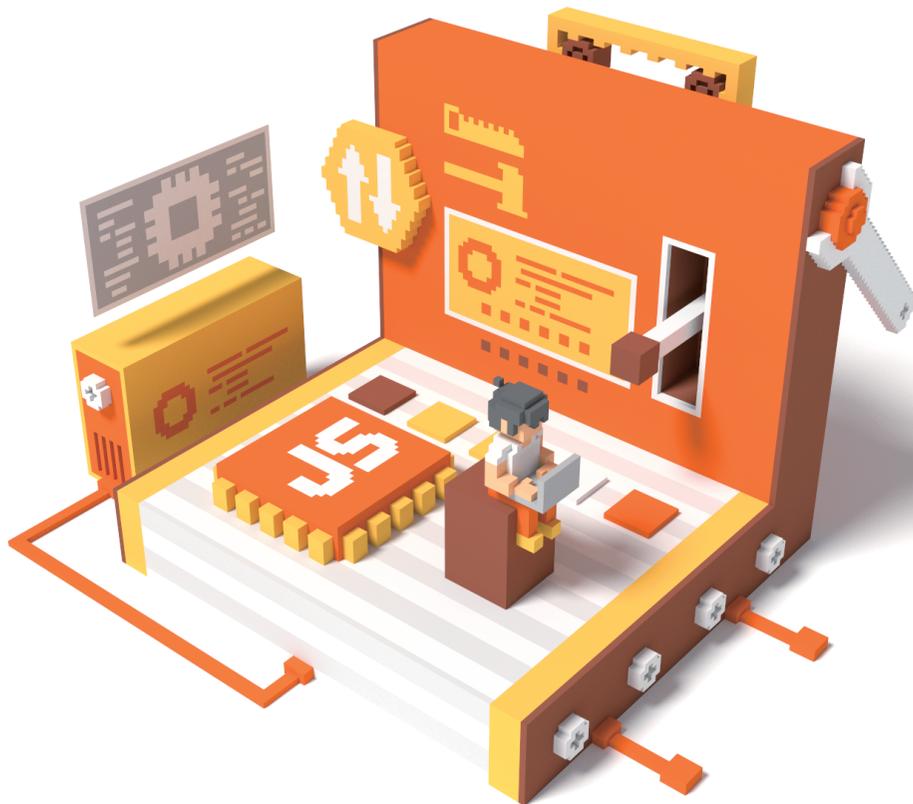


新しい JavaScript の 教科書

境 祐司 著



SCC

新しい
JavaScript
の
教科書

境 祐司 著

著者略歴

境 祐司 (Yuji Sakai)

インストラクショナルデザイナーとして講座企画、IDマネジメント、記事執筆、講演などを中心に活動。2012年5月、電子出版専門のパブリッシャーとして電子書籍のプランニング、情報設計、デリバリデザイン等を手掛ける。2016年より、ニューラルネットワークを使ったスタイル置換処理等の研究に着手。

2017年より、Adobe Community Evangelistとして主に Adobe Sensei (AdobeのAI技術の総称) やXD関連のイベント登壇、企業向け講演等を中心に活動中。

Adobe Community Evangelist ■ <https://adobe.ly/2XluCJf>

Twitter ■ <https://twitter.com/commonstyle>

著書

『実習で身につく！新しいWebデザインの教科書 ～基礎から学べるHTML&CSSデザイン～』(SCC)『Adobe Muse ランディングページ制作ガイド ～コード知識ゼロで作るWeb広告』(技術評論社) など。

サポートページのご案内

下記の本書専用サポートページでは、正誤情報や、本書に記載された項目等に関する補足情報などを、必要に応じて掲載します。

本書専用サポートページ

<https://www.scc-kk.co.jp/scc-books/b-414.html>

なお、サポートページの内容は必要に応じて随時更新されますので、ご注意ください。

本書に掲載した会社名、製品名などは、一般に各社の登録商標または商標です。本書の使用(本書のとおり操作を行う場合を含む)により、万一、直接的・間接的に損害等が発生しても、出版社および編著者は一切の責任を負いかねますので、あらかじめご了承ください。

はじめに

インターネット黎明期の代表的なブラウザだった「Netscape Navigator」のスクリプト言語として登場した「JavaScript」は、数あるプログラミング言語の中でも大変ユニークな存在です。

Microsoftが「JavaScriptと類似した」スクリプト言語「JScript」を同社のブラウザ「Internet Explorer」に搭載し、Netscapeとの熾烈なブラウザ戦争を繰り広げていた1990年代後半、世界中のWeb開発者はその互換性の低さに頭を悩ませていました。

1997年、標準化団体のEcma Internationalが、JavaScriptとJScriptの共有化できる部分を抜き出して標準化しましたが、この仕様が「ECMAScript」です。なお、実際のプログラミングでよく利用されるDOM (Document Object Model)などは、技術コミュニティの「WHATWG」が策定を行っています。

JavaScriptはこのような多層的な言語であり、進化が速く、Webのトレンドにも密接しています。

しかし、JavaScriptの学習を始めたばかりの初心者にとって、これらの関係は理解しづらいものです。そこで本書は、JavaScriptの全体像を常に意識しながら、言語仕様であるECMAScript、ブラウザに文字や画像を表示する仕組み、ブラウザとプログラムをつなぐAPIなどの技術を体系的に学べるように構成しました。

各章の冒頭では、達成目標や学習のポイント、学習の流れを理解し、「今何を学んでいるのか」を常に把握できます。章末では、練習問題や達成目標のチェックシートで理解度を自己評価し、理解が曖昧な箇所や、復習すべき学習項目を洗い出すことができます。

また、JavaScriptには20年以上の歴史があり、何度も改訂が繰り返されてきましたが、特に大きな改訂となったのがECMAScript2015 (ES6)です。ソフトウェア開発の生産性に関わる重要な仕様が盛り込まれており、これからJavaScriptを学ぶ人たちにとっての「入り口」になります。本書はこのES6に準拠しているので、より実践的なスキルを身につけることができます。

JavaScriptは他のプログラミング言語より敷居が低い（初心者向きの）言語として知られていますが、仕様や技術の全体像を理解していないと、なかなか使いこなせないものです。

本書は初心者が覚えるべき要素を厳選して解説しているので、基礎を学ぶ最初の一冊として最適です。JavaScript習得のきっかけとして、役立てていただければ幸いです。

2019年11月

境 祐司

本書の読み方

本書は全 12 章を通して、JavaScript に関するさまざまな知識を解説しています。15 時限の授業を想定して構成していますが、ご自身の学習しやすいペースで読み進めてください。

Chapter 1 Web と JavaScript	1時限 講義
Step-01 JavaScript 誕生の歴史	
Step-02 Web ブラウザーの種類	
Step-03 JavaScript と ECMAScript の関係	
達成目標	
1. JavaScript がどのような経緯で誕生したか他人に説明できる	
2. 代表的な6つのWeb ブラウザーを列挙できる	
3. レンダリングエンジンとJavaScriptの役割を理解している	
4. ECMAScript とは何か理解している	
5. JavaScript の全体像について他人にわかりやすく説明できる	

Chapter 2 準備と基本文法	2時限 講義 実習
Step-01 JavaScript を学ぶための準備をしておこう	
Step-02 HTML ファイルを作成する	
Step-03 オートコンプリートを活用してコードを記述しよう	
Step-04 プログラミングの「デバッグ」を体験してみよう	
Step-05 外部の JavaScript ファイルを作成して HTML に読み込む	
Step-06 用途に応じてコメントを記述する	
達成目標	
1. モダンなコードエディタ「Atom」の基本的な使い方がわかる	
2. 基本的なHTMLコードをコードエディタの入力支援機能を活用して記述できる	
3. 簡単なJavaScriptのコードを記述できる	
4. HTML および JavaScript のコメントを用途に応じて記述できる	
5. Google Chrome の開発者向けツールを活用して「バグ」をすばやく発見できる	

Chapter 3 変数とデータ型	3時限 実習	
Step-01 変数とは？		
Step-02 演算子の種類と優先順位		
Step-03 JavaScript のデータ型		
達成目標		
1. 変数の役割と使い方を他人に説明することができる		
2. 変数を宣言するときに必要な3種類のキーワードについて理解している		
3. 変数名を決めるときのルールや記法について理解している		
4. 演算子の種類や優先順位について理解している		
5. データ型とは何か他人にわかりやすく説明することができる		

Chapter 4 条件分岐	4時限 実習
Step-01 アルゴリズムって？	
Step-02 体格指数を算出するプログラムを if 文で書いてみよう	
Step-03 体格指数を算出するプログラムを if/else 文で改良しよう	
Step-04 曜日によって表示を変更するプログラムを switch 文で書いてみよう	
達成目標	
1. アルゴリズムとは何か他人に説明することができる	
2. if 文で使う比較演算子の使い方を理解している	
3. if/else 文の使い方を理解している	
4. if 文で使う論理演算子の使い方を理解している	
5. switch 文と if/else 文の違いについて他人に説明することができる	

Chapter 5 繰り返し	5時限 実習
Step-01 ループ文	
Step-02 while 文と do while 文を理解する	
Step-03 for 文を理解する	
Step-04 break 文と continue 文を理解する	
達成目標	
1. ループ文とは何か他人に説明することができる	
2. while 文と do while 文がどのような処理でどこが異なるのか理解している	
3. for 文がどのような処理なのか理解している	
4. break 文と continue 文がどのような処理でどこが異なるのか理解している	
5. 制御構文の分類、それぞれの特徴を他人に説明することができる	

Chapter 6 配列	6時限 実習
Step-01 配列とは何か？	
Step-02 JavaScript の配列はオブジェクト	
Step-03 連想配列	
Step-04 基本的な配列の操作	
達成目標	
1. JavaScript の配列とは何か他人に説明することができる	
2. Java や C などの配列と何が異なるのか理解している	
3. 連想配列について他人にわかりやすく説明することができる	
4. 配列に格納した要素を取り出したり入れ替えたりすることができる	
5. 配列に新しい要素を追加できる	

Chapter 7 関数		7時間 実習
Step-01	関数の基礎知識	
Step-02	関数の使い方	
Step-03	関数を使って簡単なプログラムを作る	
達成目標		
<ol style="list-style-type: none"> 1. JavaScript の関数とはどのようなものか他人に説明することができる 2. 関数を使うメリットについて他人にわかりやすく説明することができる 3. 関数を定義することができる 4. 関数のことを「メソッド」と呼ぶ理由について理解している 5. 関数リテラルについて理解している 		

Chapter 8 組み込みオブジェクト		8時間 実習
Step-01	JavaScript の組み込みオブジェクト	
Step-02	Date オブジェクトを使う	
Step-03	String オブジェクトを使う	
達成目標		
<ol style="list-style-type: none"> 1. JavaScript の組み込みオブジェクトについて他人に説明することができる 2. 組み込みオブジェクトにどのようなものがあるか理解している 3. 今日の日付を取得して「〇〇年〇〇月〇〇日」のように表示できる 4. さらに曜日を取得して「〇〇年〇〇月〇〇日(曜日)」と表示できる 5. 「東京都→東京」のように指定した範囲の文字列を取り出すことができる 		

Chapter 9 オブジェクト指向プログラミング		9時間 講義
Step-01	オブジェクト指向とは？	
Step-02	DOM オブジェクトとは？	

Step-03	DOM の操作	10時間 実習
達成目標		
<ol style="list-style-type: none"> 1. オブジェクト指向やクラス/インスタンスについて理解している 2. JavaScript がプロトタイプベースのオブジェクト指向であることを理解している 3. DOM とは何か他人に説明することができる 4. DOM ツリーが何を意味しているのか理解している 5. DOM オブジェクトを使って HTML の要素を追加したり削除したりすることができる 		

Chapter 10 Web API		11時間 講義 実習
Step-01	API とは？	
Step-02	Web ブラウザーのオブジェクト	

Step-03	その他のオブジェクト (XMLHttpRequest)	12時間 実習
達成目標		
<ol style="list-style-type: none"> 1. API とは何か他人に説明することができる 2. ネイティブオブジェクトとホストオブジェクトの違いを理解している 3. ブラウザーの API とサードパーティの API の違いを理解している 4. ブラウザーのオブジェクトでどんなことができるのか理解している 5. Ajax とは何か理解している 6. JSON とは何か他人に説明することができる 		

Chapter 11 jQuery		13時間 実習
Step-01	JavaScript のライブラリ「jQuery」	
Step-02	スライドショーを作成する	
達成目標		
<ol style="list-style-type: none"> 1. JavaScript のライブラリやフレームワークについて理解している 2. jQuery とは何か他人に説明することができる 3. jQuery CDN を利用するメリットについて理解している 4. uncompressed と minified の違いを他人に説明することができる 5. jQuery を使って Web ページの見出しの色を変更することができる 6. スライドショーのライブラリ (bxSlider) を使用することができる 		

Chapter 12 アニメーション		14時間 実習
Step-01	Vue.js の使い方	
Step-02	基本的なアニメーション表現	
Step-03	モーショントデザイン	
達成目標		
<ol style="list-style-type: none"> 1. jQuery と Vue.js の違いについて理解している 2. プログレッシブ・フレームワークという概念について理解している 3. Vue.js をインストールして簡単なプログラムを動かすことができる 4. Vue.js で文字列などをフェードイン・アウトする表現ができる 5. Vue.js でイーザング効果 (加速減速) を適用できる 6. リスト項目をアニメーションさせながら置き換えることができる 		

総合演習 (第3~10章の練習問題の復習)		15時間 実習
Chapter 3	変数とデータ型 練習問題の復習	
Chapter 4	条件分岐 練習問題の復習	
Chapter 5	繰り返し 練習問題の復習	
Chapter 6	配列 練習問題の復習	
Chapter 7	関数 練習問題の復習	
Chapter 8	組み込みオブジェクト 練習問題の復習	
Chapter 9	オブジェクト指向プログラミング 練習問題の復習	
Chapter 10	Web API 練習問題の復習	

目次

▶ はじめに	3
▶ 本書の読み方	4

Chapter 1 WebとJavaScript

11

01. JavaScript誕生の歴史	13
▶ JavaScriptとは？	13
▶ JavaScriptとJavaはどう違うの？	13
02. Webブラウザの種類	18
▶ OSの標準ブラウザとサードパーティのブラウザ	18
▶ レンダリングエンジンとJavaScriptエンジン	21
03. JavaScriptとECMAScriptの関係	23
▶ マイクロソフトのJScriptって何？	23
▶ 高校生でもわかるJavaScriptの全体像	24
■ 学習目標のチェックシート・Chapter1のまとめ	30

Chapter 2 準備と基本文法

31

01. JavaScriptを学ぶための準備をしておこう	33
▶ 学習に必要な環境とは？	33
▶ テキストエディタの種類	34
▶ プログラミングは修正の繰り返し	38
02. HTMLファイルを作成する	39
▶ プロジェクトフォルダーとHTMLファイルを新規作成する	39
03. オートコンプリートを活用してコードを記述しよう	42
▶ オートコンプリート機能とは？	42
04. プログラミングの「デバッグ」を体験してみよう	45
▶ コンソールとは？	45
05. 外部のJavaScriptファイルを作成してHTMLに読み込む	49
▶ JavaScriptファイルを新規作成する	49
06. 用途に応じてコメントを記述する	53
▶ コメントを記述してみよう	53
■ 学習目標のチェックシート・Chapter2のまとめ	56

01. 変数とは？	59
▶ 変数を宣言する	59
▶ 変数にデータを記憶する	61
▶ 変数名の付け方	63
▶ 予約語は使用できない	64
▶ 変数宣言の巻き上げとは？	66
02. 演算子の種類と優先順位	68
▶ 演算子とは？	68
03. JavaScriptのデータ型	71
▶ データ型を学ぶ	71
▶ JavaScriptのデータ型はプリミティブ型とオブジェクト型	72
■ 練習問題	74
■ 学習目標のチェックシート・Chapter3のまとめ	75
■ 練習問題の解答	76

01. アルゴリズムって？	79
▶ アルゴリズムの基本	79
02. 体格指数を算出するプログラムをif文で書いてみよう	81
▶ if文で記述する	81
03. 体格指数を算出するプログラムをif/else文で書いてみよう	86
▶ if/else文で記述する	86
▶ 痩せぎみの判定を追加する	87
04. 曜日によって表示を変更するプログラムをswitch文で書いてみよう	91
▶ switch文で記述する	91
■ 練習問題	94
■ 学習目標のチェックシート・Chapter4のまとめ	95
■ 練習問題の解答	96

01. ループ文	101
▶ ループ文とは？	101

02. while文とdo while文を理解する	104
▶ while文とdo while文で記述する	104
03. for文を理解する	108
▶ for文で記述する	108
04. break文とcontinue文を理解する	110
▶ break文とcontinue文で記述する	110
■ 練習問題	114
■ 学習目標のチェックシート・Chapter5のまとめ	115
■ 練習問題の解答	116

Chapter 6

配列

117

01. 配列とは何か？	119
▶ 配列を学ぶ	119
02. JavaScriptの配列はオブジェクト	123
▶ 組み込みオブジェクトとは？	123
03. 連想配列	126
▶ 連想配列を使ってデータを格納する	126
04. 基本的な配列の操作	129
▶ 配列の操作を学ぶ	129
■ 練習問題	133
■ 学習目標のチェックシート・Chapter6のまとめ	135
■ 練習問題の解答	136

Chapter 7

関数

137

01. 関数の基礎知識	139
▶ 関数のどこが便利なの？	139
▶ 関数はどのように定義するの？	140
▶ メソッドでもある関数とメソッドではない関数	142
02. 関数の使い方	144
▶ 独自の関数を定義して呼び出す	144
03. 関数を使って簡単なプログラムを作る	147
▶ 改良前のプログラムを再確認する	147
▶ プログラムを設計する	148
■ 練習問題	154
■ 学習目標のチェックシート・Chapter7のまとめ	156
■ 練習問題の解答	157

01. JavaScriptの組み込みオブジェクト	161
▶ 組み込みオブジェクトとその種類	161
02. Dateオブジェクトを使う	164
▶ Dateオブジェクトで日付や時刻を表示する	164
03. Stringオブジェクトを使う	168
▶ Stringオブジェクトで文字列を操作する	168
04. Numberオブジェクトを使う	172
▶ Numberオブジェクトで数値を表示する	172
■ 練習問題	175
■ 学習目標のチェックシート・Chapter8のまとめ	177
■ 練習問題の解答	178

01. オブジェクト指向とは？	181
▶ オブジェクト指向の概念を学ぶ	181
02. DOMオブジェクトとは？	186
▶ JavaScriptをWebで使うために欠かせない技術	186
03. DOMの操作	189
▶ DOMの構造を理解する	189
▶ DOMを操作する	192
▶ DOMオブジェクトの内容を確認する	199
■ 練習問題	205
■ 学習目標のチェックシート・Chapter9のまとめ	207
■ 練習問題の解答	208

01. APIとは？	211
▶ Webアプリケーション開発で必須のAPI	211
▶ ブラウザーのAPIとサードパーティのAPI	212
02. Webブラウザのオブジェクト	217
▶ Webブラウザのオブジェクトとは？	217
▶ Webブラウザの印刷ダイアログを表示する	219
▶ イベントオブジェクト	223

03. その他のオブジェクト (XMLHttpRequest)	229
▶ XMLHttpRequestを使った処理を確認する	229
■ 練習問題	239
■ 学習目標のチェックシート・Chapter10のまとめ	241
■ 練習問題の解答	242

Chapter 11 jQuery

243

01. JavaScriptのライブラリ「jQuery」	245
▶ ライブラリとフレームワークとは？	245
▶ jQueryの使い方	246
▶ ライブラリなしのJavaScriptコードとjQueryを比較する	248
▶ jQueryでh1要素の色を変更する	252
02. スライドショーを作成する	254
▶ スライドショーを作成する	254
■ 練習問題	261
■ 学習目標のチェックシート・Chapter11のまとめ	263
■ 練習問題の解答	264

Chapter 12 アニメーション

265

01. Vue.jsの使い方	267
▶ JavaScript 学習の全体像	267
▶ Vue.js をインストールする	269
▶ Vue.js を動かしてみる	271
02. 基本的なアニメーション表現	273
▶ フェードイン・アウト	273
▶ ズームイン・アウト	277
03. モーションデザイン	282
▶ フラットデザインとモーションデザイン	282
▶ モーションデザインの重要性	283
▶ CSSアニメーションライブラリの活用	285
■ 練習問題	288
■ 学習目標のチェックシート・Chapter12のまとめ	290
■ 練習問題の解答	291
▶ 索引	292

Chapter

1

Webと JavaScript

JavaScript は、クライアントサイドのスクリプト言語です。Web ブラウザーとエディタがあれば誰でもすぐにコードを書くことができます（この手軽さが JavaScript の魅力のひとつ）。初心者にとって理解しにくいのは、JavaScript の全体像です。全体像を把握するには ECMAScript や API、JavaScript エンジンなどの基礎知識が必要になります。

本章で学ぶこと

▶ 達成目標：

1. JavaScriptがどのような経緯で誕生したか他人に説明できる
2. 代表的な6つのWebブラウザを列挙できる
3. レンダリングエンジンとJavaScriptの役割を理解している
4. ECMAScriptとは何か理解している
5. JavaScriptの全体像について他人にわかりやすく説明できる

▶ 学習ポイント：

- JavaScriptが生まれた歴史的背景を学ぶことができます
- Webブラウザのレンダリングエンジンとは何か解説します
- WebブラウザのJavaScriptエンジンとは何か解説します
- JavaScriptとECMAScriptの関係について詳しく学べます
- JavaScriptの全体像をしっかりと理解することができます

▶ 学習の流れ：

- Step-01 JavaScript誕生の歴史
- Step-02 Webブラウザの種類
- Step-03 JavaScriptとECMAScriptの関係

Step 01

JavaScript 誕生の歴史

JavaScript は、Web サイトやアプリなどのあらゆる開発が手軽にできるプログラミング言語です。誕生の経緯や、関連する企業などを見てみましょう。

JavaScriptとは？

プログラミングは、英会話の習得と同じように2つの学び方があります。ひとつは「最初に文法や単語をしっかりと学習してから会話の練習に進む」方法、もうひとつは「基本をある程度理解したら後は実践しながら（失敗と学びを繰り返しながら）身につけていく」方法です。

どちらが早道になるかは学習する人の態度によって変わってきますが、JavaScriptに関しては後者のほうが「早く慣れることができる」有効な方法だと考えています。実際にコードを書いてプログラムを実行し、デバッグしながら同時に学習していくやり方です。

JavaScript (ジャバスクリプト) は、Web ブラウザーとメモ帳のような簡易エディタがあれば、すぐにコードを書いて実行できるプログラミング言語です。誰でも今すぐに始められる敷居の低さが、JavaScriptの大きな特長だといえるでしょう。HTMLやCSSのように、一般のユーザーでもある程度のレベルであれば習得可能な言語です。

プログラミング言語の中で「異端児」と呼ばれるJavaScriptがどのような経緯で登場したのを見ていきましょう。

JavaScriptとJavaはどう違うの？

JavaScriptは、1995年にネットスケープコミュニケーションズのプログラマーであるブレندان・アイク (Brendan Eich) によって開発されたプログラミング言語です。開発コードネームは「Mocha (モカ)」で、Web ブラウザーの「Netscape Navigator 2.0 ベータ版」リリースに間に合わせるためプロトタイプは10日間 (1995年5月6～15日) で開発されました。1995年9月に「LiveScript (ライブスクリプト)」という名称に変わりましたが、リリース直前になって「JavaScript」に変更されました。

Chapter

1

Chapter

2

Chapter

3

Chapter

4

Chapter

5

Chapter

6

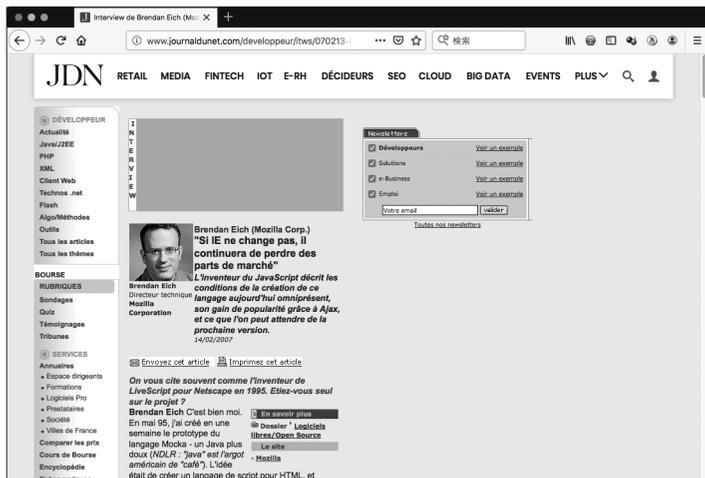
Web
&
JavaScript

▶ 参考：

- プレンダン・アイクのインタビュー記事（2007年2月14日）

たった一人で開発したこと、プロトタイプを10日間で作ったことなどを語っています。

URL : <http://www.journaldunet.com/developpeur/itws/070213-itw-mozilla-eich.shtml>



1995年4月11日、ネットスケープコミュニケーションズは、プログラミング言語「**Java (ジャバ)**」を開発していたサン・マイクロシステムズと業務提携を締結し、(Web ブラウザー開発で競っていた) マイクロソフトに対抗するための施策のひとつとして「JavaScript」を同社のWeb ブラウザーに実装しました。

JavaScriptという命名はあくまで戦略的なもので、Javaとの特別な関係はありませんが、ネットスケープとサンの共同発表（1995年12月4日）では「Javaの簡易版」だと誤解を与えるような表現をしており、大きな混乱を招きました。当時の文献を見ると誤ったJavaScriptの紹介記事が散見されます。

プレندان・アイクはJavaScript開発が評価され、1998年にはネットメディアのCNET Builder.comが主催するWeb Innovator Awardsにて、Web Innovator of the Yearを受賞しています。現在でも多くのテックメディアが「JavaScriptの父」と表現しており、その功績をたたえています。

▶ JavaScript が誕生した 1995 年の出来事

開発期間およびプレスリリース	JavaScript 開発の動向
5~9月	ブレンダン・アイクが「Mocha」を開発
9~12月	「LiveScript」と呼ばれていた期間
12月1日	サン・マイクロシステムズが「JavaScript」の商標登録を申請
12月4日	ネットスケープとサンによる「JavaScript」の共同発表

商標の正式登録 (#2416017) は2000年12月26日です。2010年1月27日、サンがオラクルに買収されたことで「JavaScript」の商標権はオラクルに移転しています。

▶ 補足 (関連企業について) :

JavaScriptはネットスケープコミュニケーションズのWebブラウザ「Netscape Navigator」のスク립ト言語です。Javaとの技術的な関係はありませんが、開発元のサン・マイクロシステムズとは業務提携していますので、ビジネス上の関連性はあったと考えてよいでしょう。

また、JavaScriptの発展には、マイクロソフトとの熾烈な開発競争も大きく影響していますので、関連企業について情報整理しておきましょう。

▶ ネットスケープコミュニケーションズ (Netscape Communications Corporation)

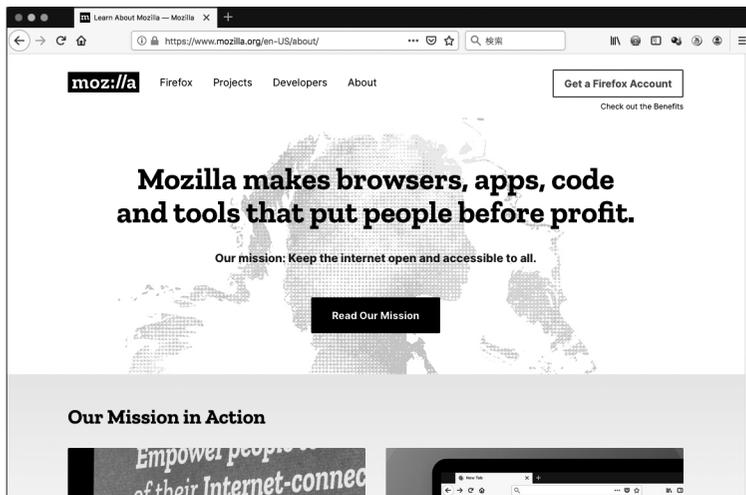
イリノイ大学のNCSA(National Center for Supercomputing Applications :米国立スーパーコンピュータ応用研究所) に在籍していたマーク・アンドリーセン (Marc Lowell Andreessen) が、実業家のジム・クラーク (James H. Clark) に誘われ、設立された企業です。マーク・アンドリーセンは、現在のWebブラウザのルーツといえる「NCSA Mosaic (モザイク)」の開発に携わっていました。

同社は、インターネット黎明期の代表的なブラウザとなる「Netscape Navigator (ネットスケープ・ナビゲーター)」を開発。1995年にリリースされたNetscape Navigator 2.0に、クライアントサイドのスク립ト言語「JavaScript」が実装されました。

1998年、同社は大手ネット企業のAOLに買収されますが、Netscapeはオープンソースソフトウェアとなり、非営利団体の「Mozilla Organization」が引き継ぎ、2003年7月15日に「Mozilla Foundation」を設立、2004年にWebブラウザ「Mozilla Firefox (モジラ・ファイアフォックス)」のバージョン1をリリース、2005年8月3日には完全子会社となる「Mozilla Corporation」を設立しました。

- Learn About Mozilla

<https://www.mozilla.org/en-US/about/>



▶ サン・マイクロシステムズ (Sun Microsystems)

1982年に設立されたコンピュータ企業で、業務用コンピュータの開発製造やソフトウェア開発、ITサービスなど事業を幅広く展開、マイクロソフトと並ぶ大企業に成長。1995年に開発されたオブジェクト指向プログラミング言語「Java」が人気を博し、1996年1月23日に「JDK (Java Development Kit) 1.0」がリリースされます。

1995年4月11日、ネットスケープコミュニケーションズとの業務提携を締結し、12月4日には「JavaScript」の共同発表を行っています。なお、JavaScriptの商標権はサンが取得していますが、2010年1月27日、米国のソフトウェア開発企業「オラクル (Oracle Corporation)」がサンを買収し、商標権はオラクルに移転しています。

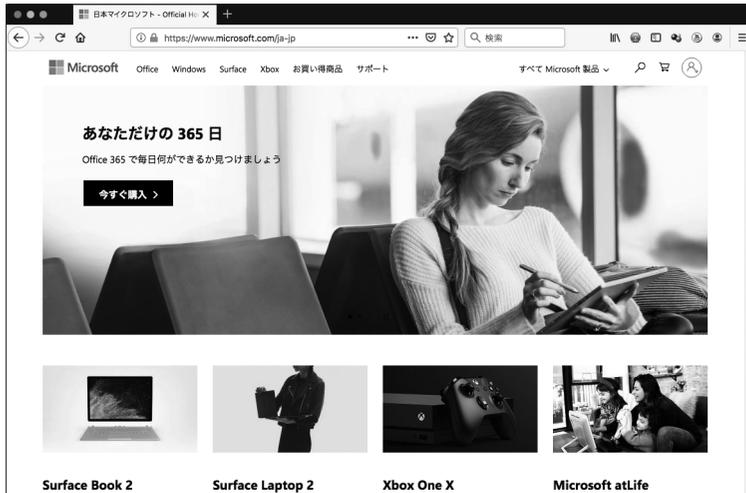
▶ マイクロソフト (Microsoft Corporation)

WindowsやOfficeなどで世界的に知られている大企業。1995年8月に「Internet Explorer (インターネット・エクスプローラー)」をWindowsの機能拡張パッケージに収録。1996年8月にリリースされたInternet Explorer 3.0には、JavaScriptと類似したスクリプト言語「JScript (ジェイ・スクリプト)」が実装されています。JavaScriptのライセンス供与が認められなかったのが独自開発の理由です。

ネットスケープコミュニケーションズとのユーザー獲得をめぐる開発競争は、Web標準の技術であるHTMLを独自に拡張するなど、Webの信頼性を低下させる大きな問題に発展していきます。この混乱期は「ブラウザ戦争」と呼ばれ、Internet Explorerが圧倒的なシェアを得る2000年頃まで続きました。

- Microsoft

<https://www.microsoft.com/ja-jp>



Internet Explorerの最終バージョンは、2013年10月に公開されたIE 11ですが、Windows 10で「Microsoft Edge」が標準ブラウザに切り替わったことで、IEのユーザー数は減り続けています。2020年1月14日にはIE 10のサポートが終了するため、IEはバージョン11のみとなります。

IEは過去のブラウザになりましたが、古いWindowsを使っているユーザーがまだ多いため、しばらくはEdgeとIE 11が混在した状態が続きます。よって、Webアプリ開発では両方の検証が必要になります。

Webブラウザの種類

Web ページを閲覧するために必要なアプリが「Web ブラウザー」です。ここでは、Web ブラウザーの種類と、プログラムを解釈して表示する「エンジン」について解説します。

OSの標準ブラウザとサードパーティのブラウザ

Webブラウザは、Web ページをスクロールしながら読む閲覧用のアプリケーションソフトです。OS にインストール済みの標準ブラウザと他の企業が提供するサードパーティのブラウザに大別することができます。

Windows が搭載されているパソコンやタブレット PC には「**Microsoft Edge**」、macOS が搭載されている Mac や iOS が搭載されている iPhone や iPad には「**Apple Safari**」、そして Android が搭載されているデバイスには「**Google Chrome**」がインストールされています。

その他、Mozilla Foundation の「**Mozilla Firefox**」や Opera Software の「**Opera**」などの Web ブラウザーが無償で提供されており、誰でも自由にインストールすることができます。

Microsoft Edge は Windows 10 以降の標準ブラウザですが、それ以前の古い OS では「Internet Explorer」が使用されています。Internet Explorer 8~10 はすでにサポートが打ち切られており、マイクロソフトは Internet Explorer の非推奨を表明しています。最終バージョンである 11 のみサポートが継続されており、利用者もいますのでしばらくは対応が必要です。

▶ 参考：

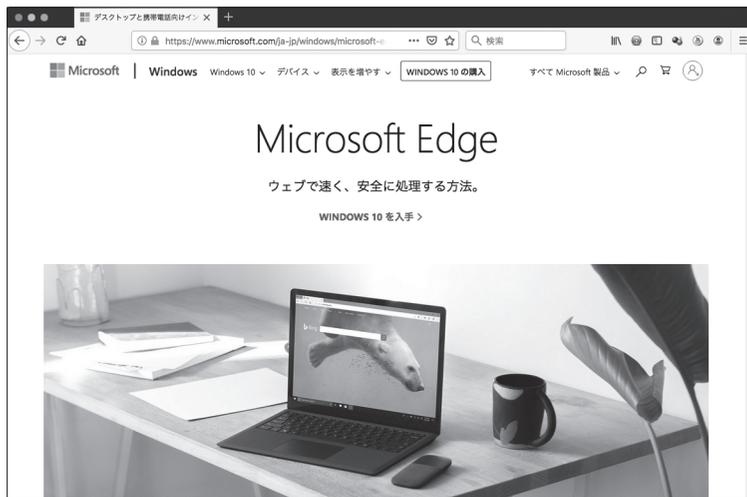
- Internet Explorer の今後について [Microsoft Developer Network]

短縮 URL : <http://bit.ly/2LJCHe4>

Web ブラウザー名	開発企業・団体	種類
Microsoft Edge (マイクロソフト・エッジ)	Microsoft	Windows 10以降の標準ブラウザ
Internet Explorer (インターネットエクスプローラー)	Microsoft	Windows 標準ブラウザ
Safari (サファリ)	Apple	macOS および iOS 標準ブラウザ
Google Chrome (グーグル・クローム)	Google	Android 標準ブラウザ (他の OS にも提供)
Mozilla Firefox (モジラ・ファイアーフォックス)	Mozilla Foundation	サードパーティ製品 (ユーザーが自分でインストールする)
Opera (オペラ)	Opera Software	サードパーティ製品 (ユーザーが自分でインストールする)

- Microsoft Edge

<https://www.microsoft.com/ja-jp/windows/microsoft-edge>



- Apple Safari

<https://www.apple.com/jp/safari/>



- Google Chrome

<https://www.google.co.jp/chrome/>



- Mozilla Firefox

<https://www.mozilla.org/ja/firefox/new/>

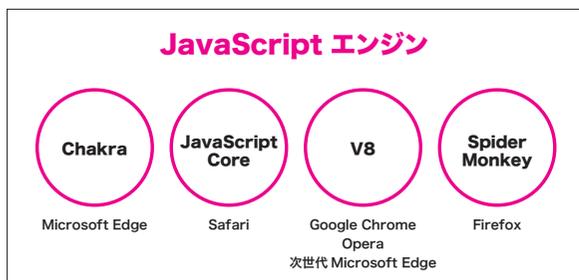


レンダリングエンジンとJavaScriptエンジン

Web ブラウザーには、HTML や CSS を解釈してブラウザーのウィンドウに文字や画像を表示するための「**レンダリングエンジン**」が搭載されています。クルマのエンジンと同様にメーカーによって差異があり、あるエンジンでは表示できても、他のエンジンでは表示できない、といった HTML や CSS の実装の違いがあります。

JavaScript のエンジンも Web ブラウザーによって異なり、Microsoft Edge は「**Chakra**」、Internet Explorer は「**Jscript9 の Chakra**」、Safari は「**JavaScriptCore**」、Google Chrome と Opera は「**V8**」、Mozilla Firefox は「**SpiderMonkey**」を搭載しています。

※ 2018 年 12 月、次世代の Microsoft Edge は Chromium ベースとして開発されることが発表され、2019 年 4 月 8 日にプレビュー版が公開されました。JavaScript のエンジンも、Chakra から V8 に切り替わります。



▶ Web ブラウザーが採用しているエンジン

Web ブラウザー名	レンダリングエンジン名	JavaScriptエンジン名
Microsoft Edge	EdgeHTML (エッジエイチ ティーエムエル)	Chakra (チャクラ)
Internet Explorer	Trident (トライデント)	Jscript9.dll Chakra
Safari	WebKit (ウェブキット)	JavaScriptCore (Nitroとも 呼ばれる)
Google Chrome	Blink (プリנק)	V8 (ブイエイト)
Mozilla Firefox	Gecko (ゲッコ) + 一部は Servo (サーボ)	SpiderMonkey (スパイダー モンキー)
Opera	Blink	V8
次世代 Microsoft Edge	Blink	V8

▶ 参考：

- **Microsoft Edge - Build 2019**からのすべてのニュース
短縮 URL : <http://bit.ly/2Y7O1r8>
- 2017年11月30日に正式リリースされた **Android 版の Microsoft Edge** には、**Blink** が採用されています。
短縮 URL : <http://bit.ly/32YnIYM>



Step 03

JavaScriptと ECMAScriptの関係

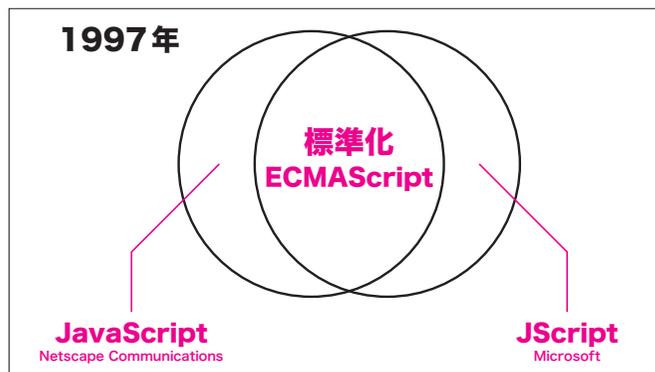
ここでは、JavaScriptを理解する上で必須である標準化規格「ECMAScript」が策定された背景から、JavaScriptの全体像までをわかりやすく解説します。

マイクロソフトのJScriptって何？

JavaScriptの文献には「ECMAScript (エクマスクリプト)」という名称がよく出てきます。初心者にとって、JavaScriptの全体像を理解するのは容易ではありません。なぜなら、歴史的背景を理解しておく必要があるからです。

Step-01では「JavaScript誕生の歴史」について学習しましたが、ネットスケープコミュニケーションズが開発した「JavaScript」と、マイクロソフトが開発した“JavaScriptと類似した”スクリプト言語「JScript」の存在を思い出してください。互換性に問題がある一見同じような言語が異なるWebブラウザに搭載されているのですから、Web開発者にとってこんな厄介なことはありません。両方のブラウザで正しく動作するプログラムを書くのは至難の業だったのですから。

この2つのスクリプトの互換性を高めるには、文法などの基本的な仕様を「標準化」しなければいけません。



1997年、情報通信システム分野における標準化団体である「Ecma International (エクマ・インターナショナル)」が、ネットスケープコミュニケーションズの「JavaScript」とマイクロソフトの「JScript」の共有化できる部分を抜き出し、標準化しました。この標準化された仕様が「ECMAScript」です。

現在のブラウザベンダーは「ECMAScript」を実装しているため、互換性の問題はほぼ解消しています。

Chapter

1

Chapter

2

Chapter

3

Chapter

4

Chapter

5

Chapter

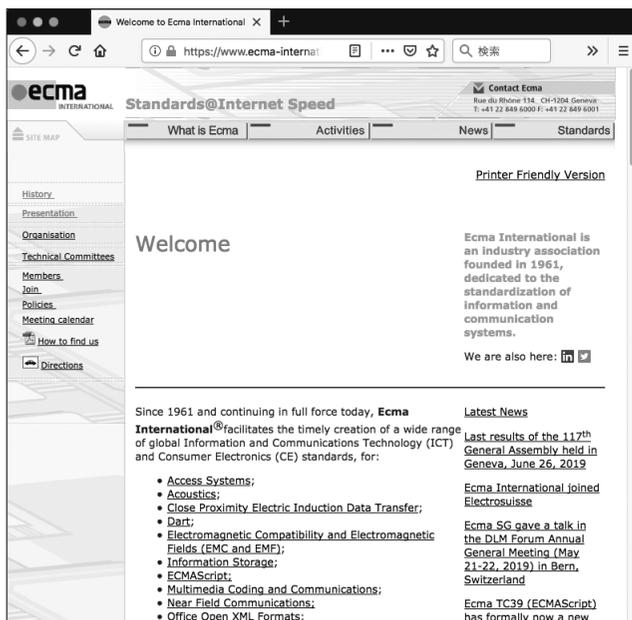
6

WebとJavaScript

※ECMAScriptは、Ecma InternationalのTC-39委員会が標準化を実行し、ECMA-262というドキュメントで公開されました。

- Ecma International

<https://www.ecma-international.org/>



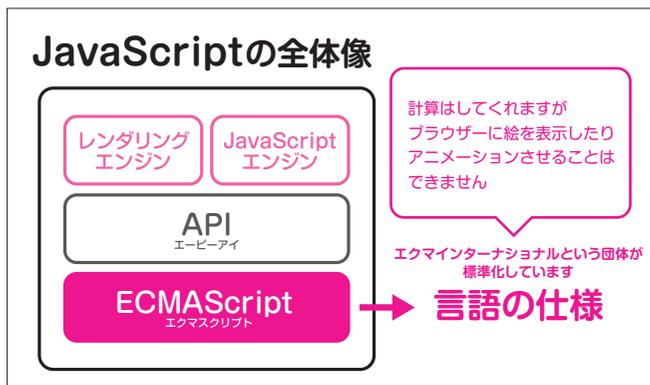
高校生でもわかるJavaScriptの全体像

JavaScriptの歴史的背景について理解できたと思いますので、続いて「JavaScriptの全体像」について学びましょう。初心者の皆さんにとっては（コーディングを習得する前に）必ずやらなくてはならない重要な学習です。全体像が理解できていないと、学習効率にも影響してしまいます。

このパートはより理解しやすいように「高校生でもわかるJavaScriptの全体像」と題して、情報をコンパクトにして進めていきます。詳しい解説はこの後の章でやっていきますので、ここでは大まかに把握することを目標とします。

① JavaScript の箱があります。中には「EcmaScript」「API」、そして2つのエンジンが入っています。

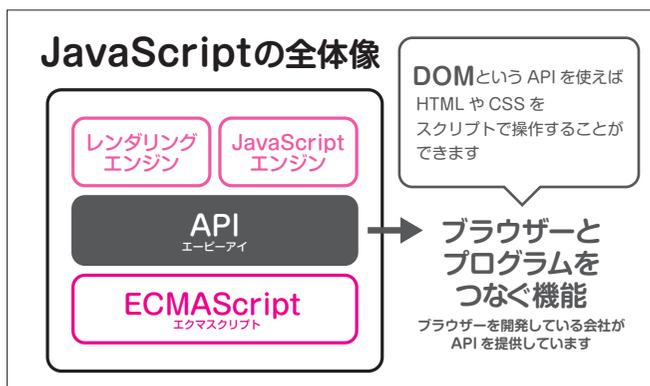
一番に下にある（基盤の部分の）EcmaScriptは「**言語の仕様**」です。計算をさせることはできますが、Webブラウザに画像を表示したり、アニメーションを設定させることはできません。



②真ん中にある「API」はWebブラウザとプログラムをつなぐ機能の集まりです。

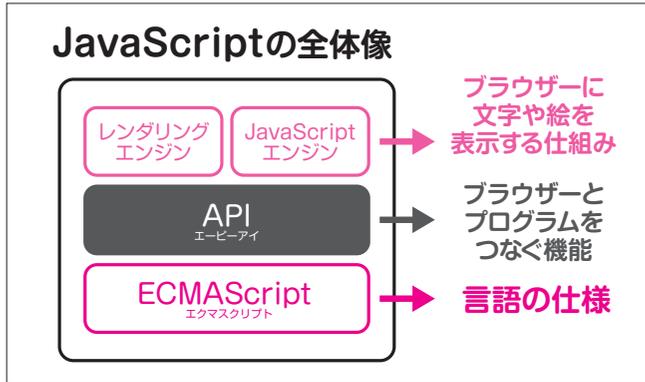
例えば、DOMというAPIを使用すればスクリプトでHTMLやCSSを操作することができます。このAPIを使えば、指定した画像を表示したり、アニメーションを設定することも可能になります。

EcmaScriptは標準化団体の「Ecma International」が策定していますが、APIはWebブラウザを開発している会社が提供しています。



③一番上の2つのエンジンは「処理」機能です。

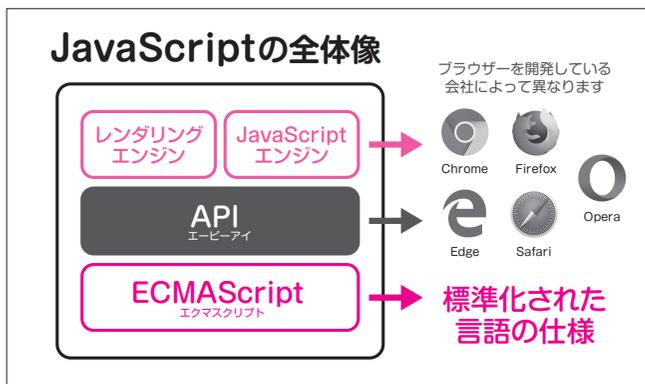
この仕組みによってWebブラウザのウィンドウにテキストや画像をリアルタイム表示したり、スクリプトを解釈して実行したりしてくれます。レンダリングエンジンとJavaScriptエンジンについてはすでに学習済みです。



④全体像についてまとめましょう。

「EcmaScript」は標準化された言語の仕様です。JavaScriptの核の部分だと捉えてください。JavaScriptの文法の学習は「**EcmaScriptの学習**」ということになりますので覚えておきましょう。

APIとレンダリングエンジン/JavaScriptエンジンはWebブラウザ固有の技術になります。



ECMAScript のバージョン

HTMLやCSSにバージョンがあるように **ECMAScriptにもバージョンがあります**。新しいバージョンは時間をかけてゆっくりと普及していきますので、まずは（現時点で）普及率の高いバージョンをしっかりと習得した上で、最新のバージョンの機能を覚えていくことになります。

2019年7月現在、習得すべきバージョンは2015年6月に公開された「**ECMAScript 2015 6th Edition**」です。略して「ES6」とも呼ばれます。

▶ ECMAScript 2015 [ES6] の Web ブラウザーの対応状況

<http://kangax.github.io/compat-table/es6/>

Feature name	Current browser	Compilers/polyfills														FF B		
		Traceur	Babel 6 ± core-js 2	Babel 7 ± core-js 2	Babel 7 ± core-js 3	Closure 2019.07	TypeScript ± core-js 3	es6-shim	Konq 4.14 ¹⁾	IE.11	Edge 17	Edge 18	FF 60 ESR	FF 67 ESR	FF 68 ESR			
Optimisation																		
▶ propept 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
Syntax																		
▶ default function parameters	7/7	4/7	4/7	4/7	4/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
▶ rest parameters	5/5	4/5	3/5	3/5	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
▶ spread syntax for iterable objects	15/15	15/15	13/15	13/15	14/15	11/15	14/15	0/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15
▶ object literal extensions	6/6	6/6	6/6	6/6	6/6	5/6	6/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
▶ for...of loops	9/9	9/9	9/9	9/9	9/9	6/9	9/9	0/9	0/9	0/9	9/9	9/9	9/9	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	4/4	2/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4
▶ template literals	7/7	6/7	6/7	6/7	6/7	5/7	5/7	0/7	0/7	0/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7	7/7
▶ RegExp "y" and "u" flags	6/6	4/6	4/6	4/6	4/6	0/6	0/6	0/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6
▶ destructuring: declarations	22/22	20/22	21/22	21/22	21/22	20/22	21/22	0/22	0/22	0/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22	22/22
▶ destructuring: assignment	24/24	23/24	24/24	24/24	24/24	22/24	24/24	0/24	0/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
▶ destructuring: parameters	24/24	19/24	21/24	21/24	21/24	20/24	21/24	0/24	0/24	0/24	23/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24
▶ Unicode code point escapes	2/2	1/2	1/2	1/2	1/2	1/2	1/2	0/2	0/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2
▶ Unicode escape escapes	3/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3

ECMAScript 2015以前はすべての仕様の合意がなければリリースできなかったため、策定に大変時間がかかっていました。バージョン3から5.xまでは10年以上もかかっています。ECMAScript 2016以降は、仕様策定のプロセスが変わり、次の図で示されているとおり毎年リリースされています。

Chapter

1

Chapter

2

Chapter

3

Chapter

4

Chapter

5

Chapter

6

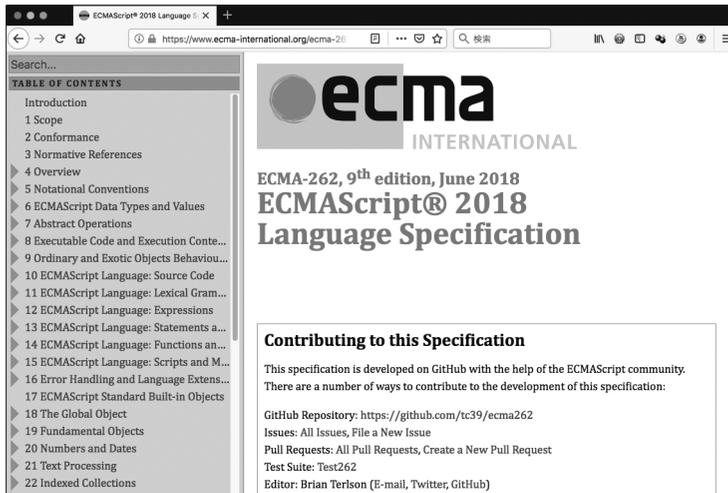
Web & JavaScript

▶ ECMAScript のバージョンとリリース日

リリース日	ECMAScript のバージョン	略称
1995年	Netscape 2.0のベータ版に最初のJavaScriptを実装	
1996年3月	Netscape 2.0が、JavaScript 1.0 を実装	
1996年8月	Netscape 3.0が、JavaScript 1.1 を実装	
1996年11月	ECMAScript 言語仕様の開発が始まる	
1997年6月	ECMAScript 1th Edition が、Ecma 総会で採択	ES
1997年7月	Netscape 4.0および4.05が、JavaScript 1.2 を実装	
1998年4月	ISO/IEC 16262 international standard 対応（仕様は変わらず）	
1998年6月	ECMA-262 2th Edition が、Ecma 総会で承認	ES2
1998年10月	Netscape 4.06および4.7が、JavaScript 1.3 を実装	
1999年12月	3th Edition が、Ecma 総会で採択	ES3
2002年6月	ISO/IEC 16262 : 2002 国際標準化の公開	
	4th Edition は破棄（仕様の一部は6th Editionへ）	ES4
2009年12月	5th Edition が、Ecma 総会で採択	ES5
2011年6月	5.1 が、Ecma 総会で採択	ES5.1
2015年6月	ECMAScript 2015 6th Edition（※大きな改訂）	ES2015 [ES6]
2016年6月	ECMAScript 2016 7th Edition	ES2016 [ES7]
2017年6月	ECMAScript 2017 8th Edition	ES2017 [ES8]
2018年6月	ECMAScript 2018 9th Edition	ES2018 [ES9]

- ECMAScript® 2018

<https://www.ecma-international.org/ecma-262/9.0/index.html>



※バージョン4は意見がまとまらず破棄されましたが、Adobe Flashのスクリプト言語「ActionScript」はこのバージョンの草案を採用しています。W3Cが策定しているHTMLもバージョン3.0が廃案になっています（バージョン3.2が勧告されています）。

▶ ECMAScript の仕様書

- ECMAScript 2015

[HTML] <http://www.ecma-international.org/ecma-262/6.0/index.html>

[PDF] <http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf>

- ECMAScript 2016

[HTML] <http://www.ecma-international.org/ecma-262/7.0/index.html>

[PDF] 短縮 URL : <http://bit.ly/2Gyrzwk>

- ECMAScript 2017

[HTML] <http://www.ecma-international.org/ecma-262/8.0/index.html>

[PDF] 短縮 URL : <http://bit.ly/2oP7qwb>

- ECMAScript 2018

[HTML] <http://www.ecma-international.org/ecma-262/9.0/index.html>

[PDF] <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>

Chapter
1

Chapter
2

Chapter
3

Chapter
4

Chapter
5

Chapter
6

Web
&
JavaScript