Engineer

JavaScript

JS ブラウザーが実行するプログラミング言語

どのようなものか

JavaScriptは、ECMAScriptとして標準化された、動的型付けのプログラミング言語です。どのウェブブラウザーでも実行でき、Node.jsなどのランタイムを通して、サーバーやビルド用マシンでも動きます。JSとも略します。標準の年次版はES2015、ES2024などと呼び、古い番号のES5、ES6も今なお使います。「vanilla JavaScript」は、フレームワークを重ねていない言語そのものを意味します。名前と波括弧を使う構文の一部を除けば、Javaとの共通点はほとんどありません。

Brendan EichがNetscapeで最初の版を書き、Navigator 2.0ブラウザーに搭載されました。Ecma Internationalは、1997年6月に標準ECMA-262の初版を採択し、委員会TC39は、ECMAScript 2016以降、毎年新しい版を発行しています。提案が仕様に入るのは、委員会の手順のstage 4に達したときだけです。この段階には、既に出荷している実装が必要なため、仕様文と実行エンジンの食い違いが長く続くことは、めったにありません。

言語自体は、外の世界についてほとんど定めません。ECMA-262は、構文、型、オブジェクト、Promise、モジュールを定義し、入出力は、それを動かす環境に任せています。ブラウザーでは、その環境はウェブプラットフォームです。文書、イベント、タイマー、ネットワーク要求があり、主にHTML Standardと関連する標準が定めています。Node.jsは、代わりにファイル、ソケット、プロセスを提供します。この分担が、よくある戸惑いを説明します。JavaScriptとして完全に正しいコードでも、ページからサーバーへ移すと壊れることがあります。言語は移っても、その環境は残ったままだからです。

ブラウザーでは、ページのスクリプトが、レイアウト、描画、入力と、一つのメインスレッドを共有します。タスクが完了するまで、ブラウザーは次のクリックに応答できません。Promiseと `async` 関数は、待ち時間でそのスレッドをふさが

ないようにします。一方、計算は、Web Workerへ移さない限り、そのスレッドに残ります。数値はIEEE 754の倍精度値なので、整数を正確に表せるのは2^53 – 1までで、それより大きな値には `BigInt` が必要です。

概念を判断に生かす

「JavaScriptとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 読み込みが速く、スマートフォンでも使え、お客様がどのブラウザーを使っていても同じように動く画面を提供します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 フルスタックのGIS地図開発を指揮 — DBからブラウザーまでのフルスタック地図開発

フルスタックのGIS地図開発を指揮 お問い合わせ →

なぜ重要なのか

実際に判断すべきなのは、サイトを読む人にどれだけ配信するかです。JavaScriptは、サーバーと往復せずにページを動かす言語ですが、配信するスクリプトは、一キロバイト残らず、配信側が管理できない機器上で、ダウンロード、解析、コンパイル、実行を経ます。開発者のノートパソコンでは瞬時に感じるバンドルの処理も、中程度の性能のスマートフォンでは、数秒かかり得ます。その分、閲覧者のバッテリーを消耗させ、閲覧者を待たせます。

スクリプトは、ページが最も攻撃にさらされる箇所でもあります。攻撃者のスクリプトが実行されれば、ページの権限で動きます。閲覧者が入力する内容を読み、ページが送信できる先なら、どこへでも送れます。その危険を抑えるのが、Content Security Policyの役割です。実行可能なJavaScriptを配信しなければ、サイト自身のブラウザースクリプトという攻撃対象はなくなりますが、信頼できない入力には、引き続き安全な処理と、厳しく制限するポリシーが必要です。パッケージのエコシステムには、もう一つの危険があります。一般的なフロントエンドのビルドは、間接的に依存する数百のパッケージを取り込み、そのすべてが、プロジェクト自身のコードと同じ権限で動きます。

一方で、利点も確かにあります。システムの両側で同じ言語を動かせば、ブラウザーとサーバーで、検証規則やデータ構造を共有できます。オフラインでの状態管理、ドラッグ＆ドロップ、地図、リアルタイムの共同作業など、多くのスクリプトを必要とする製品もあります。機能ごとに、サーバーへ問い合わせずに、閲覧者のブラウザーで実行する必要があるかを問います。なければ、サーバーかビルド時に描画し、HTMLを配信します。

関連する選択が、JavaScriptとTypeScriptです。TypeScriptは、型検査とビルドの段階を加え、通常のJavaScriptへコンパイルします。複数人が毎週変更するコードベースでは、型検査は通常、その費用に見合います。静的ページに百行の機能を加える程度なら、JavaScriptと厳格なリンターで足ります。

実際の活用例

このサイトは、実行可能なJavaScriptを含まない静的ページを閲覧者に届けます。検索のコードは残していますが、無効にしています。JavaScriptは、今もビルドと検査のツールで動きます。次の三つのページは、過去にブラウザー向けに認めていた例外と、ブラウザーの外で行う作業を示します。

以前の三言語版には、実行時のバックエンドがなく、自社製のスクリプトを一つだけ配信していました。Hugoビルドの記録は、その限定した例外を説明しています。実績一覧のフィルターで、スクリプトが動くまでは操作部を隠していました。108のテンプレートファイルには、53のパーシャルと18のショートコードが含まれ、いずれもサイトのビルド時に処理を行いました。記録は、当時の選択の代償も示しています。対話的な機能はビルド時に解決する必要があり、検索インデックスは費用を見積もって保留し、配信しませんでした。これらの数値と例外が示すのは、以前の実装であり、現在、閲覧者に配信するコードではありません。

同じコンテンツは、スクリプトを置く余地がまったくない、GeminiとGopherでも公開します。記録された実装段階では、一回のビルドが、各形式に777文書ずつ出力しました。ページと同じソースを使っています。別のビルド比較では、それらの出力形式を追加したとき、HTMLは一バイトも変わらなかったと確認しました。この結果は、その後の編集や見た目の変更を対象にしていません。このサイトは、文書を読むのに一メガバイトのスクリプトは要らないと論じています。スクリプトを保持できない形式は、その主張を最も分かりやすく示します。

このサイトのJavaScriptコードの大半は、ツール用です。リンク先の作業に記録された実装段階では、サイトの品質ゲートは、三十二本のスクリプト、10,242行のJavaScriptからなり、当時、リポジトリーで最も大きなコード群でした。一時期は、その部分だけ、整形もリンターによる検査も型検査も行っていませんでした。フォーマッターとリンターの管理下に置く作業は、フォーマッターをコードに合わせるところから始めました。リポジトリーの既定の四スペースでは、最大の検査スクリプトの2,173行を書き換えることになりましたが、対象ファイルだけ二スペースに設定すると、書式のずれとして残ったのは38行でした。その

後、リンターが13件の問題を指摘し、13件すべてを、抑制せず修正しました。検査スクリプトのバグは、失敗すべき検査が合格する形で現れます。そのため、品質ゲート自身のコードにも、ほかの対象と同じ管理を行います。

問題が起きるところ

以下の問題は、言語の仕組みから生じるもので、それぞれに既知の対処法があります。

表示はできても、一秒以上タップに反応しません。 大きなバンドルのハイドレーション、大きなJSON応答の解析、数千行の並べ替えなど、長いタスクが、入力を扱う唯一のスレッドを占有します。ページのルートごとにバンドルを分割し、残りを後回しにして、最初の操作に必要なコードを減らします。長い作業は、イベントループに制御を返す小さな単位に分け、重い計算はWeb Workerへ移します。コードを書いたノートパソコンではなく、遅いスマートフォンで計測します。

価格や識別子が少し違う値になります。 数値はすべて二進浮動小数点値なので、小数は丸められ、非常に大きな整数は、隣の整数と区別できなくなります。典型的なのが、JSONの数値として送った、64ビットのデータベースキーです。

```
0.1 + 0.2 === 0.3; // false
Number.MAX_SAFE_INTEGER; // 9007199254740991
9007199254740993 === 9007199254740992; // true
```

金額は、セントなどの補助通貨単位を使って整数で保持し、大きな識別子は文字列で送り、正確な計算が必要な箇所では `BigInt` を使います。

比較で違う値を通してしまいます。 `==` 演算子は、比較前に被演算子を変換するため、 `"" == 0` も `"0" == false` もtrueになります。一貫して `===` を使い、リンターの `eqeqeq` 規則で守らせれば、変換は、コードが明示的に求めたときだけになります。

利用者のテキストがコードとして動きます。 信頼できないテキストを `innerHTML` に代入したり、文字列の連結でマークアップを作ったり、入力を `eval` に渡したりすると、データがスクリプトになります。テキストは `textContent` で書き込

み、nonceかハッシュだけでスクリプトを許可するContent Security Policyを送ります。依存先の一覧は短く保ち、ロックファイルで固定し、各更新で取り込む内容を読んで確認します。

ブラウザーで動くコードが、サーバーでは異常終了します。 言語だけが移り、環境は残っています。`window` と `document` は、ブラウザーにしか存在しません。Node.jsは、ブラウザーが見ない設定に応じて、ファイルをCommonJSかESモジュールとして読みます。実行環境固有の呼び出しを境界に集め、`"type": "module"` を `package.json` で明示的に設定し、各コードを、実際に動かす環境で実行します。

参考資料

- ECMAScript言語仕様 — ECMA-262の継続的に更新されるドラフトです。各エンジンが実装する本文で、提案がstage 4に達するのに合わせて更新されます。
- TC39の手順 — 提案が言語に加わるまでに経る段階と、各段階の要件です。
- Ecma InternationalのECMA-262 — 公開済みの年次版です。プロジェクトが対象とする版を示すときに使います。
- HTML Standard：イベントループ — ページのスクリプトに応じて、ブラウザーがタスク、マイクロタスク、描画の実行を調整する仕組みです。
- HTML Standard：script要素 — クラシックスクリプトとモジュールスクリプト、`async` と `defer` が実行時期を変える仕組みです。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「JavaScriptとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/javascript/