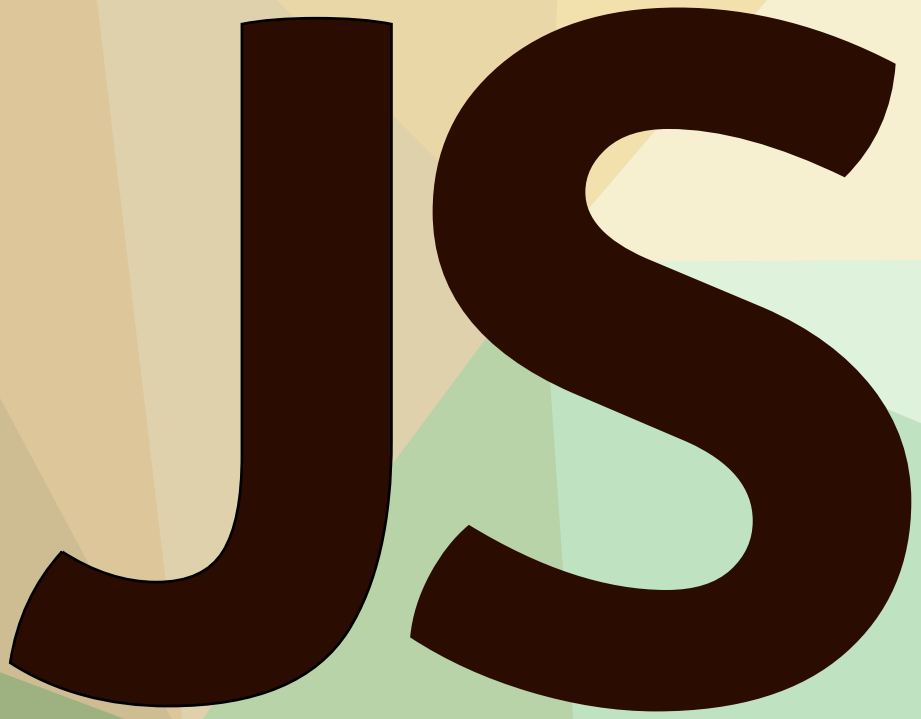
パート一

# JavaScript 言語




Ilya Kantor

**ビルド日時 2024年1月5日**

チュートリアルの最後のバージョンは https://ja.javascript.info です。.

私たちはチュートリアルを改善するために常に努力しています。間違いが見つかった場合は、github に記入してください。.

ここでは JavaScript をゼロから始め、OOP(オブジェクト指向プログラミング)のような高度なコンセプトを学びます。

ここでは、環境固有の注意事項は最小限に抑えて、言語そのものに集中しています。

# 導入

JavaScript言語とその開発環境について

## JavaScript 入門

JavaScript について、何が特別なのか、それを使ってできることや他のどの技術と上手くやるのか見てみましょう。

### JavaScript とは？

*JavaScript* は当初 *“Webページを活かすため”* に作られました。

この言語のプログラムは *スクリプト* と呼ばれます。それらはHTMLの中に書かれ、ページが読み込まれると自動的に実行されます。

スクリプトはプレーンテキストとして提供され、実行されます。 特別な準備や、実行するためのコンパイルは必要ありません。

この点において、JavaScript は Java とは大きく異なります。

> **なぜ JavaScript と言うのでしょう?**
>
> JavaScript が作られたとき, 当初は別の名前を持っていました: “LiveScript”。しかし当時 Java 言語が非常に人気であったため、Java の “弟” として新しい言語を位置づけるのが良いと判断されました。 しかし、それ以降の進化により、JavaScriptは完全に独立した言語になり、ECMAScript と呼ばれる独自の仕様を持ちました。現在 Java とは全く関係ありません。

現在、JavaScript はブラウザだけでなく、サーバ上でも実行することができます。実際には JavaScript エンジン と呼ばれるプログラムが存在するデバイスであれば実行することができます。

ブラウザにはエンジンが組み込まれており、“JavaScript 仮想マシン” と呼ばれる場合があります。

異なるエンジンは異なる “コードネーム” を持っています。例えば:

- V8 – Chrome と Opera.
- SpiderMonkey – Firefox.
- …IEはバージョンによって “Trident” や “Chakra”, Microsoft Edge 用の “ChakraCore”, Safari 用の “Nitro” や “SquirrelFish” 等のように他のコードネームもあります。

これらの用語は、インターネット上の開発者の記事で使用されているため、覚えておくと良いでしょう。 たとえば、“ある機能 X がV8でサポートされている” と言った場合、おそら

くChromeとOperaで動作します。

**エンジンはどのように動く?**

エンジンは複雑ですが、基本は単純です。

1. エンジン (ブラウザの場合は組み込まれています) はスクリプトを読み(“パース”)ます。
2. その後、スクリプトを機械語に変換(“コンパイル”)します。
3. 機械語が実行されます。非常に早く動作します。

エンジンは処理の各ステップで最適化を行います。実行時にコンパイルされたスクリプトも見ており、そこを流れるデータを分析し、それ基づいて機械語を最適化します。最終的に、スクリプトはとても速く実行されます。

## ブラウザ内のJavaScriptができることは？

モダンなJavaScriptは “安全な” プログラミング言語です。それは、メモリやCPUのような低レベルのアクセスは提供しません。なぜなら、当初はそれらを必要としないブラウザ用に作成されたものだからです。

JavaScript の機能は、実行される環境に大きく依存します。 例えば、Node.js では、JavaScriptが任意のファイルを読み書きしたりできる機能をサポートしています。

ブラウザ内のJavaScriptは、Webページの操作、ユーザやWebサーバとのやり取りに関する様々なことを実行できます。

たとえば、ブラウザ内のJavaScriptは次のようなことが可能です:

- 新たなHTMLをページに追加したり、既存のコンテンツの変更やスタイルの変更をする。
- ユーザーの操作(マウスのクリック、ポインタの動き、キーの押下など)に反応する。
- リモートサーバへネットワーク越しのリクエストを送り、ファイルのダウンロードやアップロードをする( AJAX や COMET と呼ばれる技術)。
- クッキーの取得と設定、訪問者への質問やメッセージを表示する。
- クライアント側でデータを記憶する(“ローカルストレージ”)。

## ブラウザ内のJavaScriptで出来ないことは？

ブラウザでは、JavaScriptの機能はユーザの安全のために制限されています。 その目的は、悪意のあるWebページがプライベートな情報へアクセスしたり、ユーザデータへ危害を加えることを防ぐことです。

制限の例として、次のようなものがあります:

- Webページ上のJavaScriptは、ハードディスク上の任意のファイルの読み書きや、それらのコピー、プログラムの実行をすることができません。OSのシステム機能に直接アクセスすることはできません。

現代のブラウザは、ファイルを扱うことはできますがアクセスは制限されており、ブラウザウィンドウへのファイルの “ドロップ” や、`<input>` タグを経由したファイル選択と言ったユーザの特定の操作のみを提供しています。

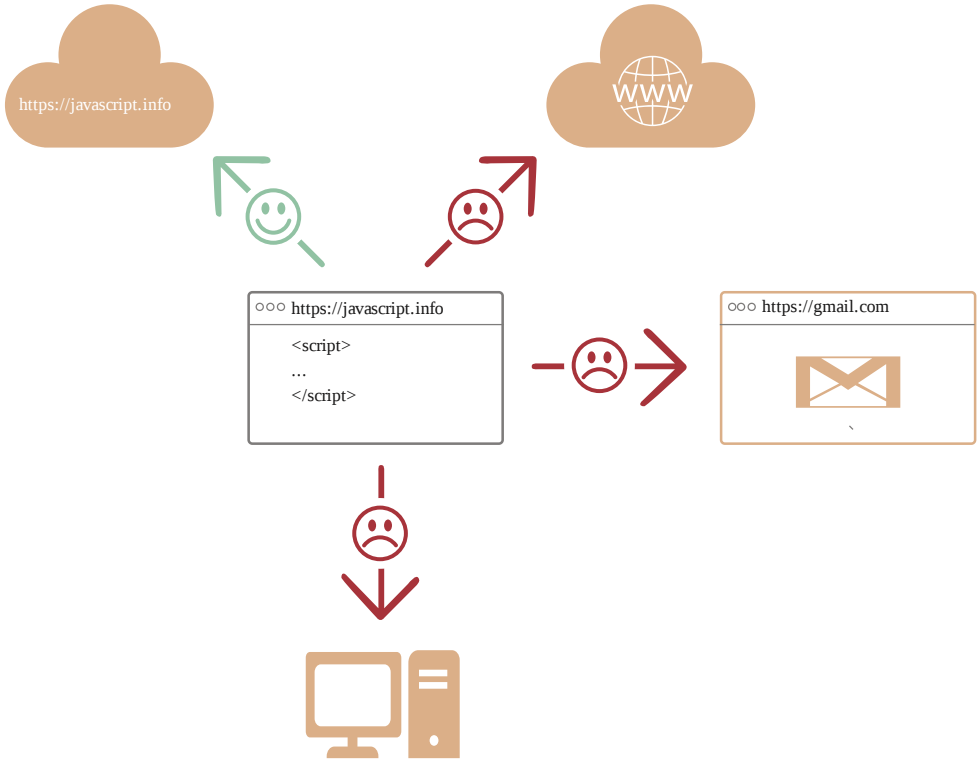
カメラ/マイクやその他デバイスとやり取りする方法はありますが、ユーザの明示的な許可が求められます。したがって、JavaScriptが有効なページがWebカメラを密かに有効にしたり、それを利用して利用者の周囲を観察したり、NSA に情報を送信すると言ったことはできません。

- 異なるタブやウィンドウは一般的にお互いについて知りません。時々、例えばあるウィンドウがJavaScriptを利用して別のウィンドウを開くケースはあります。しかし、この場合においても、あるページからのJavaScriptは、別のサイト（異なるドメイン、プロトコル、ポート）からのものである場合は、アクセスすることはできません。

  これは “Same Origin Policy” と呼ばれています。回避するためには、 *両方のページ* にデータ交換を行うための特別なJavaScriptコードを含める必要があります。

  この制限もユーザの安全のためです。ユーザが開いた `http://anysite.com` というサイトのページは、URL `http://gmail.com` の別のブラウザのタブにアクセスし、そこから情報を盗むことはできません。

- JavaScriptはネットワークを介して、現在のページがきたサーバと簡単にやり取りすることができます。しかし、他のサイト/ドメインからデータを受信することは制限されています。可能ですが、リモート側からの明示的な同意(HTTPヘッダで表現)が必要になります。繰り返しますが、これらは安全上の制限です。




JavaScriptがブラウザ外で使われる場合はこのような制限は存在しません。たとえば、サーバ上です。また、現代のブラウザは、より拡張されたアクセス権限を必要とするプラグイン/拡張機能を利用することもできます。

## なにがJavaScriptを特別なものにしている？

JavaScriptには少なくとも *3つ* の素晴らしいことがあります:

- HTML/CSSとの完全な統合
- シンプルなことはシンプルに
- すべてのメジャーブラウザでサポートされており、デフォルトで有効

JavaScriptは、これら3つのことを組み合わせた唯一のブラウザテクノロジーです。

それがJavaScriptをユニークなものにしています。だからこそ、ブラウザインターフェイスを作成するための最も普及しているツールとなっています。

とは言え、JavaScript を利用してサーバやモバイルアプリケーションなどを作成することもできます。

## JavaScriptを “覆う” 言語

JavaScriptの構文は、すべての人のニーズにマッチしている訳ではありません。人によって求める機能は異なります。

プロジェクトや要件はそれぞれ異なるため、それは自然なことです。

そのため、最近では新しい言語が数多く登場しています。これらはブラウザで実行する前にJavaScriptに *トランスパイル* (変換)されます。

最新のツールは非常に高速にトランスパイルでき、透過的です。開発者が別の言語でコードを作成し、それを自動変換することができます。

これは、そのような言語の例です:

- CoffeeScript はJavaScriptの "シンタックスシュガー"です。より短い構文を導入し、より簡潔でクリアなコードを書くことができます。たいてい、Ruby 開発者は好きです。
- TypeScript は “厳密なデータ型指定” の追加に焦点をあてています。それは複雑なシステムの開発とサポートを簡素化するためです。これは Microsoftにより開発されています。
- Flow もデータ型定義を追加しますが、その方法は異なります。Facebook により開発されました。
- Dart はブラウザ以外の環境（モバイルアプリのような）で動作する独自のエンジンを持ったスタンドアローンな言語ですが、JavaScript へトランスパイルすることもできます。Googleによって開発されました。
- Brython は JavaScript への Python トランスパイラで、JavaScript を使用することなく、純粋な Python でアプリケーションを作成することができます。

他にもあります。もちろん、上記のような言語を利用する予定だとしても、実際に行われていることを本当に理解するためにJavaScriptは知っておくのがよいです。

## サマリ

- JavaScriptは当初ブラウザ用の言語として作られました。しかし今はその他の多くの環境で利用されています。
- 現時点では、JavaScriptはHTML/CSSと完全に統合し、最も広く採用されたブラウザ言語として、独立した地位にいます。
- JavaScriptに “トランスパイル” し、特定の機能を提供する多くの言語があります。JavaScriptをマスターした後、少なくとも簡単にでも目を通しておくことをお勧めします。

## マニュアルと仕様

この本は *チュートリアル* であり、あなたが徐々に言語を学ぶのを助けることを目的としています。そのため、基本が理解できたら別の情報源が必要になってきます。

### 仕様

The ECMA-262 仕様 は、JavaScript に関して最も綿密で、詳細かつ形式化された情報を含んでいます。これが言語を定義しています。

しかし、形式張っているので最初は理解するのが難しいです。なので、言語の詳細について最も信頼できる情報源が必要であればここが正解ですが、日々使用するものではありません。

新しい仕様のバージョンは毎年リリースされます。これらのリリース間での、最新の仕様ドラフトは https://tc39.es/ecma262/ にあります。

“ほぼ標準” (いわゆる “ステージ 3”) を含む、新しい最先端の機能については、https://github.com/tc39/proposals の提案を参照してください。

また、ブラウザを対象に開発しているのであれば、チュートリアルの パート 2 で説明している、他の仕様もあります。

### マニュアル

- **MDN (Mozilla) JavaScript リファレンス** は、例やその他の情報を含むマニュアルです。ここは、個々の言語関数やメソッドなどに関する情報の詳細を知るのにとても役立ちます。

  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference にあります。日本語版は https://developer.mozilla.org/ja/docs/Web/JavaScript/Reference です。

  ですが、インターネット検索を使う方がベストなことが多いです。`parseInt` 関数を検索する場合、https://google.com/search?q=MDN+parseInt のように、クエリーを “MDN [用語]” とするだけです。

- **MSDN** – JavaScript (しばしば JScript と呼ばれます)を含む、多くの情報をもつ Microsoft のマニュアルです。Internet Explorer 固有のものが必要であれば、ここがベターです: http://msdn.microsoft.com/ (日本語版: https://msdn.microsoft.com/ja-jp/ )

  また、 “RegExp MSDN” または “RegExp MSDN jscript” といったフレーズでインターネット検索することもできます。

### 機能のサポート状況

JavaScript は開発中の言語であり、定期的に新機能が追加されます。

ブラウザベースや他のエンジン間でのサポート状況は以下で見ることができます:

- http://caniuse.com – 機能ごとのサポート状況が表で確認できます。例えば、モダンな暗号化機能をサポートしているエンジンを確認するには、http://caniuse.com/#feat=cryptography を参照してください。
- https://kangax.github.io/compat-table – 言語の機能と、それらをサポートする/しないエンジンの表です。

これらのリソースは、言語の詳細やサポートなどに関する貴重な情報が含まれているため、実際の開発に役立ちます。

特定の機能に関する詳細な情報が必要な場合は、それら（またはこのページ）を覚えておいてください。

## コードエディタ

コードエディタはプログラマが最も時間を費やす場所です。

大きく2つのタイプがあります: IDE(統合開発環境)と軽量なエディタです。多くの人々はそれぞれのタイプのツールを1つ選んで使うことが多いです。

### IDE

IDE (統合開発環境)は通常 “プロジェクト全体” での操作をする多くの機能を持ったパワフルなエディタを意味します。つまり、名前が示すように単なるエディタではなく、本格的な “開発環境” です。

IDEはプロジェクト(多くのファイル)をロードし、ファイル間のナビゲーションを可能とし、プロジェクト全体に基づいた自動補完を提供、バージョン管理システム(git のような)、テスト環境や他の “プロジェクトレベル” のものと統合します。

もしもまだIDEの選択を検討していないなら、次のようなIDEを見てみてください。

- Visual Studio Code (クロスプラットフォーム, フリー).
- WebStorm (クロスプラットフォーム, 有料).

Windows には、"Visual Studio"もあります。“Visual Studio Code” と混同しないでください。“Visual Studio” は、.NET プラットフォームに最適な、有料で強力なWindows専用エディタです。JavaScriptも得意です。無料版Visual Studio Community もあります。

多くのIDEは有料ですが、トライアル期間を持っています。それらのコストはたいてい資格をもつ開発者の給料と比べわずかなので、あなたにとってベストなものを選んでください。

### 軽量なエディタ

“軽量なエディタ” はIDEほど強力ではありませんが、速くエレガントでシンプルです。

主にファイルをすぐに開いて編集するのに使われます。

"軽量なエディタ"と"IDE"の主な違いは、IDEはプロジェクトレベルで動作するので、開始時により多くのデータをロードし、必要に応じてプロジェクト構造の分析等をします。軽量なエディタは、単一ファイルだけが必要な場合にははるかに速いでしょう。

なお、実際には軽量なエディタもディレクトリレベルの構文解析や自動補完を含む多くのプラグインを持っている場合があります。そのため、軽量なエディタとIDEの間に厳密な境界はありません。

注目に値するものとしては次のような選択肢があります:

- Atom (クロスプラットフォーム、フリー).
- Visual Studio Code (クロスプラットフォーム、フリー).
- Sublime Text (クロスプラットフォーム、シェアウェア).
- Notepad++ (Windows, フリー).
- Vim や Emacs も使い方を知っているなら良い選択肢です。

### 議論はしません

上記のリストのエディタは、私または私がよい開発者だと思っている友人が喜んで長い間利用しているものです。

この広い世界には他にも素晴らしいエディタがあります。ぜひあなたが最も好きなものを選んでください。

エディタの選択は、他のツールのようにプロジェクト、習慣や個人の趣向によります。

## 開発者コンソール

コードにエラーはつきものです。少なくともあなたが ロボット ではなく人間であるなら、*絶対に* 間違いをする、ということです。

しかしブラウザでは、ユーザはデフォルトではエラーは見えません。そのため、スクリプトが上手くいかない場合、何が悪かったのか見ることができず、直すこともできないでしょう。

エラーを確認したりスクリプトに関する多くの役立つ情報を得るために、ブラウザには “開発者ツール” が組み込まれています。

開発者の多くは、開発するのに Chrome か Firefox を利用することが多いです。なぜなら、それらのブラウザは最高の開発者ツールを持っているからです。開発者ツールを提供している他のブラウザも特別な機能を持っていますが、たいてい Chrome かFirefox の “キャッチアップ” です。なので、ほとんどの開発者は “お気に入り” のブラウザを持っており、問題がブラウザ依存の場合は他のブラウザに切り替えます。

開発者ツールは、本当に強力で多くの機能があります。開発を開始するために、私たちはどうやってそれらを開き、エラーを見て、JavaScriptコマンドを実行するのか学びましょう。

## Google Chrome

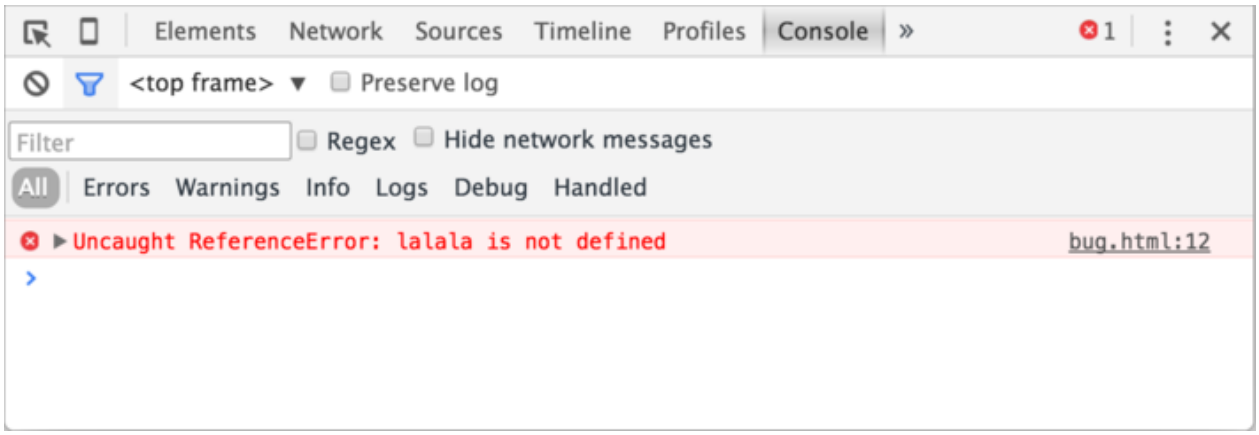
bug.htmlを開きましょう。

このJavaScriptコードにはエラーがあります。通常のサイト訪問者の目からは隠れているので、エラーを確認するために開発者ツールを開きましょう。

`F12`もしくは、Macの場合は`Cmd+Opt+J`を押します。

開発者ツールが起動します。デフォルトではConsoleタブが開かれているでしょう。

次のようになります:




開発者ツールの正確な見栄えは、利用しているChromeのバージョンに依存します。見栄えが変わることもありますが似ているはずです。

- ここでは赤色でエラーメッセージを見ることができます。このケースでは、スクリプトに未定義の “lalala” コマンドが含まれています。
- 右側にはエラーが発生した行番号とともに、ソース `bug.html：12` へのクリック可能なリンクがあります。

エラーメッセージの下に、青の `>` の記号があります。それは “コマンドライン” を意味し、JavaScriptコマンドを入力し、`Enter` を押すことでそれらを実行することができます。

これで私たちはエラーを見ることができるようになったので、開発を開始するには十分です。後ほど開発者ツールに戻り、チャプター Chrome でのデバッグ でより詳細なデバッグをカバーします。

> **複数行の入力**
>
> 通常、コンソールに1行コードを入力して `Enter` を押すと、コードが実行されます。
>
> 複数行を入力したい場合は、`Shift+Enter` を押します。この方法で長い JavaScript コードも入力できます。

## Firefox, Edge やその他

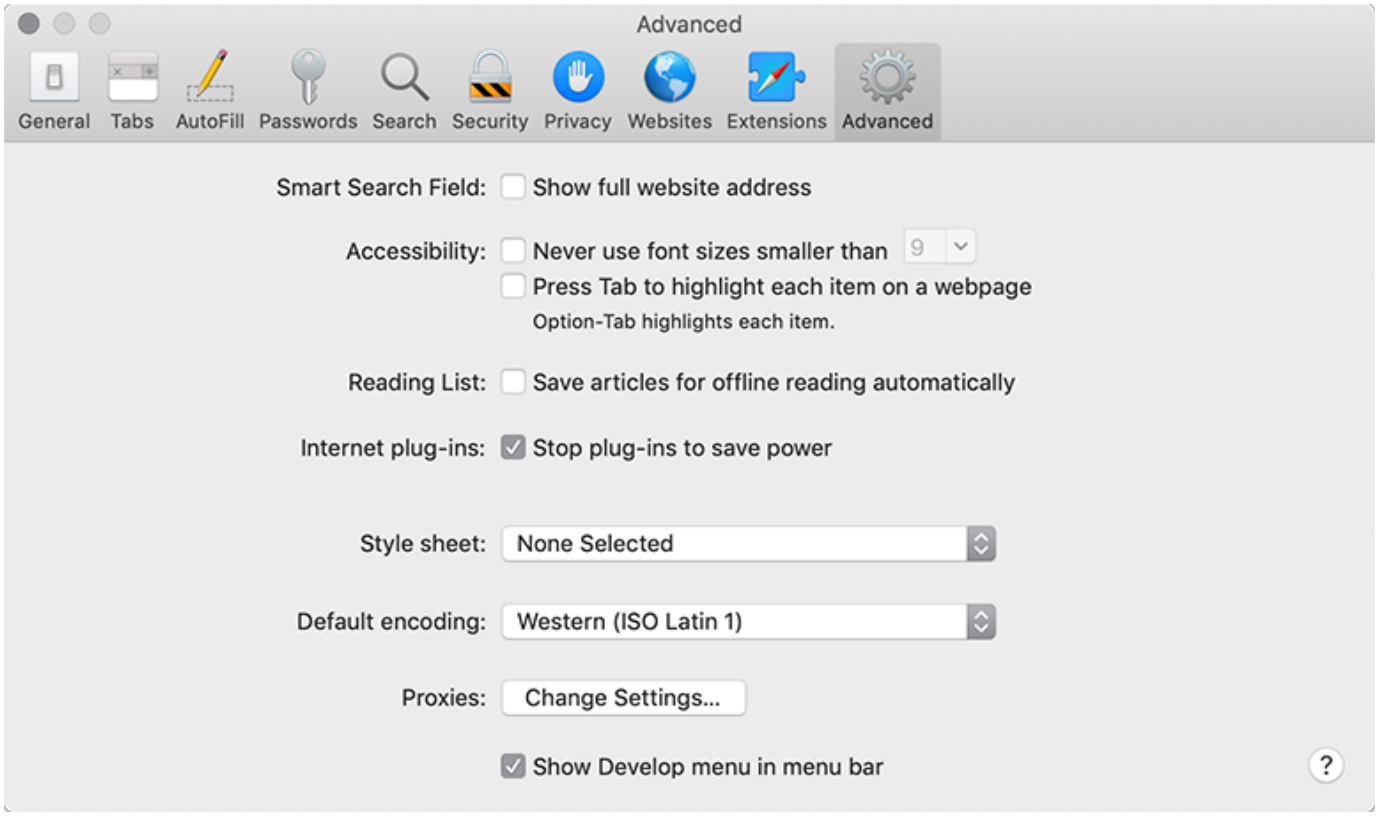
ほとんどのブラウザは開発者ツールを開くために `F12` を使います。

それらのルック＆フィールはとても似ています。それらの1つ(Chromeで始めるのが良いでしょう)の使い方を知ったら、簡単に別のものに切り替えることができるでしょう。

### Safari

Safari (Mac ブラウザ, Windows/Linux ではサポートされていません)はここでは少しだけ特別です。最初に、 “開発者メニュー” を有効にする必要があります。

Preferencesを開き、“Advanced” ペインに行きます。一番下にチェックボックスがあります。




`Cmd+Opt+C` でコンソールを切り替えることができます。また、"開発"という名前の新しいトップメニュー項目が表示されていることにも留意してください。 それは多くのコマンドとオプションを持っています。

### サマリ

- 開発者ツールは私達がエラーを見たり、コマンドを実行したり、変数を検査したりほかにも多くのことを可能にします。
- Windows下では、ほとんどのブラウザは `F12` で開くことができます。Mac用のChromeは `Cmd+Opt+J` が必要で、Safariは `Cmd+Opt+C` です(最初に有効化が必要)。

これで環境が整いました。次のセクションでは、JavaScriptの説明に入ります。