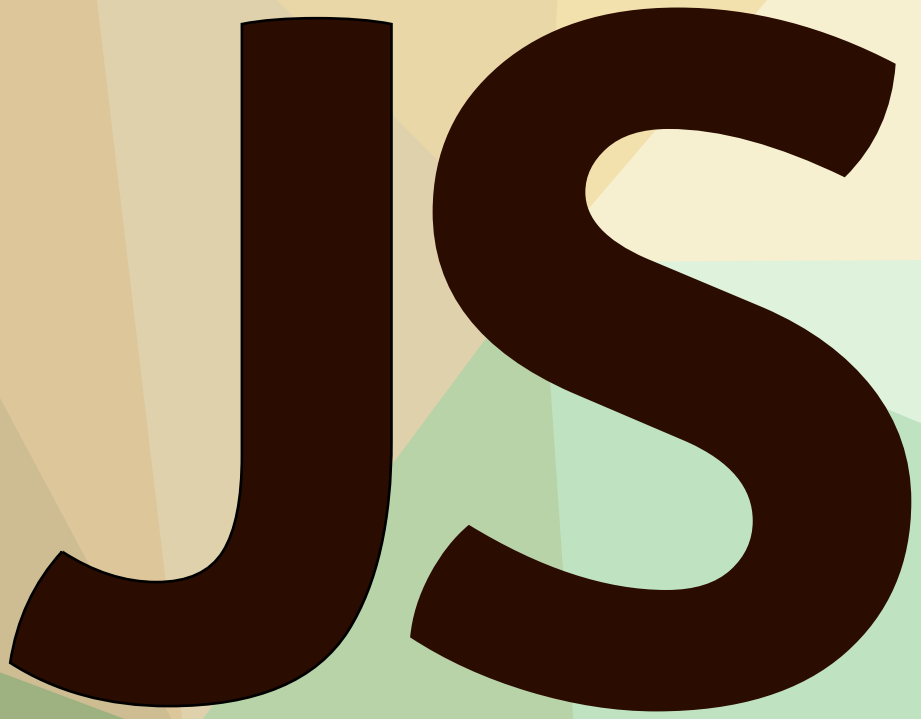
Partie 1

JavaScript le langage



Ilya Kantor

Construit à 24 avril 2024

La dernière version du tutoriel est disponible à l'adresse suivante : https://fr.javascript.info.

Nous travaillons constamment pour améliorer le tutoriel. Si vous trouvez des erreurs, merci de nous les signaler sur notre Github.

- Une introduction
  - Une Introduction à JavaScript
  - Manuels et spécifications
  - Les éditeurs de code
  - La console de développement

Dans ce guide nous allons apprendre JavaScript, en partant de zéro jusqu'à des concepts avancés comme la POO.

Nous allons nous concentrer ici sur le langage lui même, avec le minimum de notes spécifiques aux environnements.

Une introduction

A propos du langage JavaScript et de l'environnement pour le développeur.

Une Introduction à JavaScript

Voyons ce qui est spécial à propos de JavaScript, ce qu'il nous permet de faire et avec quelles autres technologies il s'accorde bien.

Qu'est-ce que JavaScript ?

JavaScript a été initialement créé pour "rendre les pages web vivantes".

Les programmes dans ce langage sont appelés *scripts*. Ils peuvent être écrits directement dans une page HTML et exécutés automatiquement au chargement des pages.

Les scripts sont fournis et exécutés en texte brut. Ils n'ont pas besoin d'une préparation spéciale ou d'une compilation pour fonctionner.

De par cet aspect, JavaScript est très différent d'un autre langage appelé Java .

Pourquoi est-il appelé JavaScript ?

Quand JavaScript a été créé, il portait initialement un autre nom : "LiveScript". Mais à cette époque le langage Java était très populaire, il a donc été décidé que positionner un nouveau langage en tant que "petit frère" de Java pourrait aider.

Mais au fur et à mesure de son évolution, JavaScript est devenu un langage totalement indépendant, avec ses propres spécifications appelées ECMAScript , aujourd'hui il n'a aucun rapport avec Java.

Aujourd'hui, JavaScript peut s'exécuter non seulement dans le navigateur, mais également sur un serveur, ou encore sur n'importe quel appareil dans lequel existe un programme appelé le moteur JavaScript .

Le navigateur a un moteur intégré, parfois il peut être également appelé "la machine virtuelle JavaScript".

Différents moteurs ont différents "nom de code", par exemple :

- V8 – dans Chrome et Opera.
- SpiderMonkey – dans Firefox.
- … Il existe d'autres noms de code comme "Chakra" pour IE, "JavaScriptCore", "Nitro" et "SquirrelFish" pour Safari etc.

Les termes ci-dessus sont bons à retenir, car ils sont utilisés dans les articles destinés aux développeurs sur Internet. Nous les utiliserons aussi. Par exemple, si "une fonctionnalité X est prise en charge par V8", cela fonctionne probablement dans Chrome, Edge et Opera.

Comment fonctionnent les moteurs ?

Les moteurs sont compliqués. Mais le fonctionnement de base est facile à comprendre.

1. Le moteur (intégré si c'est un navigateur) lit ("analyse") le script.
2. Ensuite, il convertit ("compile") le script en langage machine.
3. Enfin le code machine s'exécute, très rapidement.

Le moteur applique des optimisations à chaque étape du processus. Il surveille même le script compilé en cours d'exécution, analyse les données qui le traversent et applique des optimisations au code machine en fonction de ces informations.

Que peut faire JavaScript dans le navigateur ?

Le JavaScript moderne est un langage de programmation "sûr". Il ne fournit pas d'accès de bas niveau à la mémoire ou au processeur, parce qu'il a été initialement conçu pour les navigateurs qui n'en ont pas besoin.

Les fonctionnalités dépendent grandement de l'environnement qui exécute JavaScript. Par exemple, Node.js prend en charge les fonctions qui permettent à JavaScript de lire / écrire des fichiers arbitrairement, d'exécuter des requêtes réseau, etc.

JavaScript intégré au navigateur peut faire tout ce qui concerne la manipulation des pages Web, l'interaction avec l'utilisateur et le serveur Web.

Par exemple, JavaScript dans le navigateur est capable de :

- Ajouter un nouveau code HTML à la page, modifier le contenu existant, modifier les styles.
- Réagir aux actions de l'utilisateur, s'exécuter sur des clics de souris, des mouvements de pointeur, des appuis sur des touches.

- Envoyer des requêtes sur le réseau à des serveurs distants, télécharger et envoyer des fichiers (technologies AJAX et COMET).
- Obtenir et définir des cookies, poser des questions au visiteur, afficher des messages.
- Se souvenir des données du côté client (“stockage local”).

Qu’est-ce que JavaScript ne peut pas faire dans le navigateur ?

Les capacités de JavaScript dans le navigateur sont limitées pour la sécurité de l’utilisateur. L’objectif est d’empêcher une page Web malveillante d’accéder à des informations privées ou de nuire aux données de l’utilisateur.

Les exemples de telles restrictions sont :

- JavaScript sur une page Web ne peut pas lire/écrire des fichiers arbitrairement sur le disque dur, les copier ou exécuter des programmes. Il n’a pas d’accès direct aux fonctions du système d’exploitation.

  Les navigateurs modernes lui permettent de fonctionner avec des fichiers, mais l’accès est limité et n’est fourni que si l’utilisateur effectue certaines actions, comme «déposer» un fichier dans une fenêtre de navigateur ou le sélectionner via une balise `<input>`.

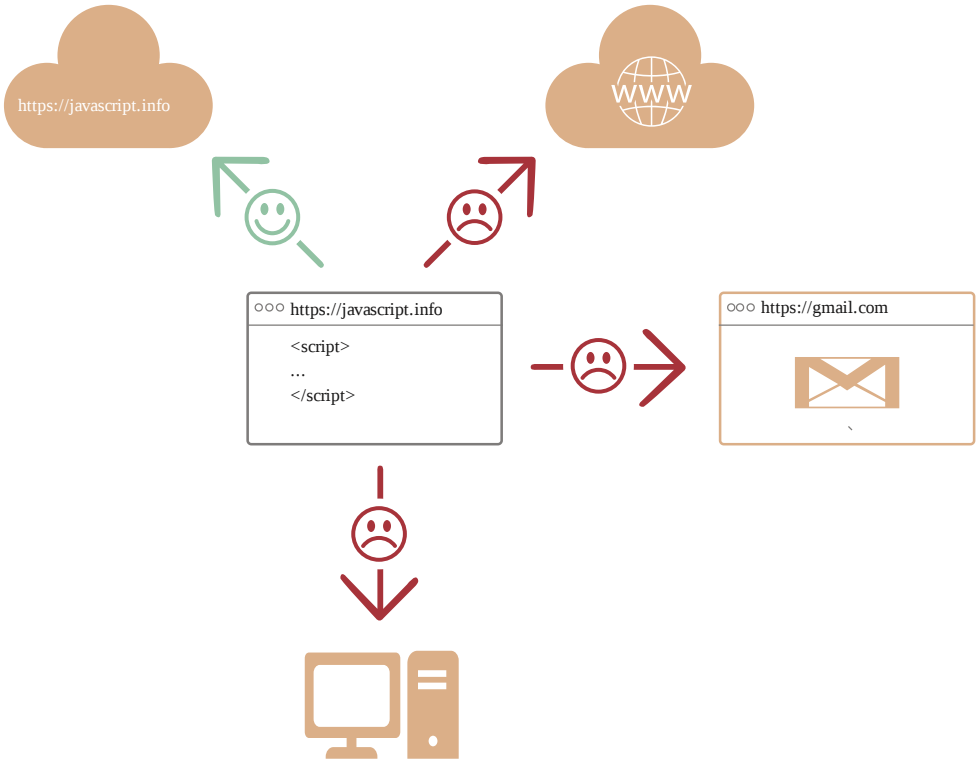
  Il existe des moyens d’interagir avec une webcam/microphone et d’autres périphériques, mais ils nécessitent une autorisation explicite de l’utilisateur. Ainsi, une page contenant du JavaScript ne permet pas d’activer une caméra Web, d’observer l’environnement et d’envoyer les informations à la NSA .

- Différents onglets / fenêtres ne se connaissent généralement pas. Parfois, ils se croisent, par exemple lorsqu’une fenêtre utilise JavaScript pour ouvrir l’autre. Mais même dans ce cas, le JavaScript d’une page ne peut pas accéder à l’autre si elle provient de sites différents (provenant d’un autre domaine, protocole ou port).

  C’est ce qu’on appelle la “politique de même origine” (“Same Origin Policy”). Pour contourner cette sécurité, *les deux pages* doivent se mettre d’accord et contenir un code JavaScript spécial qui gère l’échange de données. Nous verrons cela plus loin dans ce tutoriel.

  Cette limitation concerne également la sécurité de l’utilisateur. Une page de `http://autresite.com` qu’un utilisateur a ouvert ne doit pas pouvoir accéder à un autre onglet du navigateur avec l’URL `http://gmail.com` et y voler des informations.

- JavaScript peut facilement communiquer sur le net avec le serveur d’où provient la page. Mais sa capacité à recevoir des données d’autres sites / domaines est paralysée. Bien que possible, il nécessite un accord explicite (exprimé dans les en-têtes HTTP) du côté distant. Encore une fois, ce sont des limites de sécurité.



De telles limites n'existent pas si JavaScript est utilisé en dehors du navigateur, par exemple sur un serveur. Les navigateurs modernes permettent également l'installation de plug-ins / extensions susceptibles d'obtenir des autorisations étendues.

Qu'est-ce qui rend JavaScript unique ?

Il y a au moins trois bonnes choses à propos de JavaScript :

- Intégration complète avec HTML / CSS.
- Les choses simples sont faites simplement.
- Pris en charge par tous les principaux navigateurs et activé par défaut.

JavaScript est la seule technologie de navigateur qui combine ces trois éléments.

C'est ce qui rend JavaScript unique. C'est pourquoi il l'outil le plus répandu pour créer des interfaces de navigateur.

Cela dit, JavaScript permet également de créer des serveurs, des applications mobiles, etc.

Les langages "par dessus" JavaScript

La syntaxe de JavaScript ne convient pas aux besoins de tous. Différentes personnes veulent des fonctionnalités différentes.

Il faut s'y attendre, parce que besoins sont différents pour tous.

Donc, récemment, une pléthore de nouveaux langages sont apparus, qui sont *transpilés* (convertis) en JavaScript avant leur exécution dans le navigateur.

Les outils modernes rendent la transpilation très rapide et transparente, permettant aux développeurs de coder dans un autre langage et de le convertir automatiquement “sous le capot”.

Les exemples de ce genre de langages :

- CoffeeScript est un “sucre syntaxique” pour JavaScript, il introduit une syntaxe plus courte, permettant d’écrire du code plus précis et plus clair. Habituellement, les développeurs Ruby l’aiment bien.
- TypeScript se concentre sur l’ajout de “typage strict des données” pour simplifier le développement et la prise en charge de systèmes complexes. Il est développé par Microsoft.
- Flow ajoute également la saisie de données, mais de manière différente. Développé par Facebook.
- Dart est un langage autonome doté de son propre moteur qui s’exécute dans des environnements autres que les navigateurs (comme les applications mobiles), mais peut aussi être transpilé en JavaScript. Développé par Google.
- Brython est un transpiler (ou transpileur en bon français) Python vers JavaScript qui permet d’écrire des applications en Python pur sans JavaScript.
- Kotlin est un langage de programmation moderne, concis et sûr qui peut cibler le navigateur ou Node.js

Il en existe évidemment bien plus, cela dit, même si nous utilisons un de ces langages transpilés, nous devrions également connaître le langage JavaScript, pour bien comprendre ce que nous faisons.

Résumé

- JavaScript a été initialement créé en tant que langage de navigateur uniquement, mais il est désormais également utilisé dans de nombreux autres environnements.
- En ce moment, JavaScript occupe une position unique en tant que langage de navigateur le plus largement adopté avec une intégration complète avec HTML/CSS.
- De nombreux langages sont *transpilés* en JavaScript et fournissent certaines fonctionnalités. Il est recommandé d’y jeter un coup d’œil, au moins brièvement, après avoir maîtrisé JavaScript.

Manuels et spécifications

Ce livre est un *tutoriel*. Il vise à vous aider à apprendre progressivement le langage. Mais une fois que vous maîtriserez les bases, vous aurez besoin d’autres ressources.

Spécification

The ECMA-262 specification contient les informations les plus détaillées et formalisées sur JavaScript. C'est elle qui définit le langage.

Une nouvelle version de la spécification est publiée chaque année. La dernière version en cours est disponible à cette adresse : https://tc39.es/ecma262/ .

Pour en savoir plus sur les fonctionnalités à venir, y compris celles qui sont "presque standards" (appelées aussi "stage 3"), vous pouvez consulter les propositions à cette adresse : https://github.com/tc39/proposals .

De plus, si vous développez pour le navigateur, d'autres spécifications sont couvertes dans la seconde partie du tutoriel.

Manuels

- **La référence MDN (Mozilla) JavaScript** est le principal manuel avec des exemples et d'autres informations. C'est une excellente source pour obtenir des informations détaillées sur les fonctions linguistiques, les méthodes, etc.

  On peut la trouver à cette adresse : https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference .

  Cependant, il est souvent préférable d'utiliser une recherche sur Internet. Utilisez simplement "MDN [terme]" dans la requête, par exemple https://google.com/search?q=MDN+parseInt pour rechercher la fonction `parseInt`.

Tableaux de compatibilité

JavaScript est un langage en développement, de nouvelles fonctionnalités sont ajoutées régulièrement.

Pour voir si elles sont supportées dans les moteurs, au sein des navigateurs et autres, voir :

- http://caniuse.com – tables de prise en charge par fonctionnalité, par exemple pour voir quels moteurs supportent les fonctions de cryptographie modernes : http://caniuse.com/#feat=cryptography .
- https://kangax.github.io/compat-table – un tableau avec les fonctionnalités linguistiques et les moteurs qui les prennent en charge ou non.

Toutes ces ressources sont utiles dans le quotidien des développeurs, parce qu'elles contiennent des informations précieuses sur les fonctionnalités du langage, leur support, etc.

Veuillez vous en souvenir (ou de cette page) pour les cas où vous avez besoin d'informations détaillées sur une fonctionnalité particulière.

Les éditeurs de code

Un éditeur de code est l'endroit où les programmeurs passent la plus grande partie de leur temps.

Il existe deux archétypes: IDE et les éditeurs légers. Beaucoup de personnes se sentent à l'aise pour choisir un outil de chaque type.

IDE

Le terme IDE (Integrated Development Environment) signifie un éditeur puissant avec de nombreuses fonctionnalités qui fonctionne généralement sur un "projet entier". Comme son nom l'indique, ce n'est pas seulement un éditeur, mais un environnement de développement complet.

Un IDE charge le projet (peut contenir de nombreux fichiers), permet la navigation entre les fichiers, fournit une auto-complétion basée sur l'ensemble du projet (pas seulement le fichier ouvert), s'intègre à un système de gestion de version (comme git), un environnement de test et d'autres éléments au niveau du projet.

Si vous n'avez pas encore pensé à sélectionner un IDE, examinez les variantes suivantes :

- Visual Studio Code (cross-platform, free).

- WebStorm (cross-platform, payant).

Pour Windows, il existe également “Visual Studio”, à ne pas confondre avec "Visual Studio Code ". “Visual Studio” est un IDE payant et puissant, disponible sur Windows et Mac, bien adapté à la plate-forme .NET. C’est aussi bon en JavaScript. Il y a aussi une version gratuite Visual Studio Community .

La plupart des IDE sont payants, mais ont une période d’essai. Leur coût est généralement négligeable par rapport au salaire d’un développeur qualifié, alors choisissez le meilleur pour vous.

Les éditeurs légers

“Les éditeurs légers” ne sont pas aussi puissants que les IDE, mais ils sont rapides, élégants et simples.

Ils sont principalement utilisés pour ouvrir et éditer instantanément un fichier.

La principale différence entre un “éditeur léger” et un “IDE” réside dans le fait qu’un environnement de développement intégré fonctionne au niveau du projet. Il charge donc beaucoup plus de données au démarrage, analyse la structure du projet si nécessaire, etc. Un éditeur léger est beaucoup plus rapide si nous avons besoin d’un seul fichier.

En pratique, les éditeurs légers peuvent avoir beaucoup de plug-ins, y compris des analyseurs de syntaxe au niveau des répertoires et des autocompléteurs. Il n’y a donc pas de frontière stricte entre un éditeur léger et un IDE.

Il existe de nombreuses options, par exemple :

- Sublime Text (multiplateforme, payant).
- Notepad++ (Windows, gratuit).
- Vim et Emacs sont également cool, si vous savez comment les utiliser.

Ne discutons pas

Les éditeurs des listes ci-dessus sont ceux que moi-même ou mes amis, que je considère comme de bons développeurs, utilisent depuis longtemps et en sont satisfaits.

Il y a d’autres grands éditeurs dans notre vaste monde. Veuillez choisir celui que vous aimez le plus.

Le choix d’un éditeur, comme tout autre outil, est individuel et dépend de vos projets, de vos habitudes, de vos préférences personnelles.

L’avis personnel de l’auteur :

- J’utilise Visual Studio Code si je développe principalement du frontend.
- Sinon, s’il s’agit principalement d’un autre langage/plate-forme et partiellement d’un frontend, envisagez d’autres éditeurs, tels que XCode (Mac), Visual Studio (Windows) ou la famille Jetbrains (Webstorm, PHPStorm, RubyMine, etc., selon le langage).

La console de développement

Le code est sujet aux erreurs. Vous êtes susceptible de faire des erreurs … Oh, de quoi je parle ? Vous allez absolument faire des erreurs, du moins si vous êtes un humain, pas un robot .

Mais dans le navigateur, un utilisateur ne voit pas les erreurs par défaut. Ainsi, si quelque chose ne va pas dans le script, nous ne verrons pas ce qui ne va pas et nous ne pourrons pas le réparer.

Pour voir les erreurs et obtenir beaucoup d'informations utiles sur les scripts, les navigateurs intègrent des "outils de développement".

Le plus souvent, les développeurs se tournent vers Chrome ou Firefox pour le développement, car ces navigateurs disposent des meilleurs outils de développement. D'autres navigateurs fournissent également des outils de développement, parfois dotés de fonctions spéciales, mais jouent généralement le rôle de "rattrapage" pour Chrome ou Firefox. Donc, la plupart des gens ont un navigateur "favori" et passent à d'autres si un problème est spécifique au navigateur.

Les outils de développement sont très puissants ; ils possèdent de nombreuses fonctionnalités. Pour commencer, nous allons apprendre à les ouvrir, à examiner les erreurs et à exécuter des commandes JavaScript.

Google Chrome

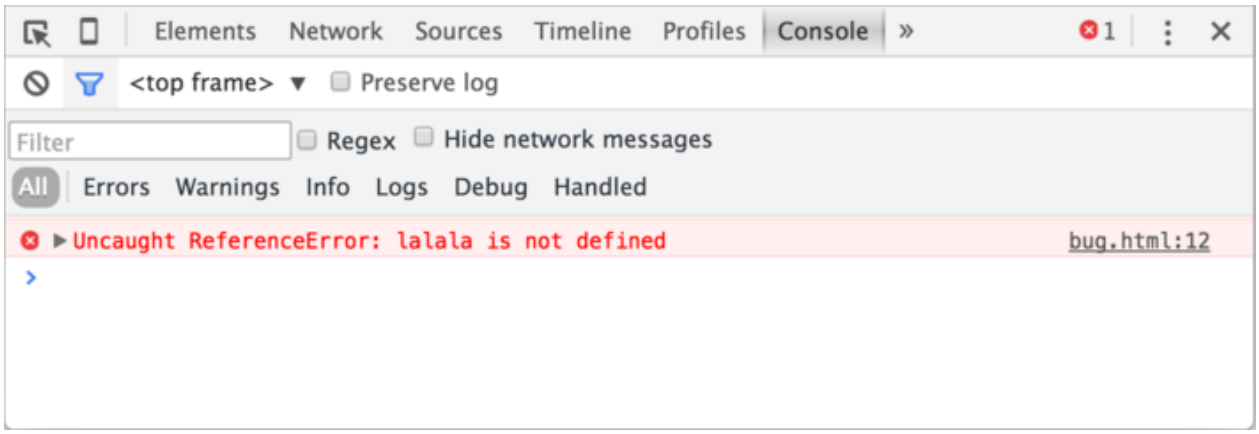
Ouvrez la page bug.html.

Il y a une erreur dans le code JavaScript. Elle est invisible à un visiteur habituel, alors ouvrons les outils de développement pour la voir.

Appuyez sur `F12` ou, si vous êtes sur Mac, utilisez la combinaison `Cmd+Opt+J`.

Les outils de développement s'ouvriront sous l'onglet Console par défaut.

Cela ressemble un peu à ceci :



L'aspect exact des outils de développement dépend de votre version de Chrome. Cela change de temps en temps, mais devrait être similaire.

- Ici, nous pouvons voir le message d'erreur de couleur rouge. Dans ce cas, le script contient une commande "lalala" inconnue.
- Sur la droite, il y a un lien cliquable vers le code source bug.html:12 avec le numéro de ligne où l'erreur s'est produite.

Sous le message d'erreur, il y a un symbole bleu `>`. Il marque une "ligne de commande" où l'on peut taper des commandes JavaScript et appuyer sur `Enter` pour les exécuter.

Nous pouvons maintenant voir les erreurs et c'est suffisant pour le début. Nous reviendrons plus tard sur les outils de développement et approfondirons le débogage plus loin dans le chapitre Débogage dans le navigateur.

Multi-line input

Habituellement, quand on met une ligne de code dans la console, puis qu'on appuie sur `Enter`, elle s'exécute.

Pour insérer plusieurs lignes, appuyez sur `Shift+Enter`. De cette façon, on peut saisir de longs fragments de code JavaScript.

Firefox, Edge et autres

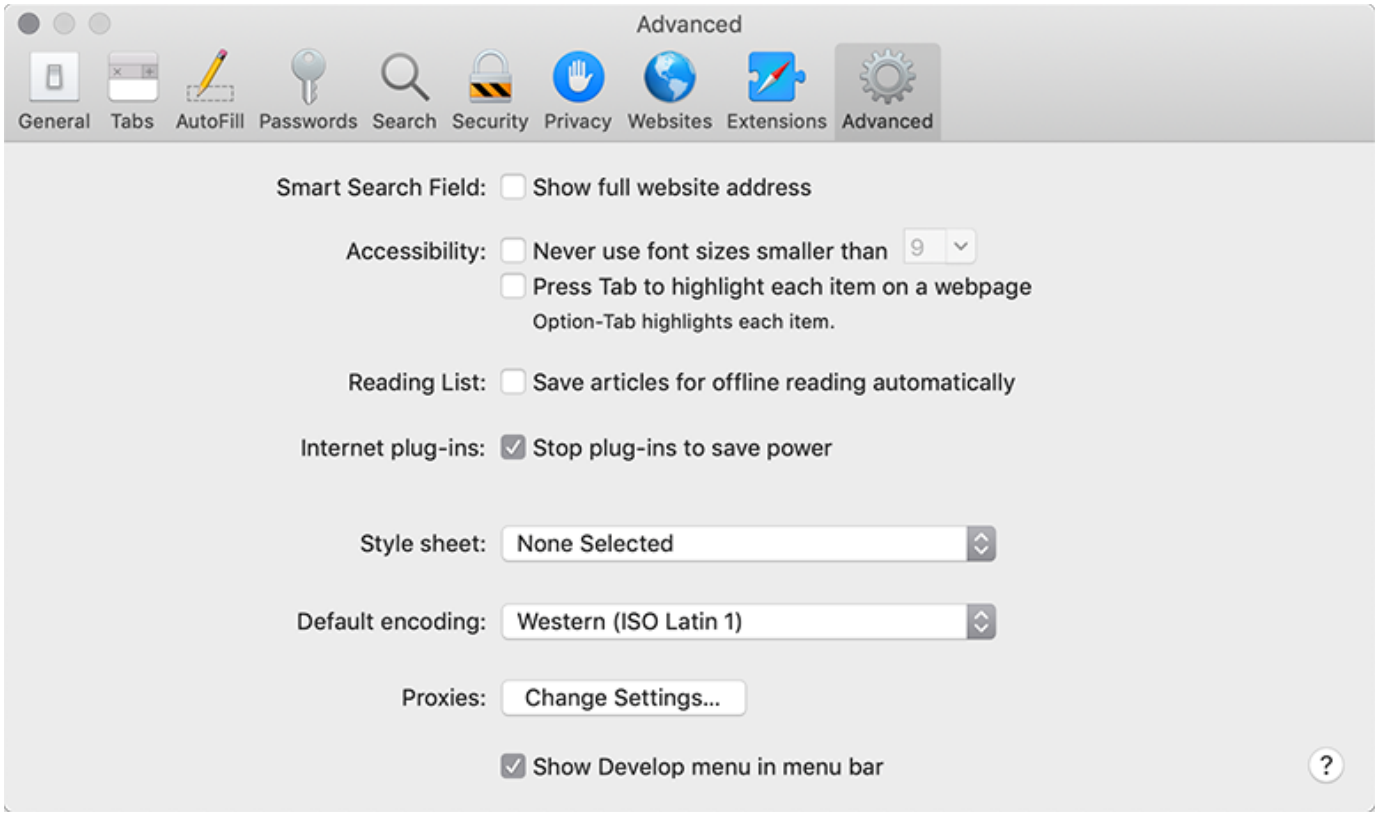
La plupart des autres navigateurs utilisent `F12` pour ouvrir les outils de développement.

Leur apparence est assez similaire. Une fois que vous savez comment utiliser l'un d'entre eux (vous pouvez commencer avec Chrome), vous pouvez facilement passer à un autre.

Safari

Safari (navigateur Mac, non pris en charge par Windows / Linux) est un peu spécial ici. Nous devons d'abord activer le menu "Développement".

Ouvrez les préférences et accédez au volet "Avancé". Il y a une case à cocher en bas :



Maintenant `Cmd+Opt+C` peut activer la console. Notez également que le nouvel élément de menu supérieur nommé “Développement” est apparu. Il a beaucoup de commandes et d’options.

Résumé

- Les outils de développement nous permettent de voir les erreurs, d’exécuter des commandes, d’examiner des variables et bien plus encore.
- Ils peuvent être ouverts avec `F12` pour la plupart des navigateurs sous Windows. Chrome pour Mac nécessite la combinaison `Cmd+Opt+J`, Safari: `Cmd+Opt+C` (doit être activé avant).

Nous avons maintenant notre environnement prêt. Dans la section suivante, nous passerons à JavaScript.