

# TOUT JavaScript

Maîtrisez les bases de la programmation JavaScript  
Démarrez avec les frameworks Node.js, React, Angular...

Pratiquez avec le site [toutjavascript.com](https://toutjavascript.com) 



A. Cazes, J. Delacroix

288 pages

2016



F. Lemaître

128 pages

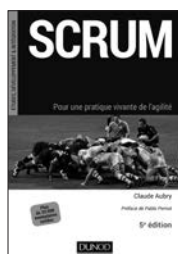
2018



P.-Y. Cloux, T. Garlot, J. Kohler

304 pages

2019



C. Aubry

384 pages

2018

Olivier Hondermarck

# TOUT JavaScript

Maîtrisez les bases de la programmation JavaScript  
Démarrez avec les frameworks Node.js, React, Angular...

Pratiquez avec le site [toutjavascript.com](https://toutjavascript.com) 

DUNOD

## Création graphique de la couverture : Hokus Pokus Créations

<p>Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage.</p> <p>Le Code de la propriété intellectuelle du 1<sup>er</sup> juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements</p>	<p>d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée.</p> <p>Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).</p>
--	--



**DANGER**  
LE PHOTOCOPIAGE  
TUE LE LIVRE

© Dunod, 2019  
11 rue Paul Bert, 92240 Malakoff  
[www.dunod.com](http://www.dunod.com)  
ISBN 978-2-10-077958-1

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2<sup>e</sup> et 3<sup>e</sup> a), d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

# Avant-propos

## ◆ *Pourquoi un nouveau livre sur le JavaScript ?*

Dix ans après mon premier ouvrage sur le JavaScript, paru chez Micro-Application, une totale réécriture était nécessaire.

J'ai écrit ce livre avec une vision plus large que le simple apprentissage du JavaScript. J'ai voulu apporter au lecteur des principes qui seront utiles dans la programmation sur tous les langages. J'ai aussi ajouté des conseils sur les facteurs de succès d'un projet web. Après une expérience de plus de douze ans en tant que créateur et éditeur d'un site web leader dans sa thématique, j'ai voulu partager mes acquis en abordant les optimisations de performances, d'ergonomie et de référencement, jusqu'au modèle économique.

## ◆ *À qui s'adresse ce livre ?*

Ce livre s'adresse à tous les développeurs web, qu'ils soient débutants ou avancés, en phase de formation à l'école, en auto-apprentissage par passion de la programmation ou en poste dans une entreprise du net.

Des connaissances préalables sur les langages HTML et CSS sont souhaitables.

Un lecteur qui découvre la programmation aura intérêt à lire cet ouvrage dans l'ordre et en intégralité, presque à l'image d'un roman. L'enchaînement des chapitres est progressif et ponctué de remarques et de conseils adaptés à la vraie vie d'un projet web.

Un développeur plus à l'aise avec le langage pourra se reporter à l'index, riche et varié, pour trouver directement les fonctionnalités du langage ou des notions plus générales qu'il souhaite approfondir ou découvrir.

Dans tous les cas, les deux premiers chapitres seront instructifs pour tous les lecteurs, avec un point de vue sur la place du JavaScript dans un projet et des rappels sur les bons outils et les bonnes pratiques de programmation.

Les lecteurs experts observeront sans doute quelques simplifications par rapport à la terminologie officielle ; elles sont destinées à rendre les notions plus facilement compréhensibles. La méthode `log()` de l'objet `console` sera par exemple présentée aussi comme une fonction.

## ◆ *Renvois vers le site [toutjavascript.com](http://toutjavascript.com)*

Afin de rendre ce livre moins volumineux, et donc moins cher, j'ai adopté une écriture assez condensée, avec les captures d'écrans strictement nécessaires et des extraits de scripts limités à la partie essentielle.

Des renvois vers les exemples complets sont proposés via des liens raccourcis au format ***tjs.ovh/nomScript***. Ces renvois, véritables compléments interactifs à ce guide, affichent :

- ✓ le rendu de l'exécution du script ;
- ✓ un émulateur de la console du navigateur ;

- ✓ le code source complet de l'exemple avec une coloration syntaxique, des commentaires et des liens vers les fiches de la référence JS de mon site ***toutjavascript.com***.

### ◆ **Erratum**

La page <http://www.toutjavascript.com/livre/erratum.php> contient les éventuels errata relevés par les lecteurs. N'hésitez pas à y déposer vos remarques pour améliorer et compléter les informations publiées.

### ◆ **Remerciements**

Je tiens à remercier les lecteurs de mon premier livre, sans qui cette version entièrement nouvelle n'aurait pas existé.

Je remercie également mon éditeur, Jean-Luc Blanc, chez Dunod, pour son professionnalisme. Il a réussi à me comprendre et à me motiver.

Sans la patience et les encouragements de Stephy, je n'aurai jamais été au bout de ce long travail.

Merci à ma mère pour la relecture attentive du manuscrit.

Mon chien, qui a supporté les longs mois d'écriture des deux livres, ne verra hélas pas l'aboutissement de cette version.

Je profite également de cet espace un peu plus personnel pour exprimer ma gratitude en tant qu'entrepreneur du web à trois sociétés incontournables. Elles ont énormément participé à la démocratisation d'Internet et donc à la réussite d'entreprises telles que la mienne :




- ✓ Google, pour avoir rendu l'accès à l'information bien plus facile aux utilisateurs ;
- ✓ Free et son patron emblématique, Xavier Niel, pour avoir créé les forfaits illimités ADSL et Mobile à tarifs imbattables ;
- ✓ OVH et son créateur, Octave Kaba, pour avoir offert des solutions d'hébergement fiables et très abordables à tous les diffuseurs de contenus.

# Table des matières

Avant-propos.....	V
-------------------	---

## PREMIÈRE PARTIE

### Les bases de JavaScript





 <b>Réflexions autour du JavaScript.....</b>	<b>3</b>
1.1 JavaScript, c'est quoi ? .....	3
1.2 Brève histoire du JavaScript .....	4
1.3 Le standard ECMAScript .....	4
1.4 Rappel sur le principe client/serveur .....	6
1.5 Les possibilités du JavaScript.....	7
1.6 Limites et sécurités du JavaScript.....	8
1.7 Impact sur les performances et le référencement .....	9
1.8 L'image du JavaScript et son avenir .....	10
1.9 Place du JavaScript dans un projet web.....	10
 <b>Hello(s) World(s).....</b>	<b>13</b>
2.1 Les exemples de Hello World .....	13
2.2 La console des navigateurs.....	17
2.3 L'extension Web Developer .....	20
2.4 L'outil Google PageSpeed.....	20
2.5 JavaScript non activé .....	20
2.6 Le bon éditeur de texte .....	22
2.7 Programmer et partager du JS en ligne .....	22
2.8 Mettre un site en ligne.....	23
2.9 Les bonnes pratiques .....	23
 <b>Structure d'un script.....</b>	<b>29</b>
3.1 Intégrer du JavaScript.....	29
3.2 Les variables.....	33
3.3 Les blocs d'instructions.....	37
3.4 Les tests conditionnels.....	38
3.5 Les conditions.....	41
3.6 Les boucles.....	45
3.7 Les fonctions .....	49

<b>4</b>	<b>Les chaînes de caractères en JavaScript</b>	57
4.1	Déclaration d'une chaîne de caractères	57
4.2	Concaténation de chaînes	59
4.3	L'objet string	60
4.4	Le traitement des caractères spéciaux	63
4.5	Exécuter une chaîne JavaScript	64
4.6	Manipulations avancées avec les expressions régulières	65
<b>5</b>	<b>Les mathématiques</b>	73
5.1	Les conversions de types de données	73
5.2	L'objet Math	76
5.3	Les nombres aléatoires	77
5.4	L'affichage formaté de nombres	79
<b>6</b>	<b>Manipulation de dates</b>	81
6.1	L'objet Date	81
6.2	Les minuteriers	85
6.3	L'affichage formaté de dates	86
<b>7</b>	<b>Les tableaux</b>	89
7.1	Utilité des tableaux	89
7.2	L'objet Array	90
7.3	Les tableaux spéciaux	94
7.4	La manipulation de tableaux	96
<b>8</b>	<b>Programmation objet et JSON</b>	105
8.1	Les principes de la programmation objet	105
8.2	Créer des objets JavaScript	108
8.3	Le format JSON	118
8.4	Résumé de la POO JavaScript	123
<b>9</b>	<b>Réintroduction au JavaScript</b>	127
9.1	Les évolutions de structure du langage	127
9.2	Les évolutions sur les objets	130
9.3	Comprendre un appel JavaScript	130
<b>10</b>	<b>L'objet maître window</b>	135
10.1	L'objet window	135
10.2	Manipulation des <i>pop-up</i>	143
10.3	Manipulation des frames	144



## DEUXIÈME PARTIE

### L'interactivité

	<b>11 La programmation événementielle.....</b>	149
	11.1 Programmation événementielle.....	149
	11.2 Le gestionnaire d'événements en détail.....	152
	<b>12 Manipuler le document.....</b>	163
	12.1 Le HTML dynamique.....	163
	12.2 Trouver les éléments du document.....	164
	12.3 Manipuler les éléments.....	170
	12.4 Créer de nouveaux éléments.....	179
	12.5 Le <i>drag and drop</i> .....	185
	12.6 La sélection de texte.....	186
	<b>13 Les formulaires.....</b>	193
	13.1 L'objet Form.....	193
	13.2 Les éléments de formulaires.....	197
	13.3 Les contrôles de saisie.....	215
	13.4 Les données côté serveur.....	215
	<b>14 Les appels AJAX.....</b>	219
	14.1 Appels asynchrones.....	219
	14.2 Le traitement côté serveur.....	223
	14.3 Les contraintes liées à AJAX.....	225
	<b>15 La gestion des erreurs.....</b>	229
	15.1 La détection d'une erreur.....	229
	15.2 L'objet Error.....	231
	<b>16 Les cookies et l'objet Storage.....</b>	235
	16.1 Les cookies.....	235
	16.2 Le stockage de données locales.....	240
	16.3 Les différences entre cookie et l'objet storage.....	241
	<b>17 La géolocalisation.....</b>	243
	17.1 Confidentialité et sécurité.....	243
	17.2 Obtenir la position actuelle.....	243
	17.3 Obtenir les positions régulièrement.....	245
	17.4 Les scripts de cartographie.....	245
	<b>18 Les notifications.....</b>	247
	18.1 Le principe des notifications.....	247
	18.2 Les notifications en JavaScript.....	248



<b>19</b>	<b>Le dessin et les canvas</b> .....	251
19.1	L'élément Canvas.....	251
19.2	Interactivités et animations.....	260

## TROISIÈME PARTIE

### Pour aller encore plus loin avec JavaScript

<b>20</b>	<b>Monétisation et publicité</b> .....	265
20.1	La monétisation.....	265
20.2	La publicité.....	266
20.3	Les bloqueurs de publicité.....	267
<b>21</b>	<b>Introduction à Node.js</b> .....	271
21.1	Installation de Node.js.....	271
21.2	Utilisation de Node.js en serveur web.....	275
21.3	Utilisation de npm.....	277
21.4	Un serveur web Node.js prêt à l'emploi.....	280
21.5	Node.js et les WebSockets.....	286
<b>22</b>	<b>Les langages dérivés du JavaScript</b> .....	291
22.1	TypeScript.....	291
22.2	Babel.....	295
22.3	JSX.....	298
<b>23</b>	<b>Bibliothèques et frameworks</b> .....	299
23.1	Frameworks.....	299
23.2	jQuery.....	301
23.3	Prototype.js.....	308
23.4	React.....	308
23.5	Angular.....	314
23.6	Vanilla.js.....	320
23.7	Quelques bibliothèques JavaScript utiles.....	320
<b>24</b>	<b>Les Web Workers</b> .....	323
24.1	Principes des Web Workers.....	323
24.2	Un cas réel de Web Worker.....	326
<b>25</b>	<b>Créer une extension de navigateur</b> .....	331
25.1	JavaScript dans un lien de la barre des favoris.....	331
25.2	Créer une extension sur Google Chrome.....	333
<b>26</b>	<b>Créer des applications</b> .....	343
26.1	Création d'applications de bureau.....	343
26.2	Création d'applications mobiles.....	345

<b>Annexe CSS</b> .....	349
Utilisation de FontAwesome .....	349
La pseudo-classe :hover .....	350
Les pseudo-classes :first-child et :last-child .....	350
Les pseudo-éléments ::before et ::after .....	351
Les boîtes fléchées avec CSS .....	352
Les animations CSS .....	352
Les transformations CSS .....	353
<b>Index</b> .....	355



## PREMIÈRE PARTIE

# Les bases de JavaScript

Dans tous les langages et tous les domaines de compétence, des bases saines sont nécessaires pour progresser de manière efficace et durable.

Cette première partie ne sera pas pour autant rébarbative. Elle permettra une bonne compréhension de l'usage du JavaScript, de ses possibilités et de ses limites.

Tous les chapitres expliquent les notions en tenant en compte de la norme ECMAScript 6. Le dernier chapitre de cette première partie regroupe toutes les nouveautés d'ECMAScript 6 et sera particulièrement utile aux lecteurs ayant besoin d'une mise à jour sur les nouveautés du langage.





# Réflexions autour du JavaScript

Objectif

Dans ce chapitre, nous allons définir les grands principes du JavaScript dans l'appel et le rendu d'une page web et, au travers de ses 25 ans d'histoire, ses possibilités, ses limites et son avenir.

## — 1.1 JAVASCRIPT, C'EST QUOI ?

### ◆ 25 ans d'histoire et encore de l'avenir

Le JavaScript est un langage de programmation sous forme de scripts, c'est-à-dire sans phase de compilation du code source vers un langage de plus bas niveau. L'exécution du script se fait directement par le navigateur à partir du code écrit par le développeur.

Le JavaScript est également un **langage événementiel**, à l'écoute de tous les événements qui peuvent se produire, comme un clic ou une minuterie. À chaque événement intercepté, une action peut être déclenchée.

Le JavaScript est aussi un langage **orienté objet**. Il utilise la **notation pointée** pour accéder aux propriétés et fonctionnalités. Chaque élément de la page est un objet de l'arborescence de l'objet maître `window`.

Le JavaScript est principalement employé dans les pages web pour les rendre interactives et dynamiques. Il est maintenant tellement puissant et compatible avec l'ensemble des navigateurs modernes que de véritables applications au format web sont utilisées par des millions d'internautes et remplacent déjà les traditionnels logiciels sous forme d'exécutables. Les interfaces en ligne des messageries, des cartographies, des éditeurs de texte ou même de tableurs sont réalisées en JavaScript et n'ont rien à envier à leurs homologues exécutables.

Le JavaScript est également de plus en plus utilisé comme plateforme serveur pour remplacer à la fois le serveur web (comme Apache, le plus utilisé à ce jour) et le langage de programmation serveur (comme PHP). Nous aborderons cette approche dans la troisième partie de ce guide avec l'introduction à Node.js.

Par convention et par son historique, si aucune précision n'est apportée, le JavaScript dont nous parlons dans ce livre se rapporte au script exécuté dans une page HTML.

Le JavaScript, fort logiquement abrégé JS, n'a pas de logo officiel. Le logo qui a tendance à s'imposer actuellement est un simple carré jaune, avec les lettres noires JS alignées en bas à droite.

Logo du langage JavaScript



## — 1.2 BRÈVE HISTOIRE DU JAVASCRIPT

Brendan Eich, ingénieur informaticien américain, a inventé le JavaScript en 1995 chez Netscape et l'a intégré dans le navigateur du même nom. Brendan Eich participera également à la création de la fondation Mozilla.

À l'origine, le nom du langage était **LiveScript**. Mais Netscape profite à cette époque de la notoriété grandissante de Java pour le renommer en JavaScript. Cet artifice marketing particulièrement malvenu entraîne des confusions sur l'utilité du langage et, près de 25 ans plus tard, porte encore préjudice au JavaScript et à tout son écosystème.

Devant le succès du langage et du navigateur Netscape, Microsoft développe de son côté le langage de script JScript et l'intègre en 1996 dans Internet Explorer 3.

Un **navigateur** est un logiciel de navigation sur l'ensemble des sites Internet du web. Nous emploierons également dans ce guide le terme original anglais de ***browser***.

## — 1.3 LE STANDARD ECMASCRIPT

Le langage JavaScript a été standardisé à partir de 1997 par l'organisation ECMA International sous le nom **ECMAScript** avec le standard numéro ECMA-262. Aujourd'hui, JavaScript, JS, ECMAScript et son abréviation ES désignent le même langage.

Depuis la première version du standard, le langage a naturellement évolué pour prendre en compte les nouvelles technologies matérielles, les nouvelles propositions et les nouveaux besoins des membres de l'organisation, et pour corriger les bugs. Les différentes versions de standards ECMAScript ont ensuite été adoptées et implémentées par les éditeurs de navigateurs, plus ou moins rapidement, et avec plus ou moins de bonne volonté. N'oublions pas que les membres de l'organisation, comme Google, Apple ou Microsoft, sont également de féroces concurrents au quotidien.



Voici un tableau récapitulatif des différentes éditions d'ECMAScript :

Nom de l'édition	Année de publication	Événements notables
1	1997	Premier standard
2	1998	Corrections
3	1999	Améliorations (String, Error, Number...)
4	Abandonnée	Aucun accord entre les membres
5	2009	Support natif du format JSON
6 ou 2015	2015	Nombreuses évolutions qui font du JavaScript un vrai langage moderne
7 ou 2016	2016	Itération annuelle
8 ou 2017	2018	Itération annuelle

Ce tableau montre les difficultés rencontrées par les membres pour tomber d'accord. Il a fallu dix ans entre la publication de la version ES3 et celle de la version ES5, avec entre-temps l'abandon notable de ES4.

Dix ans représentent une éternité sur le web. Des technologies apparaissent, des géants naissent et des champions d'hier meurent en moins d'une décennie.

1999 peut être considérée comme l'année de la véritable démocratisation de l'accès Internet en France avec la création de la société Free et sa première offre d'accès par modem RTC.

En 2009 est sorti l'iPhone 3GS. Il est déjà la troisième itération du premier iPhone, sorti en 2007.

En dix ans, nous sommes passés d'un Internet lent, facturé à la minute, visible sur des moniteurs cathodiques, au smartphone tactile en 3G, sans qu'aucune norme du langage JavaScript ne soit publiée. Il n'y a rien d'étonnant à ce que chaque éditeur ait pris sans attendre des décisions pour adapter son navigateur aux évolutions technologiques. Chaque navigateur a donc développé des fonctionnalités et des rendus qui lui étaient propres dans un esprit de conquête de parts de marché.

Cette absence de concertation a été un véritable calvaire pour les développeurs web, avec de nombreuses versions de leur site à maintenir et avec de nombreuses exceptions et incompatibilités.

Il a fallu encore six ans pour arriver à la version ES6, appelée aussi ES2015, et parvenir à une spécification du langage complète et moderne, qui peut être considérée comme entièrement disponible dans la dernière version à jour de l'ensemble des navigateurs du marché.

La version ES6 a défini de nouvelles formes de notations qui nécessitent un chapitre entier de remise à niveau pour les développeurs web habitués à la syntaxe plus classique (voir chapitre 9, page 127).

Pour éviter de nouveaux délais aussi longs et des changements aussi radicaux après chaque publication, il a été décidé que les éditions suivantes seraient moins ambitieuses et adopteraient le principe d'itérations régulières et annuelles, plus faciles à définir et à implémenter.

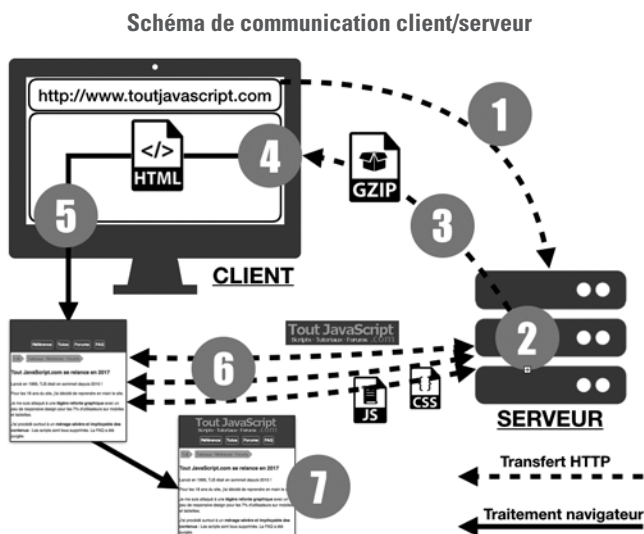
## — 1.4 RAPPEL SUR LE PRINCIPE CLIENT/SERVEUR

Pour bien comprendre le rôle et l'emploi du JavaScript, il est indispensable de maîtriser parfaitement le principe de fonctionnement en client/serveur d'une page web visualisée par un internaute dans son navigateur. L'analogie avec le serveur et son client dans le cadre d'un café est assez juste. Le client commande au serveur un expresso serré. Le serveur prépare la tasse, la pose sur son plateau avec une sucette, une cuillère et un verre d'eau. Et enfin, il dépose l'ensemble de la commande sur la table du client.

Dans le cas d'une page web, le client est le navigateur de l'internaute. Le serveur est l'ensemble de l'architecture informatique hébergée dans un ou plusieurs datacenters de la société éditrice du site Internet. Le navigateur demande par exemple au serveur de son réseau social favori de lui afficher sa page d'accueil. Le serveur construit la page adaptée à son client, avec les actualités de ses amis et ses derniers messages. Le serveur web envoie le code HTML de la page préparée, les images et les autres ressources associées vers le navigateur qui l'interprète et l'affiche à l'utilisateur.

Bien sûr, la comparaison avec une commande de boisson est simpliste. La communication entre le navigateur et le serveur est bien plus codifiée et nécessite l'utilisation stricte du protocole HTTP (HyperText Transfer Protocol) indiqué devant les adresses dans la barre de navigation du browser.

Voici un schéma des communications entre le navigateur et le serveur lors de la demande de la page d'accueil d'un site :



L'affichage d'une page se réalise en sept grandes étapes :

1. Le navigateur envoie une requête HTTP au serveur de google.fr.
2. Le serveur fabrique le code HTML de la page demandée et le compresse.
3. Le serveur envoie le code HTML dans une réponse à la requête HTTP du navigateur.
4. Le navigateur reçoit en réponse le code HTML et le décompresse.
5. Le navigateur interprète le code HTML ligne par ligne et commence l'affichage de la page.

6. Le navigateur envoie de nouvelles requêtes vers le serveur pour chacun des éléments à charger (images, styles, scripts).

7. Le rendu de la page est terminé.

Tous les objets connectés à Internet communiquent entre eux grâce à leur adresse IP, une suite de quatre nombres variant de 0 à 255. Les humains sont plus à l'aise avec les mots qu'avec les chiffres : c'est le protocole DNS (Domain Name System) qui traduit les noms de domaine en leur adresse IP. Le navigateur fait ainsi appel aux serveurs DNS du fournisseur d'accès pour trouver l'adresse IP du serveur de la page demandée.

La compression GZIP des données transportées sur le réseau par le protocole HTTP optimise les temps de rendu, car le temps gagné pour le transfert sur le réseau est largement supérieur aux temps de traitements de compression sur le serveur et de décompression par le navigateur.

Le protocole HTTP/2 est une nouvelle version majeure encore peu utilisée du protocole HTTP destinée à accélérer les transferts sur le réseau. Une seule requête peut maintenant contenir plusieurs ressources à la fois (images, vidéos, fichiers JavaScript...), réduisant fortement la latence et les temps de rendu de la page complète.

Nous verrons à plusieurs reprises dans ce livre les implications concrètes sur le développement web des différentes phases de communication entre les côtés client et serveur.

## — 1.5 LES POSSIBILITÉS DU JAVASCRIPT

Le JavaScript utilisé côté client et exécuté par le navigateur dans une page web est parfaitement adapté pour les traitements suivants :

- ✓ Assistance de saisie des formulaires
  - Contrôle et validation de saisie
  - Affichage de messages d'aide à la saisie
  - Éditeur de texte enrichi
- ✓ Sauvegarde de données sur le poste local
  - Via les cookies
  - Via une zone disque dédiée
- ✓ Gestion des nombres, dates et heures
  - Calculs mathématiques
  - Calculs sur les dates
  - Affichage dans le format du pays de l'utilisateur
- ✓ Animations graphiques
  - Menus et éléments de navigation
  - Animations esthétiques
  - Défilements, diaporamas, zooms
  - Graphiques animés et interactifs
  - Cartographies
  - Jeux

✓ Appels asynchrones vers le serveur pour actualiser la page

- Cours de Bourse
- *Chat* de messagerie
- Alerte en direct
- Sauvegarde de saisie automatique en temps réel
- Affichage de services externes
- Liste des posts d'un réseau social
- Mesure d'audience
- Campagne publicitaire

Il n'existe aucun autre langage intégré aux navigateurs qui soit aussi puissant et universellement reconnu que le JavaScript pour tous ces traitements. Si le programmeur logiciel a l'habitude d'avoir une vaste gamme de langages à choisir selon ses habitudes ou ses besoins, en revanche, le JavaScript n'a pas d'alternative !

Le JavaScript utilisé côté serveur n'a pas du tout la même finalité. Il est capable de remplir l'ensemble des fonctionnalités d'un langage serveur classique comme PHP, ASP, Python, C, Java... De plus en plus de sites sont développés en JavaScript côté serveur : forums, outils de statistiques, interfaces d'administration...

## — 1.6 LIMITES ET SÉCURITÉS DU JAVASCRIPT

Le JavaScript, et les navigateurs qui l'exécutent, ont évolué avec les menaces apparaissant progressivement sur le web. Au début, certains développeurs JavaScript abusaient des *pop-up*, qui ont été rapidement bloqués par les navigateurs. Des scripts perturbateurs tentaient de faire planter le navigateur en lançant des traitements lourds et infinis. Récemment, des scripts parasites de création de crypto-monnaies (minage) sont apparus pour voler de la puissance processeur aux utilisateurs. À chaque fois, les éditeurs de navigateurs ont intégré des parades, en anticipant la future innovation malveillante et en préparant la parade.

La sécurité sur le net est devenue un enjeu vital. Via un programme de Bug Bounty, tous les acteurs du web offrent des récompenses, allant jusqu'à plusieurs dizaines de milliers de dollars, aux chercheurs qui découvrent des failles de sécurité dans leurs logiciels.

Le JavaScript, étant exécuté localement à l'intérieur d'un navigateur qui sécurise son usage, a de nombreuses limites fonctionnelles.

Il n'est pas possible, **sans une action volontaire supplémentaire de l'utilisateur** :

- ✓ à un site d'accéder au disque dur de l'utilisateur (hormis une zone strictement réservée au stockage de données site par site) ;
- ✓ à un site d'accéder à la zone de stockage ou aux cookies d'un autre site ;
- ✓ à un site de lire l'historique de navigation, les mots de passe ou les favoris de l'utilisateur ;
- ✓ à un virus ou un logiciel espion de s'installer ou d'effacer des fichiers ;
- ✓ à un script de faire planter le navigateur, et encore moins le poste complet ;
- ✓ à un script d'abîmer ou de détruire le matériel de l'utilisateur.

Il faut bien prendre conscience que la sécurité du JavaScript et des navigateurs est garantie uniquement en l'absence d'action volontaire de l'utilisateur. Un script ne peut pas installer un logiciel sans l'affichage d'un message d'avertissement, prévenant de son intention. Mais si le visiteur clique et confirme sans lire le message, le logiciel s'installera, avec des conséquences potentiellement dramatiques, car un logiciel installé possède tous les droits d'accès à l'ensemble de l'appareil et du réseau. Le plus souvent, l'objectif des créateurs de scripts et logiciels malveillants est de gagner de l'argent. Par exemple, les **ransomwares**, littéralement « logiciels de rançon », chiffrent le contenu du disque dur et interdisent l'accès aux documents en espérant que l'utilisateur paiera une rançon en échange d'une clé de déblocage.

De la même manière, un script ne peut pas récupérer uniquement par lui-même les informations d'identification sur le site de votre banque en ligne. Mais si un script utilise la technique de **phishing**, ou « hameçonnage », et tente de se faire passer pour le site d'une banque, via des redirections invisibles et une interface identique, la sécurité native est contournée. C'est à l'utilisateur d'être attentif sur ce qu'il fait et sur quel site il le fait.

Le JavaScript ne permet pas non plus d'obtenir l'adresse IP de l'utilisateur.

Les limites imposées par le langage et sa bonne sécurité font partie des raisons qui ont permis son universalité qui le rendent incontournable.

Le JavaScript est un langage de script intégré au code HTML de la page web. Il n'est donc pas possible de cacher son code source au monde extérieur. Le navigateur en a besoin pour l'exécuter, comme il a besoin du code HTML pour interpréter et afficher la page. Des systèmes d'**obfuscation**, littéralement « obscurcissement », existent pour rendre le code difficilement compréhensible par un humain, en le compactant au maximum et en changeant entre autres le nom des fonctions et des variables. Mais l'essence du code est toujours présente et un travail de *reverse engineering*, opération inverse à l'obfuscation, est relativement simple à réaliser sur un script.

Notez que le code JavaScript côté serveur n'est pas du tout visible par le monde extérieur. Il s'exécute sur le serveur et retourne du code HTML ou des données, le plus souvent au format JSON.

## — 1.7 IMPACT SUR LES PERFORMANCES ET LE RÉFÉRENCEMENT

Il peut paraître étrange d'aborder dès le chapitre d'introduction des notions aussi éloignées de la programmation JavaScript que le **taux de rebond** ou le **référencement**. Le succès d'un site web se construit à toutes les étapes de sa vie, depuis sa conception, sa réalisation et jusqu'à son exploitation.

Le JavaScript a un impact très important sur le **temps de chargement** et de rendu de la page dans le navigateur. Pour un résultat final totalement identique, des optimisations simples permettent de gagner jusqu'à une seconde, voire davantage, pour un site particulièrement mal configuré au départ.

Comme le JavaScript joue un rôle central dans l'ergonomie d'un service web, le développeur doit l'utiliser de manière pertinente, dans le but de rendre service aux utilisateurs. Un site peu intuitif ou trop lent fait fuir ses visiteurs dès les premières secondes. La part de visiteurs qui ne consulte qu'une page en moins de quelques secondes est appelée **taux de rebond**.

Les temps de chargement et le taux de rebond sont des indicateurs officiellement reconnus des moteurs de recherche et influencent le référencement d'une page. Le moteur de recherche s'appuie sur ces critères, parmi des centaines d'autres, pour privilégier les sites apportant une réponse satisfaisante aux utilisateurs. N'oublions pas que les moteurs de recherche se livrent aussi entre eux une concurrence sévère sur un marché mondial à plusieurs dizaines de milliards de dollars annuels. Leur objectif est simplement de donner les meilleurs résultats pour satisfaire et fidéliser leurs utilisateurs.

Dans le présent paragraphe, il s'agit simplement d'alerter sur les enjeux et les impacts d'un simple bout de JavaScript, non seulement pour la partie code source, mais aussi sur son résultat. Nous aborderons dans le chapitre 2 les bonnes pratiques, les outils disponibles et la vision d'ensemble à ne pas perdre de vue dans le cadre d'un projet web.

## — 1.8 L'IMAGE DU JAVASCRIPT ET SON AVENIR

Il faut bien l'admettre, le JavaScript, et ses développeurs, ont une bien piètre image de marque. Le grand public le confond inmanquablement avec Java et lui associe son cortège de failles de sécurité et de mises à jour laborieuses. Les informaticiens, au sens large, voient en lui un langage ancien, sans grand intérêt et dépassé, tout juste capable de déclencher des animations inutiles. Les programmeurs lui reprochent sa syntaxe brouillonne et laxiste. Le milieu du web imagine qu'il suffit de le substituer à une bibliothèque de scripts ou à un framework pour répondre à tous les besoins en une ligne de code.

Bref, le JavaScript est totalement incompris, même par des développeurs web qui le connaissent et se souviennent des incompatibilités entre navigateurs, des tests interminables et des rustines répugnantes à mettre en place.

Depuis ECMAScript 6, les navigateurs se comportent bien sur l'ensemble des fonctionnalités de base. Le langage a beaucoup progressé, sans corriger toutefois certaines incohérences issues de son histoire chaotique. Mais il est maintenant possible de créer des projets incroyables. Le chapitre 9 est dédié à une « révision » (au sens de nouvelle vision) du JavaScript, alias ES6.

Le JavaScript, 25 ans après sa création, est enfin standardisé et régulièrement maintenu par une organisation puissante. Il est utilisé, sans qu'ils le sachent, par 4 milliards d'internautes dans le monde, selon des estimations de 2018<sup>1</sup>. Il est au cœur du modèle économique publicitaire du web. En l'absence de langage alternatif, il est donc incontournable. Difficile d'imaginer, sans une révolution complète de l'accès au web, que le JavaScript ne poursuive pas sa croissance aux niveaux usage, compatibilité et fonctionnalités.

## — 1.9 PLACE DU JAVASCRIPT DANS UN PROJET WEB

Une belle équipe de ninjas du JavaScript est probablement en place pour créer et maintenir l'interface d'un projet impressionnant comme Excel Online, version quasi conforme d'Excel, mais sur le web. L'ambition de ce livre n'est pas de faire de vous un ninja du JS mais de vous donner une vision complète du langage pour vous amener à l'employer du mieux possible, au service de vos utilisateurs. Le but est également d'élargir votre regard

---

1. Estimations d'Internet World Stats.

sur d'autres aspects moins liés à la stricte programmation, comme l'optimisation des performances, de l'ergonomie, du référencement ou de la rentabilité.

Soyons pourtant humbles et réalistes. Si le JavaScript est au cœur de l'ergonomie d'un projet web, il n'en demeure pas moins une toute petite partie d'un ensemble plus vaste regroupant de nombreuses compétences indispensables à son succès. Des métiers très divers interviennent au quotidien : graphiste, architecte réseau, rédacteur, modérateur, *community manager*, référenceur, commercial, logisticien, etc. Le développeur web, rarement perçu à la juste mesure de son importance, n'est toutefois qu'un des rouages de l'équipe.







# Hello(s) World(s)

## Objectif

Dans ce chapitre, nous allons immédiatement démarrer la programmation JavaScript via quelques exemples simples et progressifs. Nous mettrons en place en parallèle l'environnement de développement complet avec tous les outils nécessaires pour bien coder et prendre de bonnes habitudes.

## — 2.1 LES EXEMPLES DE HELLO WORLD

### 2.1.1 Hello World numéro 1

Rien ne vaut l'incontournable exemple Hello World, qu'on pourrait traduire par « Salut tout le monde », pour comprendre un langage et démarrer son utilisation, et le doux frisson de satisfaction quand le fameux message apparaît pour la première fois à l'écran.

Pour programmer du code JavaScript et l'exécuter, il n'y a rien à installer. Tout est déjà présent sur votre poste de travail. Pour ce premier exemple, vous avez simplement besoin d'un éditeur de texte, comme l'application Bloc-notes de Windows ou l'éditeur de texte sous macOS, et bien sûr d'un navigateur. Nous verrons plus loin dans ce chapitre comment optimiser votre environnement de travail et utiliser tous les meilleurs outils disponibles.

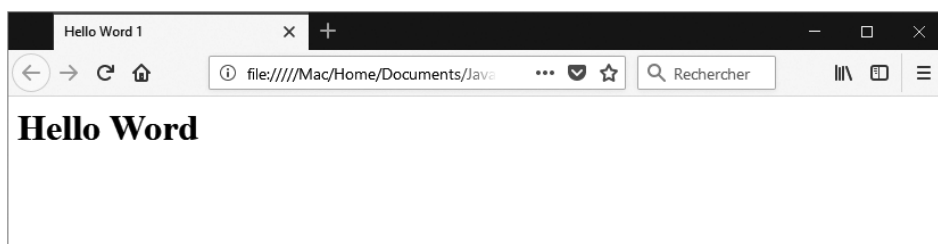
Dans l'explorateur de fichiers, choisissez un répertoire de travail, appelé par exemple **Documents/JavaScript**. Créez dans ce dossier un nouveau fichier au format texte et renommez-le en **hello1.html**. Avec un clic droit sur ce fichier, lancez l'éditeur de texte.

Nous avons un fichier vide prêt à être rempli de notre premier exemple minimaliste :

```
<html>
  <head>
    <title>Hello World 1</title>
  </head>
  <body>
```

```
<h1>
  <script type="text/javascript">
    window.document.write("Hello World");
  </script>
</h1>
</body>
</html>
```

Lancez l'exécution de **hello1.html** en double-cliquant sur le fichier dans l'explorateur de fichiers. Le navigateur par défaut s'ouvre et interprète le code HTML, exécute la ligne de JavaScript et affiche le rendu.



Le résultat équivalent en HTML est exactement ce qu'on attend :

```
<html>
  <head>
    <title>Hello World 1</title>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```

Ce premier exemple très simple est finalement riche d'enseignements. Il nous apprend que :

- ✓ Le code JavaScript est intégré au code HTML classique d'une page web.
- ✓ Le JavaScript est placé dans le bloc marqué par les balises `<script type="text/javascript"></script>`.
- ✓ Le JavaScript utilise la notation pointée `window.document.write()` pour appeler la fonction `write()` de l'objet `document`, lui-même rattaché à l'objet racine `window`.
- ✓ Le JavaScript est interprété et exécuté de manière synchrone, c'est-à-dire séquentiellement et ligne à ligne par le navigateur.
- ✓ Une instruction JavaScript se termine par un point-virgule.

### 2.1.2 Hello World numéro 2

L'exemple **hello1.html** affichait du texte dans le corps du HTML, alors que la page n'était pas encore finalisée. L'exemple **hello2.html** fonctionne différemment, car le script repère la balise `H1` avec son identifiant et lui affecte un nouveau contenu :

```

<html>
  <head>
    <title>Hello World 1</title>
  </head>
  <body>
    <h1 id="monH1"></h1>
  </body>
  <script type="text/javascript">
    document.getElementById("monH1").innerHTML="Hello World";
  </script>
</html>

```

Le résultat est exactement équivalent. À l'issue de l'exécution, la balise `h1` contient bien `Hello World`. La syntaxe est un peu plus complexe, tout en restant d'une compacité appréciable. Ici, la fonction `getElementById()` recherche et retourne l'élément HTML ayant l'identifiant `monH1` dans le document. La propriété `innerHTML` de l'élément trouvé est forcée avec notre message habituel. Nous verrons en détail dans le chapitre 12 toutes les manipulations des éléments du document.

Nous aurions dû écrire `window.document.getElementById()` mais, pour des raisons de lisibilité et de compacité du code, l'objet racine `window` peut être ignoré. Comme dans **hello1.html**, `document.write()` serait suffisant.

### 2.1.3 Hello World numéro 3

Voyons encore une autre façon d'afficher notre message de bienvenue dans **hello3.html**:

```

<html>
  <head>
    <title>Hello World 3</title>
  </head>
  <body id="monBody"></body>
  <script type="text/javascript">
    var monH1=document.createElement("h1");
    monH1.innerHTML="Hello World";
    document.getElementById("monBody").appendChild(monH1);
  </script>
</html>

```

Ici, le code HTML du fichier initial ne contient aucune balise `h1`. Il n'y a que la balise `body` avec un identifiant qui nous permettra de la trouver facilement. Le script crée une variable `monH1` avec l'instruction `var` et lui affecte un nouvel élément HTML de type `H1` grâce à la fonction `createElement()`. La propriété `innerHTML` est ensuite renseignée. Puis la fonction `appendChild()` sur la balise `body` est appelée pour ajouter dans le document l'objet HTML `monH1`.

Finalement, le document HTML contient grâce à nos trois petites lignes de JS un nouvel élément `h1`.

Rassurez-vous, nous verrons toutes les manipulations des éléments HTML dans le chapitre 12.

### 2.1.4 Hello World numéro 4

Les possibilités d'afficher notre message de bienvenue par programmation JavaScript ne sont pas encore épuisées. Voici un nouvel exemple d'affichage de message par boîte de dialogue :

```
<html>
  <head>
    <title>Hello World 4</title>
  </head>
  <body></body>
  <script type="text/javascript">
    alert("Hello World");
  </script>
</html>
```

Le rendu est différent car le message ne s'affiche pas dans la page mais dans une boîte de dialogue gérée par le navigateur :



Une boîte de dialogue est dite **modale**, c'est-à-dire bloquante, car elle impose à l'utilisateur une action pour qu'il puisse continuer sa navigation. Ici, il doit forcément valider le message en cliquant sur le bouton OK. Nous verrons dans les paragraphes suivants que les boîtes de dialogues ne sont pas fiables et sont à éviter pour les sites en ligne.

La fonction `alert()` est liée à l'objet racine `window` : on peut donc écrire `window.alert()` ou `alert()` seulement.

### 2.1.5 Hello World numéro 5

Voyons maintenant un dernier exemple d'affichage de notre message de bienvenue dans le fichier **hello5.html** :

```
<html>
  <head>
    <title>Hello World 5</title>
  </head>
  <body></body>
  <script type="text/javascript">
    var message="Hello World";
```

```

    console.log(message);
  </script>
</html>

```

À l'exécution, la page reste totalement blanche. Pourtant la fonction `log()` de l'objet `console` a bien affiché le contenu de la variable `message`. Voyons maintenant pourquoi et comment utiliser la console des navigateurs.

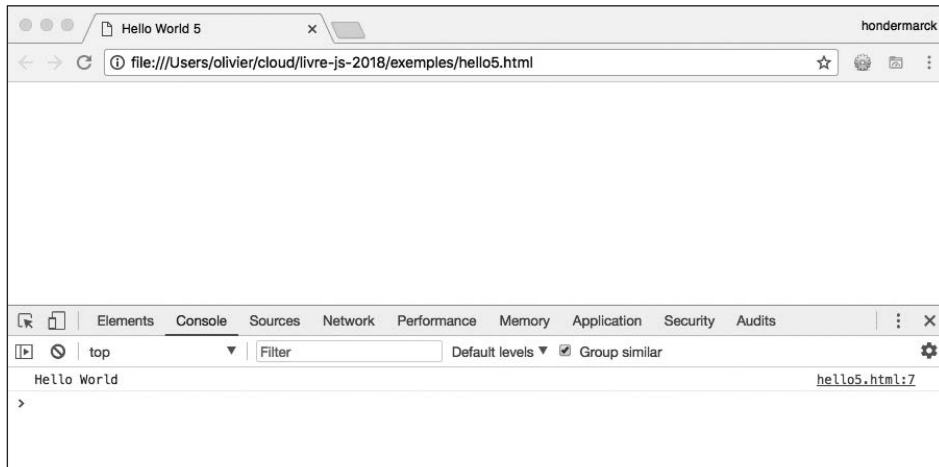
## 2.2 LA CONSOLE DES NAVIGATEURS

Tous les navigateurs modernes proposent aux développeurs web des outils d'aide à la programmation et au débogage. On parle souvent de « console JavaScript » mais la console est en fait bien plus puissante et s'appelle la plupart du temps « Outils de développement ». Son usage est bien sûr réservé aux phases de programmation et la console n'est pas affichée pour le visiteur lambda.

Une combinaison de touches suffit pour faire apparaître la console. Selon les navigateurs et le système d'exploitation, la combinaison varie. Voici les cas les plus fréquents :

- ✓ Pour Chrome sur macOS : `⌘ + ⌥ + J`.
- ✓ Pour Safari sur macOS : `⌘ + ⌥ + C`.
- ✓ Pour Chrome, Firefox et Safari sur Windows : `CTRL + MAJ + I`.
- ✓ Pour Internet Explorer et Edge : `F12`.

Reprenons notre fichier **hello5.html**, exécuté ici dans Chrome sous Mac et observons ce que la console affiche :



Sous l'écran de rendu de la page HTML, la console s'ouvre et on voit apparaître le message de bienvenue affiché avec `console.log()`. **Log** signifie « journal » en anglais. Il s'agit d'un terme très utilisé en informatique, qui désigne un ensemble de données archivées, généralement des informations techniques sur l'enchaînement d'opérations de traitements.

Notez que l'interface de la fenêtre de console est légèrement différente selon les navigateurs. Mais les principales options sont toujours présentes.

## 2.2.1 La console JavaScript

La console JavaScript affiche par défaut l'ensemble des erreurs graves détectées dans la page, qu'il s'agisse d'une erreur de script, comme ici une variable non définie, ou d'une erreur d'accès à des ressources sur Internet, comme une image introuvable :



La console est également le moyen idéal pour interagir avec la page en cours de consultation et les scripts associés. Tapez une instruction devant le curseur et la console l'exécutera dans le contexte de la page actuelle et retournera son résultat. Dans le script **hello5.html**, la variable `message` est renseignée. Vous pouvez aussi accéder au titre de la page avec la notation `document.title` :



L'objet `console` accessible en JavaScript permet d'afficher des informations en cours d'exécution dans le journal de log, dans un format lisible et tout en restant parfaitement invisible aux visiteurs. C'est très utile pour contrôler la valeur de données en cours d'utilisation et pour vérifier que le comportement réel est bien celui prévu.

Avant la généralisation de la console dans les navigateurs, les développeurs JavaScript avaient pris la mauvaise habitude d'utiliser la fonction `alert()` pour afficher le contenu d'une variable. Il faut bannir cet usage : il bloque la navigation, affiche le message à tous les visiteurs et ne peut contenir que du texte brut. De plus, le message est parfois bloqué par les navigateurs.

Explorons plus en détail les possibilités de la console JavaScript et ses options.

En cliquant sur le bouton représentant un cercle barré ou parfois une poubelle, vous videz la zone d'affichage.

En cliquant sur le bouton représentant une roue dentée d'engrenage, vous ouvrez la fenêtre de paramétrages de la console. Vous pouvez par exemple ajouter un horodatage,