

Johann Pardanaud
Sébastien de la Marck

DÉCOUVREZ LE LANGAGE

JAVASCRIPT



EYROLLES

DÉCOUVREZ LE LANGAGE

JAVASCRIPT

Vous connaissez le HTML et avez toujours rêvé d'améliorer le confort de navigation de vos sites web tout en les rendant plus attrayants pour vos visiteurs ? Ce livre est fait pour vous ! Conçu pour les débutants, il vous apprendra pas à pas la programmation en JavaScript, l'un des langages du Web le plus utilisé au monde.

QU'ALLEZ-VOUS APPRENDRE ?

- Premiers pas en JavaScript
- Les variables et les conditions
- Les boucles et les fonctions
- Les objets et les tableaux
- Déboguer le code
- TP : convertir un nombre en toutes lettres

Modeler les pages web

- Manipuler le code HTML
- Les événements et les formulaires
- Manipuler le CSS
- TP : un formulaire interactif

Les objets et les design patterns

- Les objets et les chaînes de caractères
- Les expressions régulières
- Les données numériques
- La gestion du temps
- Les tableaux et les images
- Les polyfills et les wrappers
- Les closures

L'échange de données avec Ajax

- XMLHttpRequest
- Upload via une iframe
- Dynamic Script Loading
- TP : un système d'autocomplétion

JavaScript et HTML 5

- L'audio et la vidéo
- L'élément Canvas
- L'API File et le drag & drop

À PROPOS DES AUTEURS

Sébastien de la Marck, passionné des technologies du Web et plus particulièrement du JavaScript, est l'auteur de plusieurs programmes développés avec ce langage. Il considère que le JavaScript doit être connu des webmasters, en plus du trio HTML/CSS et PHP.

Johann Pardanaud, étudiant et féru d'informatique, découvre les joies de la programmation à la fin de ses années de collège. Très vite, il se lance dans le développement web et tombe sur son langage de prédilection, le JavaScript, qu'il décide d'enseigner via OpenClassrooms.

L'ESPRIT D'OPENCLASSROOMS

Des cours ouverts, riches et vivants, conçus pour tous les niveaux et accessibles à tous gratuitement sur notre plate-forme d'e-éducation : www.openclassrooms.com. Vous y vivrez une véritable expérience communautaire de l'apprentissage, permettant à chacun d'apprendre avec le soutien et l'aide des autres étudiants sur les forums. Vous profiterez des cours disponibles partout, tout le temps : sur le Web, en PDF, en eBook, en vidéo...

www.editions-eyrolles.com

DÉCOUVREZ LE LANGAGE

JAVASCRIPT

DANS LA MÊME COLLECTION

A. BACCO. – **Développez votre site web avec le framework Symfony3.**

N° 14403, 2016, 536 pages.

M. CHAVELLI. – **Découvrez le framework PHP Laravel.**

N° 14398, 2016, 336 pages.

R. DE VISSCHER. – **Découvrez le langage Swift.**

N° 14397, 2016, 128 pages.

M. LORANT. – **Développez votre site web avec le framework Django.**

N° 21626, 2015, 285 pages.

E. LALITTE. – **Apprenez le fonctionnement des réseaux TCP/IP.**

N° 21623, 2015, 300 pages.

M. NEBRA, M. SCHALLER. – **Programmez avec le langage C++.**

N° 21622, 2015, 674 pages.

SUR LE MÊME THÈME

P. MARTIN, J. PAULI, C. PIERRE DE GEYER, É. DASPET. – **PHP 7 avancé.**

N° 14357, 2016, 732 pages.

R. GOETTER. – **CSS 3 Flexbox.**

N° 14363, 2016, 152 pages.

E. BIERNAT, M. LUTZ. – **Data science : fondamentaux et études de cas.**

N° 14243, 2015, 312 pages.

B. PHILIBERT. – **Bootstrap 3 : le framework 100 % web design.**

N° 14132, 2015, 318 pages.

C. DELANNOY. – **Le guide complet du langage C.**

N° 14012, 2014, 844 pages.

Retrouvez nos bundles (livres papier + e-book) et livres numériques sur
<http://izibook.eyrolles.com>

Johann Pardanaud
Sébastien de la Marck

DÉCOUVREZ LE LANGAGE

JAVASCRIPT

EYROLLES



ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.

© Groupe Eyrolles, 2017. ISBN Eyrolles : 978-2-212-14399-7
© OpenClassrooms, 2016

Avant-propos

Depuis sa création, le JavaScript n'a cessé de croître en popularité. Longtemps relégué au simple rang de « langage de scripts basiques », il est devenu ces dernières années un langage absolument incontournable pour toute personne s'intéressant à la création de sites web. Il est désormais entré dans la cour des grands, aux côtés d'HTML, de CSS 3 et de PHP.

Le JavaScript est donc devenu un acteur majeur dans la création de sites web. Il a été popularisé par des sites sociaux comme Facebook ou Twitter, qui en font une utilisation massive afin d'améliorer le confort de navigation des utilisateurs : moins de rechargement de pages, interactivité... Le JavaScript a aussi gagné en popularité grâce aux *frameworks* comme jQuery ou Angular.js. Ces frameworks apportent des fonctionnalités en plus, tout en simplifiant et en accélérant le développement de scripts. Cependant, si l'on veut pouvoir tirer parti de ces frameworks, il faut connaître le langage sur lequel ils s'appuient : le JavaScript.

Beaucoup de développeurs apprennent à utiliser jQuery ou Angular.js sans connaître le JavaScript. C'est comme conduire une voiture sans connaître le Code de la route : gare aux accidents ! Les frameworks permettent de faire de grandes choses, mais pas tout ! C'est pourquoi il faut souvent sortir des sentiers battus et mettre les mains dans le JavaScript.

Dans ce livre, nous mettons un point d'honneur à vous enseigner le JavaScript tel que nous aurions voulu qu'il nous soit enseigné. Apprendre JavaScript n'est pas simple ! Bien qu'inspiré de grands langages comme le C, c'est un langage pourvu d'une logique qui lui est propre et que vous ne retrouverez que dans peu d'autres langages. Mais peu importe, ce langage est formidable et passionnant, et offre des possibilités de moins en moins limitées, tant et si bien qu'il devient utilisable ailleurs que dans les pages HTML ! En effet, le JavaScript peut être employé pour réaliser des extensions pour les navigateurs, des programmes et même de la programmation pour serveurs.

Structure de l'ouvrage

Le plan de ce livre a été conçu pour faciliter votre apprentissage du JavaScript. Voici le chemin que nous allons parcourir :

- **Les bases du JavaScript** : cette partie sera consacrée à l'apprentissage des bases du langage. Nous passerons en revue tout ce qui concerne les particularités du langage et nous reprendrons toutes les notions de programmation afin que les débutants en programmation ne soient pas perdus. À la fin de cette partie, vous serez invités à réaliser une application capable d'afficher un nombre en toutes lettres ; elle réutilisera l'ensemble des notions que vous aurez acquises et mettra votre capacité de réflexion à rude épreuve.
- **Modeler les pages web** : après une première partie consacrée aux bases, vous serez ici plongés dans la modélisation de pages web « dynamiques ». Le dynamisme est une notion importante du JavaScript, c'est cela qui permet à l'utilisateur d'interagir avec votre page. Vous découvrirez aussi comment modifier le contenu d'une page web, créer et manipuler des événements et modifier le style CSS de vos éléments HTML. Comme dans la première partie, un TP est destiné à vous faire travailler sur ce que vous aurez appris au cours de votre lecture.
- **Les objets et les design patterns** : une fois les connaissances nécessaires en termes d'interaction avec une page web et les utilisateurs acquises, vous pourrez alors vous investir dans une utilisation un peu plus avancée du JavaScript. Vous y découvrirez comment créer des objets évolués et quels sont les objets natifs les plus intéressants à manipuler. Vous serez aussi initiés au traitement avancé des chaînes de caractères, des nombres et des tableaux, et vous apprendrez à gérer le temps et les images.
- **L'échange de données avec Ajax** : nous aborderons ici la notion d'échange de données entre un client et un serveur et étudierons les diverses possibilités qui s'offrent à nous afin de faire communiquer nos scripts avec un serveur sans pour autant devoir recharger la page web. Un point sera fait sur les structures de données afin que vous sachiez sous quelle forme transférer vos données et vous découvrirez comment réaliser un système d'upload de fichier. La partie se terminera avec la création d'un système d'autocomplétion, chose qui vous resservira très certainement.
- **JavaScript et HTML 5** : vous avez maintenant terminé votre apprentissage sur tout ce qui était essentiel au JavaScript, dans cette partie nous irons donc un peu plus loin et explorerons la manière dont le HTML 5 permet au JavaScript d'aller encore plus loin. Nous ferons avant tout un point sur ce qu'apporte le HTML 5 par rapport au JavaScript puis nous nous pencherons sur quelques points essentiels, à savoir : la manipulation des balises `< audio >`, `< video >` et `< canvas >`, la gestion des fichiers et la mise en place d'un système de Drag & Drop.
- **Annexes** : vous trouverez ici de quoi approfondir vos connaissances en termes de débogage de code et vous découvrirez les closures qui, bien qu'un brin complexes, vous apporteront un confort appréciable dans votre manière de développer. L'ultime chapitre vous donnera un récapitulatif de certains points essentiels du JavaScript et vous montrera que ce langage peut se rendre très utile même en dehors des navigateurs web.

Comment lire ce livre ?

Suivez l'ordre des chapitres

Lisez ce livre comme on lit un roman. Il a été conçu pour cela.

Contrairement à beaucoup de livres techniques où il est courant de lire en diagonale et de sauter certains chapitres, il est ici fortement recommandé de suivre l'ordre du livre, à moins que vous ne soyez déjà un peu expérimenté.

Pratiquez en même temps

Pratiquez régulièrement. N'attendez pas d'avoir fini de lire ce livre pour allumer votre ordinateur.



<http://odyssey.sdlm.be/javascript.htm>



Des exercices interactifs sont proposés à la fin de certains chapitres.

Les compléments web

Pour télécharger le code source des exemples de cet ouvrage, veuillez-vous rendre à cette adresse : <http://www.editions-eyrolles.com/dl/0014399>.

Remerciements

Nous tenons à remercier les personnes qui ont contribué de près ou de loin à l'écriture de ce cours. Sans elles, ce cours aurait eu du mal à voir le jour !

Commençons par trois accros du JavaScript :

Yann Logan – Golmote (<https://openclassrooms.com/membres/golmote-13169>) –, pour ses relectures et ses conseils avisés.

Xavier Montillet – xavierm02 (<https://openclassrooms.com/membres/xavierm02-20626>) –, pour ses remarques sur tout ce qui nous semblait négligeable mais qui importait beaucoup au final !

Benoît Mariat – restimel (<https://openclassrooms.com/membres/restimel-61309>) –, qui a activement participé aux débuts quelque peu chaotiques du cours.

Merci encore à vous trois, on se demande toujours ce qu'on aurait fait sans vous !

S'ensuivent certaines personnes d'OpenClassrooms :

Jonathan Baudoin – John-John (<https://openclassrooms.com/membres/john-john-35734>) –, qui a supporté nos coups de flemme pendant plus d'un an !

Pierre Dubuc – karamilo (<https://openclassrooms.com/membres/karamilo-40796>) –, pour nous avoir lancés dans ce projet immense et nous avoir aidés à mettre en place les premiers chapitres du cours.

Mathieu Nebra – M@teo21 (<https://openclassrooms.com/membres/mateo21>) –, sans qui ce cours n'aurait jamais vu le jour puisqu'il est le créateur du Site du Zéro OpenClassrooms.

Merci aussi à nos familles respectives, qui nous ont encouragés pendant plus d'un an et demi !

Merci à vous tous !

Table des matières

Première partie – Les bases du JavaScript	1
1 Introduction au JavaScript	3
Qu'est-ce que le JavaScript ?	3
<i>Un langage de programmation</i>	3
<i>Programmer des scripts</i>	4
<i>Un langage orienté objet</i>	4
<i>Le JavaScript, le langage de scripts</i>	5
<i>Le JavaScript, pas que le Web</i>	6
Petit historique du langage	6
<i>L'ECMAScript et ses dérivés</i>	7
<i>Les versions du JavaScript</i>	7
<i>Un logo inconnu</i>	8
En résumé	8
2 Premiers pas en JavaScript	9
Afficher une boîte de dialogue	9
<i>Hello world!</i>	9
<i>Les nouveautés</i>	10
<i>La boîte de dialogue alert()</i>	11
La syntaxe du JavaScript	11
<i>Les instructions</i>	11
<i>Les espaces</i>	12
<i>Indentation et présentation</i>	12
<i>Les commentaires</i>	13
<i>Commentaires de fin de ligne</i>	13
<i>Commentaires multilignes</i>	14
<i>Les fonctions</i>	14

Emplacement du code dans la page	14
<i>Le JavaScript « dans la page »</i>	15
<i>L'encadrement des caractères réservés</i>	15
<i>Le JavaScript externe</i>	16
<i>Positionner l'élément <script></i>	16
Quelques aides	17
<i>Les documentations</i>	17
<i>Tester rapidement certains codes</i>	18
En résumé	18
3 Les variables	19
Qu'est-ce qu'une variable ?	19
<i>Déclarer une variable</i>	19
<i>Les types de variables</i>	21
<i>Tester l'existence de variables avec typeof</i>	22
Les opérateurs arithmétiques	22
<i>Quelques calculs simples</i>	23
<i>Simplifier encore plus les calculs</i>	24
Initiation à la concaténation et à la conversion des types	24
<i>La concaténation</i>	24
<i>Interagir avec l'utilisateur</i>	25
<i>Convertir une chaîne de caractères en nombre</i>	26
<i>Convertir un nombre en chaîne de caractères</i>	27
En résumé	27
4 Les conditions	29
La base de toute condition : les booléens	29
<i>Les opérateurs de comparaison</i>	30
<i>Les opérateurs logiques</i>	31
<i>L'opérateur ET</i>	31
<i>L'opérateur OU</i>	32
<i>L'opérateur NON</i>	32
<i>Combiner les opérateurs</i>	32
La condition if else	33
<i>La structure if pour dire « si »</i>	33
<i>Petit intermède : la fonction confirm()</i>	34
<i>La structure else pour dire « sinon »</i>	35
<i>La structure else if pour dire « sinon si »</i>	35
La condition switch	36
Les ternaires	39
Les conditions sur les variables	40
<i>Tester l'existence de contenu d'une variable</i>	40
<i>Le cas de l'opérateur OU</i>	41

Un petit exercice pour la forme !	41
<i>Présentation de l'exercice</i>	41
<i>Correction</i>	42
En résumé	43
5 Les boucles	45
L'incréméntation	45
<i>Le fonctionnement</i>	45
<i>L'ordre des opérateurs</i>	46
La boucle while	47
<i>Répéter tant que...</i>	47
<i>Exemple pratique</i>	48
<i>Quelques améliorations</i>	49
La boucle do while	49
La boucle for	50
<i>for, la boucle conçue pour l'incréméntation</i>	50
<i>Reprenons notre exemple</i>	51
<i>Portée des variables de boucles</i>	52
<i>Priorité d'exécution</i>	52
En résumé	53
6 Les fonctions	55
Concevoir des fonctions	55
<i>Créer sa première fonction</i>	56
<i>Un exemple concret</i>	57
La portée des variables	58
<i>La portée des variables</i>	58
<i>Les variables globales</i>	59
<i>Les variables globales ? Avec modération !</i>	59
Les arguments et les valeurs de retour	60
<i>Les arguments</i>	61
Les fonctions anonymes	66
<i>Les fonctions anonymes : les bases</i>	66
<i>Retour sur l'utilisation des points-virgules</i>	67
<i>Les fonctions anonymes : isoler son code</i>	68
En résumé	71
7 Les objets et les tableaux	73
Introduction aux objets	73
<i>Que contiennent les objets ?</i>	74
<i>Le constructeur</i>	74
<i>Les propriétés</i>	74
<i>Les méthodes</i>	74

<i>Exemple d'utilisation</i>	74
<i>Objets natifs déjà rencontrés</i>	75
Les tableaux	75
<i>Les indices</i>	76
<i>Déclarer un tableau</i>	76
<i>Récupérer et modifier des valeurs</i>	77
Opérations sur les tableaux	77
<i>Ajouter et supprimer des items</i>	77
<i>Chaînes de caractères et tableaux</i>	78
Parcourir un tableau	79
<i>Parcourir avec for</i>	79
<i>Attention à la condition</i>	80
Les objets littéraux	80
<i>La syntaxe d'un objet</i>	81
<i>Accès aux items</i>	81
<i>Ajouter des items</i>	82
<i>Parcourir un objet avec for in</i>	82
<i>Utilisation des objets littéraux</i>	83
Exercice récapitulatif	83
<i>Énoncé</i>	83
<i>Correction</i>	84
En résumé	85
8 Déboguer le code	87
En quoi consiste le débogage ?	87
<i>Les bogues</i>	87
<i>Le débogage</i>	88
Les kits de développement et leur console	88
Aller plus loin avec la console	91
Utiliser les points d'arrêt	92
<i>Les points d'arrêt</i>	94
<i>La pile d'exécution</i>	96
En résumé	98
9 TP : convertir un nombre en toutes lettres	99
Présentation de l'exercice	99
<i>La marche à suivre</i>	99
<i>L'orthographe des nombres</i>	100
<i>Tester et convertir les nombres</i>	100
<i>Retour sur la fonction parseInt()</i>	100
<i>La fonction isNaN()</i>	101
<i>Il est temps de se lancer !</i>	101
Correction	102
<i>Le corrigé complet</i>	102

<i>Les explications</i>	103
Conclusion	110
Deuxième partie – Modeler les pages web	111
10 Manipuler le code HTML : les bases	113
Le Document Object Model	113
<i>Petit historique</i>	114
<i>L'objet window</i>	114
<i>Le document</i>	116
Naviguer dans le document	116
<i>La structure DOM</i>	116
<i>Accéder aux éléments</i>	117
<i>getElementById()</i>	117
<i>getElementsByTagName()</i>	118
<i>getElementsByName()</i>	119
<i>Accéder aux éléments grâce aux technologies récentes</i>	119
<i>L'héritage des propriétés et des méthodes</i>	120
<i>Notion d'héritage</i>	121
Éditer les éléments HTML	121
<i>Les attributs</i>	121
<i>Les attributs accessibles</i>	122
<i>La classe</i>	123
<i>Le contenu : innerHTML</i>	124
<i>Récupérer du HTML</i>	125
<i>Ajouter ou éditer du HTML</i>	125
innerText et textContent	126
<i>innerText</i>	126
<i>textContent</i>	127
<i>Tester le navigateur</i>	127
En résumé	129
11 Manipuler le code HTML : les notions avancées	131
Naviguer entre les nœuds	131
<i>La propriété parentNode</i>	131
<i>nodeType et nodeName</i>	132
<i>Lister et parcourir des nœuds enfants</i>	133
<i>nodeValue et data</i>	134
<i>childNodes</i>	134
<i>nextSibling et previousSibling</i>	135
<i>Attention aux nœuds vides</i>	136
Créer et insérer des éléments	137
<i>Ajouter des éléments HTML</i>	137
<i>Création de l'élément</i>	137
<i>Affecter des attributs</i>	138

Insérer l'élément	138
Ajouter des nœuds textuels	139
Notions sur les références.	141
Les références.	142
Les références avec le DOM	142
Cloner, remplacer, supprimer...	143
Cloner un élément	143
Remplacer un élément par un autre.	143
Supprimer un élément	144
Autres actions	144
Vérifier la présence d'éléments enfants	144
Insérer à la bonne place : <i>insertBefore()</i>	145
Une bonne astuce : <i>insertAfter()</i>	145
Algorithme.	146
Mini TP : recréer une structure DOM	146
Exercice 1	146
Corrigé	147
Exercice 2	148
Corrigé	149
Exercice 3	150
Corrigé	151
Exercice 4	152
Corrigé	153
Conclusion des mini TP	154
En résumé	155
12 Les événements	157
Que sont les événements ?	157
La théorie	157
Le focus.	159
La pratique.	159
Le mot-clé <i>this</i>	159
Retour sur le focus	160
Bloquer l'action par défaut de certains événements.	160
L'utilisation de <i>javascript:</i> dans les liens	161
Les événements au travers du DOM.	162
Le DOM-0	162
Le DOM-1	163
Le DOM-2	163
Les phases de capture et de bouillonnement	165
L'objet Event.	166
Généralités sur l'objet Event	166
Les fonctionnalités de l'objet Event	167
Résoudre les problèmes d'héritage des événements.	173
Le problème.	173
La solution	174

En résumé	176
13 Les formulaires	179
Les propriétés	179
<i>Un classique : value.</i>	179
<i>Les booléens avec disabled, checked et readonly.</i>	180
<i>Les listes déroulantes avec selectedIndex et options.</i>	181
Les méthodes et un retour sur quelques événements	182
<i>Les méthodes spécifiques à l'élément <form>.</i>	182
<i>La gestion du focus et de la sélection</i>	183
<i>Explications sur l'événement change</i>	184
En résumé	184
14 Manipuler le CSS	185
Éditer les propriétés CSS	185
<i>Quelques rappels sur le CSS</i>	185
<i>Éditer les styles CSS d'un élément</i>	186
Récupérer les propriétés CSS	188
<i>La fonction getComputedStyle()</i>	188
<i>Les propriétés de type offset.</i>	188
<i>La propriété offsetParent</i>	189
Votre premier script interactif !	193
<i>Présentation de l'exercice</i>	193
<i>Corrigé</i>	194
<i>L'exploration du code HTML</i>	195
<i>L'ajout des événements mousedown et mouseup</i>	195
<i>La gestion du déplacement de notre élément</i>	196
<i>Empêcher la sélection du contenu des éléments déplaçables.</i>	197
En résumé	198
15 Déboguer le code	199
L'inspecteur web	199
<i>Les règles CSS.</i>	201
<i>Les points d'arrêt</i>	202
En résumé	204
16 TP : un formulaire interactif	205
Présentation de l'exercice	205
Corrigé	207
<i>La solution complète : HTML, CSS et JavaScript.</i>	207
<i>Explications</i>	213
<i>La mise en place des événements : partie 1</i>	217
<i>La mise en place des événements : partie 2</i>	218

Troisième partie – Les objets et les design patterns 221

17 Les objets 223

- Petite problématique 223
- Objet constructeur 224
 - Définir via un constructeur 224
 - Utiliser l'objet 225
 - Modifier les données 226
- Ajouter des méthodes 227
 - Ajouter une méthode 227
 - Définir une méthode dans le constructeur 227
 - Ajouter une méthode via prototype 228
 - Ajouter des méthodes aux objets natifs 229
 - Remplacer des méthodes 231
 - Limitations 231
- Les namespaces 231
 - Définir un namespace 232
 - Un style de code 232
 - L'emploi de *this* 233
 - Vérifier l'unicité du namespace 234
- Modifier le contexte d'une méthode 235
- L'héritage 237

18 Les chaînes de caractères 241

- Les types primitifs 241
- L'objet String 243
 - Propriétés 243
 - Méthodes 243
- La casse et les caractères 243
 - toLowerCase()* et *toUpperCase()* 243
 - Accéder aux caractères 244
 - Obtenir le caractère en ASCII 244
 - Créer une chaîne de caractères depuis une chaîne ASCII 244
 - Supprimer les espaces avec *trim()* 245
- Rechercher, couper et extraire 245
 - Connaître la position avec *indexOf()* et *lastIndexOf()* 245
 - Utiliser le tilde avec *indexOf()* et *lastIndexOf()* 246
 - Extraire une chaîne avec *substring()*, *substr()* et *slice()* 247
 - Couper une chaîne en un tableau avec *split()* 248
- Tester l'existence d'une chaîne de caractères 248
- En résumé 250

19 Les expressions régulières : les bases	251
Les regex en JavaScript	251
<i>Utilisation</i>	252
Recherche de mots	253
<i>Début et fin de chaîne</i>	254
Les caractères et leurs classes	255
<i>Les intervalles de caractères</i>	255
<i>Exclure des caractères</i>	255
<i>Trouver un caractère quelconque</i>	256
Les quantificateurs	256
<i>Les trois symboles quantificateurs</i>	256
<i>Les accolades</i>	257
Les métacaractères	258
<i>Les métacaractères au sein des classes</i>	258
Types génériques et assertions	259
<i>Les types génériques</i>	259
<i>Les assertions</i>	260
En résumé	260
20 Les expressions régulières : les notions avancées	261
Construire une regex	261
L'objet RegExp	262
<i>Méthodes</i>	263
<i>Propriétés</i>	263
Les parenthèses	263
<i>Les parenthèses capturantes</i>	263
<i>Les parenthèses non capturantes</i>	265
Les recherches non greedy	265
Rechercher et remplacer	266
<i>Utiliser replace() sans regex</i>	267
<i>L'option g</i>	267
<i>Rechercher et capturer</i>	267
<i>Utiliser une fonction pour le remplacement</i>	269
Autres recherches	270
<i>Rechercher la position d'une occurrence</i>	270
<i>Récupérer toutes les occurrences</i>	270
<i>Couper avec une regex</i>	271
En résumé	271
21 Les données numériques	273
L'objet Number	273

L'objet Math	274
<i>Les propriétés</i>	275
<i>Les méthodes</i>	275
Les inclassables	277
<i>Les fonctions de conversion</i>	278
<i>Les fonctions de contrôle</i>	278
En résumé	280
22 La gestion du temps	281
Le système de datation	281
<i>Introduction aux systèmes de datation</i>	281
<i>L'objet Date</i>	282
<i>Mise en pratique : calculer le temps d'exécution d'un script</i>	284
Les fonctions temporelles	284
<i>Utiliser setTimeout() et setInterval()</i>	285
<i>Annuler une action temporelle</i>	286
<i>Mise en pratique : les animations</i>	287
En résumé	289
23 Les tableaux	291
L'objet Array	291
<i>Le constructeur</i>	291
<i>Les propriétés</i>	292
Les méthodes	292
<i>Concaténer deux tableaux</i>	293
<i>Parcourir un tableau</i>	293
<i>Rechercher un élément dans un tableau</i>	295
<i>Trier un tableau</i>	296
<i>Extraire une partie d'un tableau</i>	298
<i>Remplacer une partie d'un tableau</i>	299
<i>Tester l'existence d'un tableau</i>	299
Les piles et les files	300
<i>Retour sur les méthodes étudiées</i>	300
<i>Les piles</i>	300
<i>Les files</i>	301
<i>Quand les performances sont absentes : unshift() et shift()</i>	302
En résumé	302
24 Les images	303
L'objet Image	303
<i>Le constructeur</i>	303
<i>Les propriétés</i>	304
<i>Événements</i>	304

<i>Particularités</i>	305
Mise en pratique	305
En résumé	309
25 Les polyfills et les wrappers	311
Introduction aux polyfills	311
<i>La problématique</i>	311
<i>La solution</i>	312
<i>Quelques polyfills importants</i>	312
Introduction aux wrappers	313
<i>La problématique</i>	313
<i>La solution</i>	313
En résumé	318
26 Les closures	319
Les variables et leurs accès	319
Comprendre le problème	321
<i>Premier exemple</i>	321
<i>Un cas concret</i>	322
Explorer les solutions	322
Une autre utilité, les variables statiques	325
En résumé	328
Quatrième partie – L'échange de données avec Ajax	329
27 Qu'est-ce que l'Ajax ?	331
Introduction au concept	331
<i>Présentation</i>	331
<i>Fonctionnement</i>	332
Les formats de données	332
<i>Présentation</i>	332
<i>Utilisation</i>	333
En résumé	335
28 XMLHttpRequest	337
L'objet XMLHttpRequest	337
<i>Présentation</i>	337
<i>XMLHttpRequest, versions 1 et 2</i>	338

Première version : les bases	338
<i>Préparation et envoi de la requête</i>	338
<i>Synchrone ou asynchrone ?</i>	339
<i>Transmission des paramètres</i>	340
<i>Réception des données</i>	341
<i>Requête asynchrone : spécifier la fonction de callback</i>	341
<i>Traitement des données</i>	342
<i>Récupération des en-têtes de la réponse</i>	344
<i>Mise en pratique</i>	344
<i>XHR et les tests locaux</i>	346
<i>Gestion des erreurs</i>	346
Résoudre les problèmes d'encodage.	347
<i>L'encodage pour les débutants</i>	347
<i>Une histoire de normes</i>	347
<i>L'encodage et le développement web</i>	348
<i>L'Ajax et l'encodage des caractères</i>	349
Seconde version : usage avancé	352
<i>Les requêtes cross-domain</i>	352
<i>Une sécurité bien restrictive</i>	352
<i>Autoriser les requêtes cross-domain</i>	353
<i>Nouvelles propriétés et méthodes</i>	354
En résumé	359
29 Upload via une iframe	361
Manipulation des iframes	361
<i>Les iframes</i>	361
<i>Accéder au contenu</i>	361
Chargement de contenu	362
<i>Charger une iframe</i>	362
<i>Détecter le chargement</i>	363
Récupération du contenu	364
<i>Récupérer des données JavaScript</i>	364
<i>Exemple complet</i>	364
Le système d'upload	365
<i>Le code côté serveur : upload.php</i>	366
En résumé	367
30 Dynamic Script Loading	369
Un concept simple	369
Un premier exemple	370
Avec des variables et du PHP	370
Le DSL et le format JSON	371
<i>Charger du JSON</i>	371

<i>Petite astuce pour le PHP</i>	372
En résumé	373
31 Débuguer le code	375
L'analyse des requêtes	375
<i>Lister les requêtes</i>	375
<i>Analyser une requête</i>	376
32 TP : un système d'autocomplétion	379
Présentation de l'exercice	379
<i>Les technologies à employer</i>	379
<i>Principe de l'autocomplétion</i>	380
<i>Conception</i>	380
<i>C'est à vous !</i>	383
Corrigé	383
<i>Le corrigé complet</i>	383
<i>Les explications</i>	386
<i>Idées d'amélioration</i>	395
 Cinquième partie – JavaScript et HTML 5	 397
33 Qu'est-ce que le HTML 5 ?	399
Rappel des faits	399
<i>Accessibilité et sémantique</i>	400
<i>Applications web et interactivité</i>	400
<i>Concurrer Flash (et Silverlight)</i>	400
Les API JavaScript	401
<i>Anciennes API désormais standardisées ou améliorées</i>	401
<i>Sélecteurs CSS : deux nouvelles méthodes</i>	401
<i>Timers : rien ne change, mais c'est standardisé</i>	401
<i>Les nouvelles API</i>	401
<i>Les nouvelles API que nous allons étudier</i>	404
En résumé	404
34 L'audio et la vidéo	405
L'audio	405
<i>Contrôles simples</i>	406
<i>Analyse de la lecture</i>	408
<i>Améliorations</i>	409
La vidéo	412
En résumé	413

35 L'élément Canvas	415
Premières manipulations	415
<i>Principe de fonctionnement</i>	416
<i>Le fond et les contours</i>	417
<i>Effacer</i>	418
Formes géométriques	418
<i>Les chemins simples</i>	418
<i>Les arcs</i>	419
<i>Utilisation de moveTo()</i>	420
<i>Les courbes de Bézier</i>	420
Images et textes	422
<i>Les images</i>	422
<i>Mise à l'échelle</i>	423
<i>Recadrage</i>	423
<i>Les patterns</i>	424
<i>Le texte</i>	425
Lignes et dégradés	426
<i>Les styles de lignes</i>	426
Opérations	429
<i>L'état graphique</i>	429
<i>Les translations</i>	430
<i>Les rotations</i>	430
Animations	431
<i>Une question de « framerate »</i>	432
<i>Un exemple concret</i>	432
En résumé	434
36 L'API File	435
Première utilisation	435
Les objets Blob et File	436
<i>L'objet Blob</i>	437
<i>L'objet File</i>	437
Lire les fichiers	438
Mise en pratique	440
Upload de fichiers avec l'objet XMLHttpRequest	444
En résumé	446
37 Le drag & drop	447
Aperçu de l'API	447
<i>Rendre un élément déplaçable</i>	447
<i>Initialiser un déplacement avec l'objet dataTransfer</i>	448
<i>Définir une zone de « drop »</i>	449
<i>Terminer un déplacement avec l'objet dataTransfer</i>	451

Mise en pratique	452
<i>Le code présente un bogue majeur</i>	456
En résumé	460

Sixième partie – Annexe 461

Aller plus loin 463

Récapitulatif express	463
<i>Ce qu'il vous reste à faire</i>	463
<i>Ce que vous ne devez pas faire</i>	464
<i>Le JavaScript intrusif</i>	464
<i>Ce qu'il faut retenir</i>	465
Étendre le JavaScript	465
<i>Les frameworks</i>	465
<i>Les bibliothèques</i>	466
Diverses applications du JavaScript	467
<i>Des extensions codées en JavaScript</i>	468
<i>Des logiciels développés en JavaScript</i>	468
<i>Des applications pour smartphones en JavaScript</i>	468
<i>Du JavaScript sur le serveur</i>	469
En résumé	469

Index 471

Première partie

Les bases du JavaScript

Comme tout langage de programmation, le JavaScript possède quelques particularités : sa syntaxe, son modèle d'objet, etc. En clair, tout ce qui permet de différencier un langage d'un autre. D'ailleurs, vous découvrirez rapidement que le JavaScript est un langage relativement spécial dans sa manière d'aborder les choses. Cette partie est indispensable pour tout débutant en programmation et même pour ceux qui connaissent déjà un langage de programmation car les différences avec les autres langages sont nombreuses.

1

Introduction au JavaScript

Avant d'entrer directement dans le vif du sujet, ce chapitre va vous apprendre ce qu'est le JavaScript, ce qu'il permet de faire, quand il peut ou doit être utilisé et comment il a considérablement évolué depuis sa création en 1995.

Nous aborderons aussi plusieurs notions de bases telles que les définitions exactes de certains termes.

Qu'est-ce que le JavaScript ?

Le JavaScript est un langage de programmation de scripts orienté objet. Dans cette description un peu barbare se trouvent plusieurs éléments que nous allons décortiquer.

Un langage de programmation

Un langage de programmation est un langage qui permet aux développeurs d'écrire du code source qui sera analysé par l'ordinateur.

Un développeur, ou un programmeur, est une personne qui développe des programmes. Ça peut être un professionnel (un ingénieur, un informaticien ou un analyste programmeur) ou bien un amateur.

Le code source est écrit par le développeur. C'est un ensemble d'actions, appelées « instructions », qui vont permettre de donner des ordres à l'ordinateur afin de faire fonctionner le programme. Le code source est quelque chose de caché, un peu comme un moteur dans une voiture : le moteur est caché, mais il est bien là, et c'est lui qui fait en sorte que la voiture puisse être propulsée. Dans le cas d'un programme, c'est pareil : c'est le code source qui régit le fonctionnement du programme.

En fonction du code source, l'ordinateur exécute différentes actions, comme ouvrir un menu, démarrer une application, effectuer une recherche, etc., soit tout ce que l'ordinateur est capable de faire. Il existe énormément de langages de programmation (un bon nombre d'entre eux sont listés sur la page suivante : http://fr.wikipedia.org/wiki/Cat%C3%A9gorie%3ALangage_de_programmation).

Programmer des scripts

Le JavaScript permet de programmer des scripts. Comme nous venons de le voir, un langage de programmation permet d'écrire du code source qui sera analysé par l'ordinateur. Il existe trois manières d'utiliser du code source.

- Langage compilé : le code source est donné à un programme appelé « compilateur » qui va lire le code source et le convertir dans un langage que l'ordinateur sera capable d'interpréter : c'est le langage binaire, fait de 0 et de 1. Les langages comme le C ou le C++ sont des langages dits compilés.
- Langage précompilé : ici, le code source est compilé partiellement, généralement dans un code plus simple à lire pour l'ordinateur, mais qui n'est pas encore du binaire. Ce code intermédiaire devra être lu par ce qu'on appelle une « machine virtuelle » qui exécutera ce code. Les langages comme le C# ou le Java sont dits précompilés.
- Langage interprété : dans ce cas, il n'y a pas de compilation. Le code source reste tel quel, et si l'on veut l'exécuter, on doit le fournir à un interpréteur qui se chargera de le lire et de réaliser les actions demandées. Éventuellement, pour obtenir de significatifs gains de performances, le code peut être compilé à la volée (http://fr.wikipedia.org/wiki/Compilation_%C3%A0_la_vol%C3%A9e) pendant son exécution. C'est aujourd'hui ce que font la plupart des interpréteurs JavaScript.

Les scripts sont majoritairement interprétés. Et quand on dit que le JavaScript est un langage de scripts, cela signifie qu'il s'agit d'un langage interprété ! Il est donc nécessaire d'utiliser un interpréteur pour faire fonctionner du code JavaScript, ce que vous faites fréquemment car il est inclus dans votre navigateur web !

En effet, chaque navigateur possède un interpréteur JavaScript propre :

- pour Internet Explorer, il s'agit de *Chakra* (l'interpréteur JavaScript des versions antérieures à IE 9 est *JScript*) ;
- pour Mozilla Firefox, l'interpréteur se nomme *SpiderMonkey* ;
- pour Google Chrome, il s'agit de *V8*.

Un langage orienté objet

Intéressons-nous maintenant à la notion « orienté objet ». Ce concept est assez compliqué à définir à ce stade, il sera approfondi par la suite, notamment dans la deuxième partie du livre. Sachez toutefois qu'un langage de programmation orienté objet est un langage qui contient des éléments, appelés « objets », lesquels possèdent des caractéristiques spécifiques et peuvent être utilisés de différentes manières. Le langage fournit des objets de base comme des images, des dates, des chaînes de caractères, etc., mais

il est également possible de créer soi-même des objets pour se faciliter la vie et obtenir un code source plus clair (donc plus facile à lire) et une manière de programmer beaucoup plus intuitive (et donc plus logique).

Il est probable que vous n'ayez rien compris à ce passage si vous n'avez jamais fait de programmation, mais ne vous en faites pas : vous comprendrez bien assez vite comment tout cela fonctionne.

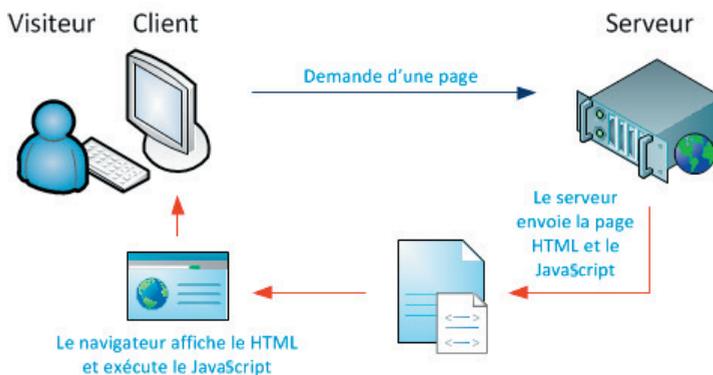
Le JavaScript, le langage de scripts

Le JavaScript est majoritairement utilisé sur Internet, conjointement avec les pages web HTML dans lesquelles il est inclus (ou dans un fichier externe). Le JavaScript permet de « dynamiser » une page HTML en ajoutant des interactions avec l'utilisateur, des animations, de l'aide à la navigation. Par exemple :

- afficher/masquer du texte ;
- faire défiler des images ;
- créer un diaporama avec un aperçu « en grand » des images ;
- créer des infobulles.

Le JavaScript est un langage dit *client-side*, c'est-à-dire que les scripts sont exécutés par le navigateur de l'internaute (le client). Cela diffère des langages de scripts dits *server-side* qui sont exécutés par le serveur web. C'est le cas des langages comme le PHP (<https://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql/>).

Cette distinction est importante car la finalité des scripts client-side et server-side n'est pas la même. Un script server-side va s'occuper de « créer » la page web qui sera envoyée au navigateur. Ce dernier va alors afficher la page, puis exécuter les scripts client-side tels que le JavaScript. Voici un schéma reprenant ce fonctionnement :



JavaScript est un langage dit client-side, c'est-à-dire interprété par le client (le navigateur)

Le JavaScript, pas que le Web

Si le langage JavaScript a été conçu pour être utilisé conjointement avec le HTML, il a depuis évolué vers d'autres destinées. En effet, il est régulièrement utilisé pour réaliser des extensions pour différents programmes : Chrome et Firefox possèdent tous deux un panel gigantesque d'extensions en partie codées en JavaScript.

Mais là où le JavaScript a su grandement évoluer ces dernières années, c'est dans la possibilité d'être exécuté sur n'importe quelle machine. Quelques projets permettent d'utiliser le JavaScript en dehors de votre navigateur, le plus connu est sans nul doute Node.js (<https://openclassrooms.com/informatique/cours/des-applications-ultra-rapides-avec-node-js>).

Le JavaScript peut aussi être utilisé pour réaliser des applications. L'interface de Firefox est notamment développée avec ce langage mais cela reste une implémentation bien particulière à la société Mozilla. Cependant, il vous est tout à fait possible aujourd'hui de créer une application en JavaScript grâce à Node.js et si vous souhaitez y ajouter une interface, d'autres projets venant se greffer à Node vous faciliteront la tâche, tel qu'Electron (<http://electron.atom.io/>) ou NW.js (<https://github.com/nwjs/nw.js>).

Petit historique du langage



Brendan Eich, le papa de JavaScript

En 1995, Brendan Eich travaillait chez Netscape Communication Corporation, la société qui éditait le célèbre navigateur Netscape Navigator, alors principal concurrent d'Internet Explorer.

Brendan développe le LiveScript, un langage de scripts qui s'inspire du langage Java, et qui est destiné à être installé sur les serveurs développés par Netscape. Netscape se met à développer une version client du LiveScript, qui sera renommée JavaScript en hommage au langage Java créé par la société Sun Microsystems. En effet, à cette époque, le langage Java était de plus en plus populaire, et appeler le LiveScript « JavaScript » était une manière de faire de la publicité à la fois au Java et au JavaScript lui-même. Mais attention, au final, ces deux langages sont radicalement différents ! N'allez pas confondre le Java et le JavaScript car ils n'ont clairement pas le même fonctionnement.

Le JavaScript sort en décembre 1995 et est embarqué dans le navigateur Netscape 2. Le langage est alors un succès, si bien que Microsoft développe une version semblable, appelée JScript, qu'il embarque dans Internet Explorer 3, en 1996.

Netscape décide d'envoyer sa version de JavaScript à l'ECMA International (*European Computer Manufacturers Association* à l'époque, aujourd'hui *European Association for Standardizing Information and Communication Systems*) pour que le langage soit standardisé, c'est-à-dire pour qu'une référence du langage soit créée et qu'il puisse ainsi être utilisé par d'autres personnes et embarqué dans d'autres logiciels. L'ECMA International standardise le langage sous le nom d'ECMAScript (<http://fr.wikipedia.org/wiki/ECMAScript>).

Depuis, les versions de l'ECMAScript ont évolué. La version la plus connue et mondialement utilisée est la version ECMAScript 5, parue en décembre 2009.

L'ECMAScript et ses dérivés

L'ECMAScript est la référence de base dont découlent des implémentations. On peut évidemment citer le JavaScript, qui est implémenté dans la plupart des navigateurs, mais aussi :

- JScript (<http://fr.wikipedia.org/wiki/JScript>), qui est l'implémentation embarquée dans Internet Explorer. C'est aussi le nom de l'ancien interpréteur d'Internet Explorer ;
- JScript.NET, qui est embarqué dans le framework .NET de Microsoft ;
- ActionScript (<http://fr.wikipedia.org/wiki/ActionScript>), qui est l'implémentation faite par Adobe au sein de Flash ;
- EX4 (<http://fr.wikipedia.org/wiki/E4X>), qui est l'implémentation de la gestion du XML d'ECMAScript au sein de SpiderMonkey, l'interpréteur JavaScript de Firefox.

La plupart de ces implémentations sont quelque peu tombées en désuétude hormis le JavaScript qui a su se frayer une place de choix.

Les versions du JavaScript

Les versions du JavaScript sont basées sur celles de l'ECMAScript (ES). Ainsi, il existe :

- ES 1 et ES 2, qui sont les prémices du langage JavaScript ;
- ES 3 (sorti en décembre 1999) ;
- ES 4, qui a été abandonné en raison de modifications trop importantes qui ne furent pas appréciées ;
- ES 5 (sorti en décembre 2009), la version la plus répandue et utilisée à ce jour ;
- ES 6 finalisé en décembre 2014 et dont l'implémentation avait déjà été commencée avant cette date au sein de plusieurs navigateurs.

Ce cours portera sur la version 5 de l'ECMAScript, la version 6 n'étant pas encore très bien supportée à l'heure où nous écrivons ces lignes.

Un logo inconnu

Il n'y a pas de logo officiel pour représenter le JavaScript. Cependant, le logo suivant est beaucoup utilisé par la communauté, surtout depuis sa présentation à la JSConf EU de 2011. Vous pourrez le trouver à cette adresse : <https://github.com/voodooatikigod/logo.js> sous différents formats. N'hésitez pas à en abuser en cas de besoin.



Ce logo non officiel est de plus en plus utilisé.

En résumé

- Le JavaScript est un langage de programmation interprété, c'est-à-dire qu'il a besoin d'un interpréteur pour pouvoir être exécuté.
- Le JavaScript est utilisé majoritairement au sein des pages web mais son utilisation en guise de serveur ou d'application commence à se répandre.
- Tout comme le HTML, le JavaScript est généralement exécuté par le navigateur de l'internaute : on parle d'un comportement client-side, par opposition au server-side lorsque le code est exécuté par le serveur.
- Le JavaScript est standardisé par l'ECMA International sous le nom d'ECMAScript, qui constitue la référence du langage.
- La dernière version standardisée du JavaScript est basée sur l'ECMAScript 5, sorti en 2009. Mais sa nouvelle version, ECMAScript 6, prend du terrain.



QCM

(<http://odyssey.sdlm.be/javascript/01/partie1/chapitre1/qcm.htm>)