

DIGIT BOOKS

# *jQuery*

*La bibliothèque qui simplifie l'interaction*

*Un site interactif en quelques lignes de JavaScript*



*Focus*

**De Didier Mouronval**

# *jQuery*

*La bibliothèque qui simplifie  
l'interaction*

---

Didier Mouronval

# *jQuery*

*La bibliothèque qui simplifie  
l'interaction*

*Focus*

**Digit Books**

**Éditeur de livres numériques**

**Brest**

*infos@digitbooks.fr*

*http://www.digitbooks.fr*

**DIGIT BOOKS**

© Digit Books, 2011

ISBN : 978-2-8150-0211-0

Prix : 15 €

Conception de la couverture : Yves Buraud

Illustration tirée de « Les cinq Les Cinq cents millions de la Bégum » de Jules Verne (1878) par Léon Benett

*<http://www.digitbooks.fr/catalogue/jquery-didier-mouronval.html>*

Les programmes figurant dans ce livre ont pour but d'illustrer les sujets traités. Il n'est donné aucune garantie quant à leur fonctionnement une fois compilés, assemblés ou interprétés dans le cadre d'une utilisation professionnelle ou commerciale.

Toute représentation ou reproduction, intégrale ou partielle, faite sans le consentement de l'auteur, de ses ayants droit, ou ayants cause, est illicite (loi du 11 mars 1957, alinéa 1er de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait une contrefaçon sanctionnée par les articles 425 et suivants du Code pénal. La loi du 11 mars 1957 autorise uniquement, aux termes des alinéas 2 et 3 de l'article 41, les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective d'une part et, d'autre part, les analyses et les courtes citations dans un but d'exemple et d'illustration.

---

# Table des matières

---

<b>Préface</b>	IV
Pourquoi ce livre	V
Prérequis et public	VI
Organisation de l'ouvrage	VI
Les exemples	VII
Les conventions utilisées	VIII
À propos des auteurs	IX
Remerciements	IX
<b>1 Présentation de jQuery</b>	1
L'essor de JavaScript	1
Les frameworks JavaScript	4
L'utilité des frameworks	4
Utilisation des frameworks JavaScript	6
Pourquoi utiliser jQuery ?	9
Fonctionnement de jQuery	12
Utiliser jQuery	14
Intégrer jQuery dans la page	14
Intégrer ses propres scripts	16
<b>2 jQuery et le DOM</b>	18
Rappels sur le DOM	18
L'arbre DOM	25
jQuery et le DOM	28
Parcourir le document	36
Retour sur filter() et find()	43
Ajouter les résultats précédents	44

---

Parcourir une sélection	45
Récupérer la sélection précédente	49
Que suis-je ?	50
La méthode map()	52
Retirer des éléments	54
<b>3 Les événements</b>	<b>55</b>
Généralités sur les événements	55
Les événements avec jQuery	63
Des événements prédictifs	72
Supprimer des événements	79
Déclencher un événement	85
L'objet Event	89
<b>4 Modification du DOM</b>	<b>92</b>
Créer un nouvel élément HTML	94
Remplacer le contenu d'une balise	100
Ajouter du contenu au document	111
Supprimer des éléments	122
Copier, déplacer et remplacer des éléments	128
Propriétés et attributs	134
L'objet Data	147
Gestion des noms de classe	148
<b>5 Cas concret : navigation par onglets</b>	<b>152</b>
Le HTML	156
Le CSS	156
Le script	157
<b>6 Les effets jQuery</b>	<b>163</b>
Les effets show() et hide()	167
Les effets fade	170
Les effets slide	172
La méthode animate()	172
Les effets stop() et delay()	174
Compléments sur les effets	175
<b>7 jQuery et AJAX</b>	<b>181</b>
Qu'est-ce qu'une requête asynchrone ?	184
AJAX vu par jQuery	185
Les requêtes inter-domaines	196
Sérialiser des données à envoyer	199
Les événements AJAX	201
L'objet Deferred()	202

<b>8 Un exemple récapitulatif</b>	205
Information sur les formats de données	206
Vérifier les valeurs des champs	210
Liens de navigation entre onglets	216
Traitement sur le serveur	217
Traitement de la réponse	218
En conclusion	219
<b>9 Addendum : la version 1.7 de jQuery</b>	221
La gestion des événements	221
Les méthodes AJAX et les traitements différés	225
Les méthodes AJAX	225
Améliorations de l'objet Deferred	225
Création de l'objet Callbacks	226
<b>Index</b>	229

---

# Préface

---

S'il est un langage de programmation particulièrement atypique, c'est bien JavaScript. Il est relativement simple à appréhender, ce qui signifie, entre autres, qu'il est très souvent utilisé (et jugé) par des personnes qui ne le connaissent pas. Pourtant, lorsque l'on va dans ses recoins, on arrive toujours à être impressionné par ses capacités. Toujours est-il qu'on lui reproche beaucoup de choses : son typage faible, sa prétention de langage orienté objet, mais sans classes, j'en passe. Pour autant, c'est malgré tout un langage qui est devenu incontournable dans le développement web (pour preuve, il suffit d'essayer de naviguer sur les sites les plus populaires, notamment les réseaux sociaux, en désactivant JavaScript dans son navigateur) et développer aujourd'hui un site sans JavaScript est désormais un non sens. Quand on connaît un peu JavaScript, on sort des idées reçues sur les lacunes du langage, mais on prend en considération ses vraies faiblesses (justifiées par son historique), notamment le manque de compatibilité entre navigateurs. Dans un premier temps, on peut opter pour se créer des bibliothèques utilitaires personnelles pour y parer. Puis, on se rend compte que d'autres ont déjà effectué ce travail ; en mieux et en plus complet. On comprend alors que « comme tout le monde », notre intérêt est d'utiliser le travail commun mis à disposition par un ensemble de développeurs compé-

---

tents et on utilise un framework. De tous ceux disponibles actuellement, celui qui sort manifestement du lot est jQuery.

Cependant, chaque framework est créé dans un certain « esprit » ; il possède sa philosophie de création et d'utilisation. Il est donc nécessaire, au-delà d'apprendre la syntaxe qui lui est propre, de prendre en considération la manière dont il est pensé.

C'est pour vous accompagner dans la découverte de ces deux aspects importants de jQuery que ce livre a été réfléchi. S'il s'était agi uniquement de syntaxe, un simple tableau recensant les méthodes disponibles aurait suffi (et des tableaux de ce type existent dans ce livre), mais l'objectif de cet ouvrage est aussi de vous montrer comment ces méthodes s'intègrent dans un principe global de fonctionnement autour duquel s'articule l'ensemble de jQuery.

## Pourquoi ce livre

Parmi toutes les bibliothèques disponibles, jQuery est de loin la plus répandue. Simple à mettre en place et à utiliser, mais surtout basée sur une « philosophie » bien adaptée aux besoins des développeurs, nombre d'entre eux l'ont rapidement adoptée pour simplifier leurs développements JavaScript. Une forte communauté s'est rapidement mise en place autour de ce projet donnant ainsi naissance à de très nombreuses extensions (plugins) qui font qu'actuellement la quasi totalité des fonctionnalités que l'on retrouve habituellement sur un site (galerie d'images, validation de formulaire avant envoi, effets de survols, etc.) sont accessibles à tous avec des scripts jQuery. Pour autant, cela ne veut pas dire que tout le monde connaît l'étendue des possibilités offertes et encore trop d'utilisateurs compliquent leurs codes par manque de connaissances.

C'est pour parer à cela que je me suis lancé dans la rédaction de ce livre.

# Prérequis et public

Cet ouvrage s'adresse aux personnes ayant déjà de bonnes bases de (X)HTML et de CSS. JavaScript, et à plus forte raison jQuery, se base sur la structure (X)HTML d'un document pour y apporter des modifications. Ces modifications portent souvent sur le style d'affichage de certains éléments. Il est donc important d'être capable de créer des pages au contenu et aux styles conformes aux standards avant de se lancer dans le dynamisme de la page. Idéalement, la connaissance des bases de JavaScript permettra de mieux comprendre les mécanismes mis en œuvre par jQuery et de rendre son utilisation plus efficace.

## Organisation de l'ouvrage

- × Le premier chapitre propose une présentation générale de jQuery. Après avoir rappelé l'importance actuelle de JavaScript dans le développement web et indiquer pourquoi utiliser une librairie telle que jQuery, nous verrons comment celle-ci fonctionne et comment l'intégrer dans une page.
- × Le second chapitre, après avoir rappelé ce qu'est le *Document Object Model* (DOM), montre comment jQuery interagit avec celui-ci et en particulier comment l'utiliser pour récupérer facilement des éléments du DOM et manipuler une collection d'éléments.
- × Le troisième chapitre traite de la gestion des événements avec jQuery. Les événements sont un aspect essentiel de la programmation JavaScript, mais certaines incompatibilités entre navigateurs rendent leur utilisation parfois complexe. jQuery offre une interface riche, uniformisée et simple d'utilisation.
- × Dans le quatrième chapitre, nous allons voir comment modifier le DOM à l'aide de jQuery. La modification du DOM permet de changer le contenu affiché dans la page, il faut donc être capable de générer du contenu (que ce soit du texte ou des portions de HTML) afin de l'utiliser pour remplacer une partie de la page ou de l'ajouter. Il faut aussi être capable de modifier des propriétés (en particulier des noms de classes ou des styles CSS) d'éléments existants.

- × Le cinquième chapitre utilise les notions déjà évoquées pour mettre en place un exemple concret : une navigation par onglets qui permet de voir comment organiser le code initial (HTML et CSS) en prenant en considération les modifications que l'on fera ensuite à l'aide de jQuery.
- × Le chapitre six aborde la question des effets. Ceux-ci sont très appréciés par les utilisateurs et apportent une ergonomie agréable sur un site. Mais JavaScript ne dispose pas à proprement parler de fonctions ou méthodes spécifiques pour les effets, jQuery propose donc nativement d'une collection d'effets préprogrammés parmi les plus usuels.
- × Le septième chapitre présente la gestion d'AJAX avec jQuery. AJAX est devenu un élément important de la programmation JavaScript et jQuery met à disposition un ensemble très complet de méthodes et de propriétés facilitant son utilisation, que ce soit pour les cas les plus simples ou les plus spécifiques.
- × Enfin, le huitième chapitre consiste en un exemple récapitulatif dans lequel nous allons modifier la navigation par onglets du chapitre cinq en le transformant en formulaire d'identification et d'inscription. Avec l'aide d'AJAX, nous récupérerons ou nous enregistrons, côté serveur, les informations liées à un utilisateur.
- × Le dernier chapitre présente, dans les grandes lignes, ce qu'apporte la version 1.7 de jQuery.

## Les exemples

Les exemples présents dans ce livre peuvent être, pour le plus grand nombre, retrouvés dans les fichiers sources joints. Ils peuvent être lancés directement depuis l'ordinateur et ne demandent aucune installation complémentaire (à l'exception des exemples concernant AJAX).

Ils nécessitent toutefois une connexion à Internet pour pouvoir charger jQuery depuis le site officiel.

Vous pouvez les télécharger depuis : <http://www.digitbooks.fr/catalogue/jquery-didier-mouronval.html>.

Ils ont, pour la plupart, été conçus pour montrer le fonctionnement d'un aspect spécifique de jQuery de façon la plus explicite possible. À l'exception des exemples récapitulatifs, la considération esthétique n'a pas été prise en compte.

## Les conventions utilisées

Voici les conventions typographiques de cet ouvrage :

### *Italique*

Met en exergue les termes nouveaux ou la signification des acronymes.

### *Noms de chemins/fichiers*

Cette couleur distingue les noms de fichiers ou les chemins.

### Menus

Désigne les libellés des menus des interfaces graphiques.

### Police à chasse fixe

Met en valeur les éléments de code dans le texte.

### Les encadrés

Deux types d'encadrés émaillent le texte.

---

Certains sont des apartés et donnent une information supplémentaire.

---

D'autres sont des astuces, des conseils. Ils sont repérables par un fond vert.

-----

Enfin, des notes éveillent votre attention sur des points précis.

-----

## À propos des auteurs

Passionné depuis toujours par l'informatique et la programmation, ce n'est que tardivement que Didier Mouronval décide d'en faire son métier. Il se spécialise dans les technologies de l'Internet et du multimédia et devient développeur et intégrateur web.

Rapidement séduit par le langage JavaScript, il s'investit au sein de la communauté de développeurs francophones de [www.developpez.com](http://www.developpez.com) afin de contribuer à améliorer l'image de ce langage. Il deviendra par la suite responsable des rubriques JavaScript et AJAX, puis des rubriques Développement web de ce site.

Il a enfin participé à la réalisation de différents ouvrages sur JavaScript et jQuery et a été formateur JavaScript et PHP pour le compte de video2brain.

## Remerciements

Tout d'abord, je tiens à remercier mon entourage et mes proches pour leurs encouragements et leur patience au cours de l'écriture de ce livre.

Mais mes principaux remerciements vont vers Dominique, qui m'a fait confiance (et il lui en a fallu aussi de la patience !) et m'a permis de réaliser ce projet que j'ai eu beaucoup de plaisir à concrétiser ; qu'elle sache que je serai toujours disponible pour revivre cette belle aventure.

Enfin, j'ai une pensée particulière pour tous mes camarades de la communauté [developpez.com](http://developpez.com) et en particulier (les « oubliés » ne m'en tiendront, je l'espère, pas rigueur : une liste exhaustive serait beaucoup trop longue) Daniel, Steven, Xavier, Pierre, Caroline et Eric.

---

# Présentation de jQuery

---

## L'essor de JavaScript

JavaScript est apparu au milieu des années 90. Il a d'abord été cantonné à des tâches assez basiques et intrusives pour l'utilisateur – ce qui a largement contribué à sa mauvaise réputation, qui lui colle encore parfois à la peau.

Pourtant, JavaScript a rapidement évolué de façon à apporter aux développeurs des interfaces toujours plus riches permettant de modifier le contenu sur le contenu de la page après que celle-ci ait été créée dans le navigateur. Sans aller jusqu'à dire que ces possibilités ont été ignorées, elles ont été en tout cas insuffisamment utilisées.

Puis est arrivée la vague AJAX et la mode du « Web 2.0 ». Termes qui ont été particulièrement galvaudés car peu ou mal définis, mais qui, au moins, ont permis de redéfinir le contexte d'utilisation de JavaScript et d'accélérer l'enrichissement des interfaces.

---

S'il fallait définir ces termes, je dirais qu'AJAX est la capacité d'un navigateur à communiquer en temps réel (et sans rechargement) avec un serveur afin de lui transmettre des données et d'en récupérer un résultat que l'on peut exploiter en JavaScript. On pressent bien que le X d'AJAX, pour XML est un peu hors sujet. De son côté, le *Web 2.0* est la capacité donnée à une page web d'interagir dynamiquement avec l'utilisateur en fonction de ses actions (avec ou sans AJAX) comme le ferait une application de bureau.

JavaScript peut être utilisé dans différents environnements. Il est ainsi divisé en deux parties distinctes : le *core* JavaScript d'un côté, l'*API* fournie par son environnement de l'autre. Le *core* JavaScript fournit les différents modèles de données et d'objets : le prototype. Mais comme souvent concernant les langages de scripts, sa véritable force réside dans la richesse des interfaces mises à sa disposition dans le cadre de l'application au sein de laquelle il évolue. À ce titre, force est de constater que le DOM (*Document Object Model* ou *Représentation du Document sous forme d'Objets*), l'interface mise à disposition de JavaScript par les navigateurs, est particulièrement riche et attrayante. Qui plus est, les différents navigateurs se livrent une lutte acharnée pour enrichir leurs moteurs JavaScript afin de les rendre plus performants et plus rapides. D'où la multiplication des versions de chacun d'entre eux, alors que ce domaine était resté relativement inerte pendant longtemps.

Surtout, l'apparition de cette notion de « Web 2.0 » a fait découvrir des techniques de mise à disposition de contenu évolutif et adapté à la demande du visiteur. À tel point qu'aujourd'hui, naviguer en désactivant JavaScript (comme cela était fréquent il n'y a encore pas si longtemps) devient presque handicapant. Essayez de désactiver JavaScript et d'aller sur les différents réseaux sociaux ou sur une page iGoogle pour vous en persuader. JavaScript est donc désormais un langage suffisamment mature pour pouvoir être utilisé efficacement sur un site afin d'y améliorer l'ergonomie de façon significative. D'autant que les meilleures performances des connexions internet permettent l'ajout de scripts, même conséquents, sans altération de la rapidité d'affichage ou presque.

---

# jQuery et le DOM

---

## Rappels sur le DOM

Nous avons déjà abordé la notion de DOM (*Document Object Model*) au chapitre précédent. Nous avons alors vu qu'il s'agit de la représentation du document sous forme d'objets JavaScript, nous allons compléter cette définition car il est important d'avoir une vision claire de ce qu'est le DOM pour pouvoir bien utiliser JavaScript et jQuery.

Comme nous l'avons évoqué, le DOM est construit parallèlement à l'interprétation du code HTML de la page. A chaque balise rencontrée, JavaScript associe un objet de type `HTMLElement`, dont les propriétés correspondent aux attributs HTML autorisés pour cette balise. L'association est faite de façon à ce que chaque modification d'une propriété a un impact sur la valeur de l'attribut associé et vice versa. Ceci a pour effet une confusion entre propriété JavaScript d'un objet du DOM et structure HTML. Pour bien comprendre la différence, nous

---

allons utiliser l'exemple suivant et suivre son comportement dans *Firebug*, l'outil indispensable de développement web pour Firefox :

### jquery-2-1.html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="iso-8859-1">
    <title>DOM et HTML</title>
    <script type="text/javascript">
      function modifDOM(){
        document.getElementById('test').value =
          'Ajouté par le DOM';
        alert(document.getElementById('test').
          getAttribute('value'));
      }
      function modifHTML(){
        document.getElementById('test').setAttribute
          ('value', 'Modifié par HTML');
        alert(document.getElementById('test').value);
      }
    </script>
  </head>
  <body>
    <div><input id="test" /></div>
    <div><button onClick="modifDOM();">Modifier le DOM</button></div>
    <div><button onClick="modifHTML();">Modifier le HTML</button></div>
  </body>
</html>
```

Dans cet exemple, nous avons juste créé un champ texte (balise `input`). Vous noterez que les attributs `type` et `value` n'ont pas été renseignés, ce sont donc les valeurs par défaut qui seront affectées (`type="text"` et `value=""`). Nous ajoutons aussi deux boutons, l'un pour modifier la valeur d'une propriété et afficher la valeur de l'attribut associé (fonction `modifDOM()`), l'autre pour le contraire (`modifHTML()`).

Les résultats sont illustrés à la figure 2-1.

---

# Les événements

---

JavaScript est essentiellement un langage événementiel, c'est-à-dire qu'il intercepte les actions de l'utilisateur et permet de leur associer des actions à effectuer (afficher des informations au survol ou au clic d'un élément, vérification du format de saisie d'un champ de formulaire, etc.) afin de rendre la page plus dynamique et attractive. L'apport de jQuery concernant les événements porte bien entendu sur la simplification de leur utilisation, que ce soit pour mettre en place des gestionnaires d'événements, les supprimer ou accéder aux informations liées à l'événement en cours, mais surtout, jQuery met en place la capacité d'affecter des comportements à des éléments qui ne sont pas encore présents dans le document.

## Généralités sur les événements

La grande majorité des scripts présents sur une page a pour but de permettre d'interagir avec l'utilisateur. Cette interaction se fait essentiellement à travers de la gestion des événements. Les événements

---

---

## Modification du DOM

---

L'intérêt majeur de JavaScript, facilité par l'utilisation de JQuery, est de pouvoir modifier l'apparence de la page pendant qu'elle est affichée. Traditionnellement, une page web est générée soit de façon statique par un fichier HTML (chaque appel de ce fichier affichera le même résultat) soit de façon dynamique par un langage serveur, par exemple PHP, qui affichera un résultat en fonction de paramètres reçus et d'un traitement spécifique sur le serveur. Mais une fois la page créée et envoyée au navigateur, elle est considérée comme fixe, c'est-à-dire que pour modifier l'affichage, il faut changer de page.

JavaScript (et donc jQuery) donne la possibilité, en fonction de certaines actions de l'utilisateur ou de certains paramètres prédéfinis, de changer l'affichage sans recharger la page. Il sera même possible avec AJAX d'échanger des informations avec le serveur (envoi de données à traiter, voire à enregistrer et / ou récupération d'informations permettant d'ajuster l'affichage).

Il est cependant important de bien comprendre ce mécanisme de création pour garder à l'esprit les limites de JavaScript.

---

C'est bien le serveur web qui stocke les fichiers HTML et les envoie au navigateur. Si vous utilisez un langage serveur, par exemple PHP, c'est encore le serveur qui stocke le fichier PHP qui va générer une page HTML et l'envoyer au navigateur. Dans ce cas, tous les traitements (vérification des données reçues, connexion à une base de données et récupération d'informations spécifiques, ...) sont faits par le serveur et invisibles par l'utilisateur. Par exemple, il est impossible de voir le code source d'un script PHP, tout ce que vous pouvez obtenir est le code source du résultat renvoyé par un script PHP. En particulier, cela signifie qu'une fois affichée, il n'existe plus aucun lien avec le fichier (statique en HTML ou dynamique en PHP) stocké sur le serveur et la page affichée. Sinon, cela voudrait dire que n'importe quel ordinateur affichant une page Web pourrait avoir un contrôle sur le serveur qui héberge la page affichée. Ce serait bien évidemment une faille de sécurité importante dont tous les pirates du Web pourraient rêver.

Surtout, cela implique que toutes les modifications que vous pourrez faire sur une page ne seront qu'éphémères et limitées au temps d'affichage de la page. Il s'agit donc principalement de choix ergonomiques créés pour être utiles à l'utilisateur afin de rendre sa navigation plus agréable et donc d'améliorer sa satisfaction sur votre site.

Bien entendu, certains feront remarquer que sur certaines pages, des informations liées à l'interaction avec l'utilisateur sont enregistrées. Cela est rendu possible grâce à AJAX (que nous verrons ultérieurement), mais permet uniquement d'ouvrir une communication avec un serveur et d'en recevoir une réponse. En tout état de cause, JavaScript (et donc jQuery) reste cantonné à l'environnement de la page Web et n'a accès ni au système de fichiers de l'ordinateur (cela donnerait trop de « pouvoir » au concepteur du script sur l'environnement de celui qui l'exécute) ni au serveur envoyant le document (cela donnerait trop de « pouvoir » à l'utilisateur sur les données d'un site). Même si les fonctionnalités apportées par le prochain HTML5 vont permettre d'aller bien au-delà des cookies actuels pour conserver des informations sur le poste client, cela restera de façon limitée.

## Cas concret : navigation par onglets

---

Avant d'aller plus loin dans l'étude de jQuery, nous allons mettre en place un exemple concret que nous allons faire évoluer dans les prochains chapitres.

La page d'exemple (il s'agit en fait d'un extrait de page, qui serait supposé être intégré à une page plus complète) montre la mise en place avec jQuery d'un système de navigation par onglets.

Le but est simple : plutôt que d'afficher toutes les informations sur la page, ce qui aurait un aspect un peu austère et pourrait rebuter certains visiteurs, nous choisissons de n'en afficher qu'une partie et de mettre des pseudos liens de navigation pour permettre de visualiser les autres.

La page se compose donc d'une barre de liens (ou d'onglets) et d'une partie contenu. Notez que l'ensemble du contenu est en fait présent sur

---

la page, c'est uniquement le clic sur un onglet qui déterminera quelles parties seront réellement affichées ou non.

D'autre part, bien qu'il soit de nos jours fantaisiste de considérer qu'il existe une part significative de visiteurs qui choisissent intentionnellement de désactiver JavaScript, des impératifs d'accessibilité voire de référencement peuvent imposer de rendre disponible l'ensemble du contenu utile si c'était le cas. Nous avons donc opté pour montrer un exemple de la pratique d'amélioration progressive, c'est-à-dire que si JavaScript est désactivé, la barre d'onglets (qui n'est pas du contenu utile) sera masquée et tout le contenu sera affiché. C'est jQuery qui se chargera de rétablir l'affichage « normal ». Pour être réellement complet, il serait bon aussi de gérer la position en `relative` et le décalage vers le haut (`top : -6px ;`) de la balise `#contenu` au chargement de la page pour éviter le risque de masquer le bas de la partie située au-dessus.

Enfin, nous avons affecté des styles CSS (en se fixant comme contrainte de ne pas utiliser d'image) pour rendre l'ensemble un peu moderne. Il n'est pas pour autant question de parler ici de design (les web-designers ne s'en remettraient pas !) mais juste d'un début de mise en page minimaliste. Ici aussi, la volonté a été d'utiliser les styles CSS selon une optique d'amélioration progressive : tout est disponible pour qu'un utilisateur ayant un navigateur à jour puisse profiter de l'ergonomie et du visuel optimum, si tel n'est pas le cas, alors certains styles ne pourront pas être rendus mais la page continuera d'être fonctionnelle.

Notez que le modèle de boîte d'Internet Explorer jusqu'à la version 7 génère un décalage entre les onglets et le contenu. Le but ici n'étant pas de détailler le code CSS, nous ne réglerons pas ce bogue (il faudrait, pour ces navigateurs, augmenter le décalage vers le haut de 15 pixels supplémentaires), d'autant qu'Internet Explorer 6 et 7 sont en voie de disparition.

Le code de la page est le suivant (le contenu en lui-même de chaque onglet a été supprimé, il s'agit de fragments de Lorem Ipsum, du texte généré automatiquement pour remplir des maquettes) :

---

## Les effets jQuery

---

Contrairement à d'autres bibliothèques, jQuery n'est pas axée particulièrement sur les effets et animations. La bibliothèque en intègre trois grandes familles permettant d'afficher ou cacher des éléments, ainsi qu'une plus générique destinée à faire varier des valeurs de propriétés CSS.

Ces effets sont amplement suffisants dans la plupart des cas. Ils sont tous basés sur le principe d'affichage / apparition d'éléments ou au contraire leur masquage / disparition. Si vous aviez besoin d'utiliser d'autres effets, je vous invite à vous renseigner sur l'extension jQuery UI (pour *user interface*) qui intègre à jQuery un large panel de fonctionnalités utiles. Sinon, en fouillant dans la collection de plugin (cherchez de préférence dans les plugins « officiels », présentés sur le site de jQuery) vous trouverez certainement votre bonheur.

L'utilisation des effets est très simple avec jQuery. Ils se mettent en place à l'aide d'une unique méthode, dont les principaux paramètres sont communs.

---

Un effet correspond à une animation. Il est donc possible de préciser la durée de cette animation grâce à un paramètre *duration*. Ce paramètre peut être soit un entier, qui correspondra à la durée en millisecondes, soit l'une des valeurs prédéfinies *slow* ou *fast*.

Une animation correspond à la modification progressive de certaines propriétés d'affichage de l'élément. Lorsque l'on crée soi-même des animations JavaScript, on utilise le plus souvent des *timers* (`setInterval` ou `setTimeout`) et à chaque appel de la fonction de rappel, on modifie des propriétés CSS d'une valeur fixe. L'effet a donc une progression linéaire. Le paramètre *easing* permet (souvent grâce à des fonctions trigonométriques) de ne pas avoir cette linéarité pour rendre l'animation plus souple visuellement. Il est possible par exemple de donner un effet d'accélération (on augmente légèrement à chaque itération la valeur de la variation), d'accélération initiale et de décélération finale, de rebonds, etc. Seules deux valeurs sont disponibles de base : *easing* (valeur par défaut) et *linear*, mais jQuery UI ainsi que plusieurs plugin permettent d'en avoir plus à disposition.

Enfin, un troisième paramètre commun est la possibilité de préciser une fonction de rappel. Si vous appelez un effet et que vous faites suivre d'autres instructions après cet appel, ces instructions seront exécutées en même temps que l'effet. La fonction de rappel permet de définir les instructions à exécuter à la fin de l'animation.

Dans la suite de cette partie, nous allons ajouter différents effets à l'exemple du chapitre précédent, dont les styles ont été légèrement modifiés. Voici le code de la page que nous allons enrichir (ici aussi, le remplissage du contenu a été supprimé), notez l'ajout d'un champ de type `select` qui permettra de choisir l'effet que l'on veut utiliser.

#### [jquery-6-1.html](#)

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8" />
  <title>Onglets</title>
  <style type="text/css">
    #onglets{
```

---

# jQuery et AJAX

---

On présente souvent jQuery (et d'autres bibliothèques similaires) comme un framework AJAX. C'est une erreur, les chapitres précédents sont là pour démontrer !

Cette affirmation tient surtout à la montagne que se font certains de la technologie AJAX, la considérant presque comme un monstre sacré accessible seulement pour une poignée d'initiés. C'est bien évidemment faux et AJAX, une fois que l'on en a compris le principe, est relativement simple à manier.

Nous allons donc voir les différentes solutions proposées par jQuery pour faciliter toutes les opérations liées à AJAX. En effet, jQuery vous permet bien sûr de simplifier la syntaxe de vos appels, mais offre aussi la possibilité d'utiliser des raccourcis de code très efficaces pour les opérations les plus courantes et gère les cas plus complexes grâce à des possibilités de paramétrage très poussées.

Tous les termes inclus dans l'appellation AJAX (*Asynchronous JavaScript And XML*) ne sont là que pour rendre l'acronyme imposant, à tel point

---

qu'un groupe de développeurs, constatant qu'AJAX n'avait pas besoin de serveur (en tout cas de langage serveur) ni de XML, ont créé par dérision un nouvel acronyme pour des requêtes simples : AHAH (*Asynchronous HTTP And HTML*). Vous noterez l'aspect volontairement moqueur de cet acronyme. Mais dans un sens, ils ont raison. En effet, il faut voir AJAX comme un équivalent aux liens hypertexte. Lorsque vous cliquez sur un lien hypertexte (balise <a>), le navigateur récupère une adresse (la valeur de l'attribut href), envoie une requête au serveur sur cette adresse, le serveur quant à lui crée si besoin une page (si l'adresse demandée correspond à un script serveur du type PHP), sinon il récupère le contenu du fichier correspondant et renvoie ce résultat au navigateur qui affiche la nouvelle page. Une requête AJAX, c'est exactement la même chose, mais dans le cadre de la partie JavaScript de la page.

Lorsqu'un événement prédéfini est lancé (et il s'agit souvent d'un clic), JavaScript récupère une adresse à interroger, ouvre une connexion avec le serveur et lui transmet la requête. Là encore, le serveur interprète le script appelé si besoin, sinon il lit le fichier demandé et renvoie la réponse à l'interpréteur JavaScript du navigateur. Bien sûr, ici, il ne s'agit plus d'afficher une nouvelle page, donc le résultat renvoyé par le serveur devra être adapté à un traitement par JavaScript qui permettra d'interpréter la réponse et si besoin de modifier l'affichage.

À titre d'exemple, une requête basique en JavaScript s'écrirait :

```
var xhr = new XMLHttpRequest();
var parametre = '?nom=' + document.getElementById('champNom').value;
    parametre+= '&prenom=' + document.getElementById
                                ('champPrenom').value;
xhr.open('GET', 'ajax.php' + parametre);
xhr.onreadystatechange = function(){
    if(xhr.readyState == 4 && xhr.status == 200){
        traitementAjax(xhr.responseText);
    }
}
xhr.send(null);
```

La première ligne crée une nouvelle instance de l'objet JavaScript XMLHttpRequest comme on le ferait avec n'importe quel objet.

---

## Un exemple récapitulatif

---

Dans cette dernière partie, nous allons reprendre notre exemple du chapitre 5 et le compléter en y ajoutant de nouvelles fonctionnalités.

Cet exemple comportant des traitements AJAX appelant des scripts PHP, vous devrez, contrairement aux autres exemples de ce livre, disposer d'un serveur pour le tester. Un serveur local de type *WAMP* (pour *Windows, Apache, MySQL, PHP*) sera amplement suffisant. Une version en ligne est toutefois également disponible à l'adresse : <http://dmouronval.developpez.com/digitbooks/jquery/>.

Notez que le but de ces exemples est de montrer le fonctionnement de jQuery, de ce fait, la partie serveur est limitée au strict minimum : le code PHP se contente de renvoyer des données prédéfinies en fonction des paramètres reçus. Il ne fait pas appel à une base de données et n'effectue que des vérifications minimalistes des entrées utilisateur. Les scripts PHP ne sont donc là que pour montrer des exemples de ce qu'est sensé renvoyer comme résultat un traitement côté serveur dans une requête AJAX.

---

L'exemple du chapitre 5 a donc été modifié en remplaçant le contenu des deux premiers onglets par des formulaires du type de ceux que l'on peut trouver dans un formulaire d'identification / inscription à un site.

Concernant la navigation des onglets, seul un effet a été conservé (celui de glissement latéral).

Le premier onglet est constitué d'un onglet de type identification. Si l'on est déjà inscrit, il suffit d'y entrer son identifiant et son mot de passe. Si ceux-ci sont corrects, alors un message de bienvenue apparaît, ainsi qu'une option de déconnexion. Si l'on n'est pas encore inscrit, un lien permet d'accéder directement à la partie inscription (en complément du clic sur les onglets).

Le deuxième onglet est donc un formulaire d'inscription. Les informations demandées sont succinctes : un identifiant, un mot de passe, le nom et le prénom, ainsi qu'une adresse mail valide.

Une fois tous les champs remplis, on envoie les informations au serveur qui renvoie une portion de page affichant ces informations.

Le troisième onglet, qui correspondrait à une partie informative (mentions légales sur la conservation des données par exemple), est resté inchangé avec du texte de type « *Lorem ipsum* ».

Le code HTML et CSS correspondant peut être consulté dans le fichier [jquery-8-1.html](#) joint.

## Information sur les formats de données

En règle générale, dans un formulaire, il est essentiel de bien définir le format des valeurs que l'on attend pour chaque champ où cela est possible. Il est donc important, si l'on fixe des règles, de les faire connaître.

L'un des soucis majeurs en termes de sécurité sur un site internet est de filtrer les données reçues de l'utilisateur. En effet, si une personne malveillante arrive sur votre formulaire d'inscription, elle se doutera

---

# Addendum : la version 1.7 de jQuery

---

La version 1.7 de jQuery, sortie après la rédaction de ce livre, apporte certains ajouts et modifications majeurs. Cet addendum vous en donne les lignes principales.

## La gestion des événements

Dans la version 1.7, jQuery simplifie et uniformise la gestion des événements.

Nous avons vu précédemment qu'il y avait trois façons distinctes d'ajouter un gestionnaire d'événements :

- × La méthode `bind()` ou ses équivalents raccourcis (`click()`, `mouseover()`, `focus()`, ...).
-

---

# Index

---

## Symboles

`$()`, fonction 28

`$` (sélecteur) 63

## A

Accessibilité 66

`add()` 227

`addClass()` 149

Afficher un élément 167

`after()` 111

AHAH 182

`ajax()` 189

appel 190

paramètres 191

Ajax

et jQuery 1.7 225

AJAX 181–204

événements 201

présentation 2

scripts PHP 205

Ajouter du contenu

au document 111

dans un élément 116

`always()` 203

Amélioration progressive 66

`andSelf()` 45

`animate()` 172

options 173

Animations 164, 172

`append()` 116

`appendTo()` 117

Arbre DOM 25

Architecture de l'information  
pour le Web 235, 237

`attr()` 136

vs. `prop()` 136

Attributs 24, 134

affecter 136

vs. propriété 23

**B**

Beucart Philippe 91  
before() 111  
Bibliothèques 3  
    présentation 5  
bind() 63

**C**

callback 223  
Callbacks 46, 185  
    désactiver 227  
    jQuery 1.7 226  
    map() 52  
CDN 15  
Champs  
    formulaire 109, 136, 207  
    texte 19  
clone() 128  
Content Delivery Network 15  
Copier, déplacer et remplacer  
    des éléments 128  
Créer un nouvel élément HTML 94  
cross browser 4  
css() 140  
CSS 29, 139  
    class 148  
    dimensions 142  
    navigation par onglets 153  
    pseudo-éléments 30  
    sélecteurs 29  
CSS3 29

**D**

data() 147  
Data 147  
dataType 186  
Deferred() 202  
    jQuery 1.7 225  
delay() 175  
delegate() 76, 209  
detach() 123  
die() 84, 222  
disable() 227  
display 167  
Dojo 5  
DOM 2, 18–54  
    arbre 25  
    modifier 92–151  
    parcourir 26  
    méthodes 39  
    récupérer les éléments 28  
DOM Inspector 26  
done() 202  
DTD 134  
duree 164

**E**

each() 47  
easing 164  
Effets 163–180  
    fondu enchaîné 177  
    système de queue 169

## Éléments

- afficher 167
    - en changeant la hauteur 172
  - ajouter
    - au document 111
    - dans un élément 116
  - copier 128
  - déplacer 128
  - dimensions CSS 142
  - entourer 117
  - HTML, créer 94
  - masquer 167
    - en changeant la hauteur 172
  - opacité 170
  - remplacer 131
  - styles 139
  - supprimer 122
- empty() 125
- end() 49
- Entourer des éléments 117
- Espace de nom 83
- evenements 222
- Événements 55–91
- Ajax 201
  - déclencher 85
    - jQuery 1.7 223
  - jQuery 63
    - liste 68
  - personnalisés 86
  - prédictifs 72
  - propagation 57
  - supprimer 79
    - jQuery 1.7 222
  - version 1.7 221
- Event 57, 89
- event.data 222
- Exemple récapitulatif 205

## ExtJS 5

**F**

- fade 170
- fail() 202
- filter() 33
  - paramètres 43
- find() 33, 37
  - paramètres 43
- fire() 227
- Firebug 19
- fire() 227
- Firefox, DOM Inspector 26
- fireWith() 227
- Fonction de rappel, jQuery 1.7 223
- Formulaires 206
  - Ajax 218
  - caractères accentués 214
  - sécurité 206
  - serveur 217
  - vérifier les champs 210
- Frameworks 4
  - choisir 10
  - utilisation 6
  - utilité 4
- Fuite de mémoire 123
- 
- G**
- get()
  - appel Ajax 34
  - DOM 34
- getAttribute() 134
- getJSON() 198

**H**

- has() 227
- Hauteur d'un élément 172
- hide() 167
- holdReady() 62
- html("") 125
- html() 100
  - vs. text() 105
- HTML
  - ajouter du contenu 111
  - balises, remplacer le contenu 100
  - créer 94
  - Internet Explorer 99
- HTML5 99

**I**

- innerHTML 98
- insertAfter() 115
- insertBefore() 115
- Instructions, chaîner 45
- Internet Explorer 3
- Introduction au Design Web 248
- is() 51
- isRejected() 204
- isResolved() 204

**J**

- JavaScript
  - bibliothèques 3
  - core 2
  - essor 1
  - Internet Explorer 3

## jQuery

- attentes 9
- DOM 18–54
- effets 163
- exemple récapitulatif 205
- fonctionnement 12
- intégrer 14
- objet
  - ajouter 44
  - modifier 36
- présentation 1–17
- propriétés gérées 90
- utiliser correctement 8
- versions 9
  - 1.7 221

## jQuery() 28

## jQuery Mobile 220

## jQuery UI 164, 220

## jqXHR 186

## JScript 3

## JSON 196

## JSONP 196

**L**

## Listeners 57

- live() 73
  - limites 75

## load() 190

## load (JavaScript) 56

## lock() 227

**M**

## map() 52

## Masquer un élément 167

memory 226

Memory leak 123

MooTools 5

mousemove 210

## N

Navigation par onglets

effets 175

exemple complet 152

Noms de classe 148

not() 54

notify() 225

notifyWith() 225

## O

off() 222

on() 222

once 226

Opacité d'un élément 170

opacity 142, 170

## P

param() 199

parametres 185

Parcourir une sélection 45

pending (état en cours) 225

PHP et Ajax 205

Plug-ins 50

prepend() 116

prependTo() 117

progress() 225

prop() 136

vs. attr() 136

Propriétés 134

affecter 136

gérées par jQuery 90

vs. attribut 23

Prototype 5

Pseudo-éléments 30

## R

ready 58

ready() 58

contrôler l'événement 62

readyState 184

Récupérer la sélection précédente 49

reject() 203

rejectWith() 203

.remove() 227

remove() 122

removeAttr() 139

removeClass() 149

removeProp() 139

replaceAll() 131

replaceWith() 131

Requêtes

actions selon le résultat 202

asynchrones 184

inter-domaines 196

resolve() 203

resolveWith() 203

Retirer des éléments 54

return 110

Rollover 70

**S**

`<script>` 16  
`script.aculo.us` 5  
Scripts  
  chargement du DOM 16  
  intégrer 16  
`scrollLeft` 173  
`scrollTop` 173  
Sécurité 206  
selecteur, paramètre 222  
Sélecteurs 31  
Sélection  
  parcourir 45  
  récupérer 49  
`selection.get()` 34  
Sérialiser des données à envoyer 199  
`serialize()` 200  
`serializeArray()` 201  
`setAttribute()` 134  
`setDisplay()` 169  
setter 100  
`show()` 167  
`siblings()` 44  
`slide` 172  
  sens du glissement 178  
`</span>` 96  
`stop()` 174  
`stopOnFalse` 227  
`style` 24, 139  
Styles 139. *Voir aussi CSS*  
Supprimer des éléments 122  
Surv. *Voir Effets*

`swapDisplay()` 169  
`swapFullDisplay()` 169

**T**

`text()` 100  
  vs. `html()` 105  
`toggle()` 71  
`toggleClass()` 150  
`trigger()` 86  
`triggerHandler()` 89

**U**

`unbind()` 79, 83, 222  
`undelegate()` 85, 222  
`unique` 226  
`unwrap()` 125  
`url` 185

**V**

`val()` 109, 136  
vanilla JavaScript 94  
Version 1.7 de jQuery 221

**W**

`wrap()` 117  
`wrapAll()` 121  
`wrapInner()` 120  
*write less, do more* 13

**Y**

YUI 5