L'HOMME DERRIÈRE LA MAGIE: ENTRETIEN AVEC JOHN RESIG de JQUERY

Visual jQuery

Le Magazine

$ TUTORIEL

Une introduction au monde de jQuery — en commençant par l'objet jQuery

$ JQUERY

Une philosophie gagnante: pourquoi l'approche de jQuery est efficace

Conception d'un plugin jQuery, de l'intérieur

Visual jQuery Magazine

Rencontre avec Dave Cardwell

Le créateur de jQBrowser et jQMinMax discute avec jQuery Magazine.

En Page 9

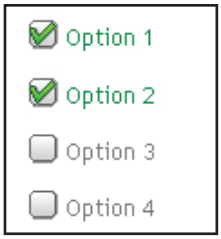
2E NUMÉRO APERÇU

Le mois prochain, une interview de **Klaus Hartl**, le créateur du célèbre plugin Tabs et un des développeurs du site Plazes.

Également à venir, un tutoriel sur les fonctionnalités **AJAX** de jQuery, les effets visuels du plugin roundup, un autre article et vos courriers. rendez vous le 18 Octobre.

ECRIVEZ-NOUS

Merci d'envoyer vos courriers à editor@visualjquery.com. Nous mettrons les lettres concernant les articles du magazine ou à propos de jQuery et sa communauté si l'espace libre le permet. Nous nous réservons le droit de ne pas publier toutes les lettres reçues.



De la part de l'éditeur

Pour respecter la tradition, j'aimerais profiter de mon premier éditorial pour souhaiter la bienvenue aux nouveaux lecteurs du **Magazine Visual Jquery**.

Il y a environ un an, après un assez grand nombre de développements web traditionnels, je me suis intéressé à un concept plus récent nommé **AJAX**. J'ai participé à un atelier conduit par l'éminent Thomas Fuchs, le créateur de **Scriptaculous** (basé sur la bibliothèque Prototype), et j'ai vraiment été impressionné par les possibilités de Prototype et de Scriptaculous, quelque chose m'attirait avec force.

Après les avoir utilisés pendant un moment, et devenant assez compétent dans l'utilisation de Prototype pour monter des **applications web** riches et réactives, je me suis rendu compte que la courbe d'apprentissage était bien trop raide. Tout en étant capable de faire toute sorte de choses puissantes, je me retrouvais constamment à réinventer la roue pour des tâches toutes simples. Bien que le Javascript AJAXien ait obtenu l'appellation de « **DOM Scripting** », je ressentais difficilement l'influence du DOM sur mes développement Prototype au jour le jour.

Heureusement, j'ai découvert **Ruby on Rails** par hasard peu de temps après ; celui-ci fait un sacrément bon boulot dans l'abstraction du concept entier de Javascript (et les modèles RJS, maintenant populaires, n'étant encore pas disponibles, je trouvais toujours les aides **Prototype** de Rails pour atténuer).

Quelques mois plus tard, j'ai découvert la bibliothèque **Interface** dans un message de forum sur les divers effets sympathiques qu'elle propose (avec une attention spéciale pour le plugin selectables), et j'ai suivi le lien jusqu'à **jQuery**.

Immédiatement je pris goût à ce framework qui semblait penser comme je programmais : en se concentrant sur les **éléments DOM** et en implantant des fonctionnalités supplémentaires au dessus d'eux, jQuery rendait le Javascript à nouveau amusant.

jQuery était de loin au dessus du lot, et je m'y plongeais rapidement à l'aide de la documentation existante, tout en essayant d'organiser mieux le wiki de documentation de jQuery, et finalement j'ai lancé le premier jet de **Visual jQuery**, l'agréable présentation visuelle de l'interface de programmation jQuery. Lorsque jQuery 1.0 a commencé à inclure la documentation intégrée, j'ai reprogrammé Visual jQuery pour qu'il en tire avantage.

Je soupçonne beaucoup des lecteurs de ce magazine d'avoir emprunté des voies parallèles pour trouver jQuery. D'autres encore, initialement horrifiés à l'idée d'apprendre Javascript, furent sans aucun doute agréablement surpris de voir combien il était **facile** d'apprendre jQuery. Si tout va bien, ce magazine séduira les deux catégories. Aux jQueryistes chevronnés, nous apportons des techniques avancées, et un **aperçu des plugins** qui vont compléter vos prochains projets.

Pour les débutants, nous inclurons des **tutoriels de base** sur le framework, qui vous aideront à bâtir votre premier projet avec une solide compréhension de ce que vous êtes en train de faire, plutôt que de vous limiter à copier/coller les extraits de code.

Sur ces bonnes paroles, je vous laisse **au magazine**. Bonne chance avec jQuery !

Yehuda Katz

Illustration: **Jörn Zaefferer**

UNE PHILOSOPHIE GAGNANTE

Pourquoi l'approche jQuery fonctionne-t-elle

Par Yehuda Katz

L'approche que retient Jquery ne consiste pas seulement en «un code propre» ou des capacités de traitements «en chaîne». Sa philosophie première, concentrée sur des ensembles d'éléments du DOM, l'amène au cœur des usages des programmeurs de Javascript.

Par contraste, les autres frameworks, tels Prototype ou Dojo, adoptent une approche fonctionnelle. Évidemment, ils sont tout à fait capables de manipuler des éléments du DOM, tout comme jQuery, mais ces frameworks reposent sur des choix de programmation complètement différents.

Prototype, pour sa part, s'applique à constituer une extension orientée objet des fonctionnalités natives (et limitées) du Javascript. En quête de l'orientation objet la plus pure, ses développeurs ont passé un temps considérable à développer des classes orientées objet pour différents types de fonctionnalités. Une classe pour les formulaires, une autre pour les éléments, une troisième pour les événements, et ainsi de suite. Il est donc tout à fait possible de produire du code Prototype propre et élégant. Et Prototype permet d'émuler une

des fonctionnalités les plus fascinantes permise par l'approche orientée DOM de Jquery : les chaînes de commandes.

Mais jQuery conçoit le développement de la programmation Javascript de la manière dont de nombreux acteurs majeurs de la communauté du Javascript commencent à l'envisager : un langage de programmation axé en priorité sur le DOM.

Pour ceux d'entre nous pour qui la programmation Javascript est axée sur les éléments de la page, et je pense que c'est la majorité d'entre nous, jQuery simplifie terriblement les choses.

Le workflow de jQuery

La plupart des commandes jQuery manipulent les éléments du DOM en s'appuyant sur la prise en compte de CSS3, XPATH, et sur un lot d'expressions propres (telles que :visible qui ne sélectionne que les éléments visibles, ou :checked qui ne retourne que les champs de formulaires sélectionnés).

Mais les choses deviennent intéressantes dès lors que vous avez sélectionné un ensemble d'éléments. Ajoutez.fadeIn("slow") et un dégradé s'appliquera lentement sur chaque élément. Mais ce n'est pas tout, ajoutez maintenant .addClass("cEstDingue") et chaque élément recevra la classe « cEstDingue ». C'est dingue non?

Et ça continue : ajoutez .click(function() { alert("Hello"); }); pour activer une alerte lorsqu'un élément est cliqué, ajoutez .append("Hello") et le mot «hello» sera ajouté à la suite de chaque élément correspondant. Cool non?

Les sélecteurs jQuery

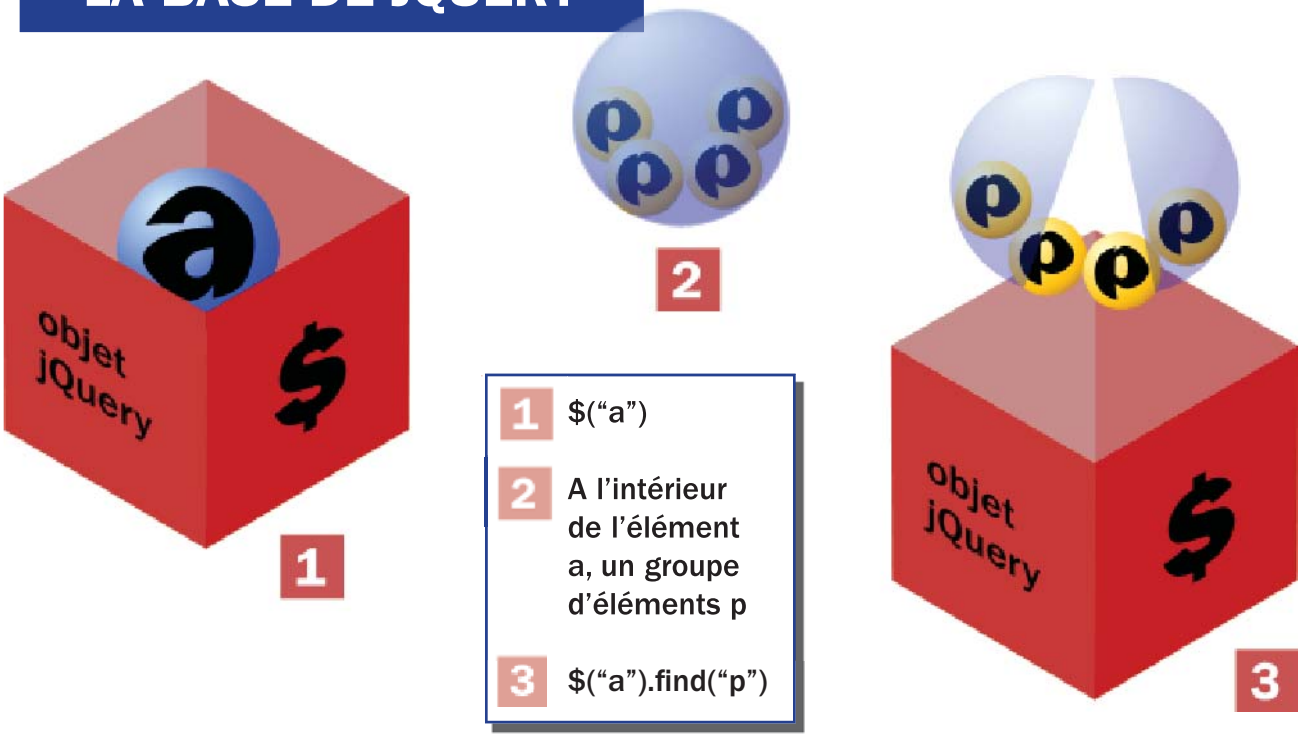
Maintenant que nous avons vu le pouvoir des commandes jQuery, intéressons nous à la manière de sélectionner préalablement un ensemble d'éléments du DOM de la page.

Heureusement, nous pouvons compter sur CSS3 (1-3), sur une gestion simplifiée du XPATH et sur quelques expressions sur mesures ajoutées pour faire bon poids. Quand on évoque CSS3, ce n'est pas une blague, jQuery gère le sélecteur ~, :not(expr), mais aussi les attributs via [@attr='quelquechose'].

La gestion de XPATH est plus limitée, mais l'essentiel y est. Les opérateurs / et // sont disponibles, de même que les notions de parents, de frère, de précédent. jQuery gère :first, :last, et :eq(n), une version simplifiée de [position() = n]. Pour finir, jQuery permet de tester le contenu des éléments via [tag].

Et puisque jQuery permet de parser le XHTML, il est possible de l'utiliser pour parser du XML simple lors d'une requête AJAX. Bien sûr, jQuery dispose de la panoplie complète des commandes liée à AJAX avec $.ajax, et $().load. Ω

LA BASE DE JQUERY



L'HOMME DERRIÈRE LA MAGIE

JOHN RESIG

Ce magazine, et jQuery lui-même, doivent leur existence à la vision forte et inébranlable de John Resig. Comme beaucoup d'enfants des années 80, John a atteint la majorité en même temps que les ordinateurs.

Son premier langage de programmation, l'omniprésent QBASIC, détermine la curiosité intellectuelle de John pour la programmation. A l'exception de Java, il est autodidacte pour tout ce qui concerne la programmation, ce qui fait de son prochain livre, Techniques Javascript professionnelles (Pro Javascript Techniques), qui devrait paraître en décembre prochain chez APress, un accomplissement.

Tout en se préoccupant peu de Java, il aime observer les programmeurs talentueux au sein de leur élément : il suit plus de 250 fils d'information du Web par jour. « Voir un programmeur étonnant dans son "habitat naturel" est toujours une vision de la beauté », dit Resig.

Parmi ses nombreuses influences, John Resig cite Alex Russel (de Dojo) et Dean Edwards (bien connu pour IE7) comme les plus intéressants sur la durée. « Si il y a des développeurs Javascript que j'admire et respecte, ce sont bien eux. »

Tout en respectant les développeurs de frameworks plus établis sur le marché — comme Dojo et Prototype —, Resig, comme David Heinemeier Hansen (Rails), a des opinions bien établies sur la direction que doit prendre son framework.

Comme Ruby on Rails, JQuery peut être vu comme un logiciel aux idées bien arrêtées, où une philosophie forte de conception donne aux développeurs un moyen simple et cohérent d'aborder des problèmes auparavant complexes.

Au-delà de cette philosophie, Resig ne lésine pas sur l'aspect communautaire des choses. « Je suis fréquemment découragé par les listes de diffusion d'autres projets, où une question simplement mal orientée obtiendra une réponse pleine de colère et de méchanceté » dit-il. « Par exemple, une question qui arriverait se rapportant à Javascript plutôt qu'à JQuery lui-même, serait facile à écarter. Mais en prenant le temps d'y répondre, vous pouvez gagner un nouvel utilisateur. »

Afin d'assurer à la communauté quantité de bonnes ressources, John s'est consacré fortement à la documentation avec, en point de mire, la version 1.0 de JQuery.

> 'Le triomphe a simplement été de voir le code diffusé.'

Pour considérer le framework comme prêt pour la production, il ne suffisait par que le code soit sans erreur ; tout devait être documenté.

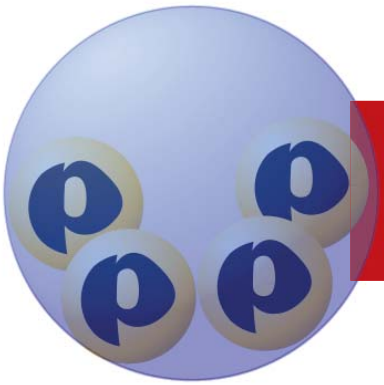
Cela a payé. Comme résultat de ses efforts de documentation innovante, la version 1.0 de jQuery engendra un site de documentation — Visual Jquery — dont la mise à jour est dynamique, faite au fur et à mesure de la base de code elle-même. Par opposition à l'ancien Visual JQuery qui nécessitait des entrées manuelles et fastidieuses, le nouveau site est devenu une place de premier recours pour la documentation JQuery la plus récente.

Prouvant que JQuery est plus qu'une tocade, Resig parlera à la conférence Ajax Experience organisée par le populaire Ajaxian.com. En plus d'un exposé sur JQuery lui-même, il présentera une discussion sur le choix d'un framework Javascript.

La progression jusqu'à JQuery 1.0 apporta son lot de difficultés mais tout autant de récompenses. « Le triomphe a simplement été de voir le code diffusé. Comme tout développeur vous le dira, ce n'est pas facile de réussir l'emballage final », dit Resig.

Et le travail qu'il a investi en alimentant la communauté JQuery a rapporté. « Il y a plutôt quelques développeurs ayant un accès [subversion] qui ont apporté leur aide, résolvant des problèmes au fur et à mesure qu'ils se présentaient à eux. » Alors que Resig est devenu récemment très occupé avec son travail et son livre sur Javascript, la communauté a repris le flambeau ; l'élan post-1.0 est en marche.

Ça ne veut pas dire que la version est sortie sans difficulté. « J'ai eu à casser l'interface dans une certaine mesure. Plusieurs noms de méthodes n'étaient tout simplement pas aussi clairs qu'ils auraient dû l'être et causaient de nombreux conflits », dit Resig. Cela dit, pratiquement tous les utilisateurs on surmonté les difficultés, et les plugins principaux, comme la bibliothèque d'effets visuels Interface, ont diffusé des versions mises à jour simultanément à la sortie officielle de jQuery 1.0. Ω



L'OBJET JQUERY

UNE BRÈVE INTRODUCTION

jQuery, en tant que « framework », est au départ un outil permettant de manipuler un ensemble d'éléments du DOM.

Vous allez vous apercevoir que cette expression revient comme un leitmotiv dans ce magazine : mais qu'est-ce qu'un ensemble d'élément du DOM en définitive? Et comment jQuery facilite-t-il la manipulation de ces ensembles?

Élément du DOM

Un élément du DOM est un nœud HTML isolé, comme un p ou un a. Il peut être vide, ou contenir du texte ou d'autres éléments.

Voici une manière de le retenir : à chaque fois que vous ouvrez une balise HTML, vous créez un élément du DOM. Tout ce qui se trouve à l'intérieur de cette balise est un enfant de cet élément du DOM.

Un élément peut être un parent, c'est-à-dire qu'il contient un autre élément, ou bien il peut être un enfant, c'est-à-dire qu'il possède un parent. Dans l'exemple présenté sur cette page, le p est un élément parent à la fois du span et du hr. Le span quant à lui est un enfant du p.

Les éléments peuvent également être frères, c'est à dire qu'ils possèdent le même parent. Le span et le hr dans l'exemple sont des frères car ils partagent le même élément p comme parent.

Ensemble d'éléments

Les programmeurs javascript ont pris l'habitude de stocker des ensembles d'éléments du DOM dans des tableaux standards.

Il était ainsi possible de déterminer la taille d'un tableau, de le parcourir par itération, et d'obtenir un élément du tableau à l'aide de son identifiant. Des fonctions habituelles de tableaux en somme.

Mais il n'existait pas vraiment de technique pour définir un tableau d'éléments du DOM avec ses propriétés spécifiques. Supposons que vous souhaitiez attribuer une classe à tous les éléments d'un ensemble : avec un tableau classique il faut parcourir tous les éléments à l'aide d'une boucle itérative ; et pour chaque élément ajouter la classe.

Ceci fonctionne parfaitement, et les travaux sur quelques-un des frameworks les plus populaires ont permis d'affiner le concept et de faciliter certaines opérations telles que l'ajout d'une classe.

Mais dans toutes les tentatives de définitions de nouvelles syntaxes, un ensemble d'éléments du DOM n'était pas différent d'un ensemble de caractères ou d'entiers.

Sur ce point, jQuery se distingue de tous ses prédécesseurs. Au lieu de considérer un ensemble d'éléments comme un tableau de plus, jQuery considère qu'un ensemble est quelque chose d'exclusivement axé sur le DOM.

Ainsi, en plus des méthodes traditionnelles pour obtenir la taille du tableau, les ensembles de nœuds du DOM contenus dans un objet jQuery peuvent accomplir un tas de choses intéressantes.

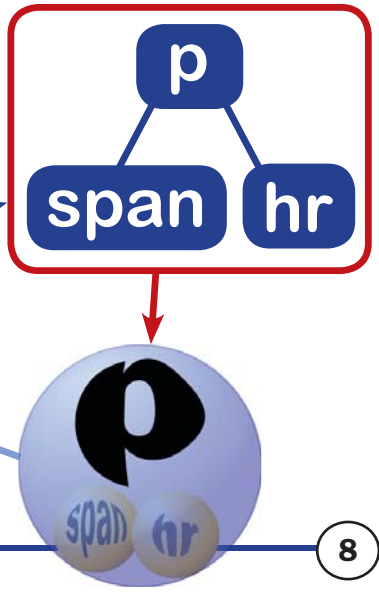
Pour commencer, vous pouvez intégrer des éléments du DOM dans un objet jQuery d'une manière très intuitive à l'aide des sélecteurs CSS.

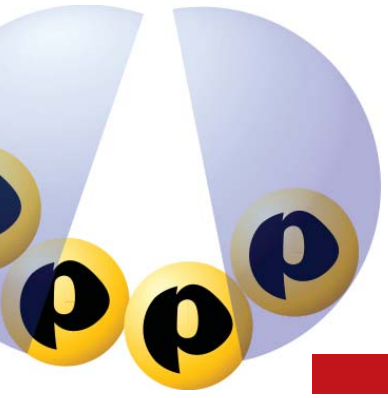
Par exemple, $("a.amusant") ; va remplir un objet jQuery avec tous les éléments a de la page qui portent la classe CSS « amusant ». De fait vous pouvez considérer un objet jQuery comme une boîte qui contient un ensemble d'éléments du DOM.

Une fois l'objet jQuery défini, on peut appliquer d'un coup toutes

Entrer dans jQuery

```
<p>
  <span>
    Hello
  </span>
  <hr />
</p>
```





‘Un objet jQuery est une boîte contenant un ensemble de nœuds du DOM’

sortes d’opérations sur l’ensemble de ses éléments.

Vous souvenez-vous que nous souhaitions précédemment ajouter une classe à tous les éléments de notre ensemble ? Avec jQuery nous pouvons adresser un message à l’objet jQuery, lui indiquant qu’il faut ajouter la classe “bonjour” à tous les éléments contenus dans la boîte. Et la syntaxe est incroyablement simple : $(“a.amusant”).addClass(“bonjour”) ; Il y a beaucoup d’énergie présente dans cette petite expression : nous récupérons dans la page tous les éléments correspondant à une définition CSS, et nous ajoutons à chaque élément correspondant la classe “bonjour”. Mais ce qui est encore plus fascinant, c’est de réaliser des chaînes en jQuery. Il est par exemple possible d’écrire $(“a.amusant”).addClass(“bonjour”).hide(). Les éléments contenus dans la boîte reçoivent en fin de chaîne la directive “hide”, ce qui aura pour effet de les faire disparaître.

Ce qui rend jQuery singulier, c’est que toutes les commandes qui appliquent des modifications à un ensemble d’éléments retournent la série d’éléments modifiée, de manière qu’elle puisse à son tour être manipulée.

Détecter des événements

Un autre besoin régulier, dans la programmation orientée DOM, est d’associer des détecteurs d’événements à divers éléments de la page.

Par exemple, vous pourriez souhaiter modifier la classe d’un élément lorsqu’il est cliqué, pour indiquer qu’il a été sélectionné. Admettons par exemple que vous souhaitiez ajouter la class “on” à tous les éléments p de la page qui portent la classe “clicable” lorsqu’ils sont cliqués.

La syntaxe courante en jQuery:

```
$(“p.clickable”).click(
  function() {
    $(this).addClass(“on”);
  });
```

On distingue ici quelques idiomes javascript et jQuery ; détaillons le code pas à pas.

D’abord, $(“p.clicable”) collecte tous les nœuds de type p de la page qui portent la classe “clicable”, et les met dans une boîte jQuery.

Ensuite, *.click()* permet de définir l’action que le navigateur doit entreprendre lorsque qu’un événement «click» est détecté sur un des éléments de la boîte.

Les actions à entreprendre, ou **callback**, sont définies par des fonctions javascript anonymes, c’est à dire des fonctions javascript dont le nom n’a pas besoin d’être défini.

A l’intérieur du « callback » d’un détecteurd’événement, le mot clé « this » se réfère à l’élément particulier sur lequel l’événement a été détecté.

Vous vous demanderez peut-être pourquoi nous avons éprouvé le besoin d’écrire $(this).addClass() plutôt que simplement this.addClass() ; souvenez-vous qu’on ne peut appliquer la commande .addClass() qu’aux objets d’une boîte jQuery.

A l’intérieur d’un détecteur d’événements, « this » se réfère à l’élément lui-même. L’écriture $(this) permet de placer l’élément dans une boite jQuery, qui devient alors éligible pour recevoir des commandes spécifiques à jQuery, telles que .addClass().

C’est parce que .click() est une commande de jQuery que vous pouvez l’enchaîner avec d’autres commandes que vous placez à la suite.

Si vous supprimez le point-virgule à la fin de la commande, alors vous pouvez ajouter des méthodes supplémentaires en passant à la ligne:

```
$(“p.clicable”).click(...)
.append(
  “<span>X</span>”
).fadeTo(“slow”, 0.5);
```

va ajouter un span à tous les éléments p qui portent la classe « clicable », puis leur appliquer un dégradé lent jusqu’a régler l’opacité à 50%. **Ω**

POINTS CLÉS

- Seuls les objets jQuery (boîte d’éléments) peuvent recevoir des commandes jQuery
- A l’intérieur d’un détecteur d’événement, « this » se réfère à l’élément sur lequel l’événement s’est produit
- Les détecteurs d’événements sont définis par des fonctions anonymes, comme suit : function() { ... }

PLUGINS

APPLICATIONS FERTILES ET GÉNÉREUSES

JQUERY ET LES PLUGINS

jQuery possède une architecture très simple pour les plugins qui permet aux développeurs de réutiliser les propriétés quasi-magiques des fonctions jQuery.

Pour créer une nouvelle méthode opérant sur un objet jQuery, les développeurs ont seulement besoin de créer une nouvelle fonction appelée jQuery.fn.foo qui doit juste renvoyer l'objet jQuery lui-même (pour garantir la possibilité d'enchainement). Pour bien fonctionner, une fonction jQuery devrait itérer automatiquement parmi les éléments, ceci afin de maintenir la cohérence entre le noyau et les plugins.

jTip par Cody Lindley

Cette page est en réalité calquée sur le plugin jTip (oui, c'est sympa comme ça). Cette solution pour infobulles vous permet de constituer des infobulles AJAX avec du pur balisage XHTML — après avoir inclus le plugin jTip, bien sûr.

Échantillon de code:

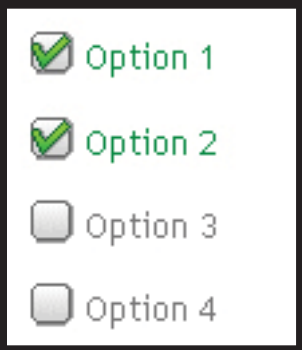
```
<a href="ajax.htm" class="jTip" id="one"
   name="Le mot de passe doit
respecter:">Texte
</a>
```

dateSelector par Kelvin Luck

Ce widget est un sélecteur de date vraiment nécessaire pour jQuery. Il résout notamment le problème des boîtes de sélection dans IE, de sorte que son calendrier de contrôle peut couvrir des boîtes de sélection. dateSelector et le contrôle de cases à cocher prouvent que jQuery peut fournir un ensemble très robuste de widgets. Il est aussi profondément adaptable par région.

Échantillon de code:

```
<input type="text" class="date-picker" name="date1"
id="date1" />
```



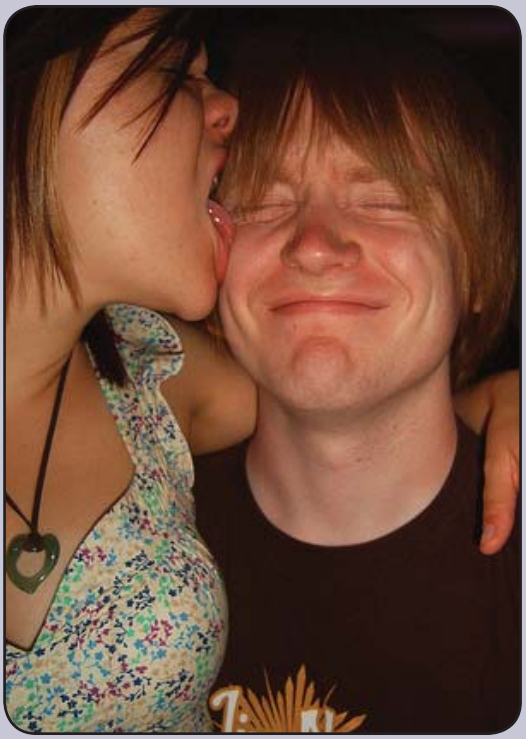
Checkbox par Kawika K.

Le plugin checkbox permet d'utiliser n'importe quelle image à la place des cases à cocher standards. On peut définir différentes images pour les cases vides, visitées (hover) ou cochées. Écrit à l'origine en éditant les propriétés CSS brutes, cette nouvelle version utilise les classes CSS pour plus de simplicité. La syntaxe, comme pour les plugins précédents, est on ne peut plus simple.

Échantillon de code:

```
$().cssCheckbox();
```

Rencontrez Dave Cardwell . . .



Dave Cardwell était programmeur bien avant que la publication de jQMinMax et JQBrowser ne lui apporte la célébrité au sein de la communauté jQuery.

« J'avais 8 ans lorsque mes parents m'achetèrent un Amstrad à écran vert et je n'ai pas tardé à me plonger dans le BASIC. »

Maintenant, après une année à l'Université de York, Cardwell se décrit comme « programmeur et designer indépendant ».

Il a publié son plugin **jQMinMax** au début d'août 2006.

« A l'époque, il y avait beaucoup de bruit à propos de la tendance vers des mises en page liquides et fluides. Je voulais créer un plugin qui soit d'une utilité immédiate pour les gens, et j'ai vu là une opportunité pour jQMinMax. » Le plugin simule max/min-height/width dans Internet Explorer.

Comme beaucoup de fans jusqu'au-boutistes de jQuery, John Resig et sa communauté toujours active furent une grosse part de ce qui y mena Cardwell.

« Ce doit être la documentation et la réactivité de la communauté qui m'ont rendu épris de jQuery. Tandis que la syntaxe et les fonctionnalités sont charmantes, il y a d'autres bibliothèques disponibles qui ont des caractéristiques comparables. Ce furent ces avantages périphériques et un sens réel du progrès dans le noyau de la bibliothèque qui m'y ramenèrent sans cesse. »

Pour Dave, JQuery fut une lumière à l'extrémité du tunnel, le tunnel étant Javascript, dont la police de la circulation ne semblait pas efficace.

« Je devenais de plus en plus frustré à essayer de dépister les développements dans la communauté Javascript. jQuery m'a apporté tout cela sous le même toit, avec une taille de fichiers qui ne remettait pas en compte la réactivité de mes sites. »

Selon Cardwell, les débutants ne doivent pas être effrayés d'expérimenter avec jQuery.

« Les sites **jQuery** et **Visual jQuery** sont indispensables, et quand vous êtes véritablement bloqués, je n'ai jamais vu une question sur la liste qui reste sans réponse. »

Cardwell a publié plusieurs autres plugins depuis qu'il a commencé à utiliser la bibliothèque jQuery. Vous trouverez plus d'information sur ces plugins sur http://davecardwell.co.uk/geekery/javascript/jquery/.

« Je suis toujours plus qu'heureux d'avoir des retours de gens dans le même état d'esprit. On peut me contacter par mon site. On peut aussi me trouver ordinairement planqué dans le canal IRC #jquery sur freenode.org. »

QUI SOMMES-NOUS

Publisher
Wycats Designs

Éditeur
Yehuda Katz

Contributors
Dave Cardwell
Klaus Hartl
John Resig
Leah Silber
Jörn Zaefferer

Traduction
Allergie
BoOz
cy_altern
Fil
Toggg

CONTRIBUTIONS

Les contributions sont bienvenues (et désirées). Nous acceptons des illustrations, des articles et des interviews. Nous nous réservons le droit d'éditer toute contribution pour la clarté et le bon goût.

S'il vous plaît, faites parvenir vos contributions au magazine par editor@visualjquery.com

PLUGINS

Si vous désirez voir votre plugin figurer dans une édition du Magazine, veuillez envoyer un lien sur le plugin, une description, quelques détails sur vous-même et, idéalement, votre photo.

Nous incluons des plugins par groupements thématiques, votre plugin peut donc paraître dans plusieurs numéros. Il pourra paraître groupé avec tout autre plugin que nous jugeons approprié. Les titres, descriptions et détails des plugins sont créés par les éditeurs du Magazine et laissés à notre discrétion.

LE JAVASCRIPT ENFIN SEXY

www.jQuery.com