Gildas P.
// Creative + code

WebDesigner
Graphiste Multimedia
Programmeur

Tél/ 06.31.15.13.04
Email/ contact@gildasp.fr
Web/ www.gildasp.fr

// Mini-guide du débutant en Javascript/JQuery

// Syntaxe Javascript générale

Le code Javascript est à placer dans le header de votre page HTML (entre <head> et </head>).
Pour délimiter la zone de texte où on parle en Javascript, on utilise les balises <script language="javascript"> et </script>

Pour importer un fichier Javascript séparé, la syntaxe est la suivante (toujours dans le header) :

```
<script language="javascript" src="js/monfichier.js"></script>
```

Là, on importe un fichier monfichier.js qui se trouve dans le dossier js/.

Attention à ne pas oublier les points-virgules après chaque ligne de Javascript !

// Commentaires en Javascript

Comme en ActionScript, on a 2 types de commentaires au choix :

```
// ceci est un commentaire d'une seule ligne
/* ça c'est pour commencer un commentaire multiligne,
qu'on doit absolument terminer par */
```

// Fonctions utilisateur

Exactement comme en ActionScript, il est possible de créer ses propres fonctions.
Ca n'a plus à voir alors avec la syntaxe JQuery, mais avec la syntaxe Javascript normale :

```
function truc(parametre){
        // le code Javascript/JQuery
};
```

Et on l'appellera par

```
truc("test");
```

// Débugger son code

La fonction Javascript alert() permet d'afficher une "boite d'alerte" dans le navigateur, et de mettre la lecture du code Javascript en standby tant qu'on n'a pas fermé le message d'alerte...
c'est le meilleur moyen de vérifier si vous récupérez bien une variable, ou localiser à quel moment votre code déraille.
si le alert() s'afiche c'est que jusque là ça va, quand il ne s'affiche pas c'est qu'il y a un problème en amont, dans le code.

alert('salut !');
vous ouvrira donc un message "salut !" avec un bouton OK, quel que soit le navigateur.
Evidement il faudra penser à les supprimer avant de mettre en ligne vos pages.

Pour débugger le code Javascript et savoir s'il y a une erreur, il convient d'adopter une bonne configuration de travail.

Voici ma configuration de travail, qui me permet de bénéficier d'outils efficaces pour tester tout ça dans de bonnes conditions :

- **Firefox** (dernière version : http://www.mozilla.org/fr/firefox/new/)
propose par défaut une "console d'erreur" déjà très précise, qui vous indiquera le fichier qui pose problème, ainsi que la ligne exacte de l'erreur.

Pour ouvrir cet outil :
CTRL(Pomme) + Shift + J
ou dans le menu :
Firefox / Développement web / Console d'erreur

La console permet de voir pas mal de chose, là ce qui nous intéresse c'est l'onglet "Erreurs".
N'hésitez pas à utiliser régulièrement le bouton "Effacer" pour vider toutes les erreurs accumulées sans que vous le sachiez sur tous les sites que vous visitez...

- **Firebug** (https://addons.mozilla.org/fr/firefox/addon/firebug/)
est un plugin de Firefox très utilisé, qui permet d'avoir beaucoup plus de détails sur ce que votre code Javascript fait à faire code HTML/CSS, en temps réel.

Une fois installé (cliquez sur le bouton "+ Ajouter..." et ça se fait très bien tout seul) et redémarré le navigateur, vous pourrez accéder à Firebug via un simple clic droit sur l'élément graphique dont le code vous intéresse... et sélectionnez "Inspecter un élément".

Le code source que Firebug vous présente alors est "dépliable", mis en couleur pour plus de lisibilité, et à jour des éventuelles injections, suppressions, modifications de contenus faites en Javascript.
Simple et très efficace !

// Installer JQuery

Déjà, pour pouvoir utiliser les fonctions très utiles de JQuery, il faut importer la librairie dans votre document HTML.

Etape 1 : Télécharger la librairie JQuery

http://code.jquery.com/jquery-1.7.1.min.js
(version de Janvier 2012, pour une version à jour par la suite : http://jquery.com/)

C'est un simple fichier texte, avec une extension .js pour signaler qu'il est écrit en Javascript.

Etape 2 : L'importer dans vos documents

Dans chaque document html/php où vous souhaitez utiliser JQuery, il faut ajouter dans le header la balise :

```
<script language="javascript" src="lib/jquery-1.7.1.min.js"></script>
```

Si vous importez plusieurs fichiers Javascript dans votre document, commencez par JQuery, au cas où un des fichiers suivants en aurait besoin pour s'exécuter...

Etape 3 : Mettre en place "l'enveloppe d'exécution"

Le principe, c'est que si on exécute notre codde JQuery avant que toute la page soit chargée, on riste d'essayer d'accéder à des balises (ou des images) qui ne sont pas encore écrites dans le document...
Pour palier au problème, on va donc saisir toutes les instructions JQuery dans une fonction spéciale qui se déclenche seulement une fois que l'ensemble de la page est chargée.

On doit donc créer le bloc suivant dans le header, après les imports de JQuery et des autres fichiers Javascript :

```
<script language="javascript">
$(document).ready(function(){
        // c'est ici que je vais écrire le code JQuery de ma page
});
</script>
```

Rien n'empêche d'utiliser cette fonction dans un fichier.js séparé, aussi.

// Les sélecteurs JQuery

En fait, les sélecteurs de JQuery sont complètement hérités des sélecteurs CSS, avec en plus le support des sélecteurs CSS3 voire 4, qui restent instables côté CSS.
Avec JQuery ça fonctionne très bien, quel que soit le navigateur (s'il a moins de 10 ans, quoi).

JQuery est une syntaxe "ciblée", on commence donc presque toujours une instruction en désignat l'élément HTML sur lequelon veut agir.
Comme en CSS, on peut sélectionner 1 ou plusieurs éléments du <body>, et de plusieurs manières.

Sélecteur JQuery	Balises HTML ciblées	Exemple HTML
$('h1')	Toutes les balises du document	...
$('#truc')	La balise unique (de n'importe quel type) dont l'attribut id est "truc"	... ou <h1 id="truc">,,,</h1>
$('.truc')	Toutes les balises de tous types dont l'attribut classe est "truc"	... <h1 class="titre truc">...</h1>

On peut cumuler les 3 syntaxes pour écrire des chemins plus complexes vers une cible mieux délimitée :

$('div#article .titre')	Sélectionne seulement les balises de classe "titre" à l'intérieur de la div dont l'id est "article"
$('h1.titre')	Sélectionne seulement les h1 de classe titre, mais pas les h2 de classe "titre" par ex
$('#article .titre a')	Sélectionne les liens (balises <a>) qui se trouvent dans une balises de la classe titre, seulement dans la div dont l'id est "article"

Quelques sélecteurs spéciaux hérités du CSS :

$('a')	Tous les liens
$('a:hover')	Le style au survol des liens
$('a:visited')	Le style des liens déjà visités

Quelques sélecteurs spécifiques de JQuery (hérités du CSS3 ou pas) :

Sélecteur JQuery	Balises HTML ciblées	Exemple HTML
$(window)	La fenêtre dans laquelle la page s'exécute	
$(document)	Le document html au complet, la page dans son entièreté	
$('body')	Tout le body du document (aussi appelé le DOM du document)	<body>...</body>
$('h1:first')	Sélectionne la première balise <h1> rencontrée	... // celle-ci ... ...
$('h1:last')	Sélectionne la dernière balise <h1> rencontrée	... ... // celle-ci
$('h1:eq(3)')	Permet de sélectionner un objet spécifique de la liste complète des balises <h1>. La liste commence au numéro 0, eq(3) va donc cibler le 4ème <h1>.	... ... // celle-ci ... ...
$('h1:contains("Test")')	Sélectionne les balises h1 seulement si elles contiennent le mot "Test"	un truc le Test n°774 // celle-ci des bidules
$('#truc').parent()	Permet d'accéder à la balise "parente" de la balise ciblée. Par parente on entend la balise qui contient la balise id="truc"	// la div au complet <h1>...</h1> <h1 id="truc">...</h1> <h1>...</h1>
$('#truc').find('p')	Permet de chercher un type de balise à l'intérieur d'une autre...	 ... blabla // ça blabla // et ça ...
$('#truc').next()	Sélectionne la balise suivante, après celle ciblée... en restant au même niveau.	// la div au complet <h1>...</h1> <h1 id="truc">...</h1> <h1>...</h1> // celle-ci
$('#truc').prev()	Sélectionne la balise précédente, avant celle ciblée... en restant au même niveau.	// la div au complet <h1>...</h1> // celle-ci <h1 id="truc">...</h1> <h1>...</h1>

Il y en a bien d'autres, ceux-là sont déjà très utiles.
Pour plus d'infos, rendez-vous sur la page officielle dédiée :
http://api.jquery.com/category/selectors/

// La syntaxe de JQuery

Une fois la cible sélectionnée, on va pouvoir lui appliquer les fonctions proposées par Jquery, comme par ex :

```
$('#truc').hide();
```

On articule donc la cible et la fonction avec un point entre les 2.

Il est possible de cumuler plusieurs fonctions sur une seule cible (avec plusieurs points, du coup), comme suit :

```
$('#truc').hide().fadeIn('slow');
// On cache #truc, et on le fait réapparaitre aussi sec en fondu.
```

Pour toute les fonction qui ne sont pas instantanées et prennent du temps (animations, chargement de contenus externes, etc) il est possible d'indiquer ce qu'on veut faire après que la présente fonction ait fini d'agir...
On peut ainsi créer des séquences de codes successifs très facilement :

```
$('#truc').fadeOut('slow', function(){ // disparition du bloc, puis...
	$('#truc').load('actu,html', function(){ // chargement d'une actu externe, puis...
		$('#truc').show('slow', function(){ // réapparition du bloc mis à jour, puis...
			alert('la séquence est finie !'); // affichage d'une boite de dialogue
		});
	});
});
```

Il suffit donc d'indiquer une fonction dans la fonction, et ça se fait tout seul.
Attention à avoir le bon nombre de }); à la fin !

Il y a une autre façon de séquencer son code dans le temps, ou simplement quand on veut attendre un certain temps avant d'exécuter une instruction :

```
$('#truc').delay(2000).hide();
// On va donc attendre 2000 millisecondes (2 secondes) avant de cacher le bloc...
```

Pour finir, un sélecteur peut renvoyer aussi bien 1 que 10 éléments, aussi il est parfois pratique de pouvoir parcourir toute la liste des balises sélectionnées, pour leur attribuer une action différente par ex.

```
$('.box').each(function(index){ // on va répéter la suite pour chaque élément, 1 par 1

	// $(this) c'est l'élément actuellement sélectionné, avant de passer au suivant
	$(this).show();
	// index c'est le numéro de cet élément dans la liste : 0 pour le 1er, 1 pour le 2ème...
	$(this).html("Ceci est la div de la classe box numéro "+index);
});
```

// Les fonctions et les familles usuelles de JQuery

Je ne prétend pas toutes les présenter, il y en a bien trop !
En voici une sélection déjà bien étendue, et pour en savoir plus :
http://docs.jquery.com/Main_Page
Le moteur de recherche de cette page est plutôt efficace, et chaque fonction a sa fiche explicative avec exemple.

Afficher/masquer un bloc

Fonctions JQuery	Description	Paramètres

.hide()	Masque un élément, via le css : display:none;	(); // instantanné
		('slow'); // lent
		('fast'); // rapide
		(2000); // dure 2 sec.

.show()	Montre un élément masqué en css ou en JQuery via hide()	// idem hide()
.toggle()	si visible -> invisible, si invisible -> visible	// idem hide()
.fadeIn()	fondu d'alpha entrant, c'est à dire pour faire apparaître un bloc	// idem hide()
.fadeOut()	fondu d'alpha sortant, c'est à dire pour faire disparaître un bloc	// idem hide()
.fadeToggle()	si visible -> invisible, si invisible -> visible	// idem hide()

.fadeTo()	vers un alpha spécifique, entre 0 et 1	(0.5, 'slow'); // vers 50%
		(0, 'slow'); // invisible

.slideDown()	effet volet roulant, pour ouvrir	// idem hide()
.slideUp()	effet volet roulant, pour fermer	// idem hide()
.slideToggle()	si ouvert -> fermé, si fermé -> ouvert	// idem hide()
.stop()	pour arrêter une animation en cours de route	

.scrollTop()	pour récupérer ou définir le scroll vertical de la page.
	c'est le nombre de pixels qui sont masqués en haut de la page.
	ce n'est pas animé.
		(); // renvoie le scroll actuel
		(150); // scrolle jusqu'à 150px

.scrollLeft()	pour récupérer ou définir le scroll horizontal de la page.
	c'est le nombre de pixels qui sont masqués en haut de la page.
	ce n'est pas animé.
		(); // renvoie le scroll actuel
		(150); // scrolle jusqu'à 150px

Styles et attributs d'un bloc

Fonctions JQuery	Description	Paramètres

.css()	permet de consulter ou modifier les attributs CSS de la cible	({'color':'#FF0000'}); // texte en rouge
		({'color':'#ff0000', 'font-size':'14px'});
		// texte en rouge, typo à 14px
		('color'); // renvoie la couleur actuelle

.animate()	permet de consulter ou modifier les attributs CSS de la cible... en interpolation sur une certaine durée !	({font-size: '14px', width: '300px'}, 500); // va prendre 0.5 sec pour aller du style actuel aux nouvelles valeurs
.addClass()	ajoute une classe CSS au bloc sélectionné. aucun pb pour avoir plusieurs classes en même temps.	('test'); // va ajouter test dans l'attribut class="..."
.removeClass()	retire une classe CSS de l'élément ciblé, si elle existait.	('test'); // le bloc n'a plus la classe test dans son attribut class="..."
.hasClass()	renvoie true si le bloc est bien associé à la classe CSS, false sinon	('test'); // regarde si le bloc à la classe test ou pas
.attr()	renvoie n'importe quel attribut HTML du bloc cible	sur <p id="truc"> $('p').attr('id'); renvoit "truc" sur <img src="truc.gif"> $('img').attr('src'); renvoit "truc.gif"
.width()	renvoie la largeur du bloc, en pixels spécifie la largeur du bloc en pixels	(); (300); // pour avoir un bloc de 300px
.height()	renvoie la hauteur du bloc, en pixels spécifie la hauteur du bloc en pixels	(); (300); // pour avoir un bloc de 300px

.offset()	renvoie ou spécifie les coordonnées x et y du bloc, en pixels.
	.left c'est x, .top c'est y
		().left; // renvoie x
		().top // renvoie y
		({top: 10, left: 30}); // pour placer le bloc à x=10 et y=30 pixels

Gestion des contenus textes

Fonctions JQuery	Description	Paramètres

.html()	renvoie ou spécifie le contenu HTML de la balise ciblée
	écrase le contenu qui existait auparavant dans le bloc
		(); // renvoie le contenu
		('blablabla'); // remplace le contenu par "blabla"

.append()	ajoute du contenu HTML à la fin du contenu existant	('blabla'); // va ajouter blabla à la fin du texte existant
.prepend()	ajoute du contenu HTML au début du contenu existant	('blabla'); // va ajouter blabla au début du texte existant
.remove()	supprime la balise sélectionnée et son contenu	
.empty()	supprime le contenu de la balise sélectionnée, mais pas la balise elle-même	

Charger de l'HTML et du Javascript externes

Fonctions JQuery	Description	Paramètres
.load()	récupère le code du document indiqué (html, php, …) et l'injecte directement dans le bloc ciblé... écrase le contenu qui existait auparavant dans le bloc	('actu.html'); // va injecter actu.html directement dans la balise ciblée...

.get()	récupère le code du document indiqué (html, php, …) après on en fait ce qu'on veut
	on peut ajouter des paramètres GET si on s'adresse à du php
	vous récupérerez le code html généré en php, et non le code php lui-même.
		('actu,html', function(data){
		// data c'est le code en retour,
		// à utiliser entre les 2 accolades
		});

.post()	récupère le code du document indiqué (html, php, …) après on en fait ce qu'on veut
	on peut ajouter des paramètres POST si on s'adresse à du php
	vous récupérerez le code html généré en php, et non le code php lui-même.
		('actu,html', function(data){
		// data c'est le code en retour,
		// à utiliser entre les 2 accolades
		});

.getScript()	pour charger via Javascript du code Javascript, et l'exécuter aussitôt ! évite d'avoir à insérer une balise <script src="...">, du coup.	$.getScript("js/monscript.js"); // va importer monscript.js et l'interpréter

Gestion des formulaires

Fonctions JQuery	Description	Paramètres

.val()	renvoie ou spécifie la valeur du champ de formulaire input, textarea, select, etc	pour <input id="test" name="test" type="text" value="blabla" />
		$('#test').val(); // renvoit "blabla"
		$('#test').val('blibli'); // change le contenu de value="..." visible à l'écran

[name="lenom"]	un sélecteur qui permet de cibler un champ <input> via son attribut name="..."	<input name="test" type="text" value="blabla" /> sera ciblée par $('input[name="test"]')
:checked	un sélecteur pour obtenir seulement les checkbox qui ont été cochées	$('input:checked') cible seulement les input où la case a été cochée
:selected	un sélecteur pour obtenir seulement la balise sélectionnée dans un champ list/menu (champ déroulant, quoi)	pour <select name="choix"> <option value="texte1"> choix 1 </option> <option value="texte2"> choix 2 </option> </select> Le choix sélectionné est accessible avec $('select:selected')
.submit()	permet de valider un formulaire sans cliquer sur le bouton submit	http://api.jquery.com/submit/

Interactivité à la souris

Fonctions JQuery	Description	Paramètres
.click()	permet d'indiquer ce qu'on veut faire quand un clic est constaté sur l'élément ciblé	$('#truc').click(function(){ $(this).hide('slow'); // $(this) c'est ici comme $('#truc') });
.mouseover()	permet d'indiquer ce qu'on veut faire quand la souris entre en survol de l'élément ciblé	$('#truc').mouseover(function(){ $(this).hide('slow'); // $(this) c'est ici comme $('#truc') });
.mouseout()	permet d'indiquer ce qu'on veut faire quand la souris sort du survol de l'élément ciblé	$('#truc').mouseout(function(){ $(this).hide('slow'); // $(this) c'est ici comme $('#truc') });
.mousedown()	permet d'indiquer ce qu'on veut faire au moment où le clic est enfoncé sur l'élément ciblé équivalent à onPress en ActionScript	$('#truc').mousedown(function(){ $(this).hide('slow'); // $(this) c'est ici comme $('#truc') });
.mouseup()	permet d'indiquer ce qu'on veut faire au moment où le clic est relâché sur l'élément ciblé équivalent à onRelease en ActionScript ne cohabite pas bien avec .click()	$('#truc').mouseup(function(){ $(this).hide('slow'); // $(this) c'est ici comme $('#truc') });
.bind()	permet de récupérer bien d'autres évènements : touch des écrans tactiles, multitouch, etc http://api.jquery.com/bind/	tous les events souris supportés : http://api.jquery.com/category/events/mouse-events/

Interactivité au clavier

Fonctions JQuery	Description	Paramètres

.keydown()	exécute la fonction indiquée au moment ou la touche est enfoncée.
	le code s'exécute une fois, même si on garde la touche enfoncée.
		$(window).keydown(function(){
		// le code Javascript / JQuery
		// à exécuter
		});

.keypress()	exécute la fonction indiquée au moment ou la touche est enfoncée.
	le code est répété ensuite tant que la touche est enfoncée.
		$(window).keypress(function() {
		// le code Javascript / JQuery
		// à exécuter
		});

.keyup()	exécute la fonction indiquée au moment ou la touche est relâchée.	$(window).keyup(function() {
		// le code Javascript / JQuery
		// à exécuter
		});

.bind()	permet de récupérer d'autres évènements http://api.jquery.com/bind/	tous les events clavier supportés : http://api.jquery.com/category/events/keyboard-events/

Evénements de la page

Fonctions JQuery	Description	Paramètres

.ready()	exécute la fonction indiquée dès que la page est chargée	$(document).ready(function(){
		  // le code Javascript / JQuery
		  // à exécuter
		});

.resize()	exécute la fonction indiquée à chaque fois que la page est redimensionée	$(window).resize(function() {
		  // le code Javascript / JQuery
		  // à exécuter
		});

.bind()	permet de récupérer d'autres évènements : scroll, erreurs de code, … http://api.jquery.com/bind/	tous les events supportés : http://api.jquery.com/category/events/browser-events/ http://api.jquery.com/category/events/document-loading/

L'ensemble du langage JQuery est référencé en détail ici :
http://docs.jquery.com/Main_Page

et il y a de quoi faire...

// Quelques librairies choisies

La grande force de JQuery (outre ses fonctions déjà très puissantes), c'est la très grande communauté de développeurs qui publient des extensions dédiées à certaines fonctionnalités compliquées.
L'extension est le plus souvent un fichier .js, parfois accompagné d'images, de css, ou de pages php, qu'il suffit d'importer dans votre page via une balise
<script language="javascript" src="js/monextension.js"></script>

Une fois cette balise saisie, vous aurez accès à de nouvelles fonctions propres à votre extension, qui viennent enrichir celles de JQuery avec la syntaxe JQuery habituelle.
Le plus souvent, les sites des librairies proposent des demos (dont vous pouvez lire le code via CTRL(Pomme) + U), voire des tutoriaux.

Vous trouverez des extensions très facilement sur google, en tapant
"jquery ce que je veux faire"
ex: "jquery diaporama", "jquery scroll animé", etc...

Le problème c'est qu'il y en a souvent des dizaines pour chaque sujet, alors voici une petite sélection de ce que je connais de mieux :

Gestion précise et animée du scroll (de la page, d'une div), en ciblant simplement une balise, ou en entrant des valeurs en pixels :
http://demos.flesler.com/jquery/scrollTo/

Personnaliser ses barres de scroll :
http://jscrollpane.kelvinluck.com/index.html#examples

Diaporamas de bon goût :
http://www.serie3.info/s3slider/demonstration.html

Diaporama pleine-page, avec ou sans boutons de navigation, et plusieurs transitions possibles d'une image à l'autre :
http://buildinternet.com/project/supersized/slideshow/3.2/demo.html

Une "lightbox" sobre et fonctionnelle pour afficher vos images en grand format au-dessus de votre page, au clic sur la miniature :
http://fancyapps.com/fancybox/#examples

Une grille qui s'adapte à la taille de la fenêtre, avec des replacements animés :
http://masonry.desandro.com/

Pour des sites scrollables et animés dans tous les sens :
http://johnpolacek.github.com/scrollorama/
http://johnpolacek.github.com/scrolldeck.js/decks/responsive/

// Quelques librairies avancées

Toute la typo qu'on veut en Javascript/css :
http://typeface.neocracy.org/

Injecter des animations Flash via JQuery :
http://jquery.lukelutman.com/plugins/flash/index.html

Une grille adaptative, avec options de tri multiples (niveau avancé) :
http://isotope.metafizzy.co/index.html

Pixelliser des images en Javascript :
http://desandro.com/resources/close-pixelate/

Formulaire d'upload de fichiers via un module Jquery :
http://www.uploadify.com/demos/

Et pour finir, pour les plus avancés, une sélection de librairies dédiées au Responsive WebDesign, autrement dit pour construire des sites qui s'adaptent quelle que soit la résolution d'écran :

http://www.webresourcesdepot.com/28-high-quality-jquery-plugins-for-building-responsive-websites/

http://cssgrid.net/

Et si vous en êtes là, c'est qu'il ne reste plus qu'à vous intéresser à l'HTML5...
http://html5doctor.com/article-archive/

...voire à Processing.js, qui est une librairie permettant de faire de l'HTML5 avec les mots de Processing :
http://processingjs.org/
http://www.gildasp.fr/playingwithpixels/
http://processing-js.github.com/processing-mobile/examples/processing-and-js/

Gildas P.
// Creative + code

WebDesigner
Graphiste Multimedia
Programmeur

Tél/ 06.31.15.13.04
Email/ contact@gildasp.fr
Web/ www.gildasp.fr

http://www.gildasp.fr
contact@gildasp.fr

Pour
http://www.licencepromultimedia.free.fr