Fonction jQuery

C'est le point d'entrée de la bibliothèque jQuery. Vous pouvez utiliser au choix l'instruction `jQuery()` ou son alias `$()`. Dans ce cours, nous utiliserons systématiquement l'alias pour limiter l'écriture.

Méthodes jQuery

La bibliothèque jQuery est constituée d'un ensemble de blocs de code autonomes appelés méthodes. Ce qui fait la puissance de cette bibliothèque, c'est avant tout la grande diversité des méthodes proposées. Pour exécuter une méthode jQuery, il suffit de préciser son nom à la suite d'un sélecteur en le séparant de ce dernier par un point : `$(sélecteur).méthode(paramètres);`.

Objet jQuery

On appelle « objet jQuery » l'entité retournée par la fonction jQuery, c'est-à-dire par `$()`. Cet objet représente un ensemble de zéro, un ou plusieurs éléments issus du document.

Le squelette HTML typique

Nous allons voir le squelette typique d'un document HTML dans lequel on utilise du code jQuery. C'est à partir de ce squelette que tous les exemples de ce tutoriel seront construits. Vous trouverez ici toutes les informations nécessaires pour comprendre son agencement.

En fait, ce n'est pas un mais deux squelettes dont nous allons parler ici : un concernant le développement et un autre concernant la production. Une seule balise HTML les différencie, mais, comme nous allons le voir, cette balise fait une grande différence.

Squelette HTML en phase de développement

Voici les instructions qui seront typiquement utilisées dans tous nos exemples :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Le titre du document</title>
    <link rel="stylesheet" type="text/css" href="feuille-de-styles.css">
  </head>
  <body>
    <!-- Une ou plusieurs balises HTML pour définir le contenu du document -->
    <script src="jquery.js"></script>
    <script src="mon-script.js"></script>
  </body>
</html>

```

Je ne vais pas m'arrêter sur chacune de ces lignes, vous devriez être en mesure de les comprendre. Je vais cependant détailler un peu les lignes 10 et 11.

La ligne 10 fait référence à la bibliothèque `jquery.js`. Ici également, étant donné qu'aucun chemin n'est spécifié dans la valeur affectée à l'attribut `src`, le fichier `jquery.js` doit se trouver dans le même dossier que le document HTML. Je ne vais pas revenir sur comment le télécharger, reportez-vous au premier chapitre si vous avez besoin.

La ligne 11 fait référence à un fichier JavaScript externe à votre page dans lequel vous placerez tout le code jQuery en rapport avec le document.

Au final, on se retrouve donc avec quatre fichiers :

1. Le document HTML lui-même ;
2. La feuille de styles externe qui définit la mise en forme du document ;
3. Le fichier `jquery.js`, qui contient la bibliothèque jQuery ;
4. Un fichier JavaScript externe qui contient tout le code jQuery que vous développerez.

Quatre fichiers pour mettre en place quelques instructions jQuery ? N'est-ce pas un peu disproportionné ?

Eh bien… tout dépend si vous voulez « bricoler » ou si vous voulez programmer en jQuery. Effectivement, la tentation est grande de placer le code CSS et jQuery dans le document HTML, afin de n'utiliser que deux fichiers : le document HTML et la bibliothèque jQuery. D'autant plus que cela fonctionne. Mais imaginez que vous deviez réaliser un site Web contenant plusieurs dizaines de pages. En externalisant les codes CSS et jQuery, vous pourrez facilement les réutiliser et ainsi être gagnants sur plusieurs tableaux :

- Le temps de développement sera réduit d'autant ;
- Les risques d'erreurs seront minimisés, puisque le même code sera utilisé dans les différents documents HTML ;
- La maintenance sera d'autant facilitée : pour effectuer une modification dans tout le site, il vous suffira d'agir sur la feuille de styles externe et sur le fichier de code JavaScript externe. Si vous aviez internalisé le code de ces deux fichiers, vous auriez dû agir sur chacune des pages HTML du site !

Squelette HTML en phase de production

Ça y est, votre code HTML/CSS/jQuery/JavaScript fonctionne à la perfection et vous êtes impatients de le tester en ligne. Avant d'utiliser un client FTP, assurez-vous que vous faites référence à la version minimisée de la bibliothèque jQuery et que vous la chargez sur un CDN. Ainsi, avec la version minimisée sur le CDN de Google, voici le code à utiliser :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Le titre du document</title>
    <link rel="stylesheet" type="text/css" href="feuille-de-styles.css">
  </head>
  <body>
    <!-- Une ou plusieurs balises HTML pour définir le contenu du document -->
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
    <script src="mon-script.js"></script>
  </body>
</html>

```

Optimisation du code

Voici quelques recommandations qui augmenteront les performances des pages qui contiennent du code jQuery :

1. Invoquez la bibliothèque `jquery.js` à la fin du document, avant la balise `</body>`.
2. Insérez le code jQuery rattaché à la page dans un fichier annexe et faites référence à ce fichier juste après la balise qui indique où se trouve la bibliothèque jQuery.

Ces positions stratégiques optimisent le temps de chargement de la page. En effet, de nombreux navigateurs bloquent l'exécution du code qui suit une balise `<script>` jusqu'à ce que cette balise ait été chargée et exécutée. En plaçant les deux balises `<script>` juste avant la balise `</body>`, l'affichage de la page n'est pas « freiné » par le code jQuery.

Attendre la disponibilité du DOM

Le langage jQuery est utilisé pour manipuler (en lecture et en écriture) le DOM, c'est-à-dire l'arborescence du document. Imaginez que ces manipulations commencent alors que l'arbre n'a pas encore été entièrement obtenu. Cette situation erratique pourrait désorganiser l'affichage, produire des erreurs, voire même… bloquer le navigateur !

Pour éviter d'en arriver là, vous devez attendre que l'arbre soit complet. En jQuery, cela se traduit par le code suivant :

```javascript
jQuery(document).ready(function() {
  // Ici, le DOM est entièrement défini
});

```

Cette écriture peut se simplifier en remplaçant `jQuery` par son alias, `$`, ce qui donne :

```javascript
$(document).ready(function() {
  // Ici, le DOM est entièrement défini
});

```

Enfin, `(document).ready` peut être omis pour arriver au code suivant :

```javascript
$(function() {
  // Ici, le DOM est entièrement défini
});
```

Ces trois instructions sont strictement équivalentes. En ce qui me concerne, j'utiliserai systématiquement la troisième tout au long de ce cours, car c'est la plus courte et la plus simple à écrire. Par contre, si vous consultez d'autres articles traitant de jQuery, vous trouverez parfois les deux autres formes.

Dans l'introduction, il a été dit que jQuery peut être utilisé pour interroger/modifier le DOM, mais aussi les styles CSS du document. L'instruction `$(function() {` s'assure que le DOM est entièrement défini. Mais qu'en est-il de la feuille de styles ?

Si vous avez fait référence à une feuille de styles entre les balises `<head>` et `</head>`, l'instruction `$(function() {` s'assure également que la feuille de styles est chargée. Vous pouvez donc l'interroger et la manipuler comme bon vous semble.

Premier script : « Hello world »

Il est une tradition bien ancrée lorsque l'on écrit un cours sur un langage de programmation : le premier exemple affiche un texte sur l'écran : « Hello world » ! Nous n'allons pas déroger à la règle. Prêts à relever le défi ?

Précédemment, j'ai plusieurs fois affirmé que jQuery était capable de manipuler les éléments du DOM. Eh bien maintenant, il est temps de le prouver. Pour cela, je vais définir une balise `<span>` dans le document, lui affecter un attribut `id` afin de l'identifier et modifier son contenu en jQuery.

hello-world.html :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World</title>
  </head>
  <body>
    Ce texte est affiché en HTML

    <span id="texteJQ"></span>
    <script src="jquery.js"></script>
    <script src="jq-hello-world.js"></script>
  </body>
</html>
```

Comme vous pouvez le constater, la ligne contenant une balise `<link>` a été enlevée de l'en-tête. En effet, l'utilisation d'une feuille de styles ne présente aucun intérêt dans cet exemple.
La ligne 12 fait référence au fichier JavaScript `jq-hello-world.js`. En voici le contenu :

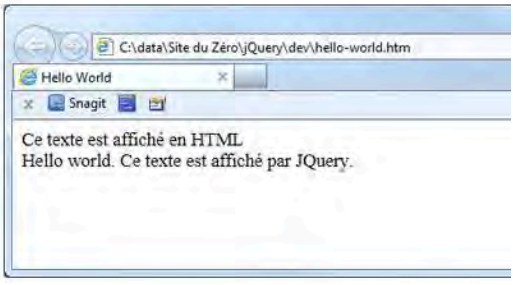
```javascript
$(function() {
  $('#texteJQ').html('Hello world. Ce texte est affiché par jQuery.');
});
```

Si vous n'avez pas la mémoire trop courte, vous reconnaissez certainement les lignes 1 et 3 qui attendent la disponibilité du DOM. La ligne 2, quant à elle, contient une instruction jQuery dont la syntaxe peut vous laisser perplexes. Décomposons-la pour mieux comprendre son fonctionnement :

- `$('#texteJQ')` : cette partie agit comme un sélecteur. Elle retrouve dans le DOM l'élément dont l'attribut `id` vaut `texteJQ`. Pourquoi l'attribut `id`, me direz-vous ? Eh bien parce que, selon les conventions du langage CSS, le caractère # représente justement l'attribut `id`.
- `html('Hello world. Ce texte est affiché par jQuery.');` : la deuxième partie indique ce qui doit être modifié. Dans cet exemple, on utilise la méthode `html()` pour demander la modification du contenu de la balise.
- Le point entre ces deux parties fait le lien entre le sélecteur et l'action.

Cette instruction sélectionne donc l'élément d'id `texteJQ` et y insère le texte « Hello world. Ce texte est affiché par jQuery. » Vous voyez, il n'y a rien de sorcier là-dedans.

Double-cliquez sur le fichier `hello-world.html` pour l'ouvrir dans votre navigateur. Vous devriez obtenir quelque chose ressemblant à l'image suivante.



SÉLECTION D'ÉLÉMENTS

Dans ce chapitre, vous allez commencer à entrevoir la puissance de jQuery en découvrant une de ses facettes : la sélection d'éléments. Après cette lecture, vous saurez sélectionner, modifier et obtenir la valeur de la plupart des éléments qui peuvent être rencontrés dans un document HTML.

Vous connaîtrez également la nature des objets retournés par les instructions jQuery et vous verrez comment réduire la taille du code jQuery en chaînant les méthodes utilisées sur un objet.

Fonctionnement de base de jQuery

jQuery repose sur une seule et unique fonction : `jQuery()`, ou son alias, `$()`. Cette fonction accepte un ou plusieurs paramètres et retourne un objet que nous appellerons « objet jQuery ». Les paramètres peuvent être d'un des types suivants :

- Une fonction, qui sera exécutée dès que le DOM est disponible. Cette technique est très largement utilisée par tous les programmeurs jQuery.
- Un sélecteur CSS : l'élément (ou les éléments) qui correspondent au sélecteur sont retournés. Nous allons nous intéresser à cette technique dans ce chapitre.
- Un élément HTML, un document ou l'objet `window` : un objet jQuery correspondant à cet élément est retourné.
- Une (ou plusieurs) balise(s) HTML : un objet jQuery correspondant à cette (ces) balise(s) est retourné. Vous pouvez lui appliquer des méthodes jQuery, par exemple pour ajouter cette (ces) balise(s) dans un élément HTML.

Sélection d'éléments

Une des grandes forces de jQuery est d'intégrer la syntaxe des sélecteurs CSS. Par cet intermédiaire, il est élémentaire de sélectionner les nœuds DOM qui nous intéressent, en utilisant la syntaxe `$(sélection)` où `sélection` représente un sélecteur CSS.

En effet, jQuery est fortement lié à trois autres langages : HTML, CSS et JavaScript. CSS est un langage consacré à la mise en forme des documents écrits en HTML. Les sélecteurs CSS sont des mots et symboles qui permettent d'identifier les éléments contenus dans un document HTML. Par exemple, `h2` représente les balises HTML `<h2>`, ou encore `p` représente les balises `<p>`. En utilisant un sélecteur CSS dans la première partie d'une instruction jQuery, le ou les éléments HTML correspondants seront sélectionnés. Par la suite, il suffira de leur appliquer une action pour modifier leur contenu ou leur apparence.

Avant de poursuivre, voici quelques sélecteurs CSS que vous devez avoir en mémoire :

- Un nom de balise, sans les caractères < et >, permet de sélectionner cette balise. Si plusieurs balises de même nom se trouvent dans le document, toutes ces balises sont sélectionnées. Par exemple, si le document contient plusieurs balises `<div>`, le sélecteur CSS `div` sélectionne toutes ces balises.
- Le signe # fait référence à l'attribut `id` (ou identifiant) d'une balise. Par exemple, si vous définissez la balise `<p id="premier">`, le sélecteur `#premier` sélectionne cette balise. Notez que deux balises ne peuvent pas avoir le même identifiant.
- Le point fait référence à l'attribut `class` d'une balise. Supposons que vous ayez défini la balise `<h2 class="rouge">`. Le sélecteur `.rouge` sélectionne cette balise. Plusieurs balises peuvent avoir la même classe. Un même traitement pourra donc être appliqué à ces deux balises.
- Pour différencier les balises `<h2>` de classe `rouge` des balises `<p>` de classe `rouge`, vous utiliserez les sélecteurs `h2.rouge` et `p.rouge`. Ce cas particulier s'applique à toutes les balises et toutes les classes. Ainsi, le sélecteur `nom_balise.nom_classe` permet de sélectionner les balises `nom_balise` de classe `nom_classe`.

Les bases de jQuery

1. Sélection d'éléments
2. Plus loin dans la sélection d'éléments
3. Modifier le contenu d'un élément
4. Insérer et remplacer des éléments dans le DOM
5. TP : Questionnaire interactif en jQuery

Quiz

Activité : Créez un générateur de formulaires

Accéder au forum

Créé par

OPENCLASSROOMS

OpenClassrooms, 1ère plateforme e-Education en Europe

- Supposons maintenant que vous ayez défini une liste à puces `<ul></ul>` et une liste numérotée `<ol></ol>`. Chacun des éléments des deux listes est repéré par des balises `<li>`. Pour différencier les éléments `<li>` de la liste à puces des éléments `<li>` de la liste numérotée, vous utiliserez un « sélecteur descendant ». Ainsi, le sélecteur `ul li` s'adresse à tous les éléments `<li>` de la liste à puces `<ul>`, et le sélecteur `ol li` s'adresse à tous les éléments `<li>` de la liste numérotée `<ol>`.

- Certaines balises HTML peuvent contenir un ou plusieurs attributs. Par exemple, la balise `<img src="chien.jpg" width="40" height="30">` contient trois attributs : `src`, `width` et `height`. Pour sélectionner toutes les balises qui contiennent un attribut `src`, vous utiliserez le sélecteur `[src]`.

- Vous pouvez même aller plus loin en sélectionnant les balises dont un attribut a une certaine valeur. Par exemple, pour sélectionner toutes les balises dont l'attribut `width` a pour valeur 40, vous utiliserez le sélecteur `[width="40"]`.

- Le caractère * représente toutes les balises du document.

Et maintenant, nous allons raisonner sur un exemple pour mieux comprendre comment utiliser les sélecteurs CSS.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <ul class="rouge">
      <li class="impair">premier élément de la liste à puces</li>
      <li class="pair">deuxième élément de la liste à puces</li>
      <li class="impair">troisième élément de la liste à puces</li>
    </ul>
    <div>
      <ul class="bleu">
        <li class="impair">premier élément de la liste à puces</li>
        <li class="pair">deuxième élément de la liste à puces</li>
        <li class="impair">troisième élément de la liste à puces</li>
      </ul>
    </div>
    <ol class="rouge">
      <li>premier élément de la liste numérotée</li>
      <li>deuxième élément de la liste numérotée</li>
      <li>troisième élément de la liste numérotée</li>
    </ol>
    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Insérer le code jQuery ici
      });
      </script>
  </body>
</html>
```

Le tableau suivant donne quelques exemples significatifs de sélecteurs et indique à quoi ils correspondent.

Sélecteur CSS	Sélecteur jQuery	Signification
`ul`	`$('ul')`	Les balises `<ul>`
ul.bleu	`$('ul.bleu')`	La balise `<ul>` de classe `bleu`
`div ul`	`$('div ul')`	La balise `<ul>` contenue dans la balise `<div>`
`div ul li[class="pair"]`	`$('div ul li[class="pair"]')`	La balise `<li>` contenue dans une balise `<ul>`, elle-même contenue dans une balise `<div>`, et dont l'attribut `class` vaut `pair`
`li[class]`	`$('li[class]')`	Les balises `<li>` qui possèdent un attribut `class`
`li[class="impair"]`	`$('li[class="impair"]')`	Les balises `<li>` qui possèdent un attribut `class` de valeur impair
`*`	`$('*')`	Toutes les balises du document

Vous avez peut-être remarqué l'utilisation des guillemets dans la colonne « Sélecteur jQuery » : `$('div ul li[class="pair"]')` et `$('li[class="impair"]')`. Ceci vient du fait qu'il est impossible d'utiliser des apostrophes à l'intérieur d'autres apostrophes. Lorsqu'un tel cas se produit, les apostrophes les plus internes sont remplacées par des guillemets.

Quelle est la nature de l'objet retourné ?

Le résultat retourné par la fonction `$()` est un objet jQuery. Cet objet ressemble à un tableau : il a une propriété `length` et les éléments sélectionnés peuvent être accédés par un indice. Par exemple :

- `$('a').length` retourne le nombre de liens hypertextes contenus dans la page.
- `$('ul.bleu').length` retourne le nombre de balises `<ul>` de classe `bleu`.
- `$('li[class="impair"]').length` retourne le nombre de balises `<li>` qui ont un attribut `class` de valeur impair.
- `$('body').length` retourne « 1 » car le document contient une seule balise `<body>`.

Pour accéder à un des éléments sélectionnés, précisez son indice entre crochets à la suite du sélecteur. Par exemple :

- `$('a')[0]` retourne le premier lien hypertexte de la page.
- `$('ul.bleu')[3]` retourne la quatrième balise `<ul>` de classe `bleu`.
- `$('body')[0]` est équivalent à `document.body`.

Appliquer une méthode à la sélection

Une fois qu'un ou plusieurs éléments ont été sélectionnés avec une instruction `$(sélecteur)`, vous pouvez leur appliquer un traitement en exécutant une méthode jQuery. Pour cela, ajoutez un point après la parenthèse fermante et indiquez la méthode à utiliser :

javascript
```
$(sélecteur).action
```

… où `$(sélecteur)` sélectionne un ou plusieurs éléments dans le document et `action` effectue un traitement sur les éléments sélectionnés. Par exemple, pour écrire un message dans une balise `<span>` d'identifiant `resultat`, vous utiliserez quelque chose comme le code suivant :

javascript

```javascript
$('#resultat').html('texte à écrire dans la balise span');
```



La méthode `html()` n'est qu'une des nombreuses méthodes utilisables en jQuery. Pour avoir un aperçu global des principales méthodes, reportez-vous à la section « Getters et setters » du chapitre « Modifier le contenu d'un élément ».

Notions indispensables

Je vais ici vous parler de deux notions indispensables : les fonctions de rappel et le chaînage de méthodes.

Fonctions de rappel

Une fonction de rappel (ou *callback* en anglais) est une fonction exécutée lorsqu'une autre fonction a terminé de s'exécuter. En jQuery, les fonctions de rappel sont essentiellement utilisées pour réaliser des animations et des appels AJAX. Nous reviendrons sur ces deux sujets dans la partie 3 du cours.

Juste pour vous mettre l'eau à la bouche, voici un exemple de fonction de rappel, appliquée aux éléments de classe `rouge` :

javascript
```javascript
$(function() {
    $(".rouge").fadeOut("slow",function(){
      $(this).fadeIn("slow");
    });
});
```

Ce code fait disparaître puis réapparaître progressivement les balises de classe `rouge`.

Chaînage de méthodes

Le chaînage est un concept très puissant et pourtant simple à comprendre. Étant donné que les méthodes jQuery retournent un objet jQuery, il est possible de les « chaîner », c'est-à-dire de les exécuter les unes à la suite des autres.

À titre d'exemple, les deux premières instructions sont équivalentes à la troisième :

javascript
```javascript
$('.rouge').css('background','red');
$('.rouge').css('color','yellow');

$('.rouge').css('background','red').css('color','yellow');
```

Nous n'avons pas encore étudié la méthode jQuery `css()`, mais je suis sûr qu'en observant la façon dont elle s'articule vous comprenez son fonctionnement. Cette méthode admet deux paramètres. Le premier est une propriété CSS et le deuxième, la valeur à affecter à cette propriété.

Ainsi par exemple, l'instruction `$('.rouge').css('background','red');` affecte un arrière-plan (`background`) de couleur rouge (`red`) aux balises de classe `rouge` (`$('.rouge')`). L'instruction `$('.rouge').css('color','yellow');`, quant à elle, affecte la couleur jaune (`yellow`) aux balises de classe `rouge`.

En chaînant ces deux instructions, les balises de classe `rouge` ont un arrière-plan de couleur rouge et des caractères de couleur jaune. Et tout cela en une seule instruction !



La méthode `css()` n'est qu'une des nombreuses méthodes utilisables en jQuery. Pour avoir un aperçu global des principales méthodes, reportez-vous à la section « Getters et setters » du chapitre « Modifier le contenu d'un élément ».

- Le sélecteur jQuery `$('sel')` utilise la syntaxe CSS. En remplaçant `sel` par un sélecteur CSS quelconque, les éléments correspondants (s'ils existent) seront sélectionnés dans le DOM. Par exemple :
    - `$('a')` sélectionne tous les liens hypertextes ;
    - `$('.rouge')` sélectionne les éléments de classe `rouge` ;
    - `$('#e2')` sélectionne l'élément d'identifiant `e2` ;
    - `$('[src]')` sélectionne tous les éléments qui possèdent un attribut `src` ;
    - `$('[width="40"]')` sélectionne tous les éléments qui ont un attribut `width` égal à 40.

- L'objet retourné par un sélecteur jQuery peut faire référence à plusieurs éléments. Pour accéder à un élément, vous pouvez préciser son index entre crochets, en ayant bien à l'esprit que le premier élément a un index égal à 0. Par exemple, le sélecteur `$('a')[2]` fait référence au troisième lien hypertexte contenu dans la page.

- Pour appliquer une méthode à un objet jQuery obtenu en utilisant un sélecteur, il suffit d'écrire cette méthode à droite du sélecteur en la séparant de ce dernier par un point. Par exemple, l'instruction suivante écrit en jaune tous les liens hypertextes contenus dans la page : `$('a').css('color', 'yellow');`.

- Pour chaîner deux méthodes jQuery, écrivez-les l'une à la suite de l'autre en les séparant par un point. Par exemple, cette instruction applique un arrière-plan de couleur rouge et un texte de couleur jaune aux éléments de classe `rouge` : `$('.rouge').css('background','red').css('color','yellow');`.

PLUS LOIN DANS LA SÉLECTION D'ÉLÉMENTS

Vous avez appris à sélectionner des balises, des identifiants et des classes dans le code HTML d'une page Web. Ces types de sélections sont très utiles et vous les utiliserez fréquemment lors de vos développements jQuery.

Cependant, il est possible d'aller plus loin en sélectionnant encore plus finement les éléments du DOM. Ce chapitre va vous montrer comment sélectionner des éléments HTML en fonction de leurs attributs, comment limiter les éléments retournés en utilisant des pseudo-sélecteurs ou encore comment utiliser des sélecteurs spécialisés pour certains types d'éléments.

Vous apprendrez également à parcourir les éléments sélectionnés pour appliquer à chacun d'entre eux un traitement spécifique. Cette technique est très importante. Elle sera souvent utilisée dans les autres chapitres de ce cours.

Sélecteurs d'attributs

Arrivés à ce point dans la lecture du cours, vous savez sélectionner les éléments qui contiennent :

- un attribut donné, en utilisant le sélecteur `$('[nom]')` ;

- un attribut donné qui a une certaine valeur, en utilisant le sélecteur `$('[nom:valeur]')`.

Le tableau suivant dresse la liste des sélecteurs d'attributs évolués auxquels nous allons nous intéresser.

Sélecteur	Éléments sélectionnés
`['nom*="valeur"']`	Éléments qui possèdent un attribut `nom` qui contient (partiellement ou totalement) la valeur spécifiée.
`['nom~="valeur"']`	Éléments qui possèdent un attribut `nom` qui contient la valeur spécifiée, délimité par des espaces.
`['nom$="valeur"']`	Éléments qui possèdent un attribut `nom` qui se termine par la valeur spécifiée.
`['nom!="valeur"']`	Éléments qui ne possèdent pas l'attribut `nom`, ou qui possèdent un attribut `nom` différent de la valeur spécifiée.
`['nom^="valeur"']`	Éléments qui possèdent un attribut `nom` qui commence par la valeur spécifiée.

Les bases de jQuery

1. Sélection d'éléments
2. **Plus loin dans la sélection d'éléments**
3. Modifier le contenu d'un élément
4. Insérer et remplacer des éléments dans le DOM
5. TP : Questionnaire interactif en jQuery

Quiz

Activité : Créez un générateur de formulaires

Accéder au forum

Créé par

OPENCLASSROOMS

OpenClassrooms, 1ère plateforme e-Education en Europe

Pour bien comprendre comment fonctionnent ces sélecteurs, nous allons travailler avec le code suivant :

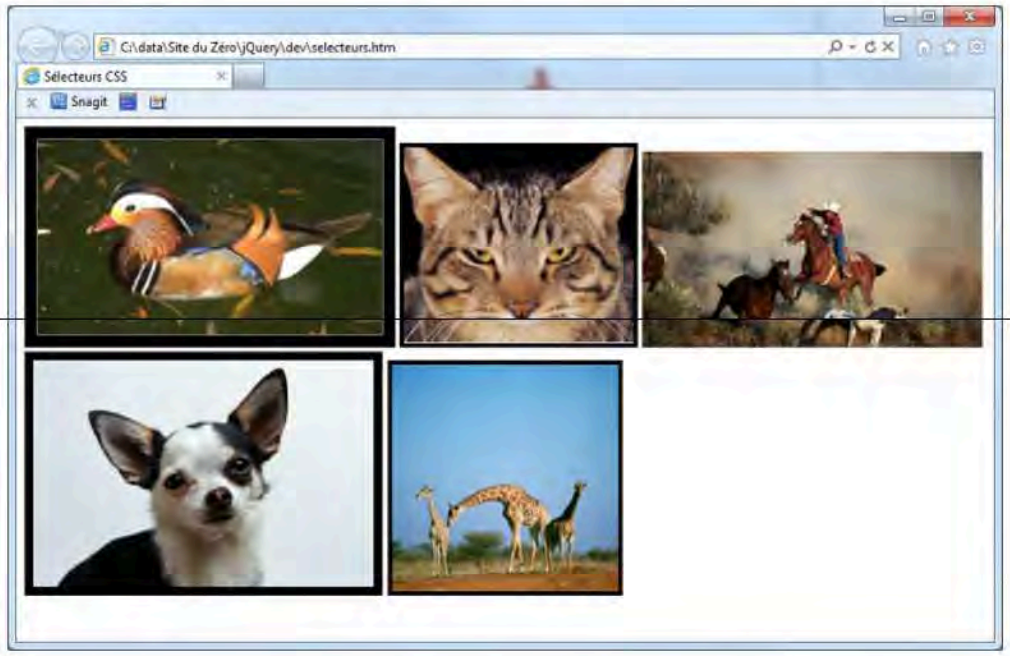
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <img src="canard.jpg" title="animal canard" border="11">
    <img src="chat.jpg" title="animal chat" border="4">
    <img src="cheval.jpg" title="cheval" border="2">
    <img src="chien.jpg" title="animal chien"  border="8">
    <img src="girafe.jpg" title="girafe" border="4">

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Le code jQuery sera inséré ici
      });
    </script>
  </body>
</html>
```

Au début de ce cours, je vous ai dit qu'il était conseillé de séparer les codes HTML, CSS et jQuery dans plusieurs fichiers. Ce conseil reste toujours d'actualité ; cependant, étant donné la faible ampleur du code, j'ai pris la liberté de mettre le code jQuery directement dans le HTML. N'ayez crainte, nous utiliserons plusieurs fichiers séparés lorsque le code deviendra un peu plus étoffé.

Le corps du document contient cinq balises `<img>`. Chacune d'entre elles a trois attributs : `src`, `title` et `border`. Nous allons utiliser plusieurs sélecteurs d'attribut évolués pour modifier la couleur de certaines bordures d'images. Après que les images ont été définies, une balise `<script>` fait référence à la bibliothèque jQuery et une autre délimite le code jQuery que nous allons écrire. L'instruction `$(function() {` attend la disponibilité du DOM. Les instructions jQuery seront placées à la place du commentaire.

L'image suivante est le résultat lorsqu'on exécute ce code dans Internet Explorer.



Le code exécuté dans Internet Explorer

Nous allons tracer une bordure rouge autour de certaines images en utilisant la méthode jQuery `css()` suivante :

```javascript
$('selecteur').css('border-color','red');
```

... où `selecteur` est un des sélecteur CSS du tableau suivant :

Critère	Sélecteur	Image modifiée
Image dont l'attribut `border` contient partiellement ou totalement la valeur « 1 ».	`$('[border*="1"]')`	1
Image dont l'attribut `title` contient le mot « animal » délimité par une espace.	`$('[title~="animal"]')`	1, 2, 4
Image dont l'attribut `src` se termine par « e.jpg ».	`$('[src$="e.jpg"]')`	5
Image qui ne possède pas un attribut `border` égal à « 15 ».	`$('[border!="15"]')`	1, 2, 3, 4, 5
Image dont l'attribut `src` commence par « ch ».	`$('[src^="ch"]')`	2, 3, 4

Sélecteurs hiérarchiques

Dans l'arborescence DOM, à l'exception de `html`, tous les éléments ont un parent, et certains éléments ont un ou plusieurs enfants. Cette section s'intéresse aux sélecteurs hiérarchiques, avec lesquels vous pourrez sélectionner les enfants d'un certain parent, l'énième enfant d'un parent, les enfants uniques, etc.

Sélecteur	Éléments sélectionnés
`('p > e')`	Éléments `e` directement descendants d'éléments `p`
`('p + e')`	Éléments `e` directement précédés d'un élément `p`
`('p ~ e')`	Éléments `e` précédés d'un élément `p`
`:empty`	Éléments qui n'ont pas d'enfant
`:first-child`	Premier enfant
`:first`	Premier élément
`:last-child`	Dernier enfant
`:last`	Le dernier élément de la sélection
`:nth-child()`	Élément qui est l'énième enfant de son parent
`:only-child`	Éléments qui sont enfants uniques de leur parent

Rien de tel qu'un peu de code pour bien comprendre comment fonctionnent ces sélecteurs. Ici, nous utiliserons des listes imbriquées :

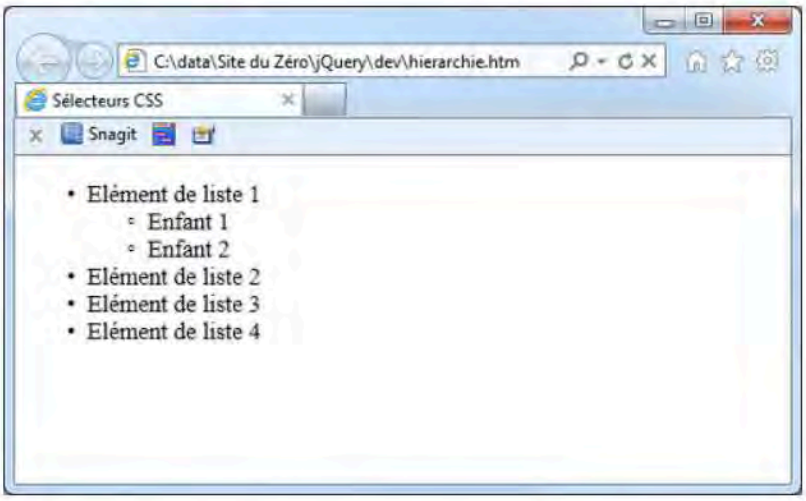
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <div id="listes">
      <ul id="ul1">
        <li> Elément de liste 1
          <ul id="ul2">
            <li> Enfant 1</li>
            <li> Enfant 2</li>
          </ul>
        </li>
        <li> Elément de liste 2</li>
        <li> Elément de liste 3</li>
        <li> Elément de liste 4</li>
      </ul>
    </div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
```

```
    </div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Le code jQuery sera inséré ici
      });
    </script>
  </body>
</html>
```

L'image suivante représente ce code exécuté dans Internet Explorer.



Le code précédent exécuté dans Internet Explorer

Nous allons afficher en rouge certains éléments de ces listes imbriquées en utilisant la méthode jQuery `css()` suivante :

javascript
```
$(function() {
    $('sel').css('color','red');
});
```

... où `sel` est un des sélecteur CSS du tableau suivant :

Critère	Sélecteur	Élément modifié
Éléments `ul` directement descendants d'éléments `li`	`$('li > ul')`	2, 3
Éléments `li` directement précédés d'un élément `li`	`$('li + li')`	3, 4, 5, 6
Premier élément `li` enfant	`$('li:first-child')`	1, 2, 3
Premier élément `li`	`$('li:first')`	1, 2, 3
Dernier élément `li`	`$('li:last')`	6
Dernier élément `li` enfant	`$('li:last-child')`	3, 6
Éléments `li` enfants uniques de leur parent	`$('li:only-child')`	aucun
Deuxième enfant `li`	`$('li:nth-child(2)')`	3, 4

Les sélecteurs $('li:first-child') et $('li:first') donnent le même résultat visuel. Cependant les causes de ces résultats diffèrent :

- Dans le cas de $('li:first-child'), le premier <li> se voit affecter une propriété CSS color initialisée à red. De même pour le premier enfant de ce premier <li>. Par contre, le deuxième enfant du premier <li> hérite de cette propriété.
- Dans le deuxième cas, le premier <li> se voit affecter une propriété CSS color initialisée à red, et ses deux enfants héritent de cette propriété.

Pseudo-sélecteurs d'éléments sélectionnés

Lorsque vous utilisez un sélecteur CSS, un ou plusieurs éléments sont sélectionnés dans le DOM. En ajoutant un pseudo-sélecteur au sélecteur, vous allez pouvoir filtrer la sélection en ne conservant que les éléments pairs, impairs, ayant un certain index, etc. Regardez le tableau suivant :

Sélecteur	Éléments sélectionnés
`:even`	Éléments pairs
`:odd`	Éléments impairs
`:eq()`	Élément dont l'index est spécifié
`:gt()`	Éléments dont l'index est supérieur à (*greater than*) l'index spécifié
`:lt()`	Éléments dont l'index est inférieur à (*lower than*) l'index spécifié

Pour varier les plaisirs, nous allons effectuer des sélections dans une série de paragraphes. Voici le code HTML utilisé :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <p>Paragraphe 1</p>
    <p>Paragraphe 2</p>
    <p>Paragraphe 3</p>
    <p>Paragraphe 4</p>
    <p>Paragraphe 5</p>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        //Le code jQuery sera inséré ici
      });
    </script>
  </body>
</html>
```

Nous allons afficher en rouge certains éléments de ces listes imbriquées en utilisant la méthode jQuery `css()` .

Critère	Sélecteur	Paragraphe modifié
Éléments `p` pairs	`$('p:even')`	1, 3, 5
Éléments `p` impairs	`$('p:odd')`	2, 4
Éléments `p` après le deuxième	`$('p:gt(1)')`	3, 4, 5
Élément `p` d'index 4	`$('p:eq(3)')`	4
Éléments `p` avant le quatrième	`$('p:lt(3)')`	1, 2, 3

Sélecteurs d'éléments particuliers

Cette section s'intéresse à des sélecteurs propres à certaines balises ou difficilement classables dans les autres catégories. Regardez le tableau suivant :

Sélecteur	Éléments sélectionnés
`:header`	Tous les titres `<h1>` à `<h6>`
`:hidden`	Éléments cachés
`:visible`	Éléments visibles
`:not()`	Éléments qui ne correspondent pas au sélecteur spécifié

Voici le code utilisé pour tester ces sélecteurs :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <h1>Titre de niveau 1</h1>
    <h2>Titre de niveau 2</h2>
    <h3>Titre de niveau 3</h3>
    <p>Un paragraphe de texte</p>
    <div>Texte dans une balise div</div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('div').hide();
        //Le code jQuery sera inséré ici
      });
    </script>
  </body>
</html>
```

Ligne 17, l'instruction jQuery dissimule la balise `<div>` à l'aide de la méthode `hide()`. Testez tour à tour les trois instructions jQuery suivantes :

Critère	Instruction jQuery	Effet
Sélection de tous les titres	`$(':header').css('color','red');`	Les titres `<h1>`, `<h2>` et `<h3>` sont affichés en rouge.
Affichage des éléments cachés	`$('div:hidden').show();`	L'élément `div` qui avait été caché à la ligne 17 est affiché.
Dissimulation de tous les titres sauf le titre `<h1>`	`$(':header:not(h1)').hide();`	Toutes les balises de titre sont cachées, à l'exception de la balise <h1>.

Pseudo-sélecteurs spécifiques aux formulaires

Pseudo-sélecteur	Éléments sélectionnés
`:input`	Tous les éléments de type `input`, `textarea`, `select` et `button`
`:button`	Éléments de type `button`
`:checkbox`	Éléments de type `checkbox`
`:checked`	Éléments qui sont cochés
`:radio`	Éléments de type `radio`
`:reset`	Éléments de type `reset`
`:image`	Tous les boutons de type `image`
`:submit`	Éléments de type `submit`
`:text`	Éléments de type `text`
`:password`	Éléments de type `password`
`:selected`	Éléments sélectionnés
`:focus`	Sélectionne l'élément s'il a le focus
`:enabled`	Éléments validés

Pour illustrer ces pseudo-sélecteurs, nous allons utiliser un formulaire très classique contenant :

- une zone de texte ;
- un mot de passe ;
- deux boutons radio ;
- une zone de texte multiligne ;
- un bouton submit et un bouton reset ;
- un bouton image.

Voici le code :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <form>
      Nom d'utilisateur
      <input type="text" name="nom"><br />

      Mot de passe
      <input type="password" name="pass"><br />

      Sexe
      Homme <input type="radio" name="sexe" value="H">
      Femme <input type="radio" name="sexe" value="F"><br />

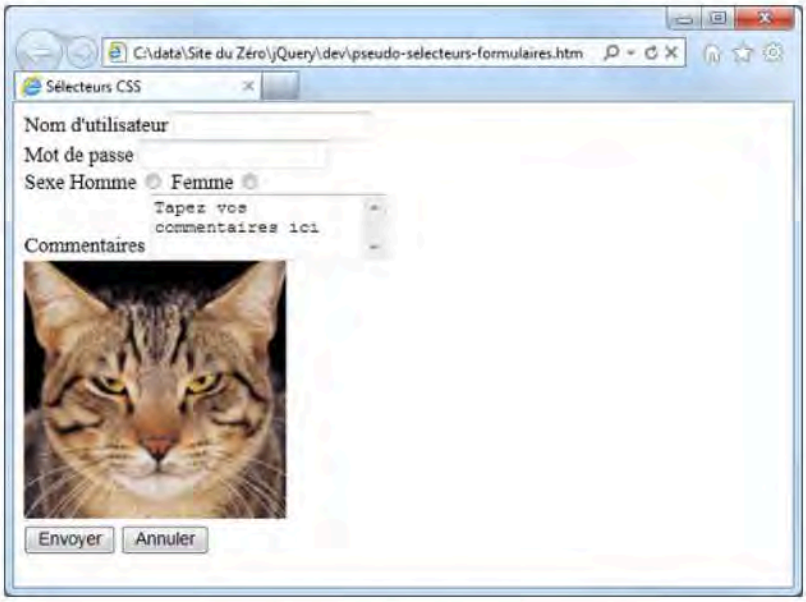
      Commentaires
      <textarea rows="3" name="commentaires">Tapez vos commentaires ici</textarea><br />

      <input type="image" src="chat.jpg"><br />

      <input type="submit" value="Envoyer">
      <input type="reset" value="Annuler">
    </form>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Le code jQuery sera tapé ici
      });
    </script>
  </body>
</html>
```

Exécuté, ce code ressemble à l'image suivante :



Le code précédent exécuté dans Internet Explorer

Nous allons modifier la couleur d'arrière-plan de certains éléments du formulaire et modifier la taille du bouton image en utilisant quelques lignes de jQuery, comme au tableau suivant :

Action à accomplir	Instruction jQuery/JavaScript	Effet
Coloration de tous les champs de saisie	`$(':input').css('background','yellow');`	Les zones de texte, boutons radio et boutons ont un arrière-plan de couleur jaune.
Coloration d'un champ de saisie particulier	`$(':password').css('background','yellow');`	Le champ de type `password` a un arrière-plan de couleur jaune.
Redimensionnement d'un champ de type `image`	`$(':image').css('width','100px');`	Le champ `input` de type `image` est redimensionné pour avoir une largeur de 100 px.
Focus au premier champ de saisie et coloration en jaune	`document.forms[0].nom.focus();` `$(':focus').css('background','yellow');`	La première instruction donne le focus au premier champ de saisie et la deuxième colore son arrière-plan en jaune.

Sélecteurs utilisés dans les tableaux

Si les tableaux n'ont pas de sélecteurs spécifiques, plusieurs des sélecteurs évoqués dans les sections précédentes sont cependant bien pratiques pour mettre en forme rapidement des tableaux HTML.

Pseudo-sélecteur	Éléments sélectionnés
`:first`	Premier élément
`:last`	Dernier élément
`:eq()`	Élément dont l'index est spécifié
`:gt()`	Éléments dont l'index est supérieur à l'index spécifié
`:lt()`	Éléments dont l'index est inférieur à l'index spécifié
`:even`	Éléments d'index pair
`:odd`	Éléments d'index impair
`:empty`	Éléments qui n'ont pas d'enfant
`:not()`	Éléments qui ne correspondent pas au sélecteur spécifié

Nous allons mettre en application ces pseudo-sélecteurs en les appliquant sur le tableau défini en HTML que voici :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <form>
      <table border=1>
      <tr><td></td><td>Salle 1</td><td>Salle 2</td></tr>
      <tr><td>Lundi</td><td>X</td><td>X</td></tr>
      <tr><td>Mardi</td><td></td><td></td></tr>
      <tr><td>Mercredi</td><td>X</td><td></td></tr>
      <tr><td>Jeudi</td><td></td><td>X</td></tr>
      <tr><td>Vendredi</td><td>X</td><td>X</td></tr>
      </table>
    </form>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('td').css('text-align','center');
        // Le code jQuery sera tapé ici
      });
    </script>
  </body>
</html>
```

Nous allons affecter la couleur jaune à l'arrière-plan de certaines cellules du tableau en utilisant la méthode jQuery `css()` .

Action recherchée	Sélecteur
Coloration de la première cellule	`$('td:first')`
Coloration de la dernière ligne	`$('tr:last')`
Coloration des cellules vides	`$(':empty')`
Coloration des lignes paires	`$('tr:even')`
Coloration des cellules paires	`$('td:even')`
Coloration des lignes d'index supérieur à 2	`$('tr:gt(1)')`
Coloration des lignes d'index supérieur à 1, et, dans le sous-ensemble correspondant, des lignes d'index inférieur à 5	`$('tr:gt(0):lt(4)')`
Coloration des cellules d'index supérieur à 5, et, dans le sous-ensemble correspondant, des cellules d'index inférieur à 11	`$('td:gt(4):lt(10)')`
Coloration de toutes les lignes à l'exception de la dernière	`$('tr:not(tr:last)')`

Parcourir les éléments sélectionnés

Tout au long de ce chapitre, vous avez vu qu'il suffisait d'appliquer une méthode jQuery aux éléments sélectionnés pour agir immédiatement sur ces éléments. Dans cette section, je vais vous montrer une autre façon d'interagir plus finement sur la sélection, en utilisant la méthode `each()` .

Supposons par exemple que trois images soient affichées dans une page Web. Ces images sont repérées par leur chemin : `images1/i1.jpg` , `images1/i2.jpg` et `images1/i3.jpg` . Comment feriez-vous en jQuery pour que ces images soient lues dans le dossier `images2` et non dans le dossier `images1` ? Au risque de vous décevoir, c'est tout bonnement impossible avec vos connaissances actuelles en jQuery ! Pour résoudre cet exercice, il faudrait pouvoir agir de façon individuelle sur l'attribut `src` de chaque balise `<img>` .

Pour arriver à nos fins, nous allons utiliser la méthode `each()` , qui passe en revue les éléments sélectionnés par une requête jQuery et permet de les modifier en utilisant des instructions JavaScript. La syntaxe de la méthode `each()` est la suivante :

```javascript
$('sel').each(function(index){
  //Une ou plusieurs instructions JavaScript
});
```

… où :

- `sel` est un sélecteur CSS, comme ceux que nous avons utilisés jusqu'ici ;
- `index` est une variable JavaScript qui représente la position de l'élément dans la sélection. Il aura pour valeurs consécutives 0, 1, 2, etc. jusqu'à ce que tous les éléments aient été passés en revue ;

Jusqu'ici, rien de bien méchant ! Cependant, une question se pose : comment accéder à l'élément courant, c'est-à-dire à l'élément à modifier ? Pour cela, vous devez utiliser le mot `this`. Si vous avez suivi ce que j'ai dit un peu plus haut, la modification se fera via l'attribut `src` de la balise `<img>`. Eh bien, dans ce cas, vous utiliserez l'expression `this.src`.

Examinons le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <img src="images1/i1.jpg">
    <img src="images1/i2.jpg">
    <img src="images1/i3.jpg">

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('img').each(function(index){
          this.src = 'images2/i' + (index+1) + '.jpg';
        });
      });
    </script>
  </body>
</html>
```

Nous retrouvons nos trois images, l'appel à jQuery ainsi que quelques instructions. Lorsque le DOM est disponible, les images du document sont sélectionnées avec le sélecteur `img` et les éléments sélectionnés sont parcourus avec la méthode `each()`. Une seule ligne est nécessaire pour modifier les adresses des images :

```javascript
this.src = 'images2/i' + (index+1) + '.jpg';
```

Le terme `this.src` correspond à l'attribut `src` de la balise `<img>` en cours de traitement. Étant donné qu'il y a trois images à traiter, la variable `index` va prendre les valeurs 0, 1 puis 2. Le terme `'images2/i' + (index+1) + '.jpg'` aura donc pour valeurs consécutives `images2/i1.jpg`, `images2/i2.jpg` et `images2/i3.jpg`, ce qui est exactement l'effet recherché.



En JavaScript, le signe + est utilisé pour *concaténer* (c'est-à-dire mettre bout à bout) deux textes, deux nombres ou un texte et un nombre. Ici, `'images2/i' + (index+1) + '.jpg'` concatène le texte `images2/i`, le nombre `(index+1)` et le texte `.jpg`. Remarquez les parenthèses autour de `index+1` ; sans elles, ce n'est pas la valeur numérique `index+1` qui aurait été concaténée, mais la valeur numérique de la variable `index` puis le chiffre 1. Ce qui aurait fabriqué les chaînes `images2/i01.jpg`, `images2/i11.jpg` et `images2/i21.jpg`.

Conversion jQuery/DOM

Conversion d'un élément du DOM en un objet jQuery

Si vous avez suivi attentivement ce tutoriel, vous savez que le simple fait d'utiliser un sélecteur jQuery convertit un élément (ou un ensemble d'éléments) du DOM en un objet jQuery. Ainsi par exemple, l'instruction `$('div')` retourne un objet jQuery qui donne accès à toutes les balises `<div>` du document. La première balise `<div>` est alors accessible avec `$("div")[0];`, la deuxième avec `$("div")[1];`, et ainsi de suite.

Conversion d'une variable JavaScript en un objet jQuery

jQuery et JavaScript font bon ménage ensemble, et il est fréquent d'entremêler des instructions jQuery et des instructions JavaScript. Fatalement, un jour ou l'autre, vous voudrez convertir une variable JavaScript en un objet jQuery pour pouvoir lui appliquer les puissantes méthodes de ce langage. Eh bien, il n'y a rien de plus simple :

```javascript
var variableJS = 'un simple texte';
var variableJQ = $(variableJS);
```

La première ligne crée une variable JavaScript en lui affectant un simple texte. La deuxième la convertit en un objet jQuery en « l'enveloppant » avec l'alias `$()`.

Conversion d'un objet jQuery en un élément du DOM

Il est parfois nécessaire d'appliquer un traitement JavaScript à un objet jQuery. Étant donné que seules les méthodes jQuery peuvent être appliquées aux objets jQuery, une conversion jQuery vers DOM est alors nécessaire. Pour cela, vous appliquerez la méthode `.get()` à un sélecteur jQuery.

Par exemple, si un document contient plusieurs balises `<span>`, il est possible de les convertir en un tableau JavaScript avec l'instruction suivante :

```javascript
var spans = $('span').get();
```

Ici, le tableau est stocké dans la variable JavaScript `spans`. La valeur stockée dans la première cellule du tableau est obtenue avec `spans[0].innerHTML`, la valeur stockée dans la deuxième cellule du tableau est obtenue avec `spans[1].innerHTML`, et ainsi de suite…

- Il est possible d'affiner la sélection d'éléments grâce aux sélecteurs d'attributs, aux pseudo-sélecteurs et aux sélecteurs hiérarchiques.
- Les formulaires ont leur propre jeu de pseudo-sélecteurs. Faciles à mémoriser, ils ont le nom des balises correspondantes et sont précédés d'un « : ».
- En utilisant un sélecteur jQuery, on obtient un objet jQuery qui contient zéro, un ou plusieurs éléments du DOM. On peut appliquer un traitement global à ces éléments en leur appliquant une méthode jQuery. Lorsqu'un traitement global n'est pas suffisant, on utilise une boucle `each`.
- Avec une boucle `each`, les différents éléments sélectionnés sont parcourus un à un. Ils sont repérés par leur index, dont le nom est spécifié en paramètre de la fonction. Le traitement est réalisé par une ou plusieurs instructions JavaScript.
- Il est parfois utile de convertir des objets jQuery en éléments du DOM et inversement. La transformation DOM vers jQuery se fait en utilisant un sélecteur, et éventuellement en le limitant à un des éléments retournés. Quant à la transformation jQuery vers DOM, elle repose sur l'utilisation de la méthode `get()`.

MODIFIER LE CONTENU D'UN ÉLÉMENT

Arrivés à ce point dans la lecture du cours, vous savez sélectionner à peu près tout ce qui peut se trouver dans un document HTML. Vous vous sentez peut-être un peu frustrés : à quoi bon effectuer des sélections si vous n'en faites rien ! Vous avez tout à fait raison.

Ce chapitre va (enfin !) vous montrer comment agir sur les éléments sélectionnés pour les transformer. Vous pourrez ainsi modifier les attributs et les propriétés CSS des éléments sélectionnés, mais aussi ajouter, réorganiser et supprimer des éléments du DOM pour modifier l'agencement de la page sur laquelle vous travaillez.

Poursuivez vite la lecture et surtout… amusez-vous bien !

Getters et setters

Plutôt que de différencier les méthodes chargées de lire (ces méthodes sont dites *getters*, de l'anglais *to get* qui signifie « obtenir ») et de modifier (ces méthodes sont dites *setters*, de l'anglais *to set* qui signifie « définir ») les valeurs HTML ou CSS, les concepteurs de jQuery ont choisi de leur donner le même nom. Mais alors, comment les différencier, me direz-vous ? C'est très simple, puisqu'il suffit d'observer les paramètres de la fonction. Raisonnons sur un exemple pour mieux comprendre de quoi il retourne. Observez les deux instructions suivantes :

javascript
```
$('h2').css('font-size');
$('h2').css('font-size','2em');
```

Dans la première instruction, aucune valeur n'est précisée. Il est impossible de modifier la valeur de la propriété CSS `font-size`. La première méthode est donc un getter ; elle retourne la valeur de la propriété `font-size`. Pour faciliter sa manipulation, vous pouvez la mémoriser dans une variable :

javascript
```
var taille = $('h2').css('font-size');
```

Dans la deuxième instruction, la valeur « 2em » est précisée dans les paramètres de la méthode `css()`. Cette valeur sera utilisée pour mettre à jour la propriété CSS `font-size` (ou la créer si elle n'existe pas) de tous les éléments retournés par le sélecteur. La deuxième méthode est donc un setter.

Ce qui vient d'être dit peut se généraliser à toutes les méthodes jQuery :

- Si aucune valeur n'est précisée dans les arguments de la méthode, il s'agit d'un getter. La méthode retourne la valeur qui correspond au premier argument.
- Si une valeur est précisée dans les arguments de la méthode, il s'agit d'un setter. Le premier argument de la méthode est initialisé avec cette valeur. S'il n'existe pas, il est créé. S'il existe, il est modifié en conséquence.

Ce que renvoie un getter

Un sélecteur jQuery peut retourner zéro, un ou plusieurs éléments. Lorsqu'aucun élément n'est retourné, le getter renvoie la valeur `undefined` (c'est-à-dire « non défini »). Lorsqu'un seul élément est retourné, le getter renvoie la valeur de cet élément. Enfin, lorsque plusieurs éléments sont retournés, le getter renvoie la valeur du premier élément.

Examinez le code suivant :

html
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <div id="listes">
      <a href="http://api.jquery.com">API jQuery</a><br>
      <a href="http://docs.jquery.com">Documentation jQuery</a><br>
    </div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        var test = $('a').attr('href');
        document.write(test);
      });
    </script>
  </body>
</html>
```


Les bases de jQuery

1. Sélection d'éléments
2. Plus loin dans la sélection d'éléments
3. **Modifier le contenu d'un élément**
4. Insérer et remplacer des éléments dans le DOM
5. TP : Questionnaire interactif en jQuery

Quiz

Activité : Créez un générateur de formulaires

Accéder au forum

Créé par

OPENCLASSROOMS

OpenClassrooms, 1ère plateforme e-Education en Europe

Deux liens hypertextes sont définis lignes 9 et 10. Le premier pointe sur la page `http://api.jquery.com` et le second sur la page `http://docs.jquery.com`. À la ligne 16, on utilise l'instruction jQuery `$('a').attr('href')` pour lire le contenu de l'attribut `href` des balises `<a>` contenues dans le document. L'objet retourné est stocké dans la variable `test`. La ligne 17 affiche cette variable. D'après vous, quelle valeur va s'afficher dans le navigateur ?

Comme il a été dit précédemment, dans le cas d'une réponse multiple, seule la première valeur est retournée par le getter. Ici, c'est donc l'adresse `http://api.jquery.com` qui s'affichera dans le navigateur.

Comme vous pouvez le voir, les deux balises `<a>` de ce document ne contiennent qu'un seul attribut : `href`. Si on utilisait l'instruction `var test = $('a').attr('class');`, la valeur retournée serait `undefined`.

Ce qui peut être passé à un setter

Les méthodes setters peuvent se présenter sous trois formes différentes :

javascript
```
$('#logo').attr('src','logo.gif');
$('#logo').attr({src: 'logo.gif', alt: 'Logo de la société', width: '200px'});
$("a").attr({target:function(){…}});
```

La première ligne se contente d'affecter la valeur « logo.gif » à l'attribut `src` de l'élément d'identifiant `logo`.
La deuxième ligne crée (s'ils n'existent pas) ou modifie (s'ils existent) plusieurs attributs dans l'élément d'identifiant `logo`. Ici, l'attribut `src` est initialisé avec la valeur « logo.gif », l'attribut `alt` avec la valeur « Logo de la société » et l'attribut `width` avec la valeur « 200px ».
Enfin, la troisième ligne utilise une fonction JavaScript pour créer ou modifier l'attribut `target` des balises `<a>` du document. Voici par exemple à quoi pourrait ressembler la fonction passée en deuxième argument de la méthode `attr()` :

javascript
```
$('a').attr('target', function() {
  if(this.host == location.host) return '_self'
  else return '_blank'
});
```

Si vous avez quelques rudiments de JavaScript, le code utilisé dans cette fonction ne devrait pas vous poser de problème. Dans le cas contraire, je vais décrire en détail les actions accomplies.

Si le lien (`this.host`) se trouve sur le même site que la page en cours (`== location.host`), l'attribut `target` est initialisé avec la valeur « _self » (`return '_self'`). Dans le cas contraire, l'attribut `target` est initialisé avec la valeur « _blank » (`else return '_blank'`). Une fois ces deux lignes exécutées, les liens hypertextes seront ouverts :

- dans l'onglet courant du navigateur s'ils renvoient vers une page située dans le même nom de domaine que la page actuelle ;
- dans une autre fenêtre (ou un nouvel onglet) du navigateur s'ils se trouvent sur un autre nom de domaine.

Accéder aux attributs HTML et aux propriétés CSS

Accéder aux attributs des balises HTML

Vous utiliserez la méthode `attr()` pour lire, créer et modifier les attributs des balises HTML. Voici quelques exemples :

- `$('#plus').attr('src');` retourne l'attribut `src` de l'élément d'identifiant `plus`.
- `$('div').attr('class');` retourne l'attribut `class` du premier `<div>`.
- `$('div').attr('class', 'madiv');` modifie ou crée l'attribut `class` dans les balises `<div>` du document et leur affecte la valeur « madiv ».
- `$('#illustration').attr('src','monimage.jpg');` modifie ou crée l'attribut `src` dans la balise d'identifiant `illustration` et lui affecte la valeur « monimage.jpg ».

Vous savez maintenant comment lire et comment créer/modifier un attribut d'une balise ou d'un ensemble de balises HTML.

Voyons maintenant comment supprimer un attribut dans une balise ou un ensemble de balises. Pour cela,

vous utiliserez la méthode `removeAttr()` :

```javascript
$(sel).removeAttr('attribut');
```

… où `sel` est un sélecteur jQuery et `attribut` est l'attribut que vous voulez supprimer. Cette méthode agit sur tous les éléments sélectionnés par le sélecteur jQuery. Par exemple, pour supprimer l'attribut `href` de toutes les balises `<a>` du document, vous utiliserez l'instruction suivante :

```javascript
$('a').removeAttr('href');
```



Pour avoir un aperçu des différentes balises HTML5 et de leurs attributs, vous pouvez consulter le site W3Schools.com.

Accéder aux propriétés CSS

Dans les chapitres précédents, nous avons utilisé à plusieurs reprises la méthode jQuery `css()` pour créer ou modifier les propriétés CSS des balises HTML. Cette méthode peut également être utilisée comme un getter, pour connaitre la valeur d'une propriété CSS. Par exemple, l'instruction suivante récupère la valeur stockée dans l'attribut `font-size` du premier élément de classe `para` et la stocke dans la variable `taille` :

```javascript
var taille = $('.para').css('font-size');
```

Cette deuxième instruction affecte la valeur « 40px » à l'attribut `font-size` de tous les éléments de classe `para` :

```javascript
$('.para').css('font-size', '40px');
```



Pour avoir un aperçu des différentes propriétés CSS3, vous pouvez consulter le site W3Schools.com.

Travailler avec l'attribut class

Comme tout programmeur (ou apprenti programmeur) Web, vous utilisez certainement l'attribut `class` pour donner la même apparence à plusieurs balises HTML. Pour accéder aux balises dont l'attribut `class` a une certaine valeur, il suffit de préciser cette valeur dans le sélecteur en la faisant précéder d'un point. Par exemple, pour sélectionner tous les éléments de classe `vert`, vous utiliserez le sélecteur jQuery `$('.vert')`.

Ajouter et supprimer des classes

Trois méthodes consacrées aux classes vont vous permettre d'aller plus loin :

- `addClass()` ajoute une classe dans les éléments sélectionnés ;
- `removeClass()` supprime (si elle existe) une classe des éléments sélectionnés ;
- `toggleClass()` accomplit deux actions : si la classe spécifiée n'existe pas dans les éléments sélectionnés, elle y est ajoutée. Si elle existe, elle est supprimée.

La théorie étant posée, nous allons l'expérimenter en utilisant le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Manipulation de l'attribut class</title>
    <style type="text/css">
      .rouge { color: red; }
      .vert { color: green; }
      .petit { font-size: 100%; }
      .grand {font-size: 250%; }
    </style>
  </head>
  <body>
    <span id="jean" class="rouge grand">Jean</span>

    <span id="pierre">Pierre</span>

    <span id="paul" class="vert grand">Paul</span>

    <span id="julia">Julia</span>

    <span id="eric" class="vert">Eric</span>

    <span id="kevin" >Kévin</span>


    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Ajouter le code jQuery ici
      });
    </script>
  </body>
</html>
```

La théorie étant posée, nous allons l'expérimenter en utilisant le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Manipulation de l'attribut class</title>
    <style type="text/css">
      .rouge { color: red; }
      .vert { color: green; }
      .petit { font-size: 100%; }
      .grand {font-size: 250%; }
    </style>
  </head>
  <body>
    <span id="jean" class="rouge grand">Jean</span>

    <span id="pierre">Pierre</span>

    <span id="paul" class="vert grand">Paul</span>

    <span id="julia">Julia</span>

    <span id="eric" class="vert">Eric</span>

    <span id="kevin" >Kévin</span>

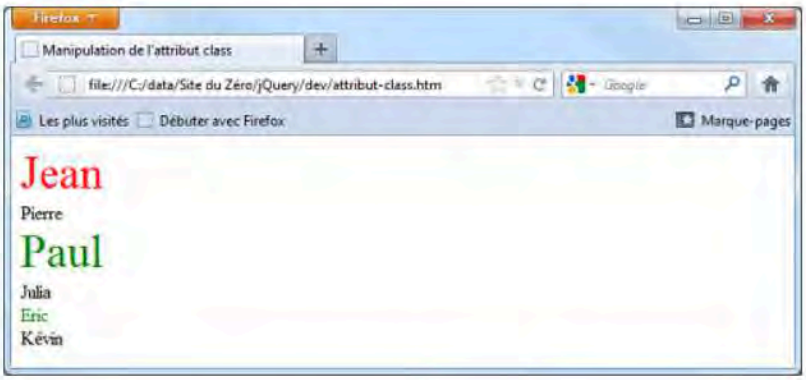

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Ajouter le code jQuery ici
      });
    </script>
  </body>
</html>
```

Les lignes 6 à 11 définissent quatre styles :

- `rouge` affiche les caractères en rouge ;
- `vert` affiche les caractères en vert ;
- `petit` affiche les caractères avec la taille par défaut (100%) ;
- `grand` affiche les caractères avec une grande taille (250%).

Les lignes 14 à 19 affichent six prénoms par l'intermédiaire de balises `<span>`. Chaque balise a un identifiant unique, et certaines balises ont un attribut `class` initialisé avec une ou deux classes.

Pour l'instant, aucun code jQuery n'a été inséré dans le document. L'image suivante montre à quoi il ressemble une fois affiché dans un navigateur.



Le code précédent exécuté dans Firefox

Nous allons expérimenter les méthodes `addClass()`, `removeClass()` et `toggleClass()` en insérant du code jQuery ligne 30.

La balise `<span>` d'identifiant `julia` ne possède aucune classe. Le prénom Julia est donc affiché en caractères noirs de taille standard. Supposons que nous voulions les afficher en rouge ; le code à utiliser est le suivant :

```javascript
$('#julia').addClass('rouge');
```

Supposons maintenant que le prénom Julia doive être affiché en grands caractères verts. Le code à utiliser est le suivant :

```javascript
$('#julia').addClass('vert grand');
```

Nous allons maintenant travailler avec le prénom Eric. Ce prénom est affiché via une balise `<span>` de classe `vert`. Supposons que vous vouliez afficher le prénom Eric en rouge. L'instruction suivante n'a aucun effet :

```javascript
$('#eric').addClass('rouge');
```

Avez-vous une idée de la raison de cet échec ? Cela vient d'un conflit entre la classe `vert` (existante) et la classe `rouge` (que l'on veut ajouter). Pour parvenir au résultat souhaité, il faudrait supprimer la classe existante et la remplacer par la classe `rouge` :

```javascript
$('#eric').removeClass('vert').addClass('rouge');
```

La première méthode supprime la classe `vert` et la deuxième ajoute la classe `rouge`.

Supposons maintenant que vous vouliez afficher le prénom Paul en petits caractères de couleur rouge. La balise `<span>` correspondante utilise deux classes : `vert` et `grand`. Pour que le texte s'affiche en caractères rouges de taille standard, vous devez :

- supprimer la classe `vert` ;
- ajouter la classe `rouge` ;
- supprimer la classe `grand` ;
- ajouter la classe `petit`.

Voici l'instruction à utiliser :

```javascript
$('#paul').removeClass('vert').addClass('rouge').removeClass('grand').addClass('petit');
```

En utilisant un autre identifiant dans le sélecteur, ce chaînage de quatre méthodes peut également s'appliquer à une quelconque balise `<span>` du document. Ainsi par exemple, cette instruction affiche le prénom Pierre en caractères rouges de taille standard :

```javascript
$('#pierre').removeClass('vert').addClass('rouge').removeClass('grand').addClass('petit');
```

Les méthodes `removeClass()` et `addClass()` peuvent également être remplacées par la méthode `toggleClass()`. Ainsi, cette instruction affiche le prénom Paul en caractères rouges de petite taille :

```javascript
$('#paul').toggleClass('vert').toggleClass('rouge').toggleClass('grand').toggleClass('petit'
```

Par contre, si vous l'appliquez au `<span>` d'identifiant `eric`, le prénom Eric est affiché en caractères rouges de grande taille :

- Le `<span>` étant de classe `vert`, cette classe est supprimée : `toggleClass('vert')` ;
- La classe `rouge` lui est ajoutée puisqu'elle n'existe pas : `toggleClass('rouge')` ;
- Le `<span>` étant de classe `petit`, cette classe est supprimée : `toggleClass('petit')` ;
- Enfin, la classe `grand` lui est ajoutée puisqu'elle n'existe pas : `toggleClass('grand')`.

Comme vous le voyez, la méthode `toggleClass()` doit être utilisée en toute connaissance de cause, sans quoi elle produira des effets aléatoires.

Pour simplifier l'écriture, il est possible d'indiquer plusieurs classes séparées par des espaces dans les méthodes `addclass()`, `removeclass()` et `toggleclass()`. Ainsi par exemple, cette instruction :

```javascript
$('#pierre').removeClass('vert').addClass('rouge').removeClass('grand').addClass('petit');
```

Peut être simplifiée comme suit :

```javascript
$('#pierre').removeClass('vert grand').addClass('rouge petit');
```

Tester l'existence de classes

La méthode `hasClass()` permet de tester si la sélection est d'une certaine classe. Supposons par exemple que la balise `<span>` suivante soit définie :

```html
<span id="jean" class="rouge grand">Jean</span><br />
```

L'instruction `$('#jean').hasClass('rouge');` renverra la valeur `true`, car le `<span>` est de classe `rouge`. L'instruction `$('#jean').hasClass('petit');` renverra la valeur `false`, car le `<span>` n'est pas de classe `petit`.

Ainsi, on pourra effectuer une action ou une autre en fonction de l'existence d'une classe :

javascript
```
if ($('#jean').hasClass('rouge'))
  alert('le span #jean est de classe rouge');
else
  alert('le span #jean n\'est pas de classe rouge');
```

Si vous devez tester l'appartenance à plusieurs classes, vous utiliserez la méthode `is()`. Raisonnons sur la balise `<span>` suivante :

html
```
<span id="jean" class="rouge grand">Jean</span><br />
```

L'instruction `$('#jean').is('.grand.rouge');` renverra la valeur `true`, car le `<span>` est de classe `grand` et `rouge`. Par contre, l'instruction `$('#jean').is('.petit.rouge');` renverra la valeur `false`, car le `<span>` n'est pas de classe `petit`. En enveloppant l'instruction jQuery par un `if`, vous pourrez effectuer une action ou une autre en fonction de l'existence de deux ou plusieurs classes :

javascript
```
if ($('#jean').is('.grand.rouge'))
  alert('le span #jean est de classe grand et rouge');
else
  alert('le span #jean n\'est pas de classe grand et/ou rouge');
```

Travailler avec les formulaires

Vous utiliserez la méthode `val()` pour tester/modifier la valeur des zones de texte, boutons radio, cases à cocher, listes déroulantes et zones de liste contenues dans un document HTML. Pour vous montrer comment utiliser cette méthode, nous allons raisonner sur un exemple :

html
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sélecteurs CSS</title>
  </head>
  <body>
    <form>
      Nom d'utilisateur
      <input type="text" id="nom"><br />

      Mot de passe
      <input type="password" id="pass"><br />

      Sexe
      H <input type="radio" id="H" name="sexe" value="H">
      F <input type="radio" id="F" name="sexe" value="F"><br />

      Fonction
      <select id="fonction">
        <option VALUE="etudiant">Etudiant</option>
        <option VALUE="ingenieur">Ingénieur</option>
        <option VALUE="enseignant">Enseignant</option>
        <option VALUE="retraite">Retraité</option>
        <option VALUE="autre">Autre</option>
      </select><br /><br />

      <input type="submit" id="envoyer" value="Envoyer">
      <input type="reset" id="annuler" value="Annuler">
    </form>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Entrer les instructions jQuery ici
      });
    </script>
  </body>
</html>
```

Ce code définit une zone de texte (`nom`), un mot de passe (`pass`), deux boutons radio (`sexe`), une liste déroulante (`fonction`), un bouton « Envoyer » (`envoyer`) et un bouton « Annuler » (`annuler`).

Le tableau suivant donne un aperçu des instructions que vous pouvez utiliser pour lire et modifier les données stockées dans le formulaire.

Instruction jQuery	Effet
`$('#nom').val()`	Lit le nom de l'utilisateur.
`$('#pass').val()`	Lit le mot de passe.
`$(':radio#H:checked').val()`	Lit l'état du bouton radio `H`. Renvoie `true` si le bouton est sélectionné, sinon `false`.
`$('#fonction').val()`	Lit l'élément sélectionné dans la liste déroulante.
`$('#nom').val('Michel')`	Écrit « Michel » dans la zone de texte `Nom d'utilisateur`.
`$('#pass').val('abcde')`	Écrit « abcde » dans la zone de texte `Mot de passe`.
`$(':radio').val(['H']);`	Sélectionne le bouton radio `H`.
`$('#fonction').val('retraite')`	Sélectionne `Retraité` dans la liste déroulante.

La syntaxe des instructions n'est pas complexe, mais rien de tel qu'un peu de pratique pour qu'elle ne vous pose plus aucun problème. Je vous suggère donc de vous entraîner à utiliser ces instructions en les tapant une à une sur la ligne 36 du code précédent !

Travailler avec les valeurs stockées dans des éléments

Lorsque vous définissez un sélecteur jQuery, vous obtenez un objet jQuery qui fait référence à zéro, un ou plusieurs éléments. Si ces éléments contiennent des valeurs textuelles, vous pouvez les lire ou les modifier en utilisant deux méthodes jQuery :

- `text()` retourne/modifie la valeur textuelle stockée dans l'élément ;
- `html()` retourne/modifie le code HTML stocké dans l'élément.

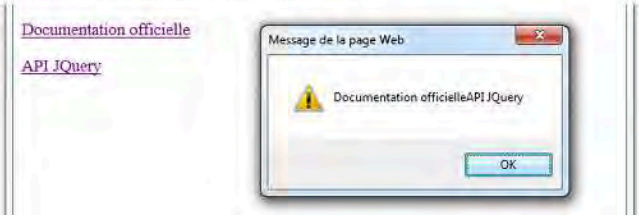
Comme toujours, nous allons utiliser un exemple pour bien comprendre le fonctionnement de ces deux méthodes. Voici le code utilisé :

html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Valeurs stockées dans les éléments</title>
  </head>
  <body>
    <h1>Documentation jQuery</h1>
    <p><a href='http://docs.jquery.com'>Documentation officielle</a></p>
    <p><a href='http://api.jquery.com'>API jQuery</a></p>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Entrer les instructions jQuery ici
      });
    </script>
  </body>
</html>
```

Le corps du document définit un titre de niveau 1 et deux paragraphes qui pointent vers la documentation officielle de jQuery et l'API jQuery. Pour récupérer le texte stocké dans les deux paragraphes, nous utiliserons l'instruction `$('p').text();`, et pour afficher ce texte sur l'écran nous utiliserons une boîte de dialogue : `alert($('p').text());`. Tapez cette instruction à la ligne 15 du code précédent, sauvegardez le document et affichez-le dans le navigateur de votre choix. L'image suivante vous montre le résultat sous Internet Explorer.



Comme vous pouvez le voir, l'instruction retourne les deux valeurs textuelles stockées dans les balises `<p>`. Ces deux valeurs sont placées l'une à la suite de l'autre. Pour accéder individuellement à la première et à la dernière valeur, le plus simple consiste à utiliser des pseudo-opérateurs :

```javascript
var premier = $('p:first').text();
var dernier = $('p:last').text();
```

Si vous voulez accéder individuellement à chacune des valeurs, vous devez définir une fonction comme paramètre de la méthode `text()`, comme ceci : `function(index, actuel)`, où `index` représente le numéro de la valeur en cours de traitement (à partir de 0), et `actuel` représente la valeur en cours de traitement.

À titre d'exemple, pour afficher la valeur contenue dans chaque paragraphe du document, vous pourriez utiliser le code suivant :

```javascript
$('p').text(function(index,actuel) {
  alert('Paragraphe ' + (index+1) + ' : '+actuel);
});
```

Ce qui vous donnerait la figure suivante.



Vous savez maintenant récupérer sous forme de texte les valeurs stockées dans des éléments HTML. Pour en obtenir une forme HTML, remplacez la méthode `text()` par la méthode `html()`. Voici quelques-unes des instructions que vous pourriez utiliser :

Instructions	Résultat
`alert($('p').html());`	Affiche le code HTML du premier élément (voir image suivante).
`alert($('p:first').html());`	Affiche le code HTML du premier élément.
`alert($('p:last').html());`	Affiche le code HTML du dernier élément.
`$('p').html(function(index,actuel) { alert('Paragraphe ' + (index+1) + ' : '+actuel);});`	Affiche individuellement le code HTML de chaque élément.

En observant les deux premiers exemples de code dans le tableau précédent, vous vous demandez certainement si une erreur ne s'est pas glissée dans la colonne « Résultat ». En effet, est-ce que les instructions `alert($('p').html());` et `alert($('p:first').html());` seraient équivalentes et renverraient toutes deux le code HTML du premier élément ? Eh bien oui, ces deux instructions sont équivalentes car, contrairement à la méthode `text()`, `html()` ne balaie pas tous les éléments, mais se contente du premier.

Les méthodes `text()` et `html()` peuvent bien évidemment être utilisées en tant que setters. Par exemple, pour que le premier paragraphe du listing précédent pointe vers le moteur de recherche Google et non vers la documentation officielle de jQuery, vous utiliserez l'instruction suivante :

```javascript
$('p:first').html('<a href="http://www.google.com">Moteur de recherche Google</a>');
```

Position et taille des éléments

Je vais vous montrer comment connaître et modifier la position et les dimensions des éléments affichés dans une page Web.
Pour gérer la position des éléments dans une page HTML, vous utiliserez les méthodes suivantes :

- `offset()` : position absolue d'un élément dans la page (getter et setter) ;
- `position()` : position relative d'un élément dans son parent (getter seulement).

Les positions retournées par ces méthodes ont deux composantes : l'abscisse `left` et l'ordonnée `top`. Vous utiliserez donc :

- `offset().left` et `offset().top` pour connaître la position absolue d'un élément.
- `position().left` et `position().top` pour connaître la position d'un élément dans son parent.

Pour montrer comment utiliser ces deux méthodes, nous allons utiliser le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Valeurs stockées dans les éléments</title>
    <style type="text/css">
      #parent {
        width: 300px;
        height:300px;
        position: absolute;
        top: 100px;
        left: 200px;
        background-color: yellow;
      }

      #enfant {
        width: 100px;
        height:100px;
        position: absolute;
        top: 150px;
        left: 100px;
        background-color: red;
      }
    </style>
  </head>
  <body>
    <div id="parent">
      Texte dans le parent
      <div id="enfant">
        Texte dans l'enfant
      </div>
    </div>
    <span id="resultat"></span>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Entrer les instructions jQuery ici
      });
    </script>
  </body>
</html>
```

Le corps du document contient deux balises `<div>` imbriquées, d'identifiants respectifs `parent` et `enfant` :

```html
<div id="parent">
  Texte dans le parent
  <div id="enfant">
    Texte dans l'enfant
  </div>
</div>
```

... ainsi qu'une balise `<span>` qui sera utilisée par la suite pour afficher les coordonnées des balises `<div>` :

```html
<span id="resultat"></span>
```

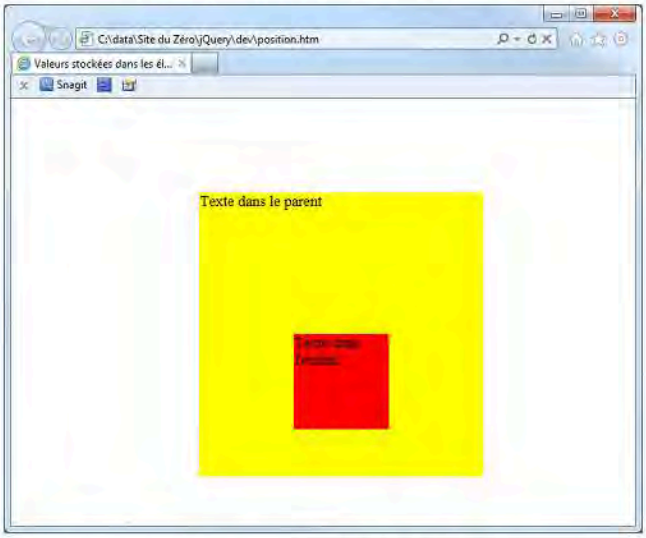
Ces balises sont mises en forme par des règles CSS, entre les lignes 7 et 23. Les dimensions de la balise d'identifiant `parent` sont fixées à 300 pixels sur 300. Cette balise est positionnée de façon absolue à 100 pixels du bord supérieur et à 200 pixels du bord gauche de la page. Enfin, la couleur d'arrière-plan est jaune :

```css
#parent {
  width: 300px;
  height:300px;
  position: absolute;
  top: 100px;
  left: 200px;
  background-color: yellow;
}
```

Les dimensions de la balise d'identifiant `enfant` sont fixées à 100 pixels sur 100. Cette balise est positionnée de façon absolue à 150 pixels du bord supérieur et à 100 pixels du bord gauche de son parent. Oui, vous avez bien lu, de son parent : la balise `enfant` étant imbriquée dans la balise `parent`, le terme `absolute` a une valeur toute… relative (sans vouloir faire un jeu de mots). En effet, le positionnement est bien absolu, si on se réfère à la balise `parent` et non au document dans son ensemble. Enfin, la couleur d'arrière-plan est rouge :

```css
#enfant {
  width: 100px;
  height:100px;
  position: absolute;
  top: 150px;
  left: 100px;
  background-color: red;
}
```

Si vous affichez ce document dans votre navigateur, vous devriez obtenir l'image suivante.



Connaître la position des éléments sur la page

Nous allons ajouter quelques instructions jQuery à partir de la ligne 38 pour afficher les coordonnées absolues des deux balises `<div>` dans la balise `<span>` :

```javascript
var posparent=$('#parent').offset();
var posenfant=$('#enfant').offset();
$('span').text('Parent : x=' + posparent.left + ', y=' + posparent.top + ' Enfant : x=' +
posenfant.left + ', y=' + posenfant.top);
```

La ligne 1 utilise la méthode `offset()` pour connaître les coordonnées absolues de la balise `<div>` `parent`. Ces coordonnées sont mémorisées dans la variable `posparent`.
La ligne 2 est très proche de la ligne 1, à ceci près qu'elle mémorise les coordonnées absolues de la balise `<div>` `enfant` dans la variable `posenfant`.

La ligne 3 affiche les coordonnées absolues des balises `parent` et `enfant` dans la balise `<span>`. La méthode `text()` est utilisée pour insérer du texte dans la balise `<span>`. Les coordonnées `left` et `top` des balises `parent` et `enfant` sont extraites des variables `posparent` et `posenfant`. Par exemple, pour l'abscisse de la balise `parent`, on utilise `posparent.left`.

Pourquoi avoir défini les variables `posparent` et `posenfant` ?

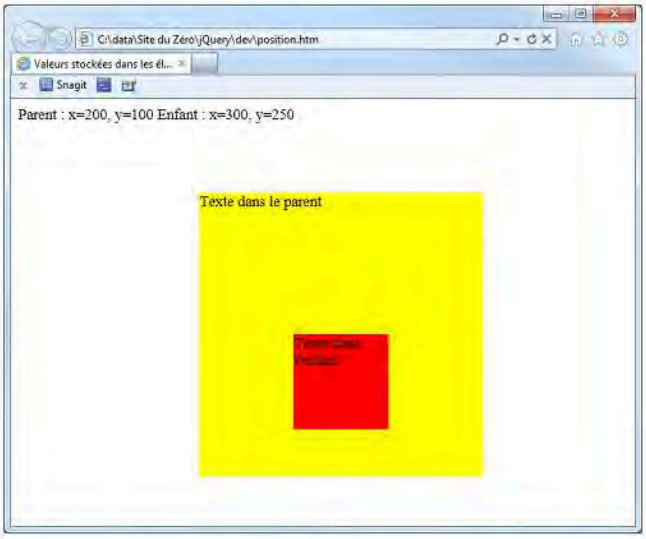
Deux objectifs ont motivé l'écriture de variables intermédiaires :

1. L'instruction `$('span').text()` est ainsi allégée. En effet, il est plus court d'écrire `posparent.left` que `$('#parent').offset().left`. Si vous n'utilisiez pas de variables intermédiaires, l'instruction deviendrait :

```javascript
$('span').text('Parent : x=' + $('#parent').offset().left + ', y=' +
$('#parent').offset().top + ' Enfant : x=' + $('#enfant').offset().left + ', y=' +
$('#enfant').offset().top);
```

2. L'instruction `posparent.left` s'exécute bien plus vite que l'instruction `$('#parent').offset().left`. Le code sera donc optimisé en utilisant des variables intermédiaires.

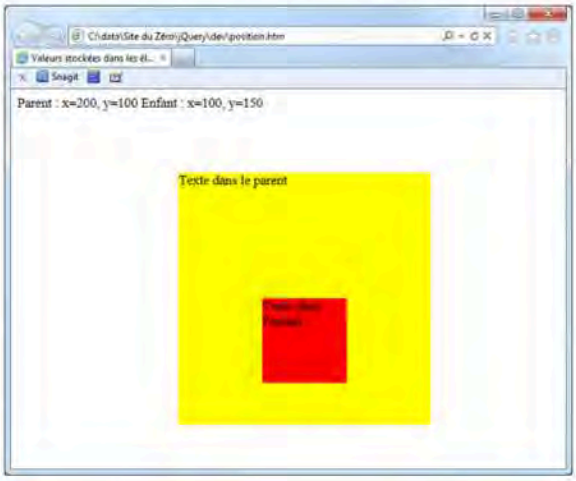
Une fois ces trois lignes de code insérées après la ligne 38, sauvegardez puis affichez le document dans un navigateur. L'image suivante représente ce que vous devriez obtenir.



Remplaçons les méthodes `offset()` par `position()`, sans toucher à l'affichage dans la balise `<span>` :

```javascript
var posparent=$('#parent').position();
var posenfant=$('#enfant').position();
$('span').text('Parent : x=' + posparent.left + ', y=' + posparent.top + ' Enfant : x=' + posenfant.left + ', y=' + posenfant.top);
```

Les coordonnées renvoyées sont relatives au parent de chaque balise. Le parent de la balise `#parent` est le document. Elles ne devraient donc pas changer. Quant au parent de la balise `#enfant`, il s'agit de la balise `#parent`. Ses coordonnées seront donc relatives à cette balise.
Une fois le code modifié et sauvegardé, exécutez le fichier. Vous devriez obtenir l'image suivante.



Modifier la position des éléments

Nous avons vu que la méthode `offset()` pouvait être utilisée en tant que setter, et donc pouvait modifier les coordonnées absolues d'un élément. Pour cela, il suffit d'indiquer les nouvelles coordonnées dans les paramètres de la méthode `offset()`. Par exemple, pour afficher la balise `<div>` `#enfant` aux coordonnées absolues (100,100), voici le code à mettre en place :

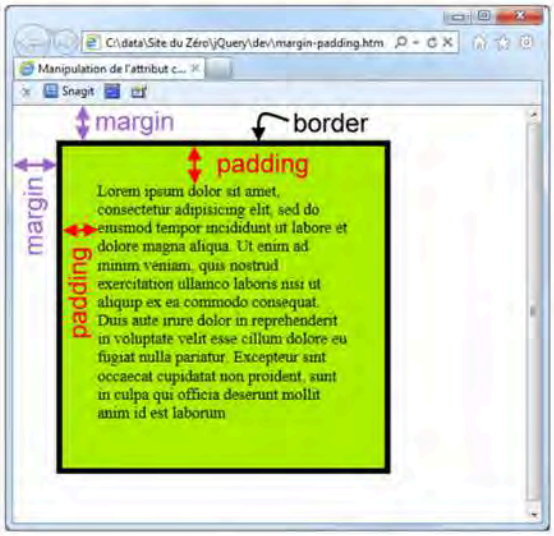
```javascript
var posenfant = $('#enfant').offset();
posenfant.top = 100;
posenfant.left = 100;
$('#enfant').offset(posenfant);
```

La première instruction crée un objet jQuery contenant les coordonnées absolues de la balise `#enfant` et le mémorise dans la balise `posenfant`. Les deux instructions suivantes définissent les nouvelles coordonnées et les affectent aux composantes `top` et `left` de l'objet `posenfant`. Enfin, la quatrième instruction utilise l'objet `posenfant` pour modifier les coordonnées absolues de la balise `#enfant`.

Connaître les dimensions des éléments

Examinez l'image suivante. Elle représente une balise `<div>` dans laquelle ont été définies :

- une marge intérieure (`padding`) ;
- une marge extérieure (`margin`) ;
- une bordure (`border`).



Plusieurs méthodes jQuery permettent de connaître les dimensions et de redimensionner les éléments de type `block` :

- `width()` : largeur de l'élément, de la fenêtre ou du document, sans inclure les marges (`padding` , `border` et `margin`). Cette méthode peut être utilisée comme getter (pour connaître la largeur d'un élément) ou comme setter (pour modifier la largeur d'un élément).
- `innerWidth()` : largeur de l'élément, en incluant le `padding` gauche et droit.
- `outerWidth()` : largeur de l'élément, en incluant le `padding` gauche et droit et `border` .
- `outerWidth(true)` : largeur de l'élément, en incluant `padding` gauche et droit, `border` et `margin` gauche et droit.
- `height()` : hauteur de l'élément, de la fenêtre ou du document, sans inclure les marges (`padding` , `border` et `margin`). Cette méthode peut être utilisée comme getter (pour connaître la hauteur d'un élément) ou comme setter (pour modifier la hauteur d'un élément).
- `innerHeight()` : hauteur de l'élément, en incluant le `padding` supérieur et inférieur.
- `outerHeight()` : hauteur de l'élément, en incluant `border` et `padding` supérieur et inférieur.
- `outerHeight(true)` : hauteur de l'élément, en incluant `border` , `padding` supérieur et inférieur et `margin` supérieur et inférieur.

Voyons comment utiliser ces méthodes en exploitant les propriétés CSS d'une balise `<div>`. Voici le code utilisé :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Manipulation de l'attribut class</title>
    <style type="text/css">
      div {
        width: 250px;
        height: 250px;
        background-color: #AEEE00;
        padding: 35px;
        margin: 35px;
        border-width : 6px;
        border-color: black;
        border-style: solid;
      }
    </style>
  </head>
  <body>
    <div>
    Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
    Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
    Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
    </div>
    <span id="resultat"></span>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Entrer les instructions jQuery ici
      });
    </script>
  </body>
</html>
```

Le corps du document accomplit deux actions :

1. Mise en place d'une balise `<div>` et insertion d'un peu de texte dans cette balise.
2. Mise en place d'une balise `<span>` d'identifiant `#resultat`, dans laquelle les dimensions de la balise `<div>` seront affichées.

Le style de la balise `<div>` est redéfini dans l'en-tête, entre les balises `<style>` et `</style>` :

- Dimensions : `width: 250px; height: 250px;`
- Couleur d'arrière-plan : `background-color: #AEEE00;`
- Marges internes : `padding: 35px;`
- Marges externes : `margin: 35px;`
- Bordure : `border-width : 6px; border-color: black; border-style: solid;`

Pour accéder aux dimensions de la balise `<div>`, nous allons insérer un peu de code jQuery après la ligne 30 :

```javascript
var dimensions = 'width=' + $('div').width() + ', innerWidth=' + $('div').innerWidth() + ', outerWidth=' + $('div').outerWidth() + ', outerWidth(true)=' + $('div').outerWidth(true);
dimensions = dimensions + ', height=' + $('div').height() + ', innerHeight=' + $('div').innerHeight() + ', outerHeight=' + $('div').outerHeight() + ', outerHeight(true)=' + $('div').outerHeight(true);
$('#resultat').text(dimensions);
```

Les deux premières lignes obtiennent les dimensions de la balise `<div>` avec les méthodes `width()`, `innerWidth()`, `outerWidth()`, `outerWidth(true)`, `height()`, `innerheight()`, `outerHeight()` et `outerHeight(true)`. Ces informations sont mémorisées dans la variable `dimensions`. La troisième ligne affiche le contenu de la variable `dimensions` dans la balise `<span>` en utilisant la méthode jQuery `text()`.

Pour améliorer la lisibilité du code, l'affectation à la variable `dimensions` a été scindée en deux. La première ligne interroge toutes les méthodes relatives à la largeur de la balise `<div>` et stocke les valeurs renvoyées dans la variable `dimensions`. La deuxième ligne interroge toutes les méthodes relatives à la hauteur de la balise `<div>` et concatène les valeurs renvoyées à la variable `dimensions`.

Le résultat se trouve à l'image suivante.



Les dimensions sont affichées sur la page

Aux imprécisions près, les valeurs retournées par les méthodes jQuery correspondent bien aux dimensions définies dans le style CSS de la balise `<div>` :

Méthode	Propriété(s) CSS	Valeur
`width()`	width	250
`innerWidth()`	width + padding gauche + padding droit	250 + 35 + 35 = 320
`outerWidth()`	width + padding gauche + padding droit + border gauche + border droit	250 + 35 + 35 + 6 + 6 = 332
`outerWidth(true)`	width + padding gauche + padding droit + border gauche + border droit + margin gauche + margin droit	250 + 35 + 35 + 6 + 6 + 35 + 37 = 402
`height()`	height	250
`innerHeight()`	height + padding supérieur + padding inférieur	250 + 35 + 35 = 320
`outerHeight()`	height + padding supérieur + padding inférieur + border supérieur + border inférieur	250 + 35 + 35 + 6 + 6 = 332
`outerHeight(true)`	height + padding supérieur + padding inférieur + border supérieur + border inférieur + margin supérieur + margin inférieur	250 + 35 + 35 + 6 + 6 + 35 + 35 = 402

Supposons maintenant que vous vouliez modifier les dimensions de la balise `<div>`. Vous utiliserez pour cela les méthodes `width()` et `height()` en tant que setters. Dans cet exemple, les dimensions de la balise `<div>` sont fixées à 400×200 pixels :

javascript

```
1 $('div').width('400px');
2 $('div').height('200px');
```

Associer des données aux balises

Utilisée en setter, la méthode jQuery `$.data()` permet d'associer une ou plusieurs données textuelles à n'importe quel élément du DOM. Oui, vous avez bien lu ! Il peut s'agir de données complémentaires quelconques que vous retrouverez par la suite en utilisant la méthode `$.data()` en getter. Rassurez-vous, ces données ne surchargent pas la mémoire de l'ordinateur : elles sont détruites lorsque l'élément auquel elles sont liées est retiré du DOM ou lorsque la page Web change.

La syntaxe de cette méthode est un peu particulière. Supposons que vous vouliez associer une donnée unique à un élément, vous utiliserez la syntaxe suivante :

javascript
```
$.data(el, 'nom', nom_don: don);
```

… où :

- `el` est le nom de l'élément concerné, sans apostrophes ;
- `nom` est le nom (entre apostrophes) de la « variable » dans laquelle sera stockée la donnée ;
- `nom_don` est le nom (sans apostrophes) associé à la donnée ;
- `don` est une donnée quelconque. S'il s'agit d'un nombre, les apostrophes sont inutiles. Par contre, s'il s'agit d'une chaîne, mettez-la entre apostrophes.

Si vous voulez associer plusieurs données à un élément, vous utiliserez une syntaxe légèrement différente :

javascript
```
$.data(el, 'nom', {nom_don1: don1, nom_don2: don2, nom_don3: don3, etc.});
```

… où :

- `el` est le nom de l'élément concerné, sans apostrophes ;
- `nom` est le nom (entre apostrophes) dans lequel sera stockée la donnée ;
- `nom_don1`, `nom_don2`, `nom_don3`, etc. sont les noms (sans apostrophes) associés aux données ;
- `don1`, `don2`, `don3`, etc. sont des données quelconques. Lorsqu'une de ces données est numérique, ne mettez pas d'apostrophes. Au contraire, lorsqu'une de ces données est une chaîne, mettez-la entre apostrophes.

Pour retrouver une donnée associée à un élément, utilisez la syntaxe suivante :

javascript
```
var uneVariable = $.data(el,'nom').nom_don;
```

… où :

- `uneVariable` est une variable quelconque ;
- `el` est le nom de l'élément (sans apostrophes) auquel une donnée a été associée ;
- `nom` est le nom (entre apostrophes) dans lequel a été stockée la donnée ;
- `nom_don` est le nom de la donnée à retrouver, sans apostrophes.

Si tout ceci est confus pour vous, un petit exemple va vous apporter la lumière ! Dans le code suivant, nous allons associer trois données à une balise `<div>`, retrouver ces données et les afficher dans trois balises `<span>` enfants du `<div>`. Voici le code :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Manipulation de l'attribut class</title>
  </head>
  <body>
  <div id="mondiv">
  <!--Les valeurs stockées dans la balise <div> sont : <span id="sp1"></span>, <span id="sp2"></span> et <span id="sp3"></span>.-->
  </div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        var div = $('div')[0];
        $.data(div, 'mesValeurs', {premier: 'bonjour', deuxieme: 12, troisieme: 'http://www.siteduzero.com'});
        var val1 = $.data(div, 'mesValeurs').premier;
        var val2 = $.data(div, 'mesValeurs').deuxieme;
        var val3 = $.data(div, 'mesValeurs').troisieme;
        $('#sp1').text(val1);
        $('#sp2').text(val2);
        $('#sp3').text(val3);
        });
    </script>
  </body>
</html>
```

Remarquez :

- Les codes HTML `<` et `>` qui apparaissent autour du mot `div`. Ces codes remplacent les caractères < et >. Sans eux, le texte « » serait interprété comme une balise, et le code ne produirait pas l'effet recherché.
- L'emplacement stratégique des balises `<span>`. En y insérant des données, elles complèteront la phrase « Les valeurs stockées […] » d'une façon naturelle, comme si la phrase avait été écrite « en dur ».

La partie intéressante du code se trouve entre les lignes 15 et 22.

Pour alléger l'écriture, la ligne 15 commence par stocker le premier élément `div` dans la variable `div`. Ainsi, il suffira d'écrire `div` à la place de `$('div')[0]` :

```javascript
var div = $('div')[0];
```

La ligne 16 attache trois données nommées « premier », « deuxieme » et « troisieme » à la variable `mesValeurs` et relie cette variable à l'unique balise `<div>` du document. Remarquez les apostrophes utilisées autour des données textuelles « premier » et « troisieme » :

```javascript
$.data(div, 'mesValeurs', {premier: 'bonjour', deuxieme: 12, troisieme: 'http://www.siteduzero.com'});
```

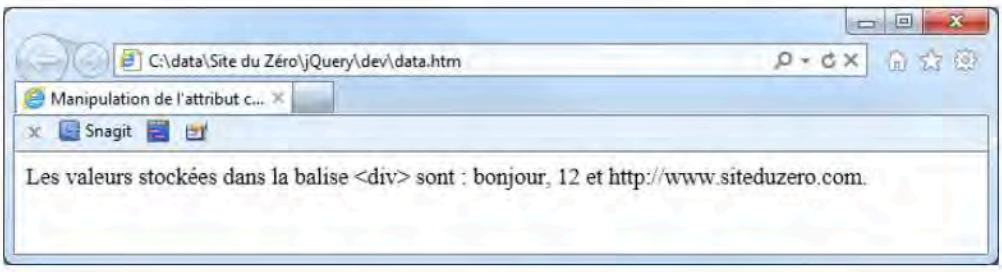
Les lignes 17 à 19 sont construites sur le même modèle. À titre d'exemple, la ligne 17 lit la valeur « premier » de la variable `mesValeurs` associée à la balise `<div>` et la mémorise dans la variable `val1` :

```javascript
var val1 = $.data(div, 'mesValeurs').premier;
```

Les lignes 20 à 22 insèrent les trois valeurs récupérées lignes 17 à 19 dans les balises `<span>` `#sp1`, `#sp2` et `#sp3` :

```javascript
$('#sp1').text(val1);
$('#sp2').text(val2);
$('#sp3').text(val3);
```

Si vous exécutez ce code dans votre navigateur, vous devriez obtenir le même résultat qu'à l'image suivante.



Le code exécuté dans le navigateur

Je vous ai dit que les données associées aux éléments ne surchargeaient pas la mémoire de l'ordinateur, car elles étaient détruites lorsque l'élément auquel elles sont liées est retiré du DOM, lorsque la page Web change ou que le navigateur est fermé.

Je voudrais vous parler d'une troisième technique, moins destructrice que les deux premières puisqu'elle se contente de supprimer les données associées à un élément sans supprimer l'élément et sans changer de page. Il s'agit de la méthode `removeData()` dont voici la syntaxe :

javascript
```
$.removeData(el,'nom');
```

… où :

- `el` est le nom de l'élément (sans apostrophes) auquel les données ont été associées ;
- `nom` est le nom (entre apostrophes) de la « variable » dans lequel les données ont été stockées.

Par exemple, pour supprimer la variable `mesValeurs` associée à la première (et unique d'ailleurs) balise `<div>` de l'exemple précédent, vous utiliserez les instructions suivantes :

javascript
```
var div = $('div')[0];
$.removeData(div, 'mesValeurs');
```

La première instruction définit la variable `div` et y stocke le premier élément `<div>`. La deuxième instruction supprime la variable `mesValeurs` qui était associée à première balise `<div>`.

- Les termes getter et setter correspondent aux méthodes de lecture (getter) et d'écriture (setter) du langage jQuery. Ils viennent des verbes anglais *to get* (obtenir) et *to set* (définir). En jQuery, un getter est une méthode qui obtient une valeur HTML ou CSS en interrogeant le code. Inversement, un setter est une méthode qui modifie une valeur HTML ou CSS dans le code du document.
- Un sélecteur jQuery peut retourner zéro, un ou plusieurs éléments. Lorsqu'aucun élément n'est retourné, le getter renvoie la valeur `undefined` (c'est-à-dire « non défini »). Lorsqu'un seul élément est retourné, le getter renvoie la valeur de cet élément. Les setters reçoivent généralement plusieurs paramètres textuels qui permettent de modifier une propriété CSS, un attribut HTML ou encore d'insérer un élément dans le code.
- Vous utiliserez la méthode `attr()` pour lire, créer et modifier les attributs des balises HTML.
- Vous utiliserez la méthode `css()` pour lire, créer et modifier les propriétés CSS attachées au document.
- La méthode `addClass()` ajoute à la sélection l'attribut `class` dont le nom est indiqué entre les parenthèses. La méthode `removeClass()` supprime un attribut `class` dans une balise. Enfin, la méthode `hasClass()` permet de tester l'existence d'une classe dans les éléments retournés par le sélecteur.
- La méthode `val()` permet de connaître et de modifier la valeur des zones de texte, boutons radio, cases à cocher, listes déroulantes et zones de liste contenues dans un document HTML.
- Les méthodes `text()` et `html()` permettent de travailler avec les valeurs stockées dans des éléments HTML : `text()` retourne/modifie la valeur textuelle stockée dans l'élément, et `html()` retourne/modifie le code HTML stocké dans l'élément.
- Pour connaître/modifier la position absolue d'une balise dans le document, vous utiliserez les méthodes `offset()` et `position()`. Pour connaître les dimensions d'un élément, vous utiliserez les méthodes `width()`, `innerWidth()`, `outerWidth()`, `height()`, `innerHeight()`, et `outerHeight()`.
- En utilisant le langage jQuery, il est possible d'associer des données aux balises HTML. Pour cela, on utilise la méthode `$.data()`. Dans un premier temps, on affecte une ou plusieurs données à la balise en utilisant la méthode `$.data()` en tant que setter. Lorsque cela est nécessaire, la ou les valeurs associées à la balise sont récupérées en utilisant la méthode `$.data()` en getter. Lorsque les données associées à une balise ne sont plus nécessaires, on peut les supprimer avec la méthode `$.removeData()`.

INSÉRER ET REMPLACER DES ÉLÉMENTS DANS LE DOM

Nous allons ici nous intéresser à une autre facette de jQuery : la modification du DOM. Les méthodes qui vont être examinées permettent d'insérer, de remplacer et de supprimer des éléments quelconques dans le DOM, et ainsi de modifier le contenu de la page affichée dans le navigateur.

Insérer du contenu

Toutes les méthodes passées en revue dans cette section seront testées sur le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insertion, copie et suppression de données</title>
  </head>
  <body>
    <h2 id="un">Lorem ipsum</h2>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
      Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </p>
    <hr>

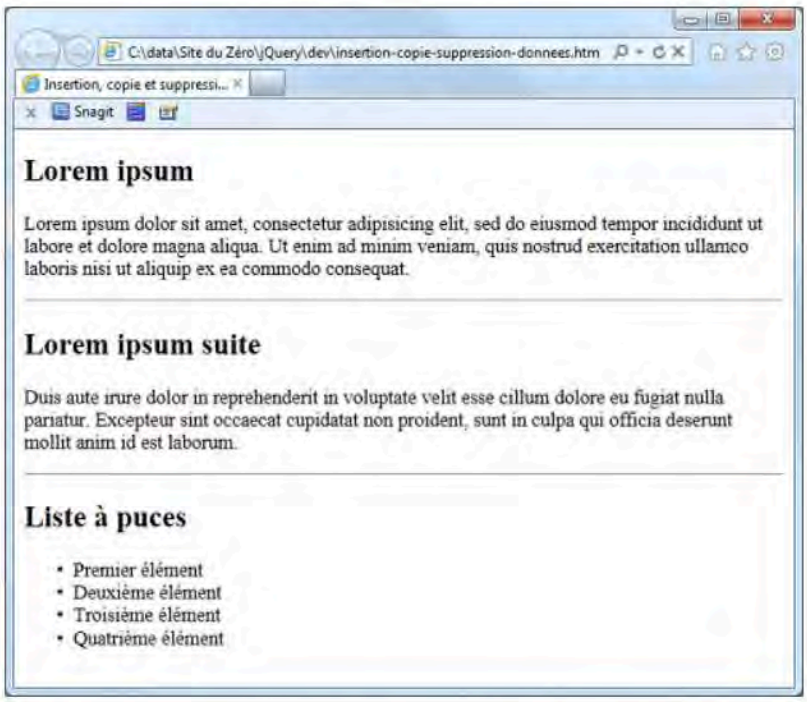
    <h2 id="deux">Lorem ipsum suite</h2>
    <p>
      Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
      Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
    </p>
    <hr>

    <h2 id="trois">Liste à puces</h2>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
      <li>Quatrième élément</li>
    </ul>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Insérer le code jQuery ici
      });
    </script>
  </body>
</html>

```

L'image suivante vous montre comment apparaît la page lorsqu'aucun code jQuery n'a été inséré.



La page HTML avant l'intervention du code jQuery

Les bases de jQuery

1. Sélection d'éléments
2. Plus loin dans la sélection d'éléments
3. Modifier le contenu d'un élément
4. **Insérer et remplacer des éléments dans le DOM**
5. TP : Questionnaire interactif en jQuery

Quiz

Activité : Créez un générateur de formulaires

Accéder au forum

Créé par

OPENCLASSROOMS

OpenClassrooms, 1ère plateforme e-Education en Europe

Plusieurs méthodes très pratiques permettent d'insérer du contenu dans ou en dehors de la sélection (entendez par là des éléments retournés par le sélecteur jQuery) :

- `append()` insère du contenu à la fin de la sélection ;
- `prepend()` insère du contenu au début de la sélection ;
- `before()` insère du contenu avant la sélection ;
- `after()` insère du contenu après la sélection.

Voici quelques exemples d'utilisation de ces méthodes.

Ajout d'une espace et de trois astérisques à la suite de chaque titre <h2>

```javascript
$('h2').append(' ***');
```

Ajout de trois astérisques et d'une espace avant chaque titre <h2>

```javascript
$('h2').prepend('*** ');
```

Ajout d'une ligne de séparation horizontale avant le titre <h2> #trois

```javascript
$('#trois').before('<hr>');
```

Insertion de deux sauts de ligne après chaque balise <hr>

```javascript
$('hr').after('


');
```

Comme vous le voyez, c'est plutôt simple ! Passons donc à la suite.

Remplacer des éléments

Pour remplacer la sélection, utilisez la méthode `replaceWith()` en précisant le nouvel élément entre les parenthèses. Par exemple, pour remplacer les balises `<hr>` par des sauts de ligne, utilisez l'instruction suivante :

```javascript
$('hr').replaceWith('


');
```

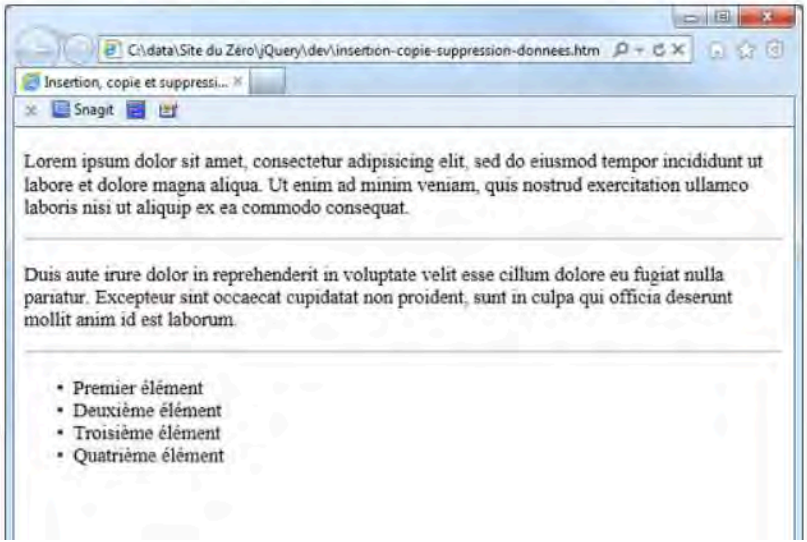
Une petite question à tout hasard : quelle instruction jQuery utiliseriez-vous pour remplacer tous les titres `<h2>` du document par des titres `<h3>` ?

…

Alors, une petite idée ? Peut-être avez-vous songé à cette instruction :

```javascript
$('h2').replaceWith('<h3>');
```

Mais quelle déception lorsque vous avez affiché le document dans votre navigateur ! En effet, vous devriez avoir quelque chose ressemblant à l'image suivante.



Les titres `<h2>` ont tout bonnement disparu ! Comprenez-vous ce qui s'est passé ? Le sélecteur `$('h2')` ne s'est pas contenté de sélectionner les balises `<h2>`, mais également leur contenu et la balise fermante `</h2>`. Les trois titres de niveau 2 ont donc été remplacés par une balise `<h3>`, sans balise fermante, ce qui a provoqué leur disparition.

Mais alors, est-ce que nous nous trouvons face à la première lacune de jQuery ? Bien sûr que non ! La solution consiste à parcourir les éléments sélectionnés avec la méthode `each()` et à effectuer un remplacement `replaceWith()` personnalisé :

javascript
```
$('h2').each(function(){
  var elemH2 = $(this);
  elemH2.replaceWith('<h3>' + elemH2.text() + '</h3>');
});

```

La première ligne sélectionne tous les éléments `<h2>` du document (`$.('h2')`) et applique une fonction à chacun d'entre eux (`each(function(){`).

Pour limiter l'écriture et améliorer les performances du code, la deuxième ligne définit la variable `elemH2` et y mémorise l'élément `<h2>` en cours de traitement.

La troisième ligne applique la méthode `replaceWith()` à l'élément jQuery en cours de traitement (`elemH2.replaceWith`) et le remplace par une balise ouvrante `<h3>`, suivie du texte contenu dans l'élément en cours de traitement (`elemH2.text()`) et d'une balise fermante `</h3>`.

La quatrième ligne met fin à la fonction et à la méthode `each()`.

Insérer des éléments

Vous avez précédemment appris à insérer du contenu dans ou en dehors des éléments sélectionnés par un sélecteur jQuery. Ici, je vais vous montrer comment insérer des éléments dans l'arborescence du DOM. Les méthodes utilisées sont les suivantes :

- `eai.appendTo(cible)` insère un élément à la fin de la cible ;
- `eai.prependTo(cible)` insère un élément au début de la cible ;
- `eai.insertBefore(cible)` insère un élément avant la cible ;
- `eai.insertAfter(cible)` insère un élément après la cible.

`eai` représente l'**é**lément **à**insérer et `cible` représente l'élément avant ou après lequel doit se faire l'insertion.



Pour décrire `eai` et `cible`, vous pouvez utiliser un sélecteur jQuery, un nom d'élément, une chaîne HTML ou un objet jQuery.

Peut-être vous demandez-vous si les méthodes `appendTo()` et `insertAfter()` ne sont pas équivalentes ? Et de même, si les méthodes `prependTo()` et `insertBefore()` ne sont pas équivalentes ?

Eh bien, `appendTo()` insère un élément à la fin de la cible, tout en restant à l'intérieur de cette dernière, alors que `insertAfter()` insère un élément après la cible.
D'une manière identique, `prependTo()` insère un élément au début de la cible tout en restant à l'intérieur de cette dernière, alors que `insertBefore()` insère un élément avant la cible.

Voici quelques exemples d'utilisation de ces méthodes pour mieux les comprendre (ces exemples se basent sur le code HTML présenté au début de ce chapitre). Les instructions jQuery sont insérées après la ligne 32.

Ajout d'un élément de liste à puces après le deuxième élément

javascript
```
$('<li>Deuxième élément bis</li>').insertAfter($('li:nth-child(2)'));

```

Le résultat se trouve à l'image suivante.



La puce « Deuxième élément bis » a été insérée après la puce « Deuxième élément »

Ajout d'une balise <hr> avant chaque titre <h2>

```javascript
$('<hr>').prependTo($('h2'));
```

Le résultat se trouve à l'image suivante.

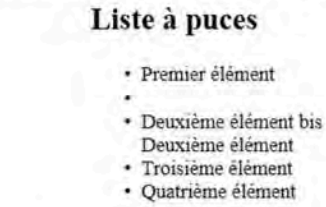


Un séparateur horizontal a été inséré avant chaque titre de niveau 2

Et maintenant, une simple petite question pour vous inciter à réfléchir sur la méthode `prependTo()`. D'après vous, que produit l'instruction suivante ?

```javascript
$('<li>Deuxième élément bis</li>').prependTo($('li:nth-child(2)'));
```

Sans trop y réfléchir, vous pensez certainement qu'une puce « Deuxième élément bis » est ajoutée avant la puce « Deuxième élément ». Pourtant, si vous exécutez le code, vous obtenez l'image suivante.



L'effet de la méthode prependTo() n'est pas celui qui était attendu

D'où vient le problème ?

Rappelez-vous ce que j'ai dit à propos des différences entre les méthodes `prependTo()` et `insertBefore()`. La première effectue une insertion *à l'intérieur* de la cible et la deuxième *avant* la cible. Après l'exécution de l'instruction jQuery, le code HTML de la liste à puces est donc le suivant :

```html
<ul>
  <li>Premier élément</li>
  <li><li>Deuxième élément bis</li>Deuxième élément</li>
  <li>Troisième élément</li>
  <li>Quatrième élément</li>
</ul>
```

Totalement incohérent d'un point de vue sémantique, ce code est mal interprété par le navigateur qui affiche… quelque chose d'assez inattendu ! Ne rejetez pas la faute sur lui : dans ce cas précis, la méthode `insertBefore()` était plus adaptée à la situation.

Déplacer du contenu

Pour déplacer un élément existant dans le document, vous utiliserez les méthodes `append()`, `prepend()`, `before()` ou `after()` :

- `$('sel').append(depl);`
- `$('sel').prepend(depl);`
- `$('sel').before(depl);`
- `$('sel').after(depl);`

… où `sel` sélectionne l'élément avant ou après lequel doit se faire le déplacement et `depl` représente l'élément à déplacer.

Les méthodes `append()` et `after()` sont comparables : elles déplacent toutes deux un élément après un autre élément. Mais attention, avec `append()` le déplacement se fait avant la balise de fin de l'élément sélectionné, alors qu'avec `after()` elle se fait après cette balise.

Les méthodes `prepend()` et `before()` sont également comparables : elles déplacent toutes deux un élément avant un autre élément. Mais attention, avec `prepend()` le déplacement se fait après la balise de début de l'élément sélectionné, alors qu'avec `before()` elle se fait avant cette balise.

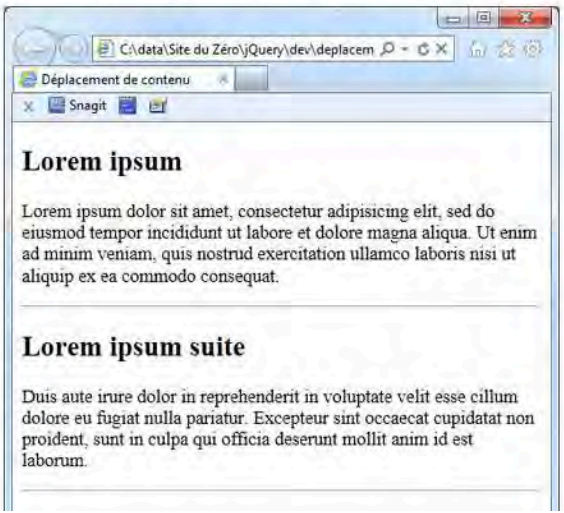
À titre d'exemple, considérons le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Déplacement de contenu</title>
  </head>
  <body>
    <h2>Lorem ipsum</h2>
    <p id="un">
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
      Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </p>
    <hr>

    <h2>Lorem ipsum suite</h2>
    <p id="deux">
      Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
      Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
    </p>
    <hr>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // insérer le code jQuery ici
      });
    </script>
  </body>
</html>
```

Ce code définit deux titres de niveau 2, chacun suivi d'un paragraphe de texte et d'un trait de séparation horizontal, comme le montre l'image suivante.



Remplacez la ligne 25 par l'instruction suivante, sauvegardez le code et rafraîchissez la page dans le navigateur.

```javascript
$('#deux').after($('#un'));

```

Le résultat est à l'image suivante.



Le paragraphe d'identifiant #deux a été cloné et copié avant le paragraphe d'identifiant #un

Si l'élément dupliqué a un ou plusieurs descendants, ils font eux aussi partie du clonage. Nous allons illustrer ce comportement en dupliquant tous les éléments qui composent la liste à puces du code suivant :

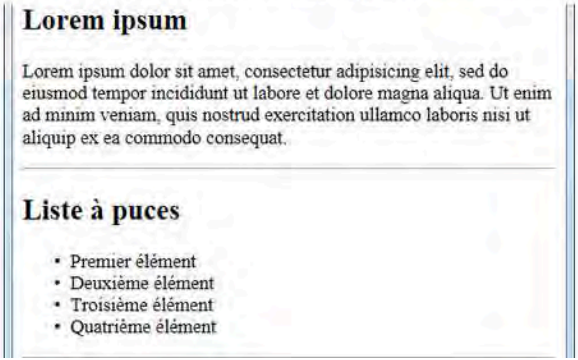
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Déplacement de contenu</title>
  </head>
  <body>
    <h2>Lorem ipsum</h2>
    <p id="un">
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
      Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
    </p>
    <hr>

    <h2 id="trois">Liste à puces</h2>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
      <li>Quatrième élément</li>
    </ul>
    <hr>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Insérer le code jQuery ici
        });
    </script>
  </body>
</html>

```

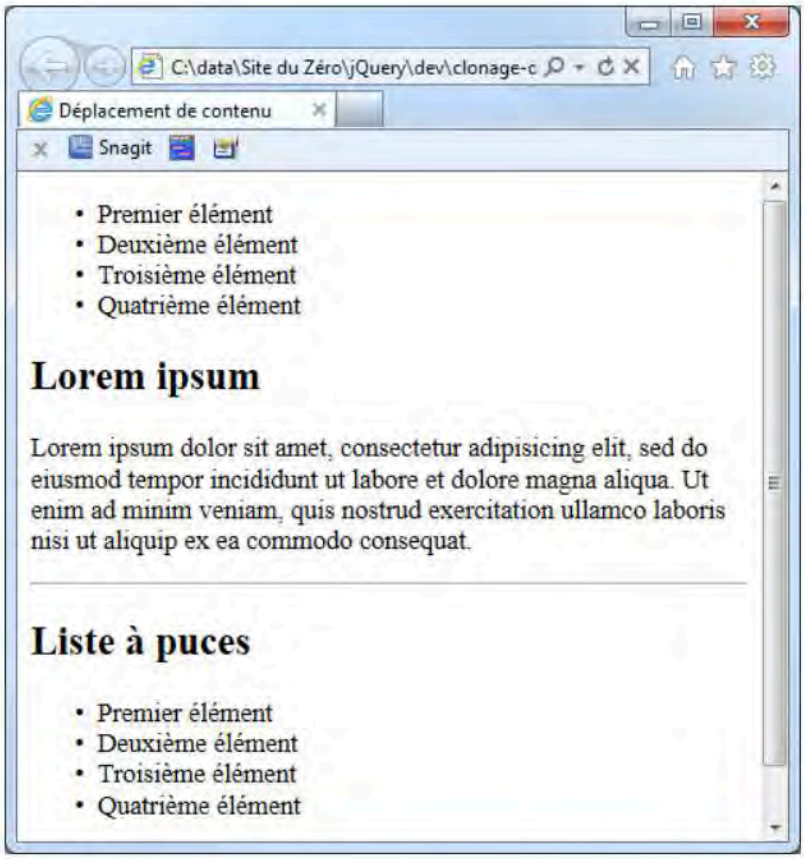
Ce code affiche un titre 2 suivi d'un paragraphe de texte et d'un trait de séparation, puis un autre titre 2 suivi d'une liste à puces composée de quatre éléments, comme le montre l'image suivante.



Insérez l'instruction suivante ligne 27 :

```javascript
$('ul').clone().insertBefore($('h2:first'));
```

Cette instruction sélectionne la balise `<ul>` (`$('ul')`), la duplique (`clone()`) et place le clone avant la première balise `<h2>` (`insertBefore($('h2:first'))`). Admirez le résultat visible à l'image suivante.



La liste à puces a été clonée et copiée avant le premier titre de niveau 2

Lorsque plusieurs éléments sont retournés par le sélecteur, ils font tous partie du clonage. Ainsi par exemple, pour définir le sommaire du document en y faisant figurer tous les titres de niveau 2, vous pourriez utiliser les instructions suivantes :

```javascript
$('<h1>Sommaire</h1>').insertBefore($('h2:first'));
$('h2').clone().insertAfter($('h1'));
```

La première instruction insère le titre de niveau 1 « Sommaire » (`$('<h1>Sommaire</h1>')`) avant le premier titre de niveau 2 du document (`insertBefore($('h2:first'))`).

La deuxième instruction sélectionne tous les titres de niveau 2 du document (`$('h2')`), les clone (`clone()`) et les insère après le titre de niveau 1 (`insertAfter($('h1'))`).

Le résultat se trouve à l'image suivante.



Entourer des éléments

La méthode `wrap()` permet d'entourer un élément par un ou plusieurs autres éléments créés à la volée. Voici sa syntaxe :

```javascript
$('sel').wrap('elwrap');
```

… où `sel` est un sélecteur jQuery quelconque et `elwrap` représente le ou les éléments (ouvrants et fermants) à insérer autour de la sélection. Ces éléments peuvent être du code HTML, un sélecteur, un élément jQuery ou un élément du DOM. Quelle que soit leur nature, ils encadrent les éléments à entourer.

Pour bien comprendre le fonctionnement de cette méthode, nous allons raisonner sur le code suivant :

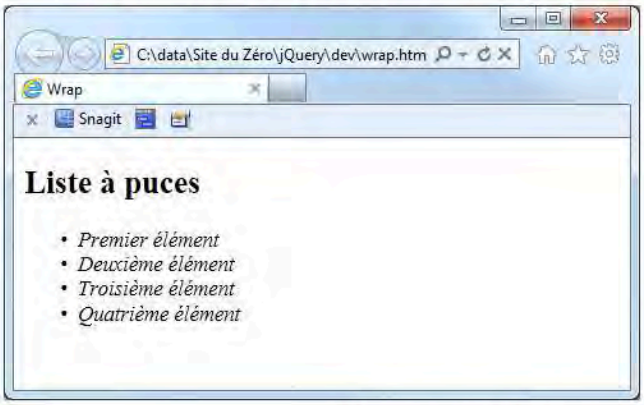
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Wrap</title>
  </head>
  <body>
    <h2 id="trois">Liste à puces</h2>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
      <li>Quatrième élément</li>
    </ul>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Insérer le code jQuery ici
        });
    </script>
  </body>
</html>
```

Ce code affiche un titre de niveau 2 et une liste à puces composée de quatre éléments. Supposons que vous vouliez afficher les éléments de la liste à puces en italique. Pour cela, il vous suffit de les entourer avec la balise `<i>` en utilisant la méthode `wrap()`. Insérez l'instruction suivante ligne 19 :

```javascript
$('li').wrap('<i></i>');
```

Sauvegardez le document et affichez-le dans votre navigateur. Les quatre éléments de la liste à puces sont maintenant affichés en italique, comme le montre l'image suivante.



Chaque élément <li> est en italique

Et si vous voulez afficher les éléments de la liste en rouge, gras, italique et souligné, vous utiliserez l'instruction suivante :

```javascript
$('li').wrap('<font color="red"><b><i><u></u></i></b></font>');
```

Passons à la vitesse supérieure en utilisant une fonction pour personnaliser les éléments insérés autour de la sélection.

La méthode `wrap()` a deux variantes :

- `wrapInner()`, pour entourer le contenu d'un élément par un autre élément créé à la volée ;
- `wrapAll()`, pour entourer d'une façon globale les éléments sélectionnés avec un autre élément créé à la volée.

Pour illustrer le fonctionnement de la méthode `wrapInner()`, supposons qu'une page HTML définisse le paragraphe suivant :

```html
<p>Le texte du paragraphe</p>
```

Si vous exécutez l'instruction jQuery suivante :

```javascript
$('p').wrapInner('<i></i>');
```

… le paragraphe se transforme en :

```html
<p><i>Le texte du paragraphe</i></p>
```

Si vous aviez utilisé la méthode `wrap()` à la place :

```javascript
$('p').wrap('<i></i>');
```

… le paragraphe se serait transformé en :

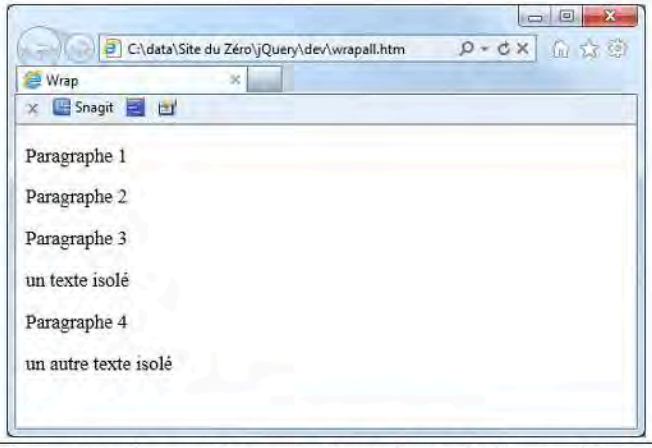
```html
<i><p>Le texte du paragraphe</p></i>
```

Pour illustrer le fonctionnement de la méthode `wrapAll()`, nous allons raisonner sur le code suivant :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Wrap</title>
    <style type="text/css">
      div { background: red;}
    </style>
  </head>

  <body>
    <p>Paragraphe 1</p>
    <p>Paragraphe 2</p>
    <p>Paragraphe 3</p>
    un texte isolé
    <p>Paragraphe 4</p>
    un autre texte isolé

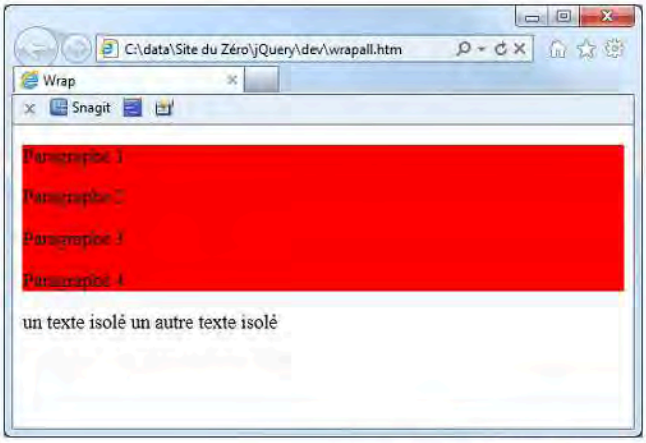
    <script src="jquery.js"></script>
    <script>
      $(function() {
        // Insérer le code jQuery ici
        });
    </script>
  </body>
</html>
```



Comme vous le voyez, les paragraphes et le texte isolé apparaissent dans l'ordre où ils ont été définis. Nous allons maintenant appliquer la méthode `wrapAll()` aux paragraphes (autrement dit aux balises `<p>`) du document et les entourer d'une balise `<div>`. Le style `div` a été défini dans le document : les éléments de ce style auront un arrière-plan rouge. Insérez l'instruction suivante en ligne 22 :

```javascript
$('p').wrapAll('<div></div>');
```

Sauvegardez le document, puis rafraîchissez l'affichage dans le navigateur. Vous devriez obtenir l'image suivante.



Les quatre balises <p> ont été rassemblées

Les paragraphes ont été rassemblés et entourés par une balise `<div>`. Le code HTML a été transformé comme suit :

```html
<div>
  <p>Paragraphe 1</p>
  <p>Paragraphe 2</p>
  <p>Paragraphe 3</p>
  <p>Paragraphe 4</p>
</div>
un texte isolé un autre texte isolé
```

Supprimer des éléments

La méthode `remove()` permet de supprimer les éléments retournés par un sélecteur jQuery. Par exemple, pour supprimer tous les titres `<h2>` du document, utilisez cette instruction :

```javascript
$('h2').remove();
```

Ou encore, pour supprimer la troisième puce dans l'unique liste à puces du document, utilisez l'instruction suivante :

```javascript
$('li:nth-child(2)').remove();
```

Un dernier exemple. Pour supprimer tous les paragraphes qui contiennent le mot « quelconque », utilisez l'instruction suivante :

```javascript
$('p').remove(':contains("quelconque")');
```

- Pour insérer du contenu dans un document, vous utiliserez les méthodes `append()`, `prepend()`, `before()` et `after()`. Pour remplacer des éléments, vous utiliserez la méthode `replaceWith()`. Pour insérer des éléments dans le DOM, vous utiliserez les méthodes `appendTo()`, `prependTo()`, `insertBefore()` et `insertAfter()`. Enfin, pour dupliquer des éléments, vous utiliserez la méthode `clone()`, chaînée à la méthode `appendTo()`, `prependTo()`, `insertBefore()` ou `insertAfter()`.
- Les méthodes `wrap()`, `wrapInner()` et `wrapAll()` permettent d'entourer un élément par un ou plusieurs autres éléments créés à la volée.
- Pour supprimer un élément, vous utiliserez la méthode `remove()`.

PREMIERS PAS AVEC AJAX

Connectez-vous ou inscrivez-vous pour bénéficier de toutes les fonctionnalités de ce cours !

jQuery et AJAX

1. Premiers pas avec AJAX
2. Plus loin avec AJAX
3. TP : Un chat en jQuery
Quiz
Activité : Simplifiez votre générateur avec des messages d'aide

Suivre ce cours

Comment ça marche ?

Accéder au forum

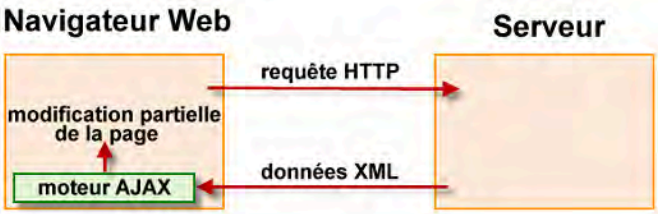
Ce chapitre va aborder un sujet qui fait souvent peur aux programmeurs Web : AJAX. Vous allez voir à quel point jQuery facilite les échanges de données AJAX et il y a fort à parier que vous utiliserez sans aucune appréhension tout ce qui sera écrit dans ce chapitre pour obtenir des pages Web dynamiques, vraiment réactives et qui soulageront dans de grandes mesures les échanges avec le serveur.

Qu'est-ce qu'AJAX ?

Lorsque vous naviguez de page en page sur un site Web traditionnel (entendez par là non-AJAX), les actions de l'internaute se traduisent par les actions suivantes :

1. Envoi d'une requête au serveur afin d'obtenir une nouvelle page.
2. Calcul de la nouvelle page par le serveur et envoi des données HTML/CSS correspondantes.





Charger un fichier

Je vais vous montrer comment charger des informations stockées sur le serveur et mettre à jour un élément de la page actuelle (et juste cet élément) avec ces informations. Pour cela, nous allons utiliser la méthode `load()`, dont voici la syntaxe :

```javascript
$('sel').load('nom_page', function() {
  //une ou plusieurs instructions
});

```

… où :

- `sel` est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
- `nom_page` est le nom d'une page Web quelconque dont le contenu sera utilisé pour effectuer la mise à jour ;
- Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

Charger une partie d'un fichier

En modifiant légèrement la syntaxe de la méthode `load()`, il est possible d'utiliser une partie seulement des données auxquelles donne accès la requête AJAX. Pour cela, il suffit de faire suivre le nom du fichier par une espace et par un sélecteur jQuery :

javascript
```
$('sel').load('nom_page sel2', function() {
  //Une ou plusieurs instructions
});

```

… où :

- `sel` est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
- `nom_page` est le nom d'une page Web quelconque dont le contenu sera utilisé pour effectuer la mise à jour ;
- `sel2` est un sélecteur jQuery quelconque, sans le signe $ ni les parenthèses, qui permettra d'isoler certaines données dans la page `nom_page` ;
- Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

À titre d'exemple, j'ai regroupé les données de mise à jour qui se trouvaient dans les fichiers `maj1.html` et `maj2.html` de l'exemple précédent, je les ai placées dans le fichier `maj.html` et je leur ai affecté un identifiant pour qu'elles soient plus faciles à isoler en jQuery :

html
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>

  <body>
    <div id="modif1">
      Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur? Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?
    </div>
    <img id="modif2" src="paysage2.jpg">
  </body>
</html>

```

Ainsi, le texte à utiliser pour mettre à jour la première balise est facilement identifiable par l'identifiant `#modif1`, et l'image à utiliser pour mettre à jour la deuxième balise est facilement identifiable par l'identifiant `#modif2`. Voici comment doivent être modifiées les deux méthodes événementielles de l'exemple précédent pour n'utiliser qu'une partie du fichier lors de la mise à jour :

javascript
```
$('#majPremier').click(function() {
  $('#premier').load('maj.html #modif1', function() {
    alert('La première zone a été mise à jour');
  });
});

$('#majDeuxieme').click(function() {
  $('#deuxieme').load('maj.html #modif2', function() {
    alert('La deuxième zone a été mise à jour');
  });
});

```

Passer des paramètres à un programme PHP

Je ne pouvais pas faire l'impasse sur les possibilités de passage de paramètres de la méthode `load()`. Cette technique est particulièrement adaptée si vous programmez en PHP. En utilisant jQuery pour créer des adresses URL contenant un ou plusieurs paramètres (`http://site.fr/page.php?id=10&p=2`), vous pourrez interroger une base de données en PHP et retourner des informations qui dépendent des paramètres passés dans l'URL.

Première forme de la méthode `load()`

Pour passer des paramètres à la suite de l'adresse URL avec la méthode `load()`, voici la syntaxe à utiliser :

```javascript
$('sel').load(url,param);
```

… où :

- `sel` est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
- `url` est le nom de la page PHP qui sera utilisée pour faire la mise à jour ;
- `param` est une chaîne qui contient un ou plusieurs couples paramètres/valeurs. Par exemple, si `param` vaut « id=5, p=14 » et `url` vaut `http://site.fr/page.php`, la page utilisée pour la mise à jour sera `http://site.fr/page.php?id=5&p=14`.

Et maintenant, je vous propose de voir comment utiliser cette version de la méthode `load()` sur un cas concret. Que diriez-vous d'afficher des proverbes chinois en utilisant quelques lignes de jQuery associées à un programme écrit en PHP ? Voici le code HTML/jQuery utilisé :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ajax - Load</title>
  </head>

  <body>
    <input type="text" id="ref">
    <button id="action">Afficher</button><br />
    <div id="r">Entrez un nombre compris entre 1 et 10 pour afficher un proverbe chinois</div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          var param = 'l=' + $('#ref').val();
          $('#r').load('http://www.proverbes.php',param);
        });
      });
      </script>
  </body>
</html>
```

Lorsque le bouton est cliqué, le contenu de la zone de texte est lu avec la méthode jQuery `val()` appliquée à la zone de texte `#ref`. La valeur ainsi obtenue est mémorisée dans la variable `param`, précédée du texte « l= » :

```javascript
var param = 'l=' + $('#ref').val();
```

Par exemple, si vous tapez « 5 » dans la zone de texte, la variable `param` contiendra la chaîne « l=5 » après l'exécution de cette instruction.

La ligne suivante passe le paramètre que vous avez saisi au programme `proverbes.php` et met à jour la balise `<div id="r">` en conséquence :

```javascript
$('#r').load('proverbes.php',param);
```

Il ne vous manque plus que le traitement PHP. Je vais vous le donner, mais je ne vais pas l'expliquer, ce n'est pas vraiment le but de ce cours. Sans plus attendre…

```php
<?php
$proverbe = array("On ne rassasie pas un chameau en le nourrissant à la cuillère.",
                  "Connaître son ignorance est la meilleure part de la connaissance.",
                  "Une maison en paille où l'on rit, vaut mieux qu'un palais où l'on pleure.",
                  "Le vrai voyageur ne sait pas où il va.",
                  "Point n'est besoin d'élever la voix quand on a raison.",
                  "Un ami c'est une route, un ennemi c'est un mur.",
                  "Un peu de parfum demeure toujours sur la main qui te donne des roses.",
                  "Si élevé que soit l'arbre, ses feuilles tombent toujours à terre.",
                  "Si ce que tu as à dire n'est pas plus beau que le silence, tais toi.",
                  "Trois coupes de vin font saisir une doctrine profonde.");
$l=$_GET["l"];
if (($l != "") && ($l>0) && ($l<11))
{
  echo "<u>Proverbe chinois N° ".$l."</u><br><br>";
  echo "<b>".$proverbe[$l-1]."</b>";
}
else
  echo "<font color=red>Entrez un nombre compris entre 1 et 10 !</font>";
?>
```

Deuxième forme de la méthode `load()`

Il est également possible de passer un objet en deuxième argument de la méthode `load()`. Par exemple, vous pourriez passer deux couples paramètres/valeurs en utilisant l'instruction suivante :

javascript
```
$('sel').load('http://www.site.com/page.php',{ id:50, nom: 'durand'});

```

Mais attention, dans ce cas, les valeurs sont passées par une requête HTTP `POST`. Elles devront donc être récupérées de la sorte par le programme PHP.



Requête HTTP `POST` ? Mais qu'est-ce que tout ce charabia ?

À chaque manipulation faite par l'utilisateur, le navigateur envoie une requête au serveur contenant une référence à une page Web (`http://www.site.com/page.php?param1=15&param2=rouge` par exemple). Le serveur effectue les calculs nécessaires et renvoie le résultat au navigateur sous forme d'une page Web. Les requêtes peuvent être de type `GET` (apparentes dans l'adresse URL) ou `POST` (absentes de l'adresse URL). Selon la méthode utilisée, les instructions permettant de récupérer les données côté serveur seront différentes.

Requêtes GET et POST

La fonction `$.get()`

En parallèle de la méthode `load()`, vous pouvez utiliser la fonction jQuery `$.get()` pour obtenir des données envoyées par le serveur en utilisant une requête HTTP `GET`. Voici la syntaxe de cette fonction :

javascript
```
$.get(adresse, données, function() {
  // Une ou plusieurs instructions
});

```

… où :

- `adresse` est le nom d'une page Web quelconque dont le contenu sera récupéré par la fonction `get()` ;
- `données` représente les données à passer à la page Web par une requête HTTP `GET` ;
- Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

Passons à la pratique. À l'aide de la fonction `get()`, nous allons modifier le code précédent pour récupérer le code HTML retourné par une page PHP et l'afficher dans une boîte de message. Voici les instructions utilisées :

html
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ajax - Get</title>
  </head>

  <body>
    <button id="action">Lancer la requête HTTP GET</button><br />

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          $.get('proverbes.php?l=9', function(data) {
            alert(data);
          });
        });
      });
      </script>
  </body>
</html>

```

contient l'adresse de la page à exécuter (ici, une page PHP). Le deuxième correspond à la fonction de

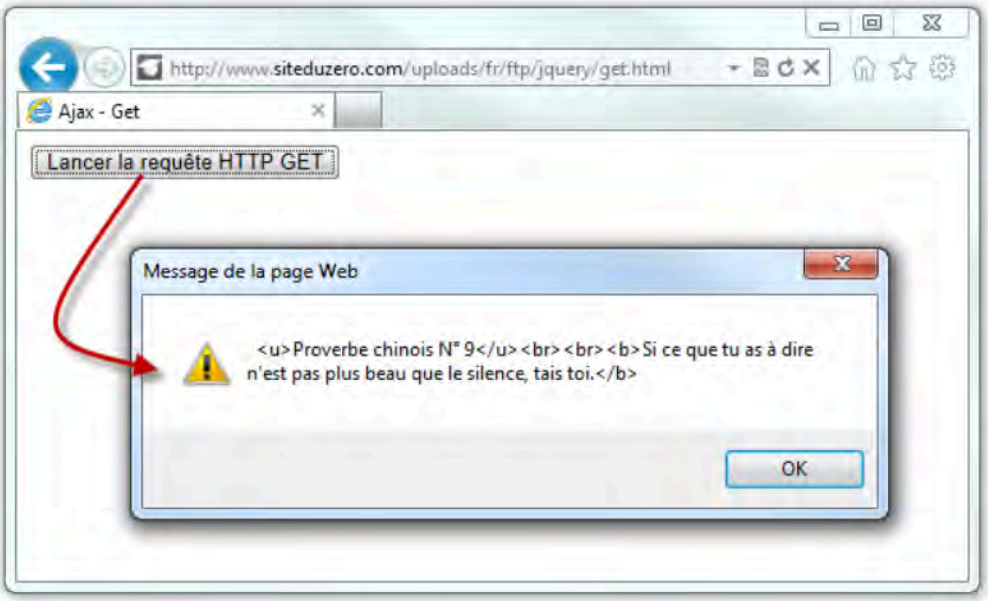
Le corps du document comporte un simple bouton de commande qui déclenchera l'exécution de la méthode `get()`.
Lorsque ce bouton est cliqué, la fonction `get()` est exécutée. Le premier paramètre de la fonction contient l'adresse de la page à exécuter (ici, une page PHP). Le deuxième correspond à la fonction de rappel grâce à laquelle les données retournées par la page PHP seront récupérées :

```javascript
$.get('proverbes.php?l=9', function(data) {

```

Le paramètre `data` ayant été passé en argument de la fonction de rappel, il suffit de l'utiliser pour récupérer les données affichées par la page PHP. Ces données sont alors affichées dans une boîte de message avec la fonction `alert()`.

La figure suivante vous montre le résultat.



Récupération des données PHP et affichage

La fonction `$.post()`

La fonction `$.post()` est toute indiquée si vous voulez envoyer des données de grande taille et/ou sensibles (entendez par là qui contiennent des mots de passe ou d'autres données du même type) au serveur. Par exemple, vous utiliserez la fonction `post()` pour envoyer des données saisies dans un formulaire, qui doivent être stockées dans la base de données du site. Voici le type d'instruction que vous pourriez utiliser :

```javascript
$.post('traiteFormulaire.php', { nom: 'Pierre34', heure: '2pm', post='Un peu de texte récupéré dans un formulaire HTML et destiné à être posté dans un forum.' },
  function(data) {
    alert(data);
  });

```

Ici, le programme `traiteFormulaire.php` est exécuté. Les données saisies dans le formulaire lui sont transmises, et les éléments affichés par le programme PHP sont affichés dans une boîte de message.

Faire patienter l'utilisateur avec une animation

Certaines requêtes AJAX peuvent demander quelques secondes pour s'exécuter. Pour faire patienter la personne qui en est à l'origine, il est courant d'utiliser une image GIF animée, comme celle présentée à la figure suivante.



L'image indique au visiteur qu'il doit attendre quelques secondes

Si vous désirez une image différente, je vous conseille de vous rendre sur le site ajaxload.info pour obtenir une image GIF en parfait accord avec la charte graphique de votre site.

Pour gérer l'affichage de cette image, il suffit d'y faire référence lorsque la requête AJAX est déclenchée, puis de l'effacer lorsque l'exécution de la requête AJAX est terminée.

À titre d'exemple, voici comment a été modifié le code de l'exemple précédent pour afficher une image d'attente entre le début et la fin de la requête AJAX :

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ajax - Load</title>
  </head>

  <body>
    <input type="text" id="ref">
    <button id="action">Afficher</button><br />
    <div id="r">Entrez un nombre compris entre 1 et 10 pour afficher un proverbe chinois</div>

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          $('#r').html('<img src="http://www.mediaforma.com/sdz/jquery/ajax-loader.gif">');
          var param = 'l=' + $('#ref').val();
          $('#r').load('http://www.mediaforma.com/sdz/jquery/data.php',param);
        });
      });
      </script>
  </body>
</html>

```

Qu'est-ce qui a changé, d'après vous ?
Le code HTML est strictement identique. C'est du côté jQuery qu'il faut chercher les différences, ou plutôt la différence, puisqu'une seule instruction a été ajoutée, ligne 17 :

```javascript
$('#r').html('<img src="http://www.mediaforma.com/sdz/jquery/ajax-loader.gif">');

```

Lorsque le bouton est cliqué par l'utilisateur, cette ligne affiche l'animation dans la balise `<div>`. Les lignes 18 et 19 lancent une requête AJAX qui met à jour le contenu de la balise `<div>` et donc efface l'image GIF animée.

- Pour mettre à jour une zone du document avec une page Web (écrite en HTML ou en PHP par exemple), le plus simple consiste à utiliser la méthode `load()`, en précisant le nom de la page entre les parenthèses, éventuellement suivi d'une fonction de rappel. Si elle est précisée, cette fonction sera appelée lorsque les données auront été chargées.

- En modifiant légèrement la syntaxe de la méthode `load()`, il est possible d'utiliser une partie seulement des données auxquelles donne accès la requête AJAX. Pour cela, il suffit de faire suivre le nom du fichier par une espace et par un sélecteur jQuery, sans le signe $, sans les parenthèses et sans les apostrophes.

- Un ou plusieurs couples paramètres/valeurs peuvent être passés à la méthode `load()`. Indiquez-les sous la forme d'une chaîne dans le deuxième paramètre de la fonction.

- Plusieurs fonctions jQuery vont vous permettre d'aller plus loin avec vos requêtes AJAX. Vous pouvez en particulier envoyer des requêtes HTTP `GET` et `POST` au serveur avec les fonctions `$.get()` et `$.post()`, charger et exécuter un script JavaScript avec la fonction `$.getScript()` et des données JSON avec la fonction `$.getJSON()`.

PLUS LOIN AVEC AJAX

Arrivés à ce point dans la lecture du cours, vous savez mettre à jour une partie d'une page Web en utilisant la méthode `load()`. Nous allons maintenant nous intéresser à des fonctions jQuery complémentaires.

Les méthodes se différencient des fonctions, car elles s'appliquent à des éléments obtenus à travers un sélecteur jQuery. Dans `$('sélecteur').meth(paramètres);`, `meth` est une méthode, alors que dans `$.fonc(paramètres);`, `fonc` est une fonction.

Charger un script et des données JSON

Charger un script

La fonction `$.getScript()` permet de charger (de façon asynchrone) puis d'exécuter un fichier JavaScript. Dans sa syntaxe la plus simple, il suffit de préciser l'adresse URL du fichier à charger :

javascript
```
$.getScript('adresse');

```

Dans la deuxième syntaxe, une fonction de retour est précisée en deuxième paramètre de la fonction. Cette fonction est exécutée lorsque le code JavaScript a été chargé et exécuté :

javascript
```
$.getScript('adresse', function() {
  // Une ou plusieurs instructions
});

```

Charger des données codées en JSON

JSON (*JavaScript Object Notation*) est un format de données textuel qui permet de représenter des informations structurées. Voici un exemple de données au format JSON :

```
{
  'menu': 'Fichier',
  'commande': [
      {
          'nomCde': 'Nouveau',
          'action':'CreateDoc'
      },
      {
          'nomCde': 'Ouvrir',
          'action': 'OpenDoc'
      },
      {
          'nomCde': 'Enregistrer sous',
          'action': 'SaveAs'
      }
      {
          'nomCde': 'Fermer',
          'action': 'CloseDoc'
      }
   ]
}
```

Comme vous pouvez le déduire en examinant ce code, un fichier JSON est composé d'un ensemble de paires `'nom':'valeur'` organisées de façon hiérarchique. Ici par exemple, les noms `menu` et `commande` se trouvent au même niveau hiérarchique. Quant aux noms `nomCde` et `action`, il s'agit des enfants du nom `commande`.

Dans vos vies de programmeurs jQuery, vous serez peut-être amenés à manipuler des données au format JSON. Pour cela, vous chargerez le fichier de données JSON avec la fonction `$.getJSON()`, puis vous travaillerez sur les différentes données qui le composent en utilisant la fonction de rappel.

Pour bien comprendre comment accéder aux données d'un fichier codé en JSON, nous allons raisonner sur un exemple simple qui comporte quatre paires `'nom':'valeur'` de même niveau :

```
{
  "nom": "Pierre Durand",
  "age": "27",
  "ville": "Paris",
  "domaine": "HTML5, CSS3, JavaScript"
}
```

Et voici le code HTML/jQuery utilisé pour manipuler ces données :

jQuery et AJAX

1. Premiers pas avec AJAX
2. Plus loin avec AJAX
3. TP : Un chat en jQuery

Quiz

Activité : Simplifiez votre générateur avec des messages d'aide

Suivre ce cours

Comment ça marche ?

Accéder au forum

Créé par

OPENCLASSROOMS

OpenClassrooms, 1ère plateforme e-Education en Europe

Et voici le code HTML/jQuery utilisé pour manipuler ces données :

```html
<button id="charger">Charger et traiter les données</button>
<div id="r">Cliquez sur "Charger et traiter les données" pour lancer la lecture et le traitement des données JSON</div>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('#charger').click(function() {
        $.getJSON('fichier.json', function(donnees) {
        $('#r').html('<p><b>Nom</b> : ' + donnees.nom + '</p>');
        $('#r').append('<p><b>Age</b> : ' + donnees.age + '</p>');
        $('#r').append('<p><b>Ville</b> : ' + donnees.ville + '</p>');
        $('#r').append('<p><b>Domaine de compétences</b> : ' + donnees.domaine + '</p>');
      });
    });
  });
</script>

```

Lorsque le bouton est cliqué, la fonction `getJSON()` est exécutée pour charger le fichier de données `fichier.json` :

```javascript
$.getJSON('fichier.json', function(donnees) {

```

Le deuxième paramètre de la fonction `getJSON()` correspond à la fonction de rappel. Cette fonction est exécutée lorsque le fichier de données a été entièrement chargé. Remarquez le mot `donnees` passé comme paramètre de la fonction de rappel. C'est par son intermédiaire que les données JSON seront accessibles.

Dans un premier temps, la valeur correspondant au nom (`donnees.nom`) est extraite du fichier de données et placée sous une forme HTML (`html()`) dans la balise `<div>` `#r` . Comme nous passons par la méthode `html()` pour remplir la balise `<div>` , il est possible d'utiliser des attributs de mise en forme. Ici, le mot « Nom » est mis en gras avec la balise HTML `<b>` :

```javascript
$('#r').html('<p><b>Nom</b> : ' + donnees.nom + '</p>');

```

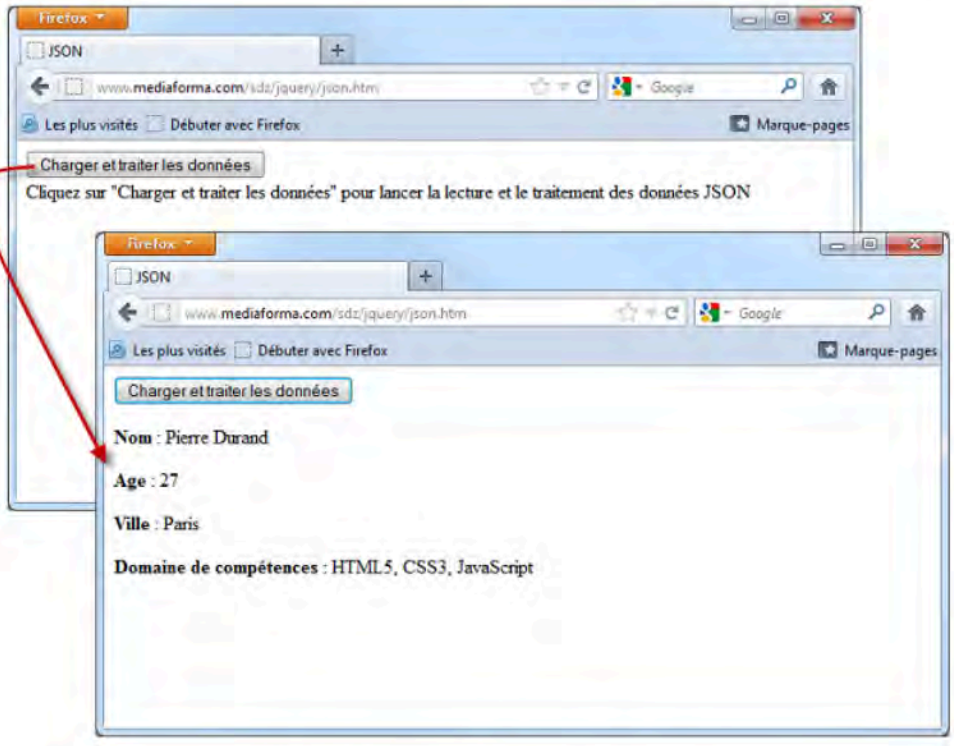
La donnée `age` (`donnees.age`) est alors extraite du fichier de données et placée à la suite du nom, dans un nouveau paragraphe. Ici aussi, le nom du champ est mis en gras en utilisant la balise HTML `<b>` .

Deux instructions similaires extraient les données `ville` et `domaine` du fichier de données JSON et les affichent à la suite du nom et de l'âge :

```javascript
$('#r').append('<p><b>Ville</b> : ' + donnees.ville + '</p>');
$('#r').append('<p><b>Domaine de compétences</b> : ' + donnees.domaine + '</p>');

```

La figure suivante représente le rendu de ce code dans un navigateur.



Récupération des données dans le fichier JSON

La fonction $.ajax()

J'ai gardé le meilleur pour la fin : la fonction `$.ajax()` ! Tout comme les méthodes et fonctions AJAX étudiées jusqu'ici, `$.ajax()` permet d'envoyer des requêtes HTTP AJAX à un serveur Web. Ce qui la différencie de ses « collègues », c'est la finesse des paramètres qui peuvent lui être communiqués. N'ayez crainte, je vais tout vous expliquer et vous jugerez en toute connaissance de cause si oui ou non vous avez besoin de cette fonction. Deux syntaxes sont possibles :

javascript
```
$.ajax(adresse, {options});

$.ajax({options});

```

… où `adresse` est l'adresse à laquelle la requête doit être envoyée, et `options` correspond à une ou plusieurs des options suivantes :

- `type` : type de la requête, `GET` ou `POST` (`GET` par défaut).
- `url` : adresse à laquelle la requête doit être envoyée.
- `data` : données à envoyer au serveur.
- `dataType` : type des données qui doivent être retournées par le serveur : `xml` , `html` , `script` , `json` , `text` .
- `success` : fonction à appeler si la requête aboutit.
- `error` : fonction à appeler si la requête n'aboutit pas.
- `timeout` : délai maximum (en millisecondes) pour que la requête soit exécutée. Si ce délai est dépassé, la fonction spécifiée dans le paramètre `error` sera exécutée.

Beaucoup d'autres options peuvent être utilisées. Pour en avoir une liste exhaustive, consultez la documentation officielle.

Que diriez-vous de passer à la pratique pour voir comment utiliser cette fonction ? Je vous propose de reprendre l'exemple du chapitre précédent et de remplacer la fonction `$.get()` par `$.ajax()` . Voici le code utilisé :

html
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ajax - La fonction ajax()</title>
  </head>

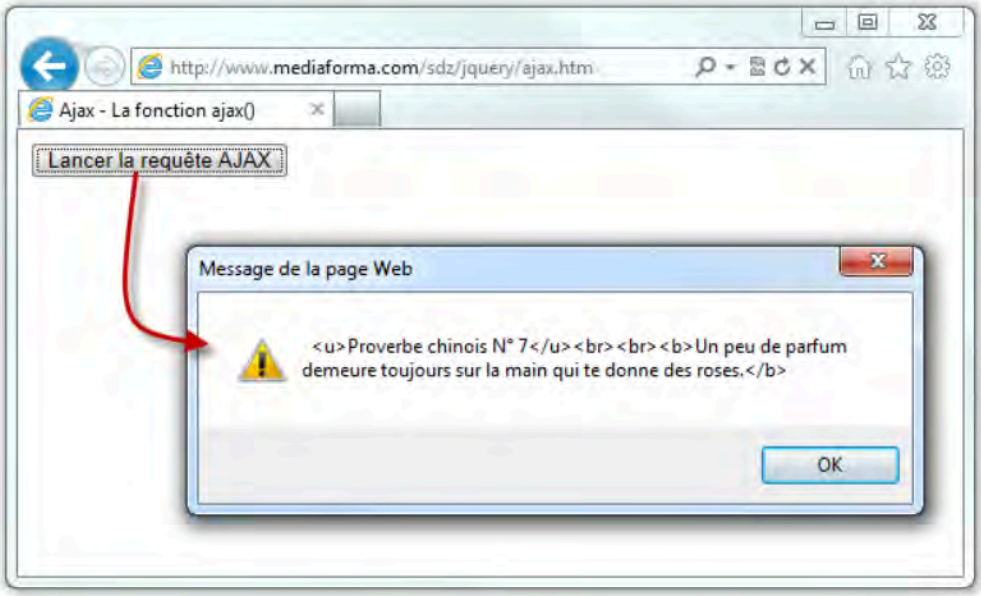
  <body>
    <button id="action">Lancer la requête AJAX</button><br />

    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          $.ajax({
            type: 'GET',
            url: 'proverbes.php?l=7',
            timeout: 3000,
            success: function(data) {
              alert(data); },
            error: function() {
              alert('La requête n\'a pas abouti'); }
          });
        });
      });
      </script>
  </body>
</html>

```

Le corps du document contient un bouton de commande qui sera utilisé pour exécuter la requête AJAX. Lorsque ce bouton est cliqué, la fonction `$.ajax()` est lancée pour exécuter la requête AJAX. Cette requête est de type `GET` (ce paramètre aurait pu être omis, puisqu'il s'agit de la valeur par défaut). La page invoquée est définie dans le paramètre `url` et le délai maximal d'exécution l'est dans le paramètre `timeout` . Ici, un délai de 3000 millisecondes est accordé au programme PHP pour fournir ce qui lui est demandé.

Si la requête aboutit, les données renvoyées par le programme PHP sont affichées dans une boîte de message. Dans le cas contraire, un message d'erreur est affiché.
Voici un exemple d'exécution de cette page :



Exécution de la fonction $.ajax() pour récupérer des données sur le serveur

Que pensez-vous de la fonction `$.ajax()` ? Personnellement, j'ai tendance à lui préférer la méthode `load()` et les fonctions `$.get()`, `$.post()`, `$.getScript()` et `$.getJSON()`. Mais ce n'est qu'une affaire de goût ! Peut-être préférerez-vous vous concentrer sur une seule fonction pour toutes vos requêtes AJAX plutôt que d'apprendre à utiliser plusieurs méthodes et fonctions.

Événements associés à une requête AJAX

Dans la section précédente, vous avez appris à utiliser la fonction `success()` pour exécuter du code lorsqu'une requête AJAX a abouti, et la fonction `error()` pour exécuter du code lorsqu'une requête AJAX n'a pas abouti. Dans cette section, vous allez découvrir une autre technique permettant d'exécuter du code à différentes étapes de l'exécution d'une requête AJAX. Cette technique repose sur la mise en place de méthodes de gestion événementielle. Le tableau suivant donne un aperçu des méthodes utilisables :

Méthode	Événement
`$(document).ajaxSend(function(ev, req, options))`	Requête sur le point d'être envoyée
`$(document).ajaxStart(function())`	Début d'exécution de la requête
`$(document).ajaxStop(function())`	Fin de la requête
`$(document).ajaxSuccess(function(ev, req, options))`	La requête a abouti
`$(document).ajaxComplete(function(ev, req, options))`	La requête est terminée
`$(document).ajaxError(function(ev, req, options, erreur))`	La requête n'a pas abouti

... où :

- `ev` représente l'événement ;
- `req` représente la requête ;
- `options` contient les paramètres passés à la requête ;
- `erreur` est le nom de l'erreur détectée par jQuery.

Voici un peu de code pour vous aider à comprendre comment utiliser ces méthodes événementielles et dans quel ordre elles sont exécutées :

```html
<button id="action">Lancer la requête AJAX</button><br /><br />
<div id="donnees" style="background-color: yellow"></div><br />
<div id="message"></div>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('#action').click(function() {
      $(document).ajaxStart(function() {
        $('#message').html('Méthode ajaxStart exécutée<br>');
      });
      $(document).ajaxSend(function(ev, req, options){
        $('#message').append('Méthode ajaxSend exécutée, ');
        $('#message').append('nom du fichier : ' + options.url + '<br>');
      });
      $(document).ajaxStop(function(){
        $('#message').append('Méthode ajaxStop exécutée<br>');
      });
      $(document).ajaxSuccess(function(ev, req, options){
        $('#message').append('Méthode ajaxSuccess exécutée<br>');
      });
      $(document).ajaxComplete(function(ev, req, options){
        $('#message').append('Méthode ajaxComplete exécutée<br>');
      });
      $(document).ajaxError(function(ev, req, options, erreur){
        $('#message').append('Méthode ajaxError exécutée, ');
        $('#message').append('erreur : ' + erreur + '<br>');
      });
      $('#donnees').load('affiche.htm');
    });
  });
</script>

```

Lorsque le bouton `#action` est cliqué, plusieurs méthodes de gestion événementielle sont mises en place. Par exemple, la méthode `ajaxStart()` capture l'événement « début d'exécution de la requête » :

```javascript
$(document).ajaxStart(function() {

```

Lorsque cet événement est déclenché, un texte est affiché dans la balise `<div id="message">` :

```javascript
$('#message').html('Méthode ajaxStart exécutée<br>');

```

Les autres méthodes de gestion événementielle sont comparables. Notez simplement l'affichage du nom du fichier dans la méthode `ajaxSend()` :

```javascript
$('#message').append('nom du fichier : ' + options.url + '<br>');

```

Et l'affichage du message d'erreur dans la méthode `ajaxError()` :

```javascript
$('#message').append('erreur : ' + erreur + '<br>');

```

Une fois les méthodes de gestion événementielle définies, le contenu du document `affiche.htm` est chargé et inséré dans la balise `<div id="donnees">` :

```javascript
$('#donnees').load('affiche.htm');

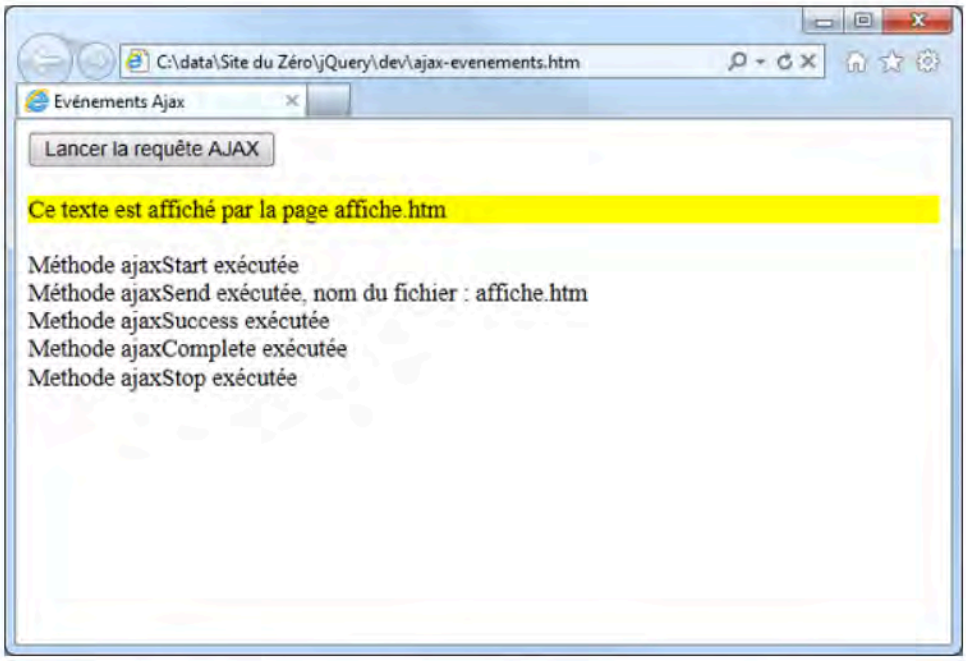
```

Le document `affiche.htm` est élémentaire : il se contente d'afficher un peu de texte dans le document :

```html
Ce texte est affiché par la page affiche.htm

```

Vous devriez obtenir quelque chose ressemblant à la figure suivante.



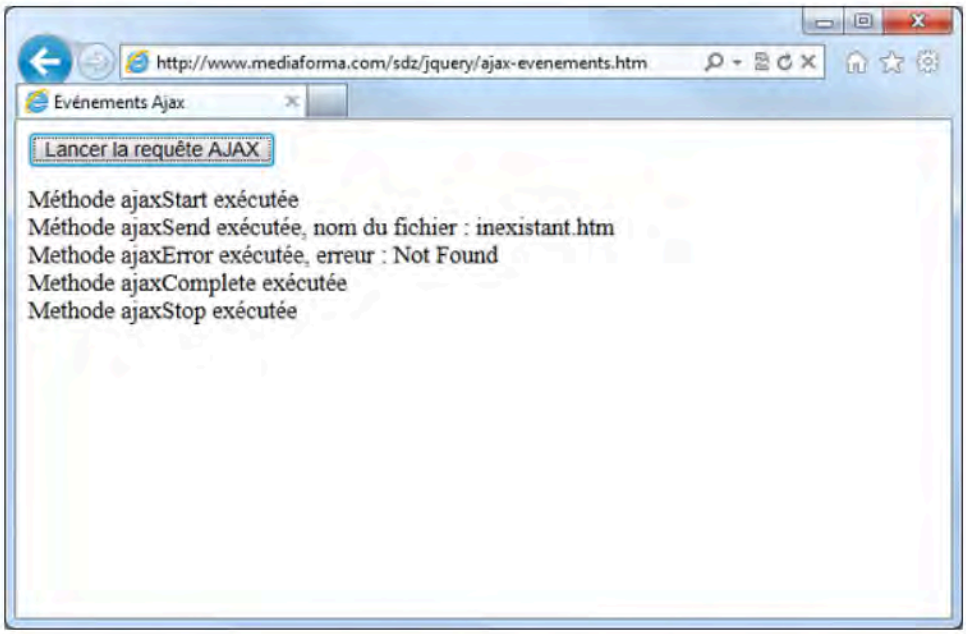
Cette exécution montre l'ordre dans lequel sont levés les événements en rapport avec la requête AJAX

Essayons de modifier le nom du fichier pour faire référence à un fichier inexistant :

```javascript
$('#donnees').load('inexistant.htm');

```

Un clic sur le bouton de commande déclenche l'exécution d'événements légèrement différents, comme le montre la figure suivante.



Le fichier inexistant.htm n'a pas été trouvé, ce qui a déclenché l'événement ajaxError

- Plusieurs fonctions jQuery vont vous permettre d'aller plus loin avec vos requêtes AJAX. Vous pouvez en particulier envoyer des requêtes HTTP `GET` et `POST` au serveur avec les fonctions `$.get()` et `$.post()`, charger et exécuter un script JavaScript avec la fonction `$.getScript()` et des données JSON avec la fonction `$.getJSON()`.

- La fonction `$.ajax()` pourra vous être utile si vous voulez paramétrer finement vos requêtes AJAX. Elle permet de définir plusieurs fonctions de rappel pour réagir à différents événements liés à la requête.

- Enfin, vous pouvez mettre en place des méthodes de gestion événementielle pour capturer les événements : `ajaxSend()`, `ajaxStart()`, `ajaxStop()`, `ajaxSuccess()`, `ajaxComplete()` et `ajaxError()`.