JQUERY

Installer jQuery

jQuery est une bibliothèque JavaScript. En d'autres termes, un fichier d'extension .js. Pour l'utiliser dans une page HTML, il vous suffit d'y faire référence en utilisant une balise <script>, comme ceci :

```
<script type="text/javascript" src="jquery.js" ></script>
```

Dans cet exemple, l'attribut src vaut jquery.js. Comme vous pouvez le voir, l'emplacement du fichier n'est pas précisé dans l'attribut. Cela signifie qu'il devra se trouver dans le même dossier que le document HTML.

Deux solutions s'offrent à nous pour installer la bibliothèque jQ:

1. Si votre ordinateur n'est pas toujours relié à Internet et/ou si votre connexion Internet n'est pas rapide, vous pouvez télécharger la bibliothèque jQuery dans un dossier quelconque et y faire référence localement.
2. Si votre ordinateur est toujours connecté à Internet, vous pouvez faire référence à la bibliothèque jQuery en indiquant une adresse Web.

Téléchargement de jQuery sur votre ordinateur

Rendez-vous sur http://code.jquery.com/ et téléchargez la dernière version en date en cliquant sur le lienjquery-version.js, dans le cadre Last Stable Versions.

*Sur le site officiel, deux versions de la bibliothèque jQuery sont proposées : jquery-*version.*js etjquery.min.js.*

Ces deux fichiers sont strictement identiques d'un point de vue fonctionnel. Par contre, le deuxième a une taille inférieure au premier. Pour cela, les espaces, tabulations, et commentaires y ont été supprimés, et les noms des variables et des fonctions ont été raccourcis, comme le montre la figure suivante.

En règle générale, vous utiliserez le fichier jquery-version.js en développement et le fichier jquery-version.min.js en production, c'est-à-dire lorsque votre code aura été testé, débogué et placé sur le Web.

Vous aussi, vous pourrez minimiser vos scripts jQuery pour qu'ils se chargent plus vite. Pour cela vous utiliserez Google Closure Compiler ou YUI Compressor.

Placer dans votre projet le fichier jquery-version.js (en générale on place ce fichier dans un sous répertoire js ou jquery).

Faire référence à jQuery en ligne

Le plus simple consiste à faire référence au fichier jquery.js sur un **CDN** (pour *Content Delivery Network*). Constitués d'ordinateurs reliés en réseau *via* Internet, ces éléments d'infrastructure coopèrent pour mettre à disposition aussi vite que possible la bibliothèque jQuery. Vous pouvez utiliser les CDN jQuery, Google ou Microsoft. Voici les adresses correspondantes :

CDN	Version non minimisée	Version minimisée
jQuery	http://code.jquery.com/jquery.js	http://code.jquery.com/jquery.min.js
Google	https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.js	https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jquery.min.js
Microsoft	https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.2.js	https://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.7.2.min.js

Les CDN jQuery et Google donnent directement accès à la dernière version de jQuery. Quant au CDN Microsoft, vous devez préciser la version à utiliser dans l'adresse. Cela peut être pratique si vous voulez utiliser une version particulière de jQuery. Le cas échéant, rendez-vous sur https://developers.google.com/speed/libraries/#libraries.

L'utilisation d'un CDN est intéressante en production, c'est-à-dire lorsque votre code jQuery a été testé et est hébergé sur un serveur Web. En effet, en faisant référence à un fichier externe à votre site, vous n'entamez pas sa bande passante. D'autre part, étant donné que les CDN ont une grande bande passante, ils sont très réactifs. Enfin, le fichier jquery.min.js issu d'un CDN est bien souvent déjà présent dans la mémoire cache du navigateur. Ces trois raisons font que votre page se chargera plus rapidement.
Par contre, en phase de développement, c'est-à-dire lorsque vous mettez au point votre code jQuery sur votre ordinateur local, il est conseillé de télécharger le fichier jquery-version.js et d'y faire référence localement. En effet, même si les CDN ont une excellente bande passante, l'utilisation d'un fichier local sera bien plus rapide.

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.

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. Avant d'utiliser jQuery il faut donc attendre que l'arbre soit complet. Cela se traduit par le code suivant :

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

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

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

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

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

Ces trois instructions sont strictement équivalentes. On pourra utiliser systématiquement la troisième, car c'est la plus courte et la plus simple à écrire.

Si l'on 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.

SÉLECTION D'ÉLÉMENTS

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.
- 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, on peut sélectionner les nœuds DOM qui nous intéressent, en utilisant la syntaxe $(sélection) où sélection représente un sélecteur CSS.

Voici quelques sélecteurs CSS :

- 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 , 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.
- Supposons maintenant que vous ayez défini une liste à puces et une liste numérotée
. 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.

Exemple pour mieux comprendre comment utiliser les sélecteurs CSS.

```
<!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 type="text/javascript" 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.bleu	$('ul.bleu')	La balise de classe bleu
div ul	$('div ul')	La balise contenue dans la balise
div ul li[class="pair"]	$('div ul li[class="pair"]')	La balise contenue dans une balise , elle-même contenue dans une balise , et dont l'attributclass vaut pair
li[class]	$('li[class]')	Les balises qui possèdent un attribut class
li[class="impair"]	$('li[class="impair"]')	Les balises 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 attributclass 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 :

```
$(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 d'identifiant resultat, vous utiliserez le code suivant :

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

La méthode html() n'est qu'une des nombreuses méthodes utilisables en jQuery.

Notions indispensables

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.

Voici un exemple de fonction de rappel, appliquée aux éléments de classe rouge :

```
$(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 :

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

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

La méthode css()permet de modifier le css d'un élément. 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.

La méthode css() n'est qu'une des nombreuses méthodes utilisables en jQuery.

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

Exemple avec des listes imbriquées :

```
<!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() {
        // Le code jQuery sera inséré ici
      });
    </script>
  </body>
</html>
```

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

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

… où sel est un des sélecteur CSS du tableau suivant :

Critère	Sélecteur	Élément modifié
Éléments ul directement descendants d'élémentsli	$('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

Critère	Sélecteur	Élément modifié
Deuxième enfant li	$('li:nth-child(2)')	3, 4

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, ett. 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

Exemple avec le code HTML suivant :

```
<!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 jQuerycss().

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

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

Exemple avec le code HTML suivant :

```
<!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 jQuerycss().

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 :

```
<!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 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>.

MODIFIER LE CONTENU D'UN ÉLÉMENT

Getters et setters

Observez les deux instructions suivantes :

```
$('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 :

```
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 :

```
<!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>
```

Deux liens hypertextes sont définis lignes 9 et 10. Le premier pointe sur la page http://api.jquery.comet 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.

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 on peut 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 :

```
$('#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() :

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

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 targetest 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').attr('class', 'madiv'); modifie ou crée l'attribut class dans les balises 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 ».

Voyons maintenant comment supprimer un attribut dans une balise ou un ensemble de balises. Pour cela, vous utiliserez la méthode removeAttr() :

```
$(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 :

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

Accéder aux propriétés CSS

La méthode jQuery css() 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 :

```
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 :

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

Travailler avec l'attribut class

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.

Exemple avec le code suivant :

```
<!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 . 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.

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

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

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

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

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

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

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 :

```
$('#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 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 :

```
$('#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 du document. Ainsi par exemple, cette instruction affiche le prénom Pierre en caractères rouges de taille standard :

```
$('#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 :

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

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

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

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 :

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

Peut être simplifiée comme suit :

```
$('#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 suivante soit définie :

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

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

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

```
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 suivante :

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

L'instruction $('#jean').is('.grand.rouge'); renverra la valeur true, car le est de classe grand et rouge. Par contre, l'instruction $('#jean').is('.petit.rouge'); renverra la valeur false, car le 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 :

```
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 :

```
<!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.

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.

Exemple :

```
<!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());

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 :

```
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 :

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

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

Instructions	Résultat
	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 :

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

Position et taille des éléments

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 :

```
<!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 imbriquées, d'identifiants respectifs parent et enfant :

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

… ainsi qu'une balise qui sera utilisée par la suite pour afficher les coordonnées des balises :

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

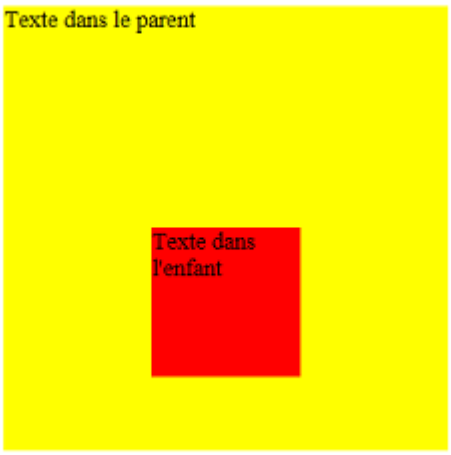
Ces balises sont mises en forme par des règles CSS, entre les lignes 7 et 23. Les dimensions de la balise d'identifiant parent sont fixées à 300 pixels sur 300. Cette balise est positionnée de façon absolue à 100 pixels du bord supérieur et à 200 pixels du bord gauche de la page. Enfin, la couleur d'arrière-plan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

```
#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. Enfin, la couleur d'arrière-plan est rouge :

```
#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.



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

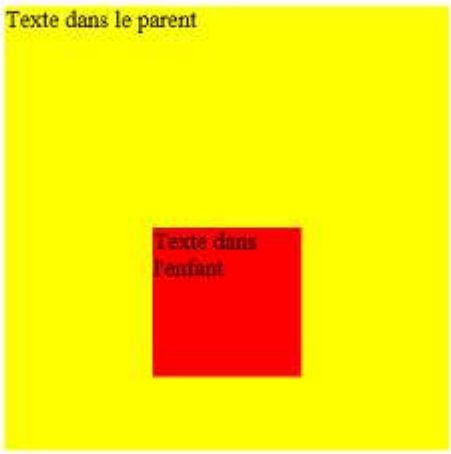
```
var posparent=$('#parent').offset();
var posenfant=$('#enfant').offset();
$('span').text('Parent : x=' + posparent.left + ', y=' + posparent.top + ' Enfant : x=' +
posenfant.left + ', y=' + posenfant.top);
```

La ligne 1 utilise la méthode offset() pour connaître les coordonnées absolues de la balise 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 enfant dans la variable posenfant.

La ligne 3 affiche les coordonnées absolues des balises parent et enfant dans la balise . La méthode text() est utilisée pour insérer du texte dans la balise . 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.

L'image suivante représente ce que vous devriez obtenir.

Parent : x=200, y=100 Enfant : x=300, y=250

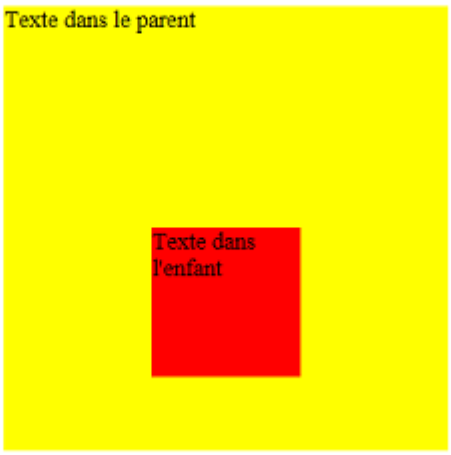


Remplaçons les méthodes offset() par position(), sans toucher à l'affichage dans la balise :

```
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.
L'image suivante représente ce que vous devriez obtenir.

Parent : x=200, y=100 Enfant : x=100, y=150



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 #enfant aux coordonnées absolues (100,100), voici le code à mettre en place :

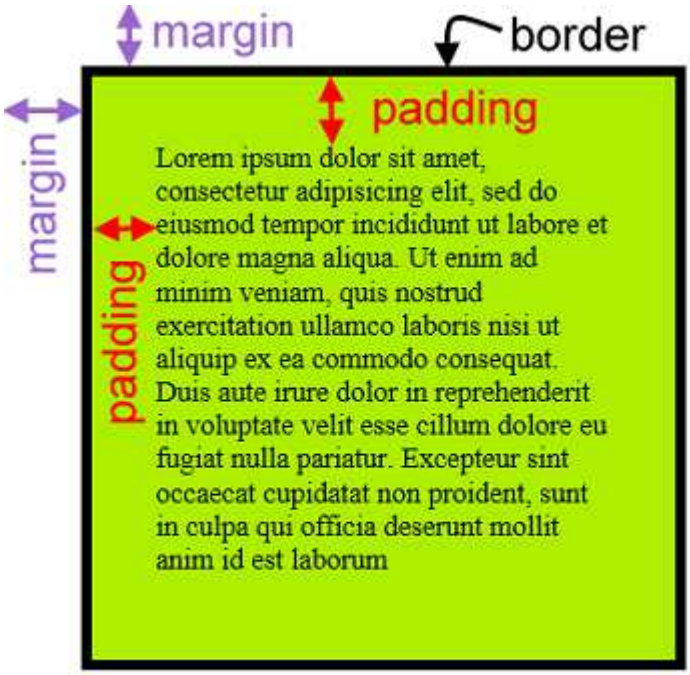
```
var posenfant = $('#enfant').offset();
posenfant.top = 100;
posenfant.left = 100;
$('#enfant').offset(posenfant);
```

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 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 etmargin 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 . Voici le code utilisé :

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

Le style de la balise 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;

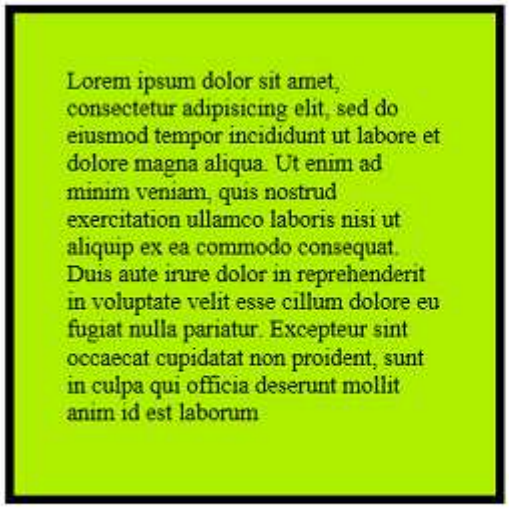
Pour accéder aux dimensions de la balise , nous allons insérer un peu de code jQuery après la ligne 30 :

```
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 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 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 et stocke les valeurs renvoyées dans la variable dimensions. La deuxième ligne interroge toutes les méthodes relatives à la hauteur de la balise et concatène les valeurs renvoyées à la variable dimensions.

Le résultat se trouve à l'image suivante :



width=250.01999999999998, innerWidth=320, outerWidth=332, outerWidth(true)=401.98,
height=250.01999999999998, innerHeight=320, outerHeight=332, outerHeight(true)=401.98

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 :

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

Méthode	Propriété(s) CSS	Valeur
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 . Vous utiliserez pour cela les méthodes width() et height() en tant que setters. Dans cet exemple, les dimensions de la balise sont fixées à 400×200 pixels :

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

INSÉRER ET REMPLACER DES ÉLÉMENTS DANS LE DOM

Insérer du contenu

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

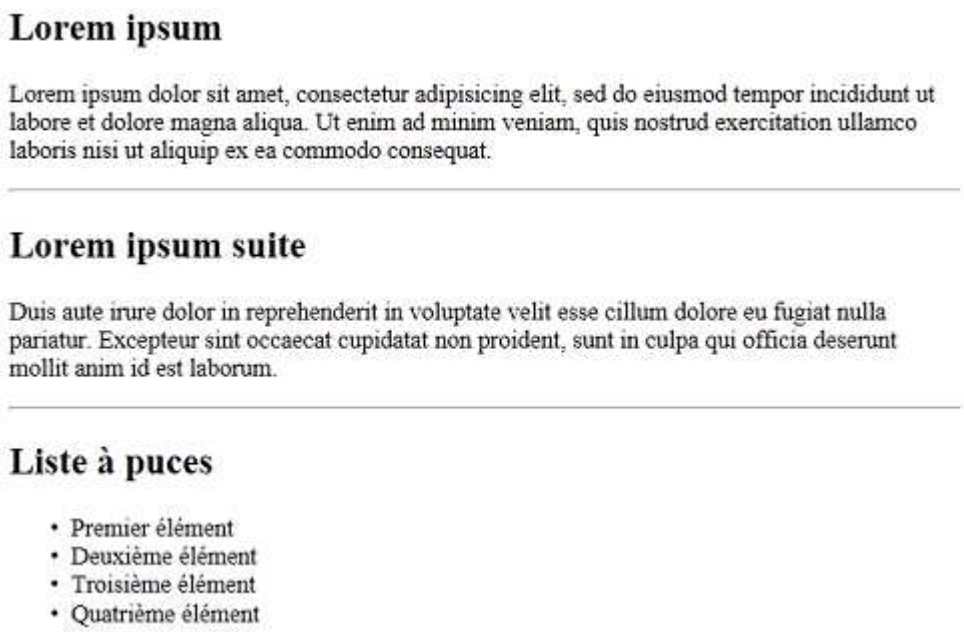
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Insertion, copie et suppression de données</title>
  </head>
  <body>
    <h2 id="un">Lorem ipsum</h2>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor
incididunt ut labore et dolore 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

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>

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

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

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

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

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

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

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

');
```

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 :

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

Pour remplacer tous les titres<h2> du document par des titres <h3> on utilisera l'instruction suivante :

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

Mais si vous avez affiché le document dans votre navigateur vous verrez que les titres ont disparus :

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.

- Premier élément
- Deuxième élément
- Troisième élément
- Quatrième élément

La méthode replaceWith() n'a pas produit l'effet escompté

Les titres <h2> ont tout disparu. 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.

La solution consiste à parcourir les éléments sélectionnés avec la méthode each() et à effectuer un remplacement replaceWith() personnalisé :

```
$('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

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 i**nsérer et cible représente l'élément avant ou après lequel doit se faire l'insertion.

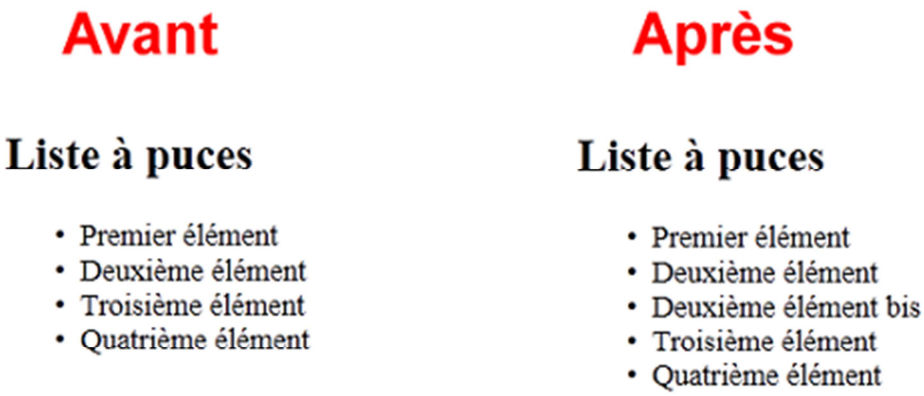
Pour décrire eai et cible, vous pouvez utiliser un sélecteur jQuery, un nom d'élément, une chaîne HTML ou un objet jQuery.

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

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

Le résultat se trouve à l'image suivante :

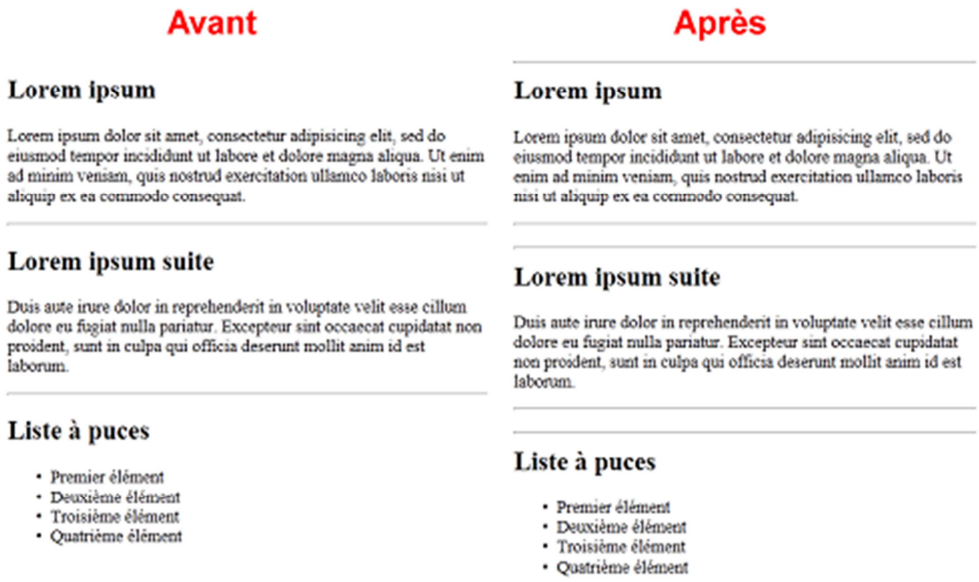


La puce « Deuxième élément bis » a été insérée après la puce « Deuxième élément »

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

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

Le résultat se trouve à l'image suivante :



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

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.

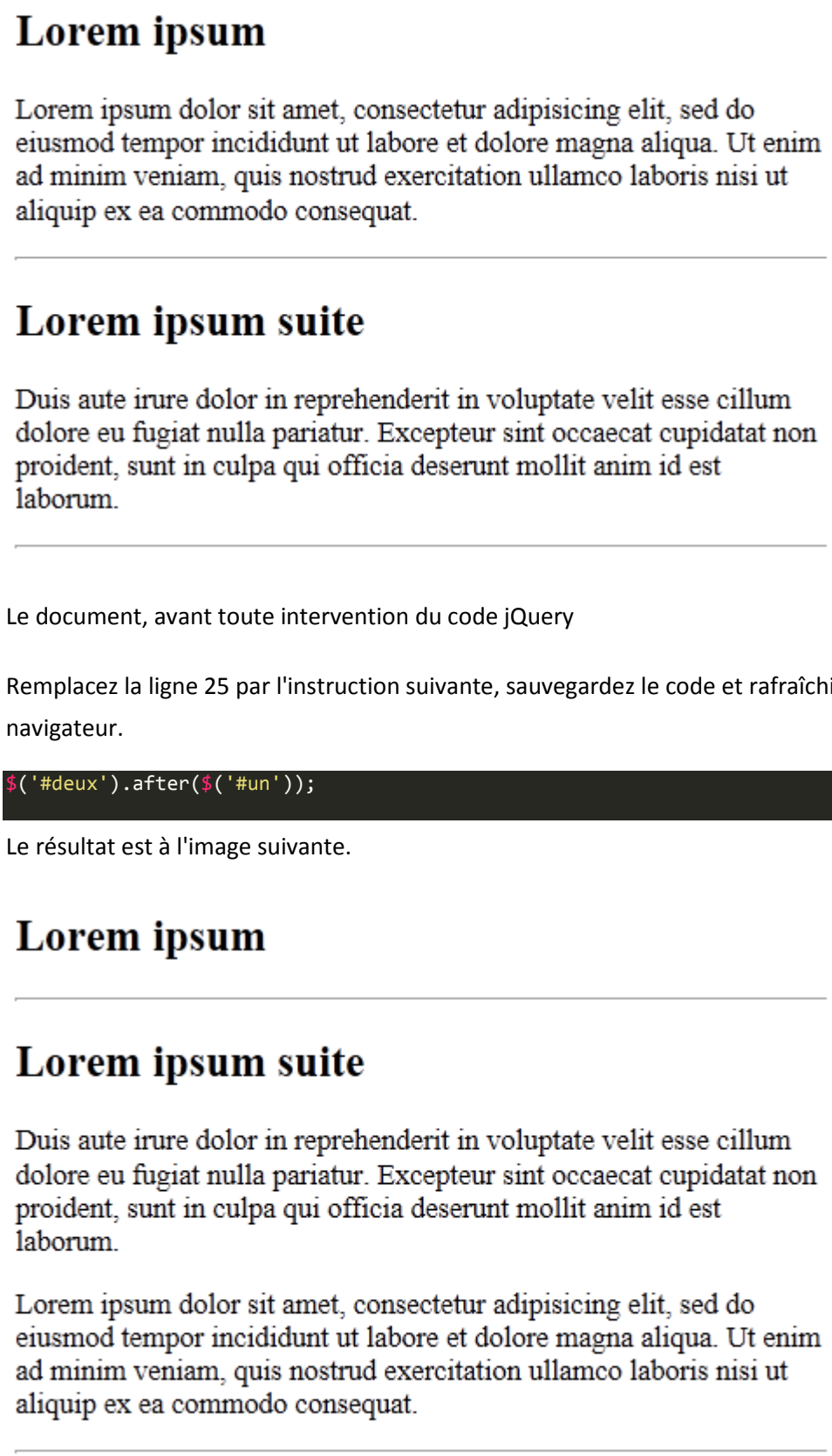
À titre d'exemple, considérons le code suivant :

```
<!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.



Le document, avant toute intervention du code jQuery

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

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

Le résultat est à l'image suivante.

Le paragraphe d'identifiant #deux a été déplacé à la suite du paragraphe d'identifiant #un

Comme vous pouvez le constater, le paragraphe d'identifiant #un n'est plus affiché après la première balise<h2>, mais après le paragraphe d'identifiant #deux. Il a donc été déplacé depuis la position qu'il occupait vers sa nouvelle position.

Dupliquer des éléments

Comme vous avez pu le constater précédemment, la méthode after() (ceci est également valable pour les méthodes append(), prepend() et before()) déplace un élément existant vers la position indiquée dans le sélecteur. Si vous voulez non pas déplacer, mais dupliquer un élément existant, vous appliquerez la méthode clone() à un sélecteur et, selon l'effet recherché, vous la ferez suivre de la méthode appendTo(), prependTo(), insertBefore() ou insertAfter().

À titre d'exemple, nous allons dupliquer le paragraphe d'identifiant #deux et l'insérer avant le paragraphe d'identifiant #un. Voici l'instruction à utiliser :

```
$('#deux').clone().insertBefore($('#un'));
```

Le résultat se trouve à l'image suivante.

Lorem ipsum

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.

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.

Lorem ipsum suite

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.

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

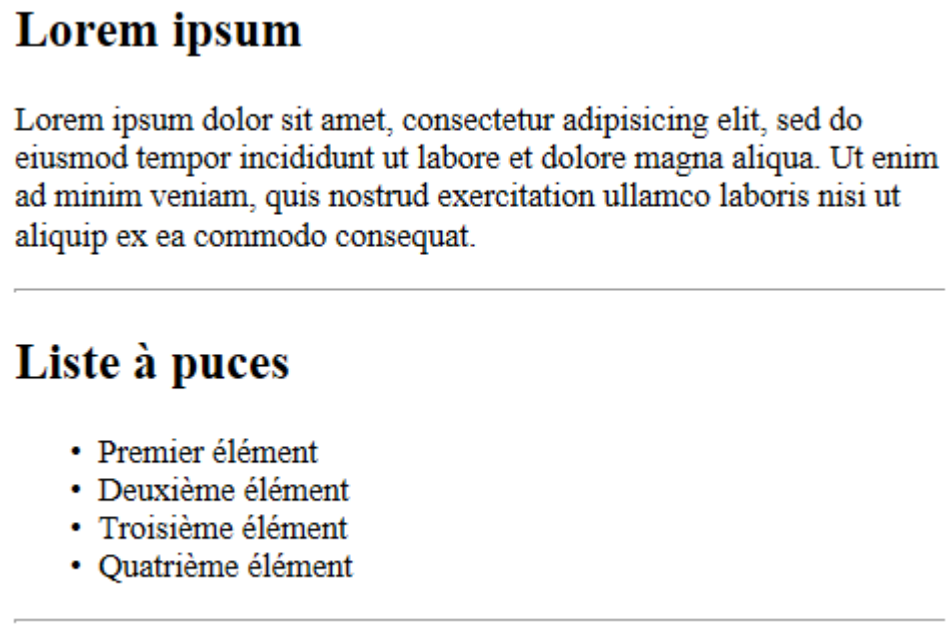
Si l'élément dupliqué a un ou plusieurs descendants, ils font eux aussi partie du clonage. Nous allons illustrer ce comportement en dupliquant tous les éléments qui composent la liste à puces du code suivant :

```
<!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>
```

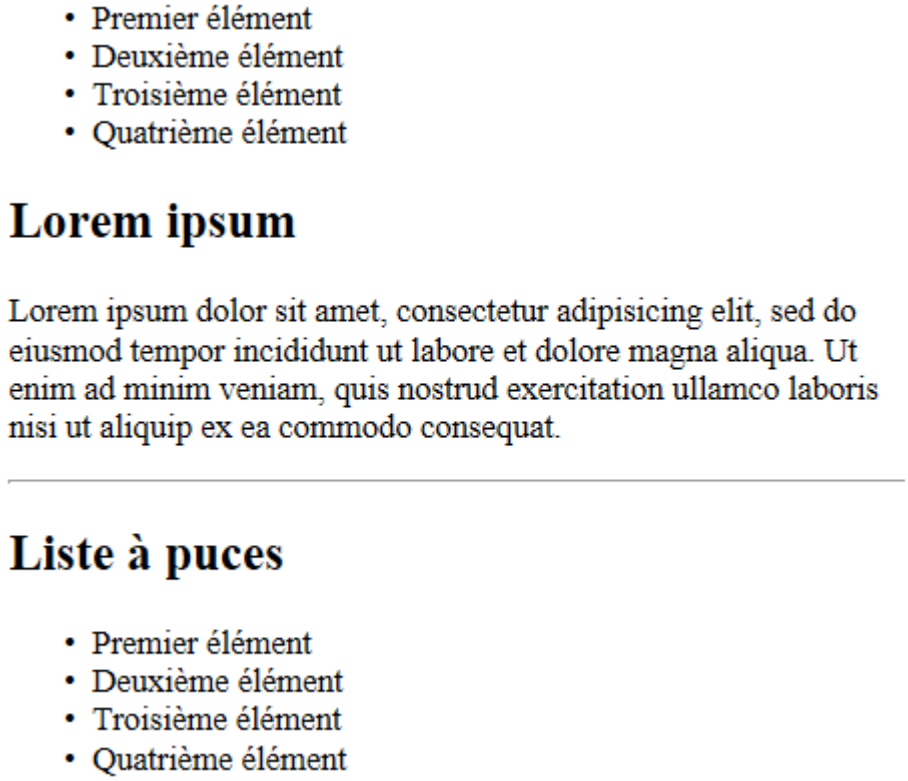
Ce code affiche un titre 2 suivi d'un paragraphe de texte et d'un trait de séparation, puis un autre titre 2 suivi d'une liste à puces composée de quatre éléments, comme le montre l'image suivante.



Insérez l'instruction suivante ligne 27 :

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

Cette instruction sélectionne la balise <ul> ($('ul')), la duplique (clone()) et place le clone avant la première balise <h2> (insertBefore($('h2:first'))).



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 :

```
$('<h1>Sommaire</h1>').insertBefore($('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.

Le sommaire a été créé automatiquement grâce à jQuery

Entourer des éléments

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

```
$('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 :

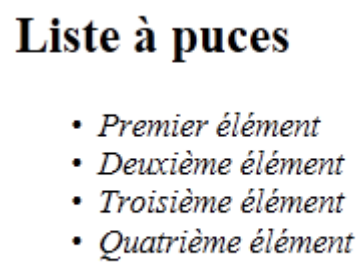
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Wrap</title>
  </head>
  <body>
    <h2 id="trois">Liste à puces</h2>
    <ul>
      <li>Premier élément</li>
      <li>Deuxième élément</li>
      <li>Troisième élément</li>
      <li>Quatrième élément</li>
    </ul>

    <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 :

```
$('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 :

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

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 :

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

Si vous exécutez l'instruction jQuery suivante :

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

… le paragraphe se transforme en :

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

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

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

… le paragraphe se serait transformé en :

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

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

```
<!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>
```

Si vous exécutez ce code dans un navigateur, vous obtiendrez l'image suivante.

Paragraphe 1

Paragraphe 2

Paragraphe 3

un texte isolé

Paragraphe 4

un autre texte isolé

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 :

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

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

un texte isolé un autre texte isolé

Les quatre balises <p> ont été rassemblées

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

```
<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 :

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

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

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

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

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

LA GESTION ÉVÉNEMENTIELLE

La souris

Avant de commencer

Quel que soit l'événement à gérer, vous devrez mettre en place une méthode qui ressemblera à ceci :

```
$(sel).mge(function() {
  // Une ou plusieurs instructions jQuery
  // pour gérer l'événement lorsqu'il se produit
}
```

… où sel est un sélecteur jQuery comme ceux que vous avez rencontrés jusqu'ici et mge est une méthode de gestion événementielle comme celles que nous allons voir.

La mise en place d'un événement concerne tous les éléments retournés par le sélecteur. Ainsi par exemple, en appliquant une gestion événementielle au sélecteur $('img'), elle concernera toutes les balises <img>du document. Ou encore, en appliquant une gestion événementielle au sélecteur $('.resultat'), elle s'appliquera à toutes les balises de classe resultat. Une seule instruction permet donc de mettre en place plusieurs gestions événementielles.

La souris est un périphérique universellement utilisé pour communiquer avec l'ordinateur. Vous pouvez désigner un élément en le pointant, sélectionner ou donner le focus à un élément en cliquant dessus, ou encore déplacer le contenu d'un élément doté d'une barre de défilement en agissant sur la roulette. Autant d'événements accessibles en jQuery. Voici la liste des évènements disponible :

Méthode	Événement géré
click()	Clic gauche
dblclick()	Double-clic
mousedown()	Appui sur le bouton gauche ou droit de la souris alors que le pointeur est au-dessus de l'élément
mouseenter() ou mouseover()	Début de survol de l'élément
mouseleave() ou mouseout()	Arrêt de survol de l'élément
mousemove()	Déplacement du pointeur au-dessus de l'élément

Méthode	Événement géré
mouseup()	Relâchement du bouton gauche ou droit alors que le pointeur est au-dessus de l'élément
scroll()	Utilisation de la roulette alors que le pointeur se trouve au-dessus d'un élément concerné par ce type d'événement

Clics et positions de la souris

Nous allons afficher une image de petite taille à une position aléatoire sur l'écran. Lorsque le joueur cliquera sur cette image, elle sera affichée à un autre emplacement. Voici le code utilisé :

```
<img id="target" src="/img/cat.png">

<script src="jquery.js"></script>
<script>
  $(function() {
    // Dimensions de la fenêtre
    var largeur = ($(window).width()) - 50;
    var hauteur = ($(window).height()) - 50;

    // Affichage de la première image en (100, 100)
    var p = $('#target').offset();
    p.top=100;
    p.left=100;
    $('#target').offset(p);

    // Gestion du clic et déplacement de l'image
    $("#target").click(function() {
      x = Math.floor(Math.random()*largeur);
      y = Math.floor(Math.random()*hauteur);
      var p = $('#target').offset();
      p.top = y;
      p.left = x;
      $('#target').offset(p);
    });
  });
</script>
```

Examinons les instructions qui composent ce document. Une balise <img> d'identifiant #target fait référence à l'image cat.png. Le reste du code utilise des instructions jQuery pour modifier l'emplacement de l'image et réagir aux clics de l'utilisateur.

Après avoir attendu la disponibilité du DOM, les dimensions de la fenêtre sont mémorisées dans les variables largeur et hauteur :

```
var largeur = ($(window).width()) - 50;
var hauteur = ($(window).height()) - 50;
```

L'image affichée a une dimension de 50×50 pixels. En soustrayant ces valeurs de la largeur et de la hauteur de la fenêtre, on s'assure que l'image sera toujours affichée dans la partie visible de la fenêtre. La méthode jQuery offset() est utilisée pour modifier l'emplacement initial de l'image, et la méthode target()pour connaître l'emplacement actuel de l'image :

```
var p = $('#target').offset();
```

En mémorisant son résultat dans la variable JavaScript `p`, on définit un objet jQuery par lequel les coordonnées de l'image pourront être modifiées. C'est ce que font les deux instructions suivantes en affichant l'image aux coordonnées (100, 100) :

```
p.top=100;
p.left=100;
```

Il ne reste plus qu'à utiliser la méthode `offset()` pour afficher l'image aux coordonnées (100, 100) :

```
$('#target').offset(p);
```

Une gestion événementielle est mise en place pour l'événement `click`, c'est-à-dire lorsque l'utilisateur clique sur le bouton gauche de la souris :

```
$("#target").click(function() {
```

Une nouvelle position aléatoire est choisie pour l'image (tout en restant dans les limites de la fenêtre) en attendant un autre clic de l'utilisateur. Un nombre aléatoire compris entre 0 et la largeur de la fenêtre est choisi et mémorisé dans la variable `x` :

```
x = Math.floor(Math.random()*largeur);
```

`Math.random()` est une fonction JavaScript qui retourne un nombre aléatoire compris entre 0 et une valeur proche de 1. Dans cet exemple, afin de simplifier les choses, nous allons admettre que le nombre retourné est compris entre 0 et 1.

En multipliant la valeur retournée par la largeur de la fenêtre, on obtient un nombre compris entre 0 et la largeur de la fenêtre. Enfin, en appliquant la fonction JavaScript `Math.floor()` à ce nombre, on obtient la valeur entière la plus proche de ce nombre. C'est exactement l'effet recherché.

La ligne suivante utilise la même technique pour choisir un nombre aléatoire compris entre 0 et la hauteur de la fenêtre. Ce nombre est mémorisé dans la variable `y`.

Pour déplacer l'image, nous utilisons la technique traditionnelle. Après avoir obtenu un objet jQuery qui correspond à la position actuelle de l'image :

```
var p = $('#target').offset();
```

… les coordonnées de l'image sont modifiées en utilisant les coordonnées tirées aléatoirement dans l'étape précédente :

```
p.top = y;
p.left = x;
```

Puis l'image est déplacée en utilisant la méthode offset() :

```
$('#target').offset(p);
```

Vous pouvez évidemment remplacer la méthode `click()` par une autre de votre choix. Par exemple, pour réagir au début du survol de l'image, vous utiliserez cette instruction :

```
$("#target").mouseenter(function() {
```

La méthode scroll

Utilisation de la méthode `scroll()` :

```
<style type="text/css">
  div {
    width: 200px;
    height: 200px;
    overflow: scroll;
    background-color: yellow;
    border: 2px black solid;
  }
</style>

<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>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('div').scroll(function() {
      alert('Utilisation de la roulette dans la balise <div>');
    });
    $(window).scroll(function() {
      alert('Utilisation de la roulette dans le document');
    });
  });
</script>
```

Pour détecter l'utilisation de la roulette dans la balise `<div>`, il suffit de sélectionner la balise et de lui appliquer la méthode `scroll()` :

```
$('div').scroll(function() {
```

La détection d'un mouvement de roulette déclenche l'affichage d'une boîte de message :

```
alert('Utilisation de la roulette dans la balise <div>');
```

Pour détecter l'utilisation de la roulette dans le document on utilise le mot `window` dans le sélecteur, sans le mettre entre apostrophes :

```
$(window).scroll(function() {
```

Ici aussi, la détection d'un mouvement de roulette déclenche l'affichage d'une boîte de message :

```
alert('Utilisation de la roulette dans le document');
```

Si la fenêtre est trop grande, vous ne pourrez pas tester l'utilisation de la roulette. Pensez à redimensionner la fenêtre pour qu'un *scrolling* soit possible.

which et type

Dans certains cas particuliers, il peut être nécessaire de savoir quel bouton de la souris a été pressé. Pour cela, vous ferez appel à la méthode `event.which`, qui renvoie l'une des valeurs suivantes :

- 1 : bouton gauche pressé ;
- 2 : bouton central pressé ;
- 3 : bouton droit pressé.

Pour connaître le type d'événement qui a été levé par la procédure de gestion événementielle, vous utiliserez la méthode `event.type`. La valeur renvoyée pourra être `click`, `dblclick`, `mousedown`, `mouseenter`, `mouseover`, `mouseleave`, `mouseout`, `mousemove` ou `mouseup`.

Voici comment utiliser ces deux méthodes en pratique.

```
Cliquez sur l'image avec un des boutons de la souris.<br ></code>
<img id="target" src="/img/cat.png"><br />
<span id="rapport"></span>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('#target').mousedown(function(e){
      $('#rapport').html('Événement : ' + e.type + '. Bouton pressé : ' +  e.which );
    });
  });
</script>
```

Le code jQuery met en place un gestionnaire événementiel en rapport avec la balise d'identifiant `#target`, c'est-à-dire l'image. Ce gestionnaire capture l'événement `mousedown`. Remarquez le paramètre `e` passé à la fonction :

```
$('#target').mousedown(function(e){
```

Les méthodes `e.type` et `e.which` sont utilisées pour indiquer le type d'événement levé et le bouton qui a été pressé. Ces informations sont affichées dans la balise `<span>` d'identifiant `#rapport` :

```
$('#rapport').html('Événement : ' + e.type + '. Bouton pressé : ' +  e.which );
```

Le clavier

Le clavier est également un périphérique fondamental pour communiquer avec l'ordinateur. Sur le Web, il est essentiellement utilisé pour saisir des données textuelles dans des formulaires. jQuery est en mesure de capturer trois événements en rapport avec le clavier.

Méthode	Événement géré

Méthode	Événement géré
`keydown()`	Appui sur une touche du clavier
`keyup()`	Relâchement d'une touche du clavier préalablement enfoncée
`keypress()`	Maintien d'une touche du clavier enfoncée

À titre d'exemple, nous allons afficher un petit rectangle de couleur verte chaque fois qu'un caractère sera ajouté dans une balise `<textarea>`. Ce rectangle deviendra blanc lorsque la touche sera relâchée. Voici le code utilisé :

```
<style type="text/css">
  #lumiere {
    width: 10px;
    height: 10px;
    background-color: white; }
</style>

<div id="lumiere"></div>
<textarea id="target"></textarea>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('#target').keydown(function(){
      $('#lumiere').css('background-color', 'green');
    });
    $('#target').keyup(function(){
        $('#lumiere').css('background-color', 'white');
    });
  });
</script>
```

Le code jQuery met en place deux procédures événementielles : une relative à l'événement keydown et l'autre à l'événement keyup. Lorsqu'une touche du clavier est enfoncée, la couleur d'arrière-plan de la balise devient verte. Lorsque la touche est relâchée, la balise redevient blanche.

Dans certains programmes écrits en jQuery, il peut être nécessaire de savoir quelle touche du clavier a été pressée. Pour cela, vous ferez appel à la méthode event.which qui renvoie précisément cette information. Pour connaître le type d'événement qui a été levé par la procédure de gestion événementielle, vous utiliserez la méthode event.type. La valeur renvoyée pourra être keydown, keypress ou keyup, en fonction de la méthode événementielle utilisée.

```
<form>
  Laissez aller votre imagination : saisissez quelques mots<br />
  <textarea id="saisie"></textarea>
</form><br />
Caractère saisi : <span id="unelettre"></span>

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

```
    $('#saisie').keypress(function(e) {
      $('#unelettre').text(e.which);  //keyCode
    });
  });
</script>
```

L'utilisateur est invité à taper quelques mots dans la zone de texte multilignes. Chacun des caractères tapés est alors affiché en dessous de la zone de saisie. Le code jQuery met en place un gestionnaire événementiel sur la balise d'identifiant `#saisie`, c'est-à-dire sur le `<textarea>`. La touche frappée est récupérée et affichée dans la balise `<span>`, comme le montre la figure suivante.

La méthode e.which retourne le code de la touche frappée

Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII
Espace	32	3	51	F	70	Y	89	l	108
!	33	4	52	G	71	Z	90	m	109
"	34	5	53	H	72	[	91	n	110
#	35	6	54	I	73	\	92	o	111
$	36	7	55	J	74	]	93	p	112
%	37	8	56	K	75	^	94	q	113
&	38	9	57	L	76	_	95	r	114
'	39	:	58	M	77	`	96	s	115
(	40	;	59	N	78	a	97	t	116
)	41	<	60	O	79	b	98	u	117
*	42	=	61	P	80	c	99	v	118
+	43	>	62	Q	81	d	100	w	119

Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII	Caractère	ASCII
,	44	?	63	R	82	e	101	x	120
-	45	@	64	S	83	f	102	y	121
.	46	A	65	T	84	g	103	z	122
/	47	B	66	U	85	h	104	{	123
0	48	C	67	V	86	i	105	|	124
1	49	D	68	W	87	j	106	}	125
2	50	E	69	X	88	k	107	~	126

Avec keydown() et keyup(), il s'agit d'une version simplifiée du code ASCII dans laquelle les caractères minuscules et majuscules sont confondus.

Touche	Code	Touche	Code	Touche	Code	Touche	Code
Retour arrière	8	6	54	v	86	F3	114
Tab	9	7	55	w	87	F4	115
Entrée	13	8	56	x	88	F5	116
Maj	16	9	57	y	89	F6	117
Ctrl	17	a	65	z	90	F7	118
Alt	18	b	66	Windows gauche	91	F8	119
Pause	19	c	67	Windows droit	92	F9	120
Verr Maj	20	d	68	Sélection	93	F10	121
Echap	27	e	69	0 pavé num.	96	F11	122
Page Préc	33	f	70	1 pavé num.	97	F12	123
Page Suiv	34	g	71	2 pavé num.	98	Verr Num	144

Touche	Code	Touche	Code	Touche	Code	Touche	Code
Fin	35	h	72	3 pavé num.	99	Arrêt Defil	145
Origine	36	i	73	4 pavé num.	100	;	186
Gauche	37	j	74	5 pavé num.	101	=	187
Haut	38	k	75	6 pavé num.	102	,	188
Droite	39	l	76	7 pavé num.	103	-	189
Bas	40	m	77	8 pavé num.	104	.	190
Inser	45	n	78	9 pavé num.	105	/	191
Suppr	46	o	79	*	106	`	192
0	48	p	80	+	107	[	219
1	49	q	81	-	109	\	220
2	50	r	82	.	110	]	221
3	51	s	83	/	111	Espace	222
4	52	t	84	F1	112	-	-
5	53	u	85	F2	113	-	-

Si vous voulez obtenir non pas le code du caractère mais le caractère lui-même, assurez-vous que vous utilisez la méthode keypress() :

```
$('#saisie').keypress(function(e) {
```

Et remplacez la ligne suivante par :

```
var c = String.fromCharCode(e.which);
$('#unelettre').text(c);
```

La première instruction récupère le code tapé au clavier (e.which), le convertit en un caractère (String.fromCharCode) et le stocke dans la variable c. La deuxième instruction affiche ce caractère dans la balise d'identifiant #unelettre, c'est-à-dire dans le .

Les éléments

Vous trouverez regroupé ici les méthodes événementielles en rapport avec le gain et la perte de focus, la modification de la taille et du contenu, et la sélection d'un élément.

Méthode	Événement géré
`focus()`	Réception de focus par l'élément
`blur()`	Perte de focus par l'élément
`focusin()`	Réception de focus par l'élément ou un de ses enfants
`focusout()`	Perte de focus par l'élément ou un de ses enfants
`resize()`	Redimensionnement d'un élément
`change()`	Modification d'un élément

Les méthodes `focus()` et `blur()` détectent respectivement la réception de focus et la perte de focus par un élément dans un formulaire. Cela peut se produire suite à l'appui sur une touche ou une combinaison de touches du clavier (**Tab** ou **Maj** + **Tab** par exemple) ou par un clic de souris.

Les méthodes `focusin()` et `focusout()` sont comparables aux méthodes `focus()` et `blur()` et peuvent les remplacer. Cependant, elles détectent également la réception et la perte de focus d'un élément parent.

focus() et blur()

Méthodes `focus()`et `blur()`.

```
<form>
  Cliquez sur les zones de texte<p>
  <input type="text" class="f" id="Zone-de-texte-1"><p>
   <input type="text" class="f" id="Zone-de-texte-2"><br />
</form><br />

  Focus : <span id="resultat"></span><br />
  Perte de focus : <span id="resultat2"></span>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('.f').focus(function() {
      $('#resultat').text($(this).attr('id'));
    });
```

```
    $('.f').blur(function() {
      $('#resultat2').text($(this).attr('id'));
    });

  });
</script>
```

Le corps du document contient essentiellement deux zones de texte et deux balises . Lorsque l'utilisateur donne le focus à l'une des zones de texte, le contenu des deux est modifié. Le premier indique l'identifiant du contrôle qui a reçu le focus et le deuxième indique l'identifiant du contrôle qui a perdu le focus.

La procédure événementielle est responsable de l'affichage dans le premier . La méthode utilisée est focus(). L'événement déclencheur sera donc la réception du focus :

```
$('.f').focus(function() {
```

Examinez le sélecteur. Toutes les balises de classe f sont concernées, à savoir les deux zones de texte. Lorsque cette fonction événementielle est exécutée, l'identifiant (attr('id')) de la balise qui a déclenché l'événement ($(this)) est affiché (text) dans la balise d'identifiant #resultat($('#resultat')), c'est-à-dire dans la première balise :

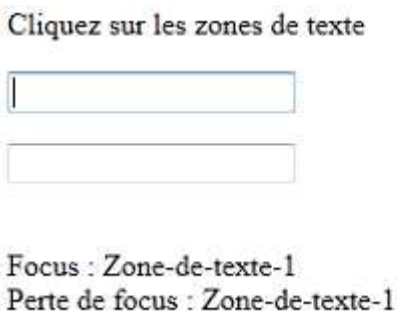
```
$('#resultat').text($(this).attr('id'));
```

La deuxième procédure événementielle est responsable de l'affichage dans le deuxième `<span>`. La méthode utilisée est `blur()`. L'événement déclencheur sera donc la perte du focus :

```
$('.f').blur(function() {
```

Cette méthode concerne les balises de classe `f`, et donc les deux zones de texte. Lorsque cette fonction événementielle est exécutée, l'identifiant (`attr('id')`) de la balise qui a déclenché l'événement (`$(this)`) est affiché (`text`) dans la balise d'identifiant `#resultat2` (`$('#resultat2')`), c'est-à-dire dans la deuxième balise `<span>`.

La figure suivante montre la page Web après avoir donné le focus à la deuxième zone de texte, puis à la première.



Les deux sont mis à jour en fonction de l'élément qui a le focus

focusin() et focusout()

Nous allons maintenant nous intéresser aux méthodes `focusin()` et `focusout()`, et montrer leurs différences par rapport aux méthodes `focus()` et `blur()`. Pour cela, deux balises `<fieldset>`contenant chacune deux balises `<input type="text">` vont être créées. Le gain et la perte de focus seront testés au niveau des balises `<fieldset>`. En donnant le focus à une zone de texte, l'événement sera répercuté jusqu'à la balise `<fieldset>` parent qui affichera des informations en conséquence.

```
<form>
  Cliquez sur les zones de texte<p>
  <fieldset id="premier">
    <legend>Premier groupe</legend>
    <input type="text" class="f" id="Zone-de-texte-1"><p>
    <input type="text" class="f" id="Zone-de-texte-2"><br />
  </fieldset>

  <fieldset id="deuxieme">
    <legend>Deuxième groupe</legend>
    <input type="text" class="f" id="Zone-de-texte-3"><p>
    <input type="text" class="f" id="Zone-de-texte-4"><br />
  </fieldset>
</form><br />

Focus : <span id="resultat"></span><br />
Perte de focus : <span id="resultat2"></span>

<script src="jquery.js"></script>
<script>
  $(function() {
    $('fieldset').focusin(function() {
      $('#resultat').text($(this).attr('id'));
    });
    $('fieldset').focusout(function() {
      $('#resultat2').text($(this).attr('id'));
    });
  });
</script>
```

Le corps du document contient deux balises `<fieldset>` d'identifiant `#premier` et `#deuxieme`. Chacune de ces balises contient une légende et deux zones de texte. À la suite des deux balises`<fieldset>`, deux balises `<span>` sont utilisées pour indiquer quelle balise `<fieldset>` gagne le focus et quelle balise `<fieldset>` le perd.

La première procédure événementielle teste le gain de focus. La méthode `focusin()` est appliquée aux éléments `fieldset`, c'est-à-dire aux deux balises `<fieldset>` :

```
$('fieldset').focusin(function() {
```

Lorsqu'une balise `<fieldset>` ou un de ses enfants (les balises `<legend>` et `<input type="text">`) gagne le focus, cette méthode événementielle est exécutée. L'identifiant (`attr('id')`) de la balise`<fieldset>` parent (`$(this)`) est alors affiché (`text`) dans la balise d'identifiant `#resultat`(`$('#resultat')`), c'est-à-dire dans la première balise `<span>` :

```
$('#resultat').text($(this).attr('id'));
```

Un traitement similaire affiche dans la deuxième balise `<span>` le nom de la balise `<fieldset>` qui a perdu le focus :

```
$('fieldset').focusout(function() {
  $('#resultat2').text($(this).attr('id'));
});
```

On peut remplacer les méthodes `focusin()` et `focusout()` par `focus()` et `blur()` et expérimentez le nouveau code afin de voir la différence entre ces deux jeux de méthodes.

resize()

Nous allons maintenant nous intéresser à la méthode événementielle `resize()`. Cette méthode est exécutée chaque fois que la fenêtre change de taille. Nous allons l'utiliser pour afficher dans une balise `<span>` les dimensions de la fenêtre chaque fois qu'elle est exécutée :

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

<script src="jquery.js"></script>
<script>
  $(function() {
    $(window).resize(function() {
      var taille = 'Taille de la fenêtre : ' + $(window).width() + 'px x ' +
$(window).height() + 'px';
      $('#resultat').text(taille);
    });
  });
  </script>
```

Le corps du document ne comporte qu'une balise `<span>` dans laquelle nous afficherons les dimensions de la fenêtre. Quant au traitement, dans un premier temps, les dimensions de la fenêtre (`$(window).width` et `$(window).height`) sont mémorisées dans la variable `taille` :

```
var taille = 'Taille de la fenêtre : ' + $(window).width() + 'px x ' + $(window).height() +
'px';
```

Puis le contenu de la variable `taille` est copié (`text(taille)`) dans la balise `<span>` d'identifiant `#resultat` (`$('#resultat')`) :

```
$('#resultat').text(taille);
```

change()

Pour en terminer avec les méthodes événementielles relatives aux éléments, nous allons nous intéresser à la méthode `change()`. Cette méthode est exécutée chaque fois que le contenu de l'élément concerné change. Elle peut être utilisée sur les balises `<input>`, `<textarea>` et `<select>`. À titre d'exemple, nous allons détecter les modifications dans une liste déroulante et afficher un message en conséquence.

```
<form>
  Sélectionnez une valeur dans la liste déroulante
  <select>
    <option>J'aime jQuery</option>
    <option>J'adore jQuery</option>
    <option>Je raffole de jQuery</option>
    <option>jQuery ? Jamais entendu parler !</option>
  </select>
</form><br />

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

```
<script src="jquery.js"></script>
<script>
  $(function() {
    $('select').change(function() {
        $('#resultat').text('Vous venez de sélectionner "' + $(this).val() +'".');
    });
  });
</script>
```

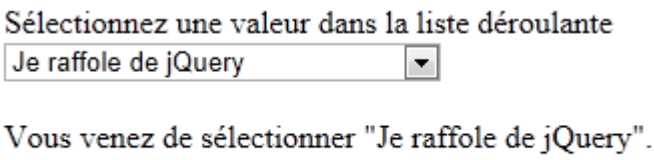
Le corps du document met en place une liste déroulante qui contient quatre éléments. L'élément sélectionné dans la liste sera indiqué dans la balise `<span>` d'identifiant `#resultat`. La méthode événementielle `change()` est appliquée à la balise `<select>`. Chaque fois que l'utilisateur sélectionne une valeur dans la liste, cette méthode est exécutée :

```
$('select').change(function() {
```

Le texte de l'élément sélectionné dans la liste (`$(this).val()`) est alors affiché dans la balise `<span>`d'identifiant `#resultat` (`$('#resultat').text`) :

```
$('#resultat').text('Vous venez de sélectionner "' + $(this).val() +'".');
```

La figure suivante vous montre un exemple d'exécution.



Le texte est mis à jour en fonction du choix dans la liste

Les pages

Appliquée à l'élément `window`, la méthode événementielle `load()` permet de tester le complet chargement d'une page, en incluant les textes, images et autres objets qui la composent. Quant à la méthode `unload()`, elle est déclenchée lorsque l'internaute a demandé un changement de page. Voyons comment utiliser ces deux méthodes :

```
<img src="/img/cat.jpg"><br />
<a href="http://www.google.fr">Cliquez ici pour aller sur Google</a>

<script src="jquery.js"></script>
<script>
  $(function() {
    alert('Le DOM est chargé');
    $(window).load(function() {
        alert('La page est entièrement chargée');
    });
    $(window).unload(function() {
        alert('Vous avez demandé à changer de page');
    });
  });
</script>
```

Le corps du document contient une image et un lien qui pointe vers Google. Lorsque le DOM est disponible, une boîte de dialogue est affichée :

```
alert('Le DOM est chargé');
```

Le contenu de la page est alors chargé. Lorsque l'image et le lien sont en mémoire, la méthode événementielle `$(window).load()` s'exécute. Une autre boîte de dialogue est alors affichée :

```
alert('La page est entièrement chargée');
```

Enfin, quand l'utilisateur clique sur le lien « Cliquez ici pour aller sur Google », puis clique sur `Page précédente` ou `Page suivante` du navigateur ou lorsqu'il ferme ce dernier, la méthode événementielle `$(window).unload()` est exécutée, ce qui produit l'affichage d'une troisième boîte de dialogue :

```
alert('Vous avez demandé à changer de page');
```

La méthode `unload()` est toujours appliquée à l'élément `window`, c'est-à-dire à la fenêtre du navigateur. Par contre, la méthode `load()` peut être appliquée à un autre élément auquel est associé une URL : une balise `<img>`, `<script>`, `<frame>` ou `<iframe>`. Dans ce cas, le code associé à cette méthode est exécuté lorsque l'élément correspondant et ses enfants (s'ils existent) sont entièrement chargés.

Par exemple, vous utiliserez les instructions suivantes pour afficher les dimensions d'une image après son complet chargement :

```
$('#image1').load(function() {
  alert(this.width + ' x ' + this.height);
}
```

… où `#image1` est l'identifiant de l'image.