JQUERY S4

Stéphane Fossé
Stephane.fosse@u-bordeuax.fr

Table des matières

Introduction 2
Syntaxe de base 2
Les événements 3
Les évènements des formulaires 5
Gérer les évènements avec la méthode jQuery on() 6
Supprimer un gestionnaire d'évènements avec off() 8
DOM et jquery 9
Méthode .parent() 9
Parcourir les descendants 9
Manipuler le texte 12
Supprimer du contenu 13
Manipuler le CSS 13
Cacher / Afficher 15
Les fondus 16
Animation 16
Chaîner deux animations 17
Ajouter un délais 18
Ajax 19

Introduction

jQuery est un framework Javascript sous licence libre qui permet de faciliter des fonctionnalités communes de Javascript. L'utilisation de cette bibliothèque permet de gagner du temps de développement lors de l'interaction sur le code HTML d'une page web, l'AJAX ou la gestion des évènements.
Pour pouvoir utiliser la bibliothèque jQuery, vous pouvez la télécharger sur le site jQuery.com, puis la charger dans votre page.

```
<head>
<script src="jquery-3.2.1.min.js"></script>
</head>
```

Vous pouvez également l'importer directement depuis un CDN, par exemple celui de Google

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

Syntaxe de base

Tout le fonctionnement de la bibliothèque se base sur la fonction jQuery(). Cette fonction est généralement écrite de manière abrégée $().

Par exemple, pour accéder à tous les éléments **p** de notre page HTML, nous écrirons : **$('p')** ou pour sélectionner tous les éléments portant la class para, en **$('.para')**

```
<html>
    <head>
     <title>Premier script </title>
    </head>
    <body>
        <h1>Découverte de jQuery</h1>
        <p class='para'>paragraphe</p>
        <p>paragraphe<p>

        <!--On inclut la librairie jQuery-->
        <script src='https://code.jquery.com/jquery
3.1.0.js'></script>

        <!--On écrit notre code JavaScript / jQuery-->
        <script>
            //On utilise ready en jQuery
            $(document).ready(function(){
                //On place tout notre code jQuery ici
             $('.para').hide();
            });
        </script>
    </body>
</html>
```

Les événements

Le jQuery simplifie la gestion d'évènements en mettant à notre disposition différentes méthodes prêtes à l'emploi.
Ces méthodes vont tout simplement généralement porter le nom des évènements auxquels on souhaite réagir, comme par exemple la méthode click() qui va nous permettre de gérer l'évènement click.

```
<p>Un paragraphe</p>
<script src='https://code.jquery.com/jquery-3.1.0.js'></script>

        <!--On écrit notre code JavaScript / jQuery-->
```

```
        <script>
            $(document).ready(function(){

                $('p').click(function(){
                    $(this).hide();
                });

            });
        </script>
  </body>
```

Plus généralement, voici quelques exemple d'événements facile à comprendre

```
$("p").mouseenter(function(){
    $("p").css("background-color", "yellow");
});
```

```
$("p").mouseleave(function(){
    $("p").css("background-color", "red");
});
```

ou plus simplement :

```
$("p").hover(function(){
    $(this).css("background-color", "yellow");
    }, function(){
    $(this).css("background-color", "pink");
});
```

Remarquez l'utilisation constante de fonctions anonymes.

Les méthodes **keydown**(), **keyup**() et **keypress**() vont nous permettre de répondre à des évènements relatifs au clavier de l'utilisateur.

```
<html>
<head>
<script src="jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#keyboard").keydown(function(){
```

```
    alert("keydown event occured! The Ascii value of pressed key is:
" + event.keyCode);
  });
});
</script>
</head>
<body>
Enter keyboard keys: <input type="text" id="keyboard">
</body>
</html>
```

Les évènements des formulaires

Dans cette partie, nous allons étudier les évènements les plus communs liés aux formulaires et voir comment les prendre en charge en jQuery.

```
<html>
<head>
<script src="jquery-1.4.2.min.js"></script>
<script>
$(‘document’).ready(){function(){
   $('#focus').focusin(function(){
   $(this).addClass('focused').val('Focus In');
});
$('#focus').focusout(function(){
       $(this).removeClass('focused').val('Focus Out');
});
});
}
</script>
   <style>
   .focused{background-color : #00f0ff;}
    </style>
</head>
<body>
<input type="text" name="focus" id="focus" />
</body>
</html>
```

Un autre exemple :

```
<select name="bonbon" multiple="multiple">
  <option>Chocolat</option>
  <option selected="selected">Candy</option>
  <option>Taffy</option>
  <option selected="selected">Caramel</option>
  <option>Cookie</option>
</select>
<div></div>

<script>
$( "select" ).change(function () {
    var str = "";
    $( "select option:selected" ).each(function() {
      str += $( this ).text() + " ";
    });
    $( "div" ).text( str );
  })
</script>
```

L'évènement submit va se déclencher lorsqu'un utilisateur tente d'envoyer un formulaire.

```
<script>
            $(document).ready(function(){
                $('form').submit(function(){
                    alert('Formulaire envoyé');
                });

            });
</script>
```

Gérer les évènements avec la méthode jQuery on()

Depuis la version 1.7 de jQuery, la documentation officielle recommande d'utiliser on() plutôt que les notations raccourcies précédentes.
La méthode on() va prendre en arguments le nom d'un évènement et une fonction à exécuter en cas de déclenchement de l'évènement passé en premier argument.

```
<script>
```

```
$(document).ready(function(){
    $("p").on("mouseover mouseout", function(){
        $(this).toggleClass("intro");
    });
});
</script>
```

L'intérêt est également d'assigner des événements à des éléments du DOM créés à la volée.

```
<script>
$(document).ready(function(){
    $("div").on("click", "p", function(){
        $(this).slideToggle();
    });
    $("button").click(function(){
        $("<p>This is a new paragraph.</p>").insertAfter("button");
    });
});
</script>
</head>
<body>

<div style="background-color:yellow">
  <p>This is a paragraph.</p>
  <p>Click any p element to make it disappear. Including this
one.</p>
  <button>Insert a new p element after this button</button>
</div>
```

Pour forcer le déclenchement d'un évènement. : trigger() et triggerHandler()

Ces deux méthodes vont prendre en argument le nom d'un évènement. On peut également leur passer d'autres arguments optionnels.
trigger() va déclencher l'évènement spécifié en argument ainsi que le comportement par défaut de l'évènement tandis que triggerHandler() ne va pas déclencher le comportement par défaut de l'évènement.
trigger() va être exécutée pour tous les éléments possédant les caractéristiques ciblées tandis que triggerHandler() ne va s'exécuter que pour le premier élément ciblé rencontré.

```
<script>
    $(‘document’).ready(function()
    {
        $("#old").click(function () {
            $("input").trigger("focus");
        });
        $("#new").click(function () {
            $("input").triggerHandler("focus");
        });
        $("input").focus(function () {
            $("<span>Focused!</span>").appendTo("body");
        });
    });
</script>
</head>
<body>
<button id="old">.trigger("focus")</button>
<button id="new">.triggerHandler("focus")</button><br/><br/>
<input type="text" value="To Be Focused"/>
```

Supprimer un gestionnaire d’évènements avec off()

La méthode jQuery off() va nous permettre de supprimer des gestionnaires d’évènements attachés avec la méthode on().

```
function changeSize() {
    $(this).animate({fontSize: "+=10px"});
}
function changeSpacing() {
    $(this).animate({letterSpacing: "+=5px"});
}
$(document).ready(function(){
    $("p").on("click", changeSize);
    $("p").on("click", changeSpacing);
    $("button").click(function(){
        $("p").off("click", changeSize);
    });
});
</script>
</head>
```

```
<body>


<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<p>Click any p element to increase size and letterspacing.</p>


<button>Remove the changeSize() event handler</button>
```

DOM et jquery

La manipulation du DOM est encore plus simple car des méthodes jQuery ont été créées pour servir à la fois de getter ou de setter selon le type d'arguments qu'on va leur passer.

Méthode .parent()

API : http://api.jquery.com/parent/
La méthode .**parent**() retourne l'élément parent direct de l'élément sélectionné. Cette méthode ne traverse qu'un seul niveau dans l'arborescence DOM.

```
$(‘.interne’ ).parent().addClass('box');
```

Méthode .**parents**() API : http://api.jquery.com/parents/
La méthode . **parents**() retourne tous les éléments ancêtres de l'élément sélectionné.
Méthode .**parentsUntil**() API : http://api.jquery.com/parentsUntil/
La méthode .parentsUntil() retourne tous les éléments ancêtres entre deux arguments donnés.

```
$('.interne’).parentsUntil(‘externe’).addClass('clear');
```

Méthode .**closest**() API : http://api.jquery.com/closest/
La méthode .closest() retourne le plus proche élément ancêtre de l'élément sélectionné.
$('.interne').closest('section').addClass('clearfix');
le résultat est :

```
<main>
    <section class="clear">
        <div>
            <div class="interne ">…</div>
        </div>
    </section>
</main>
```

Parcourir les descendants

Dans cette partie, nous verrons les méthodes children() et .find().
Méthode .children() API : http://api.jquery.com/children/
La méthode .**children**() retourne tous les enfants directs de l'élément sélectionné. Cette méthode ne traverse qu'un seul niveau dans l'arbre DOM.

```
<main>
    <section>
        <div class="box">
            <div>
                <div>…</div>
            </div>
        </div>
    </section>
</main>
$('.box').children().addClass(‘interne’);
```

Méthode .find() API : http://api.jquery.com/find/
La méthode .find() retourne tous les enfants passés en paramètre.

```
$(‘.box’).find(div).addclass(‘red’) ;
```

Parcourir les frères
Dans cette partie, nous verrons les
méthodes .siblings(), .next(), .nextAll(), .nextUntil(), .prev(), .prevAll() et .prevUntil()
Méthode .siblings() API : http://api.jquery.com/siblings/
La méthode .siblings() retourne tous les éléments frères de l'élément sélectionné.

```
<ul class="list">
    <li>…</li>
    <li>…</li>
    <li>…</li>
    <li class="item">…</li>
    <li>…</li>
</ul>
$('.item').siblings().addClass('selected');
```

Méthode .next() API : http://api.jquery.com/next/
La méthode .next() retourne le frère suivant de l'élément sélectionné.

```
$('.item').next().addClass('selected');
```

Méthode .nextAll() API : http://api.jquery.com/nextAll/
La méthode .nextAll() retourne tous les frères suivants de l'élément sélectionné.

```
$('.item').nextAll().addClass('selected');
```

Méthode .nextUntil()
API : http://api.jquery.com/nextUntil/
La méthode .nextUntil() retourne tous les frères suivants entre les éléments sélectionnés.

```
<ul class="list">
```

```
    <li>…</li>
    <li class="item">…</li>
    <li>…</li>
    <li>…</li>
    <li class="other-class">…</li>
</ul>
$('.item').nextUntil('.other-class').addClass('selected');
```

Méthode .prev() API : http://api.jquery.com/prev/
La méthode .prev() retourne le frère prédédent l'élément sélectionné.

```
$('.item').prev().addClass('selected');
```

De même pour les méthode prevAll() et prevUntil()
Les filtres
Dans cette partie, nous verrons les méthodes .first(), .last(), .eq(), .filter() et .not().
Méthode .first() API : http://api.jquery.com/first/
La méthode .first() retourne le premier élément de l'élément sélectionné.
L'exemple suivant ajoute class="selected" au premier élément
de class="item".

```
<ul class="list">
    <li class="item">…</li>
    <li class="item">…</li>
    <li class="item">…</li>
</ul>
$('.item').first().addClass('selected');
```

De ml^me pour la méthode last() API : http://api.jquery.com/last/
Méthode .eq() API : http://api.jquery.com/eq/
La méthode .eq() retourne un élément avec un numéro d'index spécifique. Les numéros d'index commencent à 0.

```
<ul class="list">
    <li class="item">…</li>
    <li class="item">…</li>
    <li class="item">…</li>
    <li class="item">…</li>
</ul>
$('.item').eq(1).addClass('selected');
```

Méthode .filter() API : http://api.jquery.com/filter/
La méthode .filter() permet de spécifier un critère. Les éléments qui ne correspondent pas aux critères sont retirés de la sélection, et ceux qui correspondent seront retournés.

```
<ul class="list">
    <li class="item">…</li>
```

```
    <li class="item">…</li>
    <li class="item other-class">…</li>
    <li class="item">…</li>
    <li class="item other-class">…</li>
    <li class="item">…</li>
    <li class="item">…</li>
</ul>
$('.item').filter('.other-class').addClass('selected');
```

Méthode .not() API : http://api.jquery.com/not/
La méthode .not() retourne tous les éléments qui ne correspondent pas aux critères. C'est le contraire de la méthode .filter()

```
$('li').not('.item').addClass('selected');
```

Manipuler le texte

Dans cette partie, nous verrons les méthodes .text(),.html() et .val().
Méthode .text() API : http://api.jquery.com/text/
La méthode .text() récupère ou remplace le contenu en texte brut **sans prendre en compte les balises HTML.**

```
<div class="box">Je suis magicien</div>
$('.box').text('Hello world');
```

Méthode .html() API : http://api.jquery.com/html/
La méthode .html() récupère ou remplace le contenu HTML de l'élément.

```
<div class="box">Je suis magicien</div>
$('.box').html('<h1>Hello world</h1>');
```

Méthode .val() API : http://api.jquery.com/val/
La méthode .val() récupère ou remplace la value d'un élément de formulaire input text, textarea etc. Dans les champs de type radio et checked elle renvoie :checked ou pas.

```
<form role="form" action="#"
    <div>
        <label for="#name">Nom :</label>
        <input type="text" name="name" id="name">
    </div>
    <input id="submit" type="submit" value="Envoyer">
</form>
$('#submit').click(function(){
    alert($('#name').val());
});
```

Supprimer du contenu

Dans cette partie, nous verrons les méthodes .remove() et .empty().
Méthode .remove() API : http://api.jquery.com/remove/
La méthode .remove() supprime du DOM l'élément sélectionné.

```
<div id="main">
    <div class="other">Je suis ingénieur informaticien</div>
    <div class="box">Je suis magicien</div>
</div>
$('.box').remove();
```

Méthode .empty() API : http://api.jquery.com/empty/
La méthode .empty() vide le contenu de l'élément sélectionné.

```
<div id="main">
    <div class="other">Je suis ingénieur informaticien</div>
    <div class="box">Je suis magicien</div>
</div>
$('.box').empty();
```

Manipuler le CSS

Dans cette partie, nous verrons les
méthodes .addClass(), .removeClass(), .toogleClass() et .css().
Méthode .addClass() API : http://api.jquery.com/addClass/
Les méthodes .addClass(), removeClass et ToggleClass permetent d'ajouter,d'enlever une classe sur l'élément selectionné ou d'alterner la présence d'une classe (toggle)

```
$('#a').click(function(){
  $('#b').toggleClass('actif');
});
```

Méthode .css() API : http://api.jquery.com/css/
La méthode .css() affecte les styles CSS à l'élément sélectionné.
Elle agit aussi comme un getter et permet donc d'obtenir la chaîne de caractères de la valeur CSS demandée.

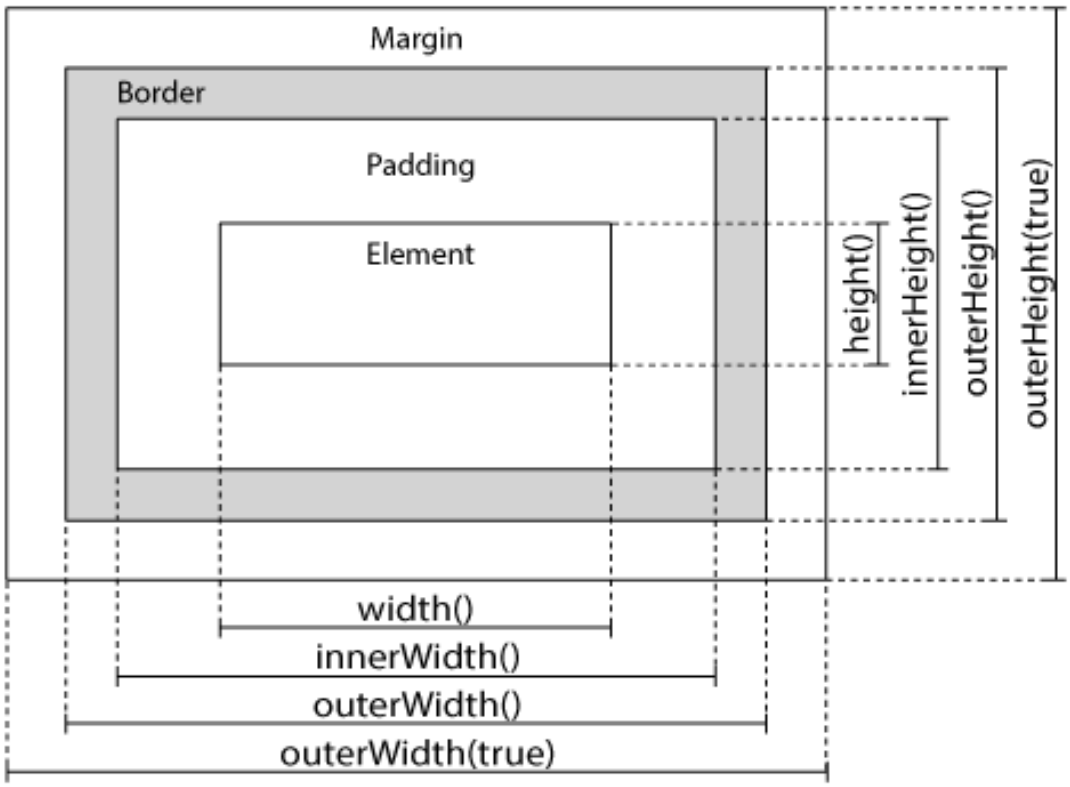
```
$('.box').css('background-color','red');
```

Nous pouvons bien sûr ajouter plusieurs styles CSS en même temps sous forme d'un objet avec des paramètres.

```
$('.box').css({
    'background-color': 'red',
    'font-size': '20px',
    'color': 'yellow'
});
```

Getter

```
<div class="box">…</div>
<div class="pod">…</div>
.pod { background: red }
var $color= $('.pod').css('background-color');
```



Méthode	Description
.width()	Définit ou retourne la largeur d'un élément. Les padding, border et les margin ne sont pas compris dans le calcul.
.innerWidth()	Définit ou retourne la largeur d'un élément. Les padding sont compris dans le calcul mais pas les border et les margin.
.outerdWidth()	Définit ou retourne la largeur d'un élément. Les padding et les border sont compris dans le calcul, mais pas les margin.
.outerdWidth(true)	Définit ou retourne la largeur d'un élément. Les padding, border et les margin sont compris dans le calcul.

Méthode	Description
.height()	Définit ou retourne la hauteur d'un élément. Les padding, border et les margin ne sont pas compris dans le calcul.
.innerHeight()	Définit ou retourne la hauteur d'un élément. Les padding sont compris dans le calcul mais pas les border et les margin.
.outerdHeight()	Définit ou retourne la hauteur d'un élément. Les padding et les border sont compris dans le calcul, mais pas les margin.
.outerdHeight(true)	Définit ou retourne la hauteur d'un élément. Les padding, border et les margin sont compris dans le calcul.

```
<button>Increase height of div with 200 px</button><br><br>

<div style="height:100px;border:4px solid;"></div>

$(document).ready(function(){
    $("button").click(function(){
        $("div").height($("div").height()+200);
    });
});
```

Cacher / Afficher

Dans cette partie, nous verrons les méthodes .hide(), .show() et .toggle().
Elles permettenet de faire apparaître ou disparaître des éléments avec une animation.
Ces méthodes peuvent prendre comme paramètres :
une vitesse

```
$('#hide').click(function(){
    $('#a').hide();
    $('#b').hide(1000);
    $('#c').hide('slow');
    $('#d').hide('slow', function(){
        console.log('Élément #d est caché');
    });
});
```

ou afficher/cacher alternativement

```
$('#toggle').click(function(){
    $('#a').toggle();
    $('#b').toggle(1000);
    $('#c').toggle('slow');
    $('#d').toggle('slow', function(){
        console.log('Élément #d est caché/affiché');
    });
});
```

Les fondus

Dans cette partie nous verrons les méthodes .fadeIn(), .fadeOut(), .fadeToggle()et.fadeTo()`. Elles permettent de faire apparaître ou disparaître des éléments avec une animation de fondu. La méthode .fadeTo() permet de modifier l'opacité d'un élément avec un effet de transition.

```
$('#fadeToggle').click(function(){
    $('#a').fadeToggle();
    $('#b').fadeToggle(1000);
    $('#c').fadeToggle('slow');
    $('#d').fadeToggle('slow', function(){
        console.log('Élément #d est apparu / à disparu');
    });
});
```

Un autre exemple :

```
$('#d').fadeTo('slow', 0.25, function(){
        console.log('Élément #d à une opacité de 25%');
```

Animation

Dans cette partie, nous verrons la méthode .animate() utilisée pour créer des animations personnalisées.
Méthode .animate() API : http://api.jquery.com/animate/

```
$('#play').click(function(){
    $('.pod').animate({left: '250px'}, 500);
});
```

```
$('#play').click(function(){
    $('.pod').animate({
```

```
        left: '250px',
        width: '+=150px'
    }, 500, function(){
        alert("L'animation est terminée");
    });
});
```

Chaîner deux animations

Nous avons la possibilité avec la méthode .animate() de chaîner les animations, c'est-à-dire de lancer une animation puis une autre.
Continuons de complexifier l'exemple du dessus. À la fin de l'animation class="pod" prend une opacité de 25%.

```
$('#play').click(function(){
    $('.pod').animate({
        left: '250px',
        width: '+=150px'
    }, 500)
    .animate({
        opacity: '0.25'
    }, 1000);
});
```

Nous pouvons aussi lancer les deux animations en même temps en fixant le paramètre queue à false. Cela aura pour effet de ne plus mettre l'animation ciblée à la queue.

```
$('#play').click(function(){
    $('.pod').animate({
        left: '250px',
        width: '+=150px'
    }, 500)
    .animate({
        opacity: '0.25',
    }, {
        'queue': false,
        'duration': 1000
    };
});
```

Stopper une animation / un effet
Dans cette partie, nous verrons la méthode .stop() servant à stopper une animation ou un effet.

Méthode .stop()
API : http://api.jquery.com/stop/
$(selecteur).stop(stopAll,goToEnd)
Cette méthode prend 2 paramètres :

Paramètre	Description
StopAll	Optionnelle : la valeur par défaut est false. Valeur booléenne indiquant true ou false. false permet d'arrêter toutes les animations en file d'attente.
goToEnd	Optionnelle : la valeur par défaut est false. Valeur booléenne indiquant true ou false. false permet de finir toutes les animations immédiatement.

Stop sans paramètres
Prenons l'exemple d'un carré qu'on anime avec la méthode .animate() au clic sur le bouton Play. Si nous cliquons sur le bouton stop, l'animation en cours se stoppe grâce à la méthode .stop() sans paramètres. Au click sur play, l'animation reprend là où elle en était.

```
// Déclaration de l'animation
$('#play').click(function(){
    $('.pod').animate({
        left: '250px',
        width: '+=150px'
    }, 1500)
    .animate({
        opacity: '0.25',
    }, 3000);
});

// Déclaration de la méthode .stop() au click
$('#stop').click(function(){
   $('.pod').stop();
});
```

Ajouter un délais

Dans cette partie, nous verrons la méthode .delay() servant à ajouter un délais avant l'exécution d'un effet.
Méthode .delay()
API : http://api.jquery.com/delay/

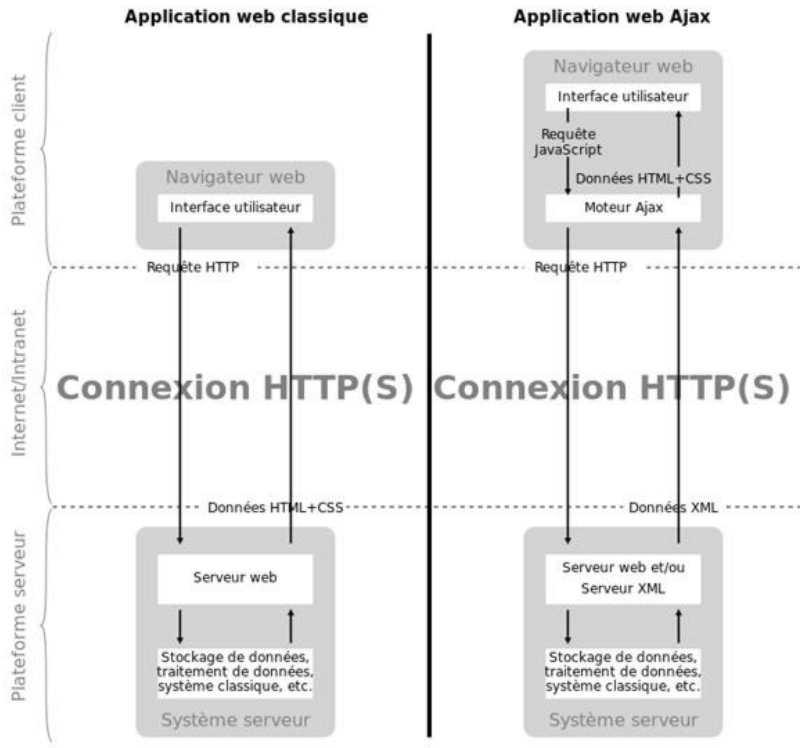
```
$('#b').slideUp().delay(1000).slideDown();
```

Pour information la méthode .delay() ne s'applique qu'aux effets. Pour temporiser d'autres méthodes, **il convient d'employer la fonction JavaScript setTimeout**.

Ajax

Ajax ou Asynchronous JavaScript and XML est une technologie se basant sur l'objet XMLHttpRequest.
XMLHttpRequest est un objet ActiveX ou JavaScript qui permet d'obtenir des données au format XML, JSON, mais aussi HTML, ou encore texte simple à l'aide de requêtes HTTP.
Ajax va pouvoir récupérer ou transmettre sur un serveur des données sans rafraichir notre navigateur.
Exemples d'applications utilisant AJAX : Gmail, Google Maps, Youtube et les onglets Facebook.
Le principe Source : Wikipédia

Dans une application Web, la méthode classique de dialogue entre un navigateur et un serveur est la suivante : lors de chaque manipulation faite par l'utilisateur, le navigateur envoie une requête contenant une référence à une page Web, puis le serveur Web effectue des calculs, et envoie le résultat sous forme d'une page Web à destination du navigateur. Celui-ci affichera alors la page qu'il vient de recevoir. Chaque manipulation entraîne la transmission et l'affichage d'une nouvelle page. L'utilisateur doit attendre l'arrivée de la réponse pour effectuer d'autres manipulations.
En utilisant Ajax, le dialogue entre le navigateur et le serveur se déroule la plupart du temps de la manière suivante : un programme écrit en langage de programmation JavaScript, incorporé dans une page web, est exécuté par le navigateur. Celui-ci envoie en arrière-plan des demandes au serveur Web, puis modifie le contenu de la page actuellement affichée par le navigateur Web en fonction du résultat reçu du serveur, évitant ainsi la transmission et l'affichage d'une nouvelle page complète.
La méthode classique de dialogue utilise des mécanismes propres au World Wide Web, qui sont incorporés dans tous les navigateurs ainsi que les robots d'indexation, et ne nécessite pas de programmation. Au contraire, le fonctionnement d'Ajax nécessite de programmer en JavaScript les échanges entre le navigateur et le serveur Web. Il nécessite également de programmer les modifications à effectuer dans la page Web à la réception des réponses, sans quoi les dialogues se font à l'insu de l'utilisateur.
En Ajax, comme le nom l'indique, les demandes sont effectuées de manière asynchrone : le navigateur Web continue d'exécuter le programme JavaScript alors que la demande est partie, il n'attend pas la réponse envoyée par le serveur Web et l'utilisateur peut continuer à effectuer des manipulations pendant ce temps.



La méthode .load()API : http://api.jquery.com/load/

La méthode .load() permet de récupérer du contenu à insérer dans notre page. Nous pouvons charger des fichiers du type : HTML, PHP, TXT, XML et JSON.
Elle est un raccourci de la fonction .ajax().

$(selecteur).load(url,data,function(reponse,status,xhr)));

Paramètre	Description
url	Requis Une chaîne contenant l'URL vers laquelle la demande est envoyée
data	Optionnel Spécifie les données à envoyer vers le serveur en même temps que la demande
function(reponse,status,xhr)	Optionnel Indique une fonction à exécuter quand la méthode est terminée Paramètres supplémentaires : - data - contient les données résultant de la demande - status - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror") - xhr - contient l'objet XMLHttpRequest

Exemples
Charger un document

```
$('#result').load('result.txt');
```

Faisons de même avec un fichier PHP mais cette fois-ci au clic.

```
$('#a').click(function(){
    $('#result').load('inc/result.php');
});
```

Bien sûr, nous pouvons ajouter des effets de fondu. La méthode consiste à cacher le contenu, le charger puis faire le fondu en callback.

```
$('#a').on('click', function(){
    var box = $('#result');
    box.hide().load('inc/result.php', function() {
        box.fadeIn('750');
    });
});
```

Passer des paramètres à un fichier PHP

Il nous est possible de demander en PHP à la base de données de nous retourner des informations nécessitant un paramètre.
Dans l'exemple ci-dessous, nous récupérons la valeur d'un champ #search, puis nous la passons en paramètre.

```
$('#a').click(function(){
    var param = 'id=' + $('#search').val();
    $('#result').load('inc/result.php', param);
});
```

La méthode .get()API : http://api.jquery.com/jQuery.get/

La méthode .get() permet de recevoir des données. Contairemement à la méthode .load(), elle nous permet non pas de recevoir du contenu directement dans l'élément ciblé mais de nous laisser le choix sur les actions à faire.
La méthode en détail
$.get(url, data, function(data,status,xhr), dataType);

Paramètre	Description
url	Requis Une chaîne contenant l'URL vers laquelle la demande est envoyée
data	Optionnel Spécifie les données à envoyer vers le serveur en même temps que la demande
function(data,status,xhr)	Optionnel Indique une fonction à exécuter lorsque la méthode est terminée

Paramètre	Description
	Paramètres supplémentaires : - data - contient les données résultant de la demande - status - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror") - xhr - contient l'objet XMLHttpRequest
dataType	Optionnel Spécifie le type de données attendu. Par défaut jQuery effectue une estimation automatique (HTML, PHP, TXT, XML et JSON)

Exemples

Charger un document

Prenons l'exemple suivant, le fichier 'inc/test.html' se charge dans #result.

```
$.get('inc/test.html', function( data ) {
    $('#result').html( data );
});
```

Elle est un raccourci de la fonction .ajax() suivante. Nous la verrons plus en détails dans la prochaine partie.

```
$.ajax({
    url: 'inc/test.html',
    success: function (data) {
        $('#result').html( data );
    }
});
```

Maintenant, affichons le statut dans la console.

```
$.get('inc/test.html', function( data, status ) {
    $('#result').html( data );
    console.log(status);
});
```

Charger des données de type JSON

Voici un exemple de chargement de données issues du fichier JSON à l'aide de .get().

Prenons l'exemple de fichier JSON.

```
{
    "name": "Jean-Michel",
    "email": "jeanmich@caramail.com"
}
Voici le script nécessaire à l'affichage des informations.
$.get('inc/user.json', function( data ) {
```

```
    $('#result').html( data.name + ' : ' +  data.email );
}, 'json');
```

JQuery à aussi une méthode raccourcie pour le JSON .getJSON()

```
$.getJSON('inc/user.json', function( data ) {
    $('#result').html( data.name + ' : ' +  data.email );
});
```

Ce dernier script est le raccourci de la méthode ajax suivante.

```
$.ajax({
    dataType: "json",
    url: 'inc/test.html',
    success: function (data) {
        $('#result').html( data.name + ' : ' +  data.email );
    }
});
```

La méthode .post()API : http://api.jquery.com/jQuery.post/

La méthode .post() permet d'envoyer des données. Par exemple, vous pouvez l'utiliser pour envoyer des données saisies dans un formulaire.
La méthode consiste à envoyer des données vers un fichier php qui se chargera de les transmettre au serveur. Cette méthode peut également retourner des informations en callback dans la page.

```
$.post('send.php',
    {
        name: 'Jean-Michel',
        email: 'jeanmich@caramail.com'
    }, function(data) {
        alert(data);
});
```

Paramètre	Description
url	Requis Une chaîne contenant l'URL vers laquelle la demande est envoyée

data	Optionnel Spécifie les données à envoyer vers le serveur en même temps que la demande
function(data,status,xhr)	Optionnel Indique une fonction à exécuter lorsque la méthode est terminée Paramètres supplémentaires : - data - contient les données résultant de la demande - status - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror") - xhr - contient l'objet XMLHttpRequest
dataType	Optionnel Spécifie le type de données attendu. Par défaut jQuery effectue une estimation automatique (xml, json, script, ou html)

Exemple d'envoi de données

Dans l'exemple suivant, nous envoyons une requête permettant de faire une recherche puis nous affichons le résultat de la recherche.

Cet exemple ne contient pas le fichier PHP de traitement (ce n'est pas le sujet de ce cours).

La structure html

```
<form action="/" id="searchForm">
    <input type="text" name="s" placeholder="Rechercher…">
    <input type="submit" value="Ok">
</form>
<div id="result"></div>
```

Le script jQuery de traitement

```
$('#searchForm').submit(function(event) {

  // Stop la propagation par défaut
  event.preventDefault();

  // Récupération des valeurs
  var $form = $(this),
       term = $form.find( "input[name='s']" ).val(),
       url = $form.attr( "action" );

  // Envoie des données
  var posting = $.post( url, { s: term } );
```

```
  // Reception des données et affichage
  posting.done(function(data) {
    var content = $(data).find('#content');
    $('#result').empty().append(content);
  });
```

});

La méthode .ajax()
La méthode .ajax() est l'artillerie lourde. Elle permet de maîtriser l'ensemble des paramètres de requête. Les autres méthodes vues juste avant sont seulement des raccourcis d'.ajax().
Cette méthode peut s'écrire de deux façons:

```
$.ajax(url, {options});
```

ou

```
$.ajax({options});
```

Paramétrer sa requête
Voici les options les plus utilisées :

Option	Description
URL	Adresse à laquelle la requête est envoyée
type	Le type de requête GET (par défaut) ou $POST
data	Les données à envoyer au serveur
datatype	Le type de données pouvant être transmises au serveur : php, html, script, json et xml
success	La fonction à appeler si la requête a abouti
error	La fonction à appeler si la requête n'a pas abouti
timeout	Le délai maximum en millisecondes de traitement de la demande. Passé ce délai, elle retourne paramètre error.

Bien sûr, il existe bien d'autres options que vous pouvez retrouver dans l'API jQuery.
Chargement de fichier
Voici un exemple simple permettant de comprendre le fonctionnement.
// Au clic sur le bouton #search je lance la fonction

```
$('#search').on('click', function(){

    // J'initialise le variable box
    var box = $('#result');

    // Je définis ma requête ajax
    $.ajax({
```

```
        // Adresse à laquelle la requête est envoyée
        url: '../inc/search.php',

        // Le délai maximun en millisecondes de traitement de la
demande
        timeout: 4000,

        // La fonction à apeller si la requête aboutie
        success: function (data) {
            // Je charge les données dans box
            box.html(data);
        },

        // La fonction à appeler si la requête n'a pas abouti
        error: function() {
            // J'affiche un message d'erreur
            box.html("Désolé, aucun résultat trouvé.");
        }

    });

});
```