
jQuery
write less, do more.

Table des matières

Chapitre 1 - PRÉSENTATION....5
1.1 - jQuery....6
1.2 - Téléchargement et installation....7
1.3 - Exemple d'arborescence de site....11
1.4 - Les pré-requis pour utiliser la bibliothèque jQuery....12
1.5 - jQuery : une seule fonction....13
1.6 - Exemple : afficher ou masquer une photo....17
1.7 - Autre exemple : FlipFlop....20
1.8 - « Écrire » dans la page....21
1.8.1 - Récupérer ou modifier le contenu d'un élément de formulaire....21
1.8.2 - Récupérer ou modifier le contenu d'un élément qui n'est pas un élément de formulaire....21
1.8.3 - Récupérer ou modifier la valeur d'un attribut....21
1.8.4 - Exercice : affichez les saisies (AuthentificationV1_EXO.html)....21
1.9 - Les conflits avec d'autres bibliothèques....22
Chapitre 2 - LES PARAMÈTRES ET SÉLECTEURS....23
2.1 - Un objet : l'objet document et la méthode ready()....24
2.2 - Les sélecteurs CSS....25
2.3 - Les sélecteurs de filtre....28
2.4 - Les sélecteurs de formulaire....29
2.4.1 - Syntaxes et exemples....29
2.4.2 - Exercice (ControlesSaisies) :....31
2.5 - Les sélecteurs hiérarchiques....32
2.6 - Les sélecteurs spécifiques de jQuery....33
2.7 - L'absence de getElementsByName....34
2.8 - Les méthodes....35
2.9 - Les « textes » dans la page HTML....36
Chapitre 3 - GESTION ÉVÉNEMENTIELLE....37
3.1 - Les bases....38
3.2 - La liste des événements gérés par jQuery....41
3.3 - Création et destruction d'événements....42
3.4 - Inhiber un événement....44
3.5 - Délégation d'événement....47
Chapitre 4 – GESTION DU CSS....48
4.1 - css()....49
4.2 - Positionnement....51
4.3 - Dimensionnement....52
4.4 - Exemple : modifications des propriétés de style....53
Chapitre 5 – GESTION DES EFFETS ET DES ANIMATIONS....55
5.1 - Tableau résumé des effets....56
5.2 - Les différentes méthodes....57

5.2.1 - La méthode show()....57
5.2.2 - La méthode hide()....59
5.2.3 - La méthode toggle()....60
5.2.4 - La méthode slideDown()....61
5.2.5 - La méthode slideUp()....61
5.2.6 - La méthode slideToggle()....62
5.2.7 - La méthode fadeIn()....63
5.2.8 - La méthode fadeOut()....63
5.2.9 - La méthode fadeTo()....64
5.2.10 - Exemple récapitulatif....65
5.2.11 - La méthode animate()....68
5.2.12 - La méthode stop()....74
Chapitre 6 – GESTION DU DOM....75
6.1 - Présentation....76
6.1.1 - Sélectionner un élément....77
6.2 - Récupérer une valeur saisie....78
6.3 - Activer/Désactiver un élément....79
6.4 - Afficher/Masquer un mot de passe....80
6.4.1 - Exemple....80
6.4.2 - Exercice : changer aussi le texte....80
6.5 - Récupérer le parent, les enfants, ...81
6.6 - Sélectionner un ou plusieurs éléments....82
6.6.1 - Sélectionner un élément....82
6.6.2 - Sélectionner plusieurs éléments....83
6.6.3 - Récupérer la valeur d'un attribut d'un élément....84
6.6.4 - Affecter une valeur à un attribut d'un élément ou à un texte....86
6.6.5 - Le cas de l'attribut classe....87
6.6.6 - Travailler avec l'élément <select>....88
6.6.7 - Travailler avec une case à cocher....91
6.6.8 - Travailler avec un bouton radio....91
6.7 – Ajouts....92
6.7.1 - Ajouter un élément....92
6.7.2 - Ajouter une série d'éléments....94
6.7.3 - Insérer un élément....96
6.7.4 - Ajouter un élément qui en enveloppe un autre (Wrap)....97
6.7.5 - Ajouter un attribut....99
6.7.6 - Exercice : créer dynamiquement des éléments de formulaire....100
6.8 - Suppressions....101
6.8.1 - Supprimer un élément....101
6.8.2 - Supprimer tous les éléments enfants....101
6.8.3 - Supprimer un élément enfant....102
6.8.4 - Supprimer un attribut....103
6.8.5 - Exercice : transférer des items d'une liste vers une autre....104

6.9 - Clonage....105
6.10 - Itération....107
6.10.1 - Présentation....107
6.10.2 - Exercice : récupérer les valeurs saisies dans un formulaire....114
6.10.3 - Exercice : récupérer les éléments multiples d'un formulaire....116
6.10.4 - Exercice : CRUD panier....119
6.11 - Gestion générique d'un événement sur les items d'une liste....127
6.11.1 - Objectif....127
6.11.2 - Principe et démarche....127
6.11.3 - Codes....128
Chapitre 7 - ANNEXES....130
7.1 - Le CSS du site du cours....131
7.2 - Le modèle HTML pour le site du cours....133
7.3 - Les inclusions pour le site du cours....134
7.4 - Menu coulissant avec icône Hamburger....135
7.5 - Détecter l'orientation....141
7.6 - Storages....142
7.7 - Corrigés des exercices....142
7.8 - Liens, bibliographies....143
7.9 - TODO....144

CHAPITRE 1 - PRÉSENTATION

1.1 - jQuery

« jQuery est une bibliothèque JavaScript rapide, petite et riche en fonctionnalités. Elle permet la manipulation d'un document HTML, la gestion des événements, l'animation et de faire de l'Ajax de façon plus simple avec une API qui fonctionne sur une multitude de navigateurs. Avec une combinaison de polyvalence et d'extensibilité, jQuery a changé la façon dont des millions de gens écrivent du JavaScript. »
Traduction de la citation http://jquery.com/

jQuery est une bibliothèque JavaScript Open source qui porte sur l'interaction entre JavaScript, HTML et CSS.

La première version date de janvier 2006.
Au 1er février 2013 la version était la 1.9.0.
Au 18 avril 2013 la version était la 2.0.0.
Au 1er juillet 2015 c'est la version 2.1.4.
Au 1er mars 2016 c'est la version 2.2.1.
Au 21 mars 2017 c'est la version 3.2.1.
Au 1er mars 2018 c'est la version 3.3.1.
Au 1er juin 2020 c'est la version 3.5.1.

La bibliothèque apporte les fonctionnalités suivantes :

- ✓ Gestion de l'arbre DOM,
- ✓ Support de base de XPath (recherche dans un arbre DOM),
- ✓ Support de CSS3,
- ✓ Gestion des événements,
- ✓ Gestion des effets et des animations,
- ✓ Gestion AJAX.

Il existe une bibliothèque UI (User Interface) et de très nombreux plugins (TableSorter, …, plus de 2000).

Cinq documents couvrent ce module :

jquery.odt (Présentation et éléments de base),
jquery_ui.odt (User Interface : création d'interfaces riches),
jquery_pi.odt (PlugIns : création d'interfaces riches),
jquery_ajax.odt (Récupération de données serveur),
jquery_et_les_tableaux.odt (Document annexe).

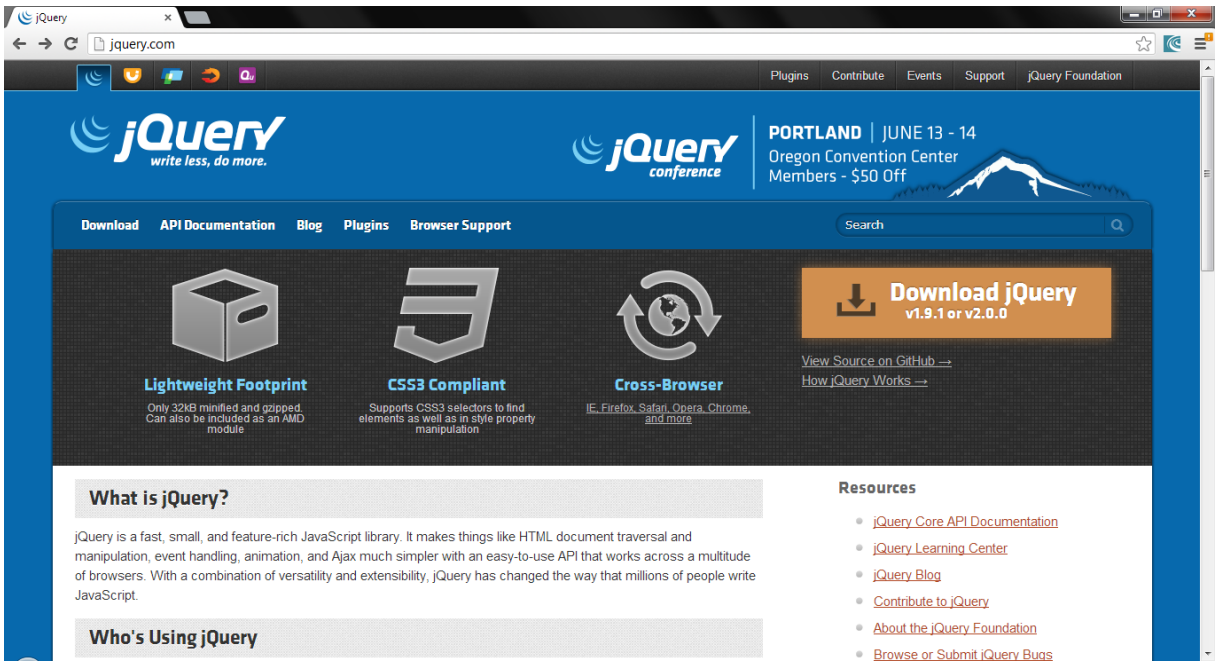
Les exercices sont principalement dans le document Ajax et feront appels aux éléments vus dans les autres documents.
Plus un petit TD nommé … ça dépend des années.

A partir du moment où vous utilisez jQuery vous ne devez plus gérer le DOM avec JavaScript (donc plus de document.getElementById(), …, de window.onload, ...

1.2 - Téléchargement et installation

- **Site de jQuery**

http://jquery.com/



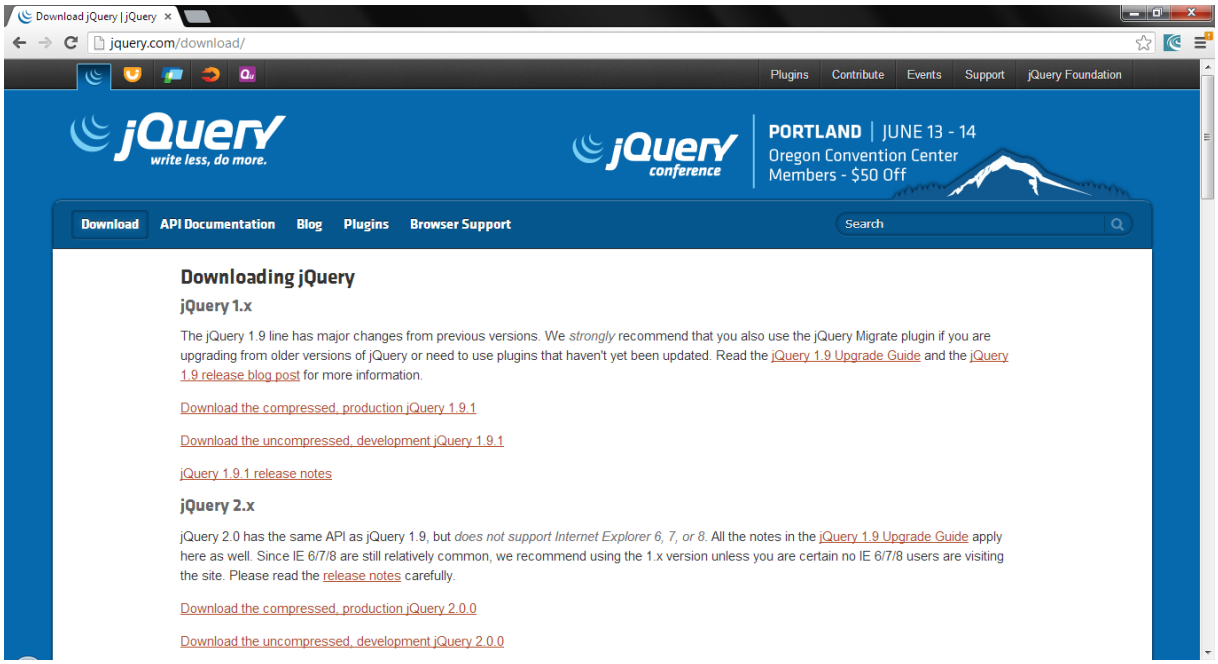
Copie d'écran du 01/05/2013.

- **Téléchargement**

http://jquery.com/download/

Il existe 2 versions :

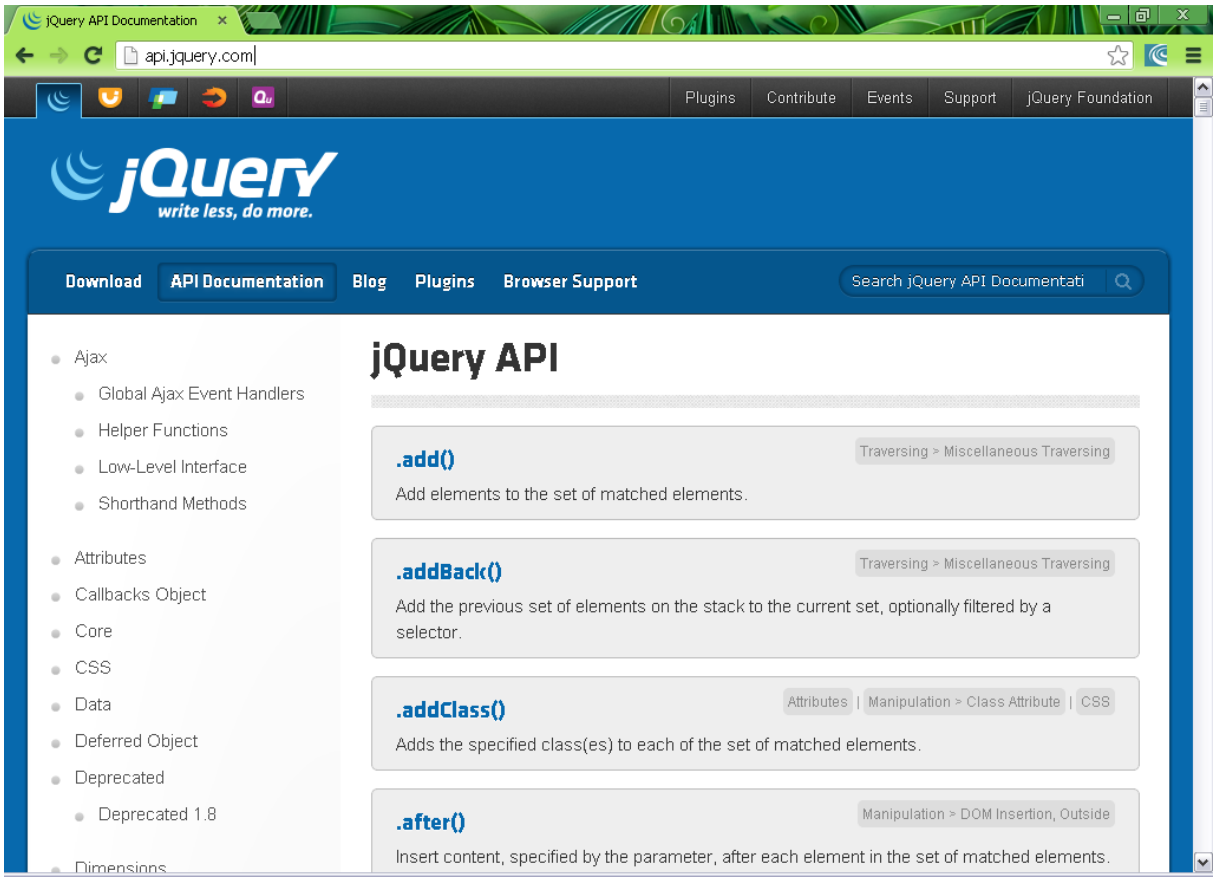
Minified : recommandée en production (≈ 90 ko).
Uncompressed : recommandée en période de développement (≈ 230 ko).



Copie d'écran du 01/05/2013.

- **La documentation de l'API de référence**

http://api.jquery.com/



Copie d'écran du 01/05/2013

La documentation sous forme de zip :

jquery-api-browser.zip

- **Installation**

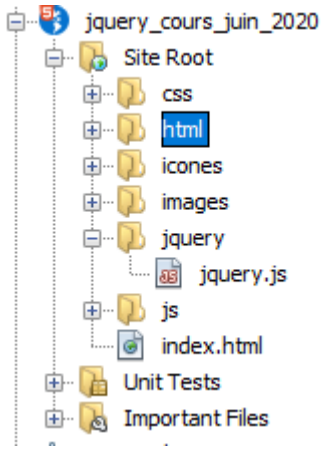
Vous avez créé un projet NetBeans de type HTML5/JS Application nommé « jquery_cours ».
Dans ce projet vous créez les dossiers suivants : html, css, js, images, icones + un dossier nommé « jquery ».

Une fois la bibliothèque téléchargée, copiez le fichier (jquery-x.x.x.js ou jquery.js) dans l'arborescence de votre site dans le dossier nommé **jquery**.

Éventuellement renommez le fichier jquery-x.x.x.js en jquery.js.

1.3 - Exemple d'arborescence de site

Dans le dossier du site JavaScript, un dossier jquery pour la production jQuery, un dossier pour les exemples du cours, un dossier pour les exercices.



1.4 - Les pré-requis pour utiliser la bibliothèque jQuery

Pour utiliser jQuery dans une page HTML vous devez inclure au moins le "noyau". Le plus souvent comme première « inclusion ».

```
<script src="../jquery/jquery.js "></script>
```

Le chemin étant fonction de votre installation sur votre serveur.

Un des modèles de page utilisé :

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <link rel="icon" type="image/png" href="../icones/favicon.png">
        <!-- Vos CSS -->
        <link rel="stylesheet" type="text/css" href="../css/style.css" />
        <title>_modeleJQ.html</title>
    </head>

    <body>
        <!-- Votre code HTML ICI -->

        <!-- Votre code JS ICI -->
        <!-- Chargement de la bibliothèque jQuery -->
        <script src="../jquery/jquery.js"></script>

        <!-- Vos scripts externes -->
        <script src="../js/scripts.js"></script>

        <!-- Vos scripts internes -->
        <script>
            // ------------
            function init() {
                console.log("init");
            } /// init

            // -------------
            function autre() {

            } /// autre

            // Au chargement
            $(document).ready(init);
        </script>
    </body>
</html>
```

Note : les éléments <script> sont le plus souvent insérés à la fin de l'élément <body>.

1.5 - jQUERY : UNE SEULE FONCTION

jQuery repose sur **une seule fonction** : **jQuery()** ou **$()**.

Elle accepte divers types de paramètres (un objet, une chaîne CSS dit sélecteur CSS, une chaîne jQuery dit sélecteur jQuery, une fonction, etc) et retourne un objet ou un tableau d'objets.
Elle possède de nombreuses méthodes. Cette phrase est bizarre ! Ce qui tend à attester que cette fonction est un objet ou une classe. Dans tous les cas elle renvoie un objet ou un ensemble d'objets, qui eux ont accès à des méthodes.

Remarque : il sera donc toujours nécessaire d'identifier un élément HTML. Et aussi de le nommer s'il fait partie d'un formulaire.

Notes :
Exemples d'objets : document, document.body, document.head, un objet que vous avez crée, ...
Exemples de sélecteurs CSS : "#titre", ".rouge", "img", ...

- **Syntaxes de la fonction jQuery() ou $()**

Pour créer un objet jQuery (ou plusieurs).

```
let objet = jQuery("sélecteur");
```

ou

```
let objet = $("sélecteur");
```

ou

```
let objets = $("sélecteur");
```

Un sélecteur peut être un identifiant, une classe, une balise, un objet (pour ce dernier ne mettez pas de quote ou de double-quote).

Ceci correspond à **document.getElementById("id")** ou à **document.getElementsByTagname("balise")** de JavaScript ou encore **document.getElementsByClassname("classe")** ou encore **document.getElementsByName("classe")** …
L'argument peut aussi être un objet (document par exemple) ou un sélecteur spécifique à jQuery (cf plus loin).

Pour solliciter une méthode de jQuery.

```
jQuery("sélecteur").méthode([argument(s)]);
```

ou

```
$("sélecteur").méthode([argument(s)]);
```

ou

```
let objet = $("sélecteur");
objet.méthode([argument(s)]);
```

Note : dorénavant nous utiliserons $.

L'événement ready.

jQuery manipule le DOM. Or avant toute manipulation il faut s'assurer que le document HTML, plus exactement le DOM, soit entièrement chargé.

```
$(document).ready(fonction);
```

```
$(document).ready(function() {
    $("#btAfficher").on("click", afficher);
    $("#btMasquer").on("click", masquer);
} );
```

- **Exemples**

```
// Renvoie un objet jQuery via un sélecteur CSS de type identifiant
let oJQuery = $("#titre");
```

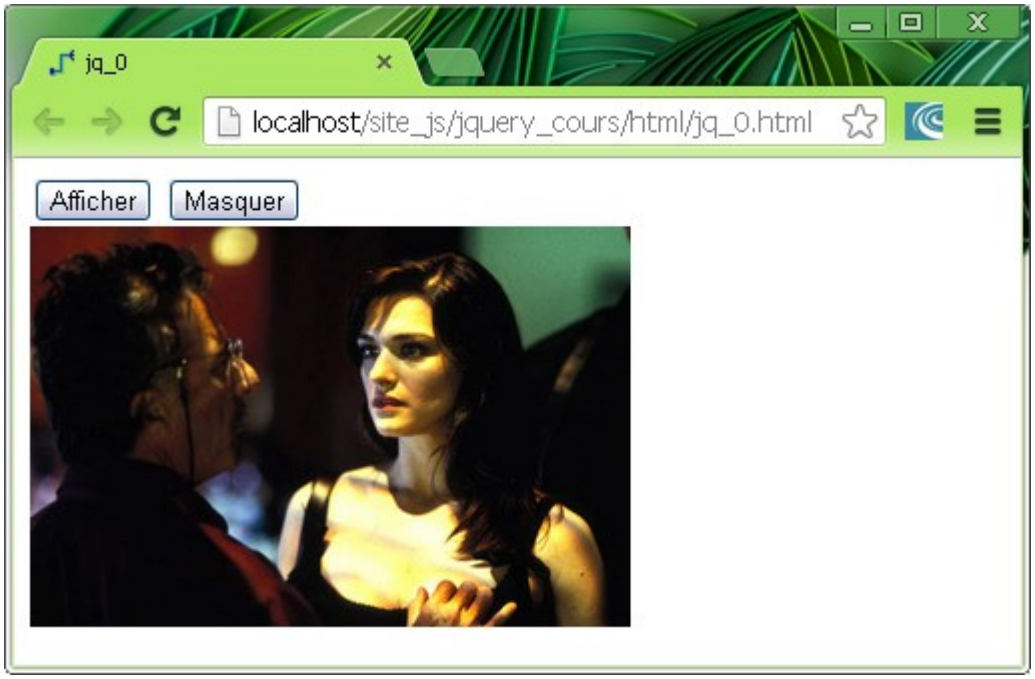
```
// Renvoie un tableau d'objets jQuery via un sélecteur CSS de type balise
let objets = $("img");
```

```
// Exécute une méthode de jQuery
// Masque toutes les images via un sélecteur CSS de type balise
$("img").hide();
```

```
// Exécute la fonction init() à la fin du chargement de la page HTML via
un sélecteur de type objet
$(document).ready(init);
```

1.6 - Exemple : afficher ou masquer une photo

Objectif



Démarche résumée :

créez un document HTML,
créez une balise <script> pour inclure la bibliothèque jQuery,
créez les éléments HTML en les identifiant,
créez une balise <script> pour coder la fonction init() et les autres fonctions dont vous aurez besoin (afficher() et masquer()).
Dans la fonction init() reliez les éléments d'interaction (boutons, …) aux fonctions.
Exécuter la fonction init() au chargement du document.

Démarche détaillée

La bibliothèque de jQuery est incluse dans le script.

```
<script src="../jquery/jquery.js"></script>
```

L'image et les boutons sont identifiés.
Il est possible de stocker l'image dans une variable et d'appliquer ensuite la méthode show() à cette variable en lui passant éventuellement le paramètre "slow" ou "fast".
Notez que l'on référence l'objet avec une notation CSS (#id).

```
let oJQ = $("#img1");
oJQ.show("slow");
```

```
$("#img1").show("slow");
```

Argument	Description
"slow"	600 millisecondes
"normal"	400 millisecondes
"fast"	200 millisecondes
n	n millisecondes
	Instantané

Le rattachement d'une fonction à un événement d'un élément s'effectue via la syntaxe suivante :

```
$("#identifiant").on("événement", fonction)
```

```
$("#btAfficher").on("click", afficher);
```

signifie que la fonction afficher() sera exécutée lorsque l'internaute cliquera sur le bouton identifié par btAfficher.

Les événements peuvent être click, change, load, unload, blur, focus, … cf plus loin le chapitre sur les événements.

Le rattachement (bind : liaison) de l'événement à l'élément est réalisé dans une fonction d'initialisation - par exemple nommée init() – celle-ci étant appelée par la méthode ready() de l'objet document :

```
$(document).ready(init); // Exécute la fonction init() au chargement
```

Note : les fonctions passées en argument peuvent être des fonctions anonymes (cf plus loin).

Script complet

```
<!DOCTYPE HTML>
<html>
<head>
     <meta charset="utf-8" />
     <link rel="icon" type="image/png" href="../icones/favicon.png">
     <title>jq0.html</title>
</head>

<body>
     <input type="button" id="btAfficher" value="Afficher" />
     <input type="button" id="btMasquer"  value="Masquer" /><br />
     <img src="../images/0.jpg" alt="Affiche" id="img1" />

     <script src="../jquery/jquery.js"></script>

     <script>
           // ------------
           function init() {
                 $("#btAfficher").on("click", afficher);
                 $("#btMasquer").on("click", masquer);
           } /// init

           // ----------------
           function afficher() {
                 $("#img1").show("slow");
           } /// afficher

           // ---------------
           function masquer() {
                 $("#img1").hide("slow");
           } /// masquer

           // Exécute la fonction init() a la fin du chargement du
document HTML
           $(document).ready(init);
     </script>

</body>
</html>
```

1.7 - Autre exemple : FlipFlop

Lorsque l'on passe la souris sur une image, elle change de source; idem lorsque la souris sort du champ de l'image.

Événements : mouseover et mouseout.
Méthode : objet.attr("attribut", valeur)

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <link rel="icon" type="image/png" href="../icones/favicon.png">
        <title>FlipFlopJQ.html</title>
    </head>

    <body>
        <img id="photo" src="../images/0.jpg" alt="Image" width="100"/>

        <script src="../jquery/jquery.js"></script>
        <script>
            // ------------
            function init() {
                $("#photo").on("mouseover", function() {
                    $("#photo").attr("src", "../images/1.jpg");
                });
                $("#photo").on("mouseout", function() {
                    $("#photo").attr("src", "../images/0.jpg");
                });
            } /// init

            $(document).ready(init);
        </script>

    </body>
</html>
```

Note : il est possible de factoriser en créant une fonction afficher(element, image). Cf les annexes.

1.8 - « Écrire » dans la page

1.8.1 - Récupérer ou modifier le contenu d'un élément de formulaire

Donc possédant l'attribut « value ».

Méthode	Affectation (set)	Récupération (get)
val	$("#id").val("valeur");	let mdp = $("#id").val();

1.8.2 - Récupérer ou modifier le contenu d'un élément qui n'est pas un élément de formulaire

Donc ne possédant pas l'attribut « value ».

Méthode	Affectation (set)	Récupération (get)
html	$("#id").html("valeur");	let message = $("#id").html();
text	$("#id").text("valeur");	let message = $("#id").text();

La méthode text() est équivalente à la méthode html() pour un document HTML mais pour un document XML il faut utiliser la méthode text().

1.8.3 - Récupérer ou modifier la valeur d'un attribut

Méthode	Affectation (set)	Récupération (get)
attr	$("#id").attr("attribut", "valeur");	let type = $("#id").attr("attribut");

1.8.4 - Exercice : affichez les saisies (AuthentificationV1_EXO.html)

1.9 - LES CONFLITS AVEC D'AUTRES BIBLIOTHÈQUES

L'objectif est de pallier les conflits avec d'autres bibliothèques (Prototype, Mootools, …) qui utilise aussi le symbole $ comme raccourci :

- ✓ Chargez jQuery (Si vous travaillez avec prototype.js, jQuery doit être chargé en dernier),
- ✓ Utilisez la méthode jQuery.noConflict() qui inhibe le raccourci $ et permet de substituer un autre symbole (j, $j, …).

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <link rel="icon" type="image/png" href="../icones/favicon.png">
    <title>JQConflict</title>
</head>

<body>
    <input id="btTest" type="button" value="Test" />
    <label id="lblMessage"></label>

    <script src="../../prototype/prototype.js"></script>
    <script src="../jquery/jquery.js"></script>
    <script>
       var j = jQuery;

       // ------------
       function init() {
           j("#btTest").on("click", autre);
       } /// init
       // -------------
       function autre() {
          j("#lblMessage").html("CLIC");
       } /// autre

       jQuery.noConflict();
       j(document).ready(init);
    </script>

</body>
</html>
```

CHAPITRE 2 - LES PARAMÈTRES ET SÉLECTEURS

La référence : http://api.jquery.com/category/selectors/

2.1 - Un objet : l'objet document et la méthode ready()

La méthode **ready()** appliquée à l'objet **document** exécute la fonction passée en argument à la fin du chargement de la page dans le navigateur, plus précisément, lorsque le DOM est chargé.

L'exécution d'une fonction.

```
$(document).ready(init); // Exécute la fonction init() au chargement
```

L'exécution d'une fonction anonyme pour exécuter plusieurs instructions.

```
$(document).ready(function() {
    $("#btAfficher").on("click", afficher);
    $("#btMasquer").on("click", masquer);
} );
```

Note :
cf aussi les objets document.body, document.head et vos objets.

2.2 - Les sélecteurs CSS

Les sélecteurs CSS permettent de travailler sur un élément DOM (via un identifiant) ou plusieurs éléments DOM (via une balise, une classe, etc).

Sélecteur	Description
#id	Un identifiant
*	Tous les éléments
balise	Les éléments correspondant à une balise
balise1,balise2, …	Les éléments correspondant à plusieurs balises
.classe	Une classe
balise.classe	Les éléments <balise> de type "classe"
balise balise ...	Les éléments de … (les th des tr d'un thead ...)

Exemples :

```
// Un objet
let oJQuery = $("#image1"); // Une image identifiée
```

```
// Un tableau d'objets
let toJQuery = $("img"); // Toutes les images donc un tableau
```

```
// Un tableau d'objets
let toJQuery = $("#menu a"); // Les ancres d'un menu identifié
```

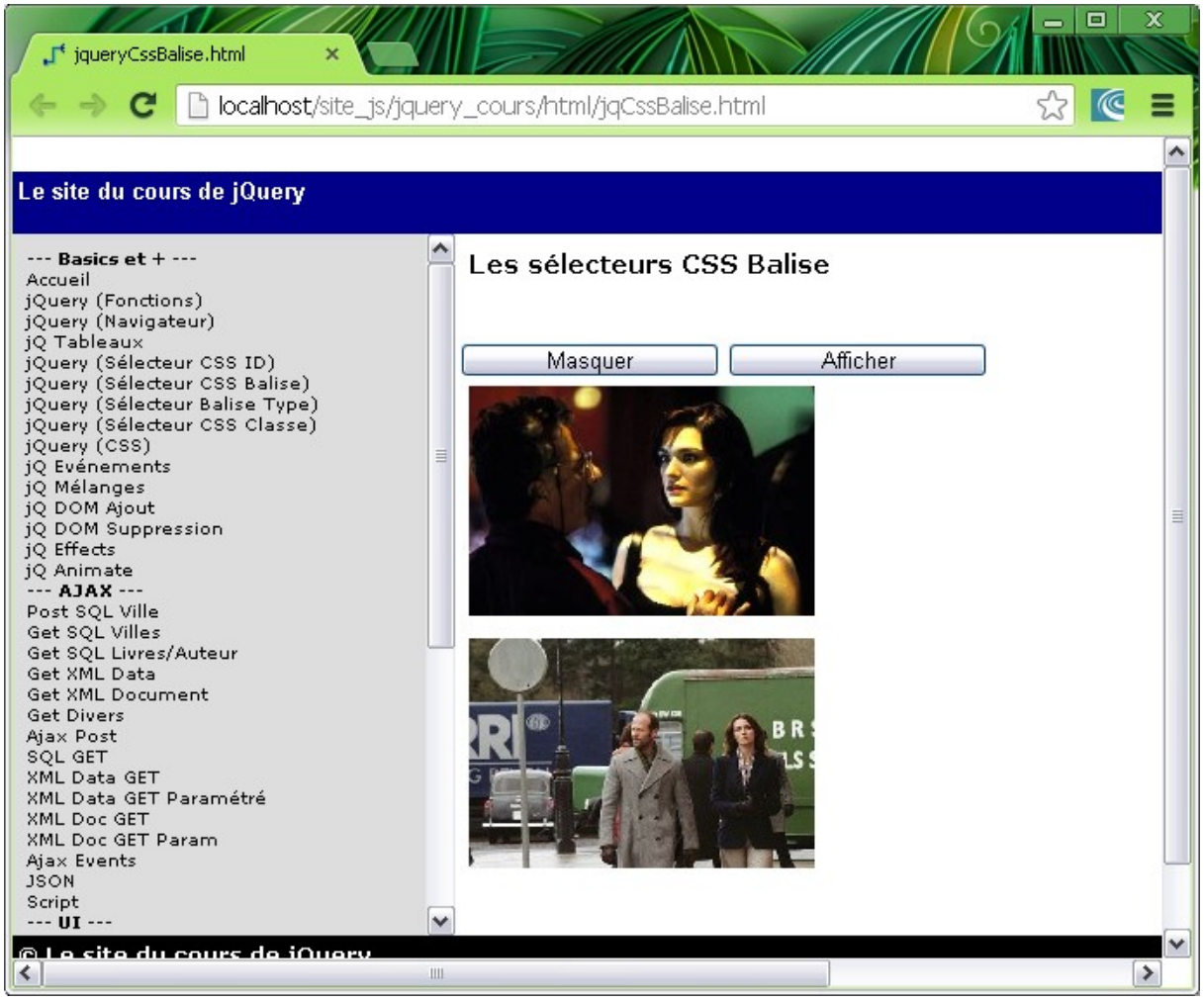
```
// Un tableau d'objets
let toJQuery = $(".rouge"); // Tous les éléments de classe .rouge
```

```
// Un tableau d'objets
let toJQuery = $("p.rouge"); // Tous les paragraphes de classe .rouge
```

```
// Un tableau d'objets
let toJQuery = $("h3,p"); // Tous les <h3> et les <p>
```

```
// Un tableau d'objets
let ths = $("thead tr th"); /// Le cellules d'un header de table
```

Exemple : afficher ou masquer toutes les photos.



Pour travailler sur tous les éléments de même type l'argument de jQuery est une balise.

```
<head>
   <meta charset="UTF-8">
   <link rel="icon" type="image/png" href="../icones/favicon.png">
   <title>CssBaliseJQ.html</title>
</head>

<body>
    <input type="button" id="btAfficher"  value="Afficher" />
    <input type="button" id="btMasquer"   value="Masquer" /><br />
    <p><img src="../images/0.jpg" height="150" alt="1" /></p>
    <p><img src="../images/1.jpg" height="150" alt="2" /></p>

    <script src="../jquery/jquery.js"></script>

    <script>
        // ----------------
        function afficher() {
           $("img").show(3000); // Correspond à 3 secondes
       } /// afficher

       // ---------------
       function masquer() {
           $("img").hide(3000);
       } /// masquer

       $(document).ready(function() {
            $("#btAfficher").on("click", afficher);
            $("#btMasquer").on("click", masquer);
        } );
</script>

</body>
```

2.3 - Les sélecteurs de filtre

Les sélecteurs de filtre permettent de travailler sur des éléments HTML ayant certaines caractéristiques de contenu.

Sélecteur	Description
$("élément:empty")	Les éléments vides
$("élément:contains('valeur')")	Les éléments contenant une valeur particulière
$("élément:has(élément)")	Les éléments contenant d'autres éléments
…	

Exemples :

```
// Affiche Vide et colorie en rouge les <td> vides
$("td:empty").text("Vide!").css('background', 'rgb(255,0,0)');
```

```
// Souligne le contenu des paragraphes qui contiennent le mot Dupont
$("p:contains('Dupont')").css("text-decoration", "underline");
```

```
// Colorie en vert les divisions qui contiennent des paragraphes
$("div:has(p)").css('background', 'rgb(0,185,0)');
```

2.4 - Les sélecteurs de formulaire

2.4.1 - Syntaxes et exemples

Les sélecteurs de formulaire permettent de sélectionner des éléments de formulaire.

Sélecteur	Description
:input	Les <input>, <textarea>, <button>, <submit> et <reset>.
:text | :password | :radio | :checkbox | :submit | :image | :reset | :button | :file | :hidden	Les <input> du type en question.
:enabled :disabled :checked :selected	Les éléments ayant l'attribut en question. Pour enabled (qui n'existe pas en HTML) ce sont les éléments qui n'ont pas l'attribut disabled. Pour checked renvoie checked et selected.

Pour stocker dans une variable, pour un usage ultérieur, une référence vers un bouton submit.

```
// Sélectionne le(s) bouton(s) Submit
let oJQuery = $(":submit");
```

Pour sélectionner tous les <input type="text" /> et pour affecter une chaîne vide à l'attribut value de tous ces éléments :

```
// ---------------
function effacer() {
    $(":text").val("");
}
```

Pour décocher toutes les cases à cocher et tous les boutons radio :

```
$(":checkbox").attr("checked", false);
$(":radio").attr("checked", false);
```

Pour tester que toutes les zones de saisie sont remplies sur le clic d'un bouton :

```
// Tous les <input> de type text
var tInfos = $(":text");
var ok = true;

for (let i = 0; i < tInfos.length; i++) {
    let info = tInfos[i];
    if ($(info).val() === "") {
        ok = false;
    }
}

if (!ok) {
    console.log("Toutes les saisies sont obligatoires !");
    $("#lblMessage").html("Toutes les saisies sont obligatoires !");
} else {
    console.log("Jusque là tout va bien !");
    $("#lblMessage").html("Jusque là tout va bien !");
}
```

2.4.2 - Exercice (ControlesSaisies) :

ControlesSaisies_EXO.html et ControlesSaisies_EXO.js

Mettez en place ces éléments de script pour valider le formulaire « InsertUtilisateur » qui correspond à la table « Utilisateurs » ayant les colonnes suivantes : user, pwd, email, phone. Dans ce cas seule la colonne phone est facultative.

2.5 - Les sélecteurs hiérarchiques

Ce sont les sélecteurs qui permettent de parcourir l'arborescence.
Principalement les enfants, les parents, les frères, ...

Sélecteur	Description
:first	Premier élément (*)
:last	Dernier élément (*)
:first-child	Premier enfant
:last-child	Dernier enfant
:nth-child(i)	i ème enfant

Cf aussi les méthodes first(), last(), ...

2.6 - Les sélecteurs spécifiques de jQuery

Il existe des sélecteurs spécifiques à jQuery (odd, eq, lt, gt, …) qui ne trouvent pas d'équivalents en CSS.

Par exemple :

```
let oJQuery = $("tr:odd td"); // Les td dans les tr impairs
let oJQuery = $("p:eq(4)");   // Le cinquième paragraphe
let oJQuery = $("p:lt(3)");   // Les trois premiers paragraphes
```

cf http://docs.jquery.com/DOM/Traversing/Selectors#Custom_Selectors

2.7 - L'absence de getElementsByName

Il n'y a pas avec jQuery d'équivalent pour getElementsByName.

La solution la plus simple est de "classer" vos éléments et de les récupérer avec le sélecteur de classe.

```
<input type="text" name="produit[]" class="produit" value="Badoit" />
<input type="text" name="produit[]" class="produit" value="Evian" />
<input type="text" name="produit[]" class="produit" value="Vichy" />
```

```
let listeProduits = $(".produit");
```

2.8 - Les méthodes

- **Appel direct**

L'objet jQuery possède des méthodes qui permettent la manipulation du DOM, du CSS, des événements et des effets visuels.

Par exemple pour masquer les éléments d'une page, on peut utiliser la méthode hide() - qui rend la place - ou la méthode fadeOut() - qui ne rend pas la place - soit sans passer de paramètre, soit en précisant une durée en **millisecondes** ou encore à l'aide des chaînes "**slow**", "**fast**" et "**normal**" :

```
$("#élément").fadeOut();
```

- **Callback**

Certaines méthodes (comme la méthode fadeOut() qui joue sur l'opacité) acceptent une fonction en paramètre. Cette dernière sera exécutée à la fin de l'exécution de la première.

Cela permet, par exemple, de créer des animations :

```
// Enchaînement
$("img").fadeOut(5000, function(){ $("img").fadeIn(5000); });
```

2.9 - Les « textes » dans la page HTML

Déjà vu !

- **Présentation**

La manipulation des textes d'une page HTML peut être réalisée de différents façons en fonctions du type d'élément (élément ayant l'attribut value ou pas).

La méthode **val()** permet de récupérer ou d'affecter une value.

La méthode **html()** permet de récupérer ou d'affecter une valeur à un élément ne disposant pas de l'attribut value (div, p, span, ...).

La méthode text() est équivalente à la méthode html() pour un document HTML mais pour un document XML il faut utiliser la méthode text().

Ces méthodes sont des getters et des setters. La présence ou l'absence d'argument détermine leur nature. Avec un argument c'est un setter, sans argument c'est un getter. Un setter est une procédure, un getter est une fonction.

- **Syntaxes**

Méthode	Affectation (set)	Récupération (get)
val	$("#id").val("valeur");	let variable = $("#id").val();
html	$("#id").html("valeur");	let variable = $("#id").html();
text	$("#id").text("valeur");	let variable = $("#id").text();

CHAPITRE 3 - GESTION ÉVÉNEMENTIELLE

3.1 - Les bases

- **Généralités**

Permet la gestion des événements et plus précisément relie (bind) une fonction (pas le résultat d'une fonction) ou plusieurs fonctions à un événement.

- **Syntaxes**

Pour relier une seule fonction sans paramètre à un événement d'un élément identifié :

```
$(élément).on("événement", nomDeFonction);
```

Pour relier un événement sur un élément à plusieurs fonctions et/ou passer des arguments à une fonction d'un élément identifié il faut passer par une fonction anonyme.

```
$(élément).on("événement", function() {
   nomDeFonction([argument(s)]);
}
);
```

- **Exemples**

Une seule fonction est appelée et ne possède pas de paramètres : l'argument de la fonction événement est le nom de la fonction à exécuter sans parenthèses.

Au chargement du document.

```
$(document).ready(init);
```

Sur clic.

```
$("#btValider").on("click", choixListe);
```

Sur changement (Lorsque la valeur change dans un <input type="text" /> ou dans un <select>, etc).

```
$("#lbEffets").on("change", choixListe);
```

Les fonctions appelées, init() et voir(), ne possèdent pas de paramètres : l'argument de la fonction événement est obligatoirement une fonction anonyme.

```
$(document).ready(function() {
    init();
    voir();
} );
```

La fonction appelée POSSÈDE DES PARAMETRES : l'argument de la fonction événement est obligatoirement une fonction anonyme.

```
$("#btElements").on("click", function() {
    getElements("input");
});
```

En résumé :

Pas de fonction anonyme si **la** fonction liée ne possède pas d'argument.

Une fonction anonyme si :

- ✓ la fonction liée possède au moins un argument,
- ✓ plusieurs instructions doivent être exécutées,
- ✓ plusieurs fonctions, avec ou sans argument, doivent être exécutées.

3.2 - La liste des événements gérés par jQuery

Liste des événements que l'on peut "triggered".

Événement	Description
change	Changement de valeur
click	Clic
dblclick	Double clic
select	Sélection dans une zone de saisie
focus	Prise de focus
blur	Perte de focus
keydown	Touche enfoncée
keyup	Touche relâchée
keypress	Touche enfoncée
submit	Soumission de formulaire
error	Erreur

Tous ces événements peuvent avoir un argument qui fera une liaison (bind) entre l'événement et la fonction à exécuter lors du déclenchement de celui-ci.

Liste des événements qui sont nécessairement paramétrés pour être liés.

Événement	Description
ready(fonction)	Chargement du document
load(fonction)	Chargement … d'un fichier JSON par exemple
unload(fonction)	Déchargement
mousedown(fonction)	Clic
mouseup(fonction)	Relâchement du bouton de la souris
mouseover(fonction)	Lorsque le curseur de la souris entre dans le champ d'un élément
mousemove(fonction)	Lorsque le curseur de la souris se déplace dans le champ d'un élément
mouseout(fonction)	Lorsque le curseur de la souris sort du champ d'un élément
resize(fonction)	Redimensionnement d'un élément
scroll(fonction)	Défilement

3.3 - Création et destruction d'événements

Les méthodes on() et off()

http://api.jquery.com/on/
http://api.jquery.com/off/

- **Objectif**

Utiliser les méthodes on() et off().

La fonction on() permet – comme les fonctions $(élément).événement(nomDeFonction); - de lier un élément à une fonction ou à une fonction anonyme.
La différence fondamentale c'est qu'elle permet de lier un événement à un élément créé à la volée et de lier plusieurs éléments à une même fonction et même plusieurs événements.

La méthode off() permet de supprimer la liaison entre un ou plusieurs éléments et une fonction.

- **Syntaxes**

```
$("élément").on("événement"[, "sélecteur"], [data,] function(e) { … });
```

```
$("élément").off("événement" [, "sélecteur]);
```

- **Exemples**

```
$(document).on("click", "#btValider", valider);
```

```
$("#btValider").on("click", valider);
```

```
$(document).on("keydown", "#paysDeDepart", keydownPaysDepart);
```

- **Exemples**

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>jqEvenementsOnOff.html</title>
        <meta charset="utf-8" />
    </head>

    <body>
        <div id="centre">
            <h3>Ev&eacute;nements avec On et Off</h3>

            <input type="button" id="btON" value="ON Event" /><br />
            <input type="button" id="btOFF" value="OFF Event" /><br />

            <img src="../images/amira_casar_1.jpg" width="200"
alt="Confidences" />
            <img src="../images/amira_casar_2.jpg" width="200"
alt="Braquage" />

            <p id="pResultat"></p>
        </div>

        <script src="../jquery/jquery.js"></script>

        <script>
            // ------------
            function init() {
                $("#btON").on("click", ajouterEvenement);
                $("#btOFF").on("click", enleverEvenement);
            } /// init

            // ------------------------
            function ajouterEvenement() {
                $(document).on("click", "img", function(e) {
                    $("#pResultat").text(this.alt + "-" + e.pageX + "," +
e.pageY);
                });
            }

            // ------------------------
            function enleverEvenement() {
                $("#pResultat").text("");
                $(document).off("click", "img");
            }

            $(document).ready(init);
        </script>

    </body>
</html>
```

3.4 - INHIBER UN ÉVÉNEMENT

- **Objectif**

Annuler un événement.
Le plus souvent utilisé pour inhiber le comportement par défaut d'un élément HTML (<a>, bouton submit, …) et ensuite programmer un comportement spécifique.

- **preventDefault**

Utiliser la méthode preventDefault() de l'objet event pour inhiber le comportement standard d'une balise.

Dans cet exemple on inhibe le fait que le clic sur un lien permette de requêter vers l'url définie.

```
$("a").on("click", function(event) {
   // Cette methode inhibe la reaction evenementielle standard
   event.preventDefault();
});
```

- **on**

Utilisez la méthode on() et renvoyez false dans la fonction anonyme pour inhiber le comportement par défaut.

Pour un formulaire particulier :

```
$("#formSaisie").on("submit", function() {
    return false;
});
```

Pour tous les formulaires :

```
$("form").on("submit", function() {
    return false;
});
```

Exemple avec la validation conditionnelle d'un formulaire.

La réaction événementielle standard du bouton submit est inhibé.
A la place c'est le click qui est géré. Lors du clic les 2 zones de saisie doivent être remplies. Si c'est le cas le formulaire est soumis via la méthode submit() appliquée au formulaire. Si ce n'est pas le cas on affiche un message.

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>FormulaireSoumission.html</title>
</head>

<body>
    <div>
        <h1>formulaireSoumission</h1>
        <form action="../php/FormulaireSoumission.php" method="GET"
id="formSaisie">
            <label>ID : </label>
            <input type="text" name="id" id="id" value="" />
            <label>Nom : </label>
            <input type="text" name="nom" id="nom" value="Tournesol" />
            <input type="submit" id="btSubmit" />
        </form>
        <span id="spMessage"></span>
    </div>

    <script src="../jquery/jquery.js"></script>
    <script src="../js/FormulaireSoumission.js"></script>
</body>

</html>
```

FormulaireSoumission.js

```
/**
 *
 * @returns {undefined}
 */
function init() {
    $("#btSubmit").on("click", function(event) {
        // Inhibe la réaction événementielle standard
        event.preventDefault();
        valider();
    });
} /// init

/**
 *
 * @returns {undefined}
 */
function valider() {
    var lsMessage = "";
    if($("#id").val() === "") {
        lsMessage += "L'ID doit être saisi<br>"
    }
    if($("#nom").val() === "") {
        lsMessage += "Le nom doit être saisi<br>"
    }
    if($("#id").val() !== "" && $("#nom").val() !== "") {
        $("#formSaisie").submit();
    }

    $("#spMessage").html(lsMessage);
} /// valider

$(document).ready(init);
```

3.5 - Délégation d'événement

- **Objectif**

Déléguer un événement c'est simuler un événement.
La fonction trigger() permet de déléguer un événement.

- **Syntaxe**

```
$("élément").trigger("événement");
```

- **Exemple**

```
// ------------
function init() {
     // Quand on clique sur le bouton btCacher on exécute la fonction
cacher()
     $("#btCacher").click(cacher);
     // Quand on clique sur le bouton btTriggerEvent on Exécute la
fonction triggerEvent() qui elle-même va Exécuter le click du bouton
btCacher
     $("#btTriggerEvent").click(triggerEvent);
}

// --------------
function cacher() {
     $("img").hide(3000);
}

// --------------------
function triggerEvent() {
     // Cette fonction "simule" un clic sur le bouton btCacher
     $("#btCacher").trigger("click");
}
```

Ceci peut être utile si deux actions utilisateur doivent déclencher la même action (par un clic sur un bouton et un double-clic sur une liste) et que cette action exécute plusieurs actions élémentaires et que l'action sur l'événement du premier élément est une fonction anonyme.

Note :

$("#btCacher").trigger("click");
est équivalent à
$("#btCacher").click();

CHAPITRE 4 – GESTION DU CSS

4.1 - css()

Rappel : le lien CSS dans le <head>.

```
<link rel="stylesheet" type="text/css" href="../css/style.css" />
```

La fonction css() permet de récupérer ou de modifier la valeur d'une propriété CSS.

- **Récupérer une valeur de style d'un élément HTML**

```
Chaîne = $("élément").css("propriété");
```

```
// Récupère la taille de la marge du haut
let margeTop = $("#img1").css("marginTop"); // ou "margin-top"
```

Note : jQuery accepte la valeur de "l'attribut" façon CSS (mots séparés par des tirets) ou façon JavaScript (Camélisée).

- **Modifier une valeur de style d'un élément html**

```
$("élément").css("propriété", "valeur");
```

```
// Modifie la taille de la marge interne
$("#img1").css("padding", "20px");
```

- **Modifier un ensemble de valeurs de style d'un élément HTML**

```
$("élément").css( { propriété:"valeur", propriété:"valeur" } );
```

```
// Modifie la taille de la marge interne et la bordure
$("#img1").css( { padding:"20px", border:"1px solid dimgray" } );
```

Note : notez que le nom de la propriété CSS peut être entourée de double quotes ou pas parce que c'est ici un tableau JSON.

cf aussi, pour plus de détails, la documentation officielle pour les méthodes suivantes :

Méthode	Description
hasClass("nomDeClasse")	Renvoie un booléen
addClass("nomDeClasse")	Ajoute l'attribut class avec une valeur à un élément ou à plusieurs et modifie ainsi le style et le rendu à la volée
removeClass("nomDeClasse")	Supprime l'attribut class avec la valeur passée en argument d'un élément ou de plusieurs et modifie ainsi le style et le rendu à la volée

```
// Teste le fait qu'un element est "classé"
console.log($("#lblMessage).hasClass("rouge"));
```

```
// "Detache" un element d'une classe
// L'element X de couleur Y devient noir
$("#pRouge").removeClass("rouge");
```

```
// "Attache" un element a une classe
// L'element X de couleur Y devient rouge
$("#pRouge").addClass("rouge");
```

4.2 - Positionnement

Récupérer la position d'un élément.

Les fonctions position() et offset() renvoient la position et le déplacement sous forme d'un objet (un point de coordonnées x et y).

position() renvoie la position absolue.
offset() renvoie le décalage relatif.
Les propriétés left et top de cet objet contiennent les positions x et y de l'objet.

```
let position = $("#img1").position();
console.log("x : " + position.left + " - y : " + position.top);
```

Positionner un élément.

Pour positionner un élément il faut utiliser la fonction css().

Par exemple :

```
$("#img1").css("left", "300px");
```

ou

```
$("#img1").css({left: "500px"});
```

Note : **il faut que l'élément soit en position:absolute ou en position:relative**.
Dans le premier cas cela déplacera l'élément **à** la position passée en paramètre et dans le deuxième cas cela déplacera l'élément **de** la position (de l'offset) passée en paramètre.

4.3 - DIMENSIONNEMENT

Ces fonctions permettent de gérer la largeur et la hauteur d'un élément.

- **Récupérer la largeur d'un élément**

```
let largeur = $("#élément").width();
```

- **Récupérer la hauteur d'un élément**

```
let hauteur = $("#élément").height();
```

- **Modifier la largeur d'un élément**

```
$("#élément").width(valeur);
```

```
$("#img1").width(200);
```

- **Modifier la hauteur d'un élément**

```
$("#élément").height(valeur);
```

```
$("#img1").height(100);
```

cf aussi les fonctions innerHeight(), outerHeight(), innerWidth (), outerWidth().
Les fonctions outer ajoutent le padding et le border.

4.4 - Exemple : modifications des propriétés de style

Le script modifie le padding et le border.

Cf jqCss.html

```
<style type="text/css">
#img1{ margin:5px; padding:5px; border:5px solid #000000; }
</style>
```

```
<div id="centre">
    <h3>Le CSS</h3>
    <input type="button" id="btModifier" value="Modifier" /><br>

    <img id="img1" src="../images/1.jpg" height="150" alt="3a" />
</div>
```

```
// ----------------
function modifier() {
     //$("#img1").height(100);
     //$("#img1").css("padding","20px");
     $("#img1").css( { padding:"20px", border:"1px solid dimgray" } );
}
```

CHAPITRE 5 – GESTION DES EFFETS ET DES ANIMATIONS

5.1 - Tableau résumé des effets

Les méthodes pour créer des effets.

Méthode	Description
Objet.show([vitesse [, callback]])	Montre un élément
Objet.hide()	Masque un élément
Objet.toggle()	"Levier bascule" des précédents
Objet.slideDown()	Affiche du haut vers le bas
Objet.slideUp()	Masque du bas vers le haut
Objet.slideToogle()	"Levier bascule" des précédents
Objet.fadeIn()	Fait apparaître en fondu
Objet.fadeOut()	Masque avec un fondu
Objet.fateTo(vitesse, opacité [, callback])	Fondu jusqu'à opacité (de 0 à 1)
Objet.animate()	Permet de créer une animation personnalisée
Objet.stop()	Arrête toutes les animations

Notes : Double effets

```
// Disparaît puis réapparaît
$("#photo").fadeOut(3000).fadeIn(2000);
```

5.2 - Les différentes méthodes

5.2.1 - La méthode show()

- **Fonctionnalité**

Affiche un élément.

- **Syntaxe**

```
$("selecteur").show([vitesse [, callback]]);
```

Vitesse : "slow" (600 millisecondes), "normal" (400 millisecondes), "fast" (200 millisecondes) ou en millisecondes.

- **Exemples**

```
$("#photo").show();
```

```
$("#photo").show("slow");
```

```
$("#photo").show(5000);
```

```
<!DOCTYPE html>
<!--
ShowPlus.html
-->
<html>
    <head>
        <title>ShowPlus</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <style>
            body{background: black; color: white;}
            #photo{display:none; z-index:10;}
        </style>
    </head>

    <body>
        <h1>ShowPlus</h1>
        <img id="photo" src="../images/amira_casar_2.jpg" alt="Photo
d'accueil" title="Photo d'accueil" />
        <br>
        <label id="lblMessage"></label>

        <script src="../jquery/jquery.js"></script>
        <script>
            function fin() {
                $("#lblMessage").html("Animation terminée !!!");
            }
            $("#photo").fadeIn(5000, fin);
        </script>

    </body>
</html>
```

5.2.2 - La méthode hide()

- **Fonctionnalité**

Masque un élément. La place est restituée.

- **Syntaxe**

```
$("selecteur").hide([vitesse [, callback]]);
```

- **Exemple**

```
$("#photo").hide();
```

5.2.3 - La méthode toggle()

- **Fonctionnalité**

Inverse les précédents.

- **Syntaxe**

```
$("selecteur").toggle();
```

- **Exemple**

```
$("#photo").toggle();
```

5.2.4 - La méthode slideDown()

- **Fonctionnalité**

Affiche du haut vers le bas.
Ne fonctionne pas sur des balises <img>. Donc mettre une image dans un conteneur identifié (<p>, , etc).

- **Syntaxe**

```
$("selecteur").slideDown(["vitesse" | durée en millisecondes [, callback]]);
```

- **Exemple**

```
$("#pPhoto").slideDown(5000); // Affiche du haut vers le bas
```

Note : la photo ne doit être affichée.

5.2.5 - La méthode slideUp()

- **Fonctionnalité**

Affiche du bas vers le haut.

- **Syntaxe**

```
$("selecteur").slideUp(["vitesse" | durée en millisecondes [, callback]]);
```

- **Exemple**

```
$("#pPhoto").slideUp(5000); // Masque du bas vers le haut
```

5.2.6 - La méthode slideToggle()

- **Fonctionnalité**

Inverser les précédents.

- **Syntaxe**

```
$("selecteur").slideToggle(["vitesse" | durée en millisecondes [, callback]]);
```

- **Exemple**

```
$("#pPhoto").slideToggle(); // Inverse l'etat
```

5.2.7 - La méthode fadeIn()

- **Fonctionnalité**

Affiche par fondu.

- **Syntaxe**

```
$("selecteur").fadeIn(["vitesse" | durée en millisecondes [, callback]]);
```

- **Exemple**

```
$("#photo").fadeIn(3000); // Affiche
```

5.2.8 - La méthode fadeOut()

- **Fonctionnalité**

Masque par fondu.

- **Syntaxe**

```
$("selecteur").fadeOut(["vitesse" | durée en millisecondes [, callback]]);
```

- **Exemple**

```
$("#photo").fadeOut(3000); // Masque
```

5.2.9 - La méthode fadeTo()

- **Fonctionnalité**

Crée un effet de fondu jusqu'à une certaine opacité.

- **Syntaxe**

```
$("selecteur").fadeTo(["vitesse" | durée en millisecondes [, opacité[, callback]]]);
```

Opacité entre 0 et 1.
Définition de l'opacité : propriété qu'ont certains corps de s'opposer au passage de la lumière. Avec une opacité de 0 (donc aucune) la lumière passe et la photo devient invisible.

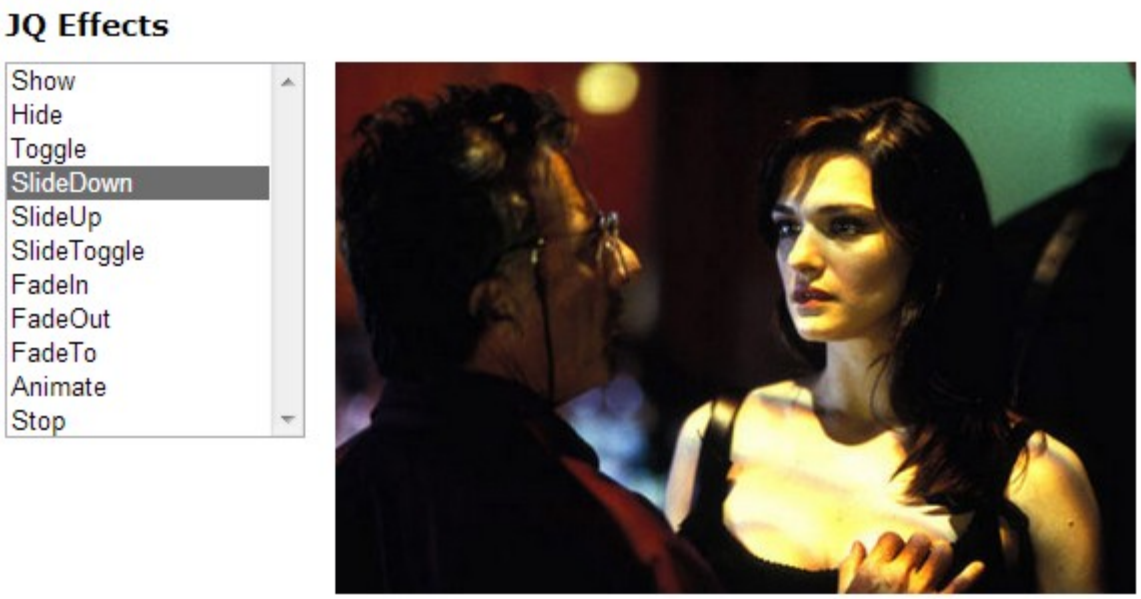
- **Exemple**

```
$("#photo").fadeTo(3000, 0.3);
```

5.2.10 - Exemple récapitulatif

- **Objectif**

Mettre en place les différents effets sur une image.



Remarques :

Nous allons utiliser la méthode [] : $("#element")["fonction"]();

Pour cela il faut que les values des options soient camélisées et que l'image soit dimensionnée pour slideUp et slideDown.
La méthode fadeTo() est un cas particulier puisqu'elle nécessite un argument.

- **Le script complet**

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <style>
       .flottantGauche{float: left; padding: 10px;}
       .nettoyeur{clear: both;}
    </style>
    <title>jqEffets.html</title>
</head>

<body>
    <div id="centre">

        <h3>JQ Animations</h3>

        <p class="flottantGauche">
            <select name="lbEffets" id="lbEffets" size="9">
                <option value="show">Show</option>
                <option value="hide">Hide</option>
                <option value="toggle">Toggle</option>
                <option value="slideDown">SlideDown</option>
                <option value="slideUp">SlideUp</option>
                <option value="slideToggle">SlideToggle</option>
                <option value="fadeIn">FadeIn</option>
                <option value="fadeOut">FadeOut</option>
                <option value="fadeTo">FadeTo</option>
            </select>
        </p>

        <p class="flottantGauche" id="pImage">
            <img id="photo" src="../images/1.jpg" alt="Photo du film
Confidences" width="400" />
        </p>

        <p class='nettoyeur'>
            <label id='lblMessage'></label>
        </p>
    </div>

    <script src="../jquery/jquery.js"></script>

    <script>
          // ------------
          function init() {
                $("#lbEffets").on("change", choixListe);
          }
          // ------------------
          function choixListe() {
                let lsEffet = $("#lbEffets").val();
                $("#lblMessage").html(lsEffet);

                if (lsEffet === "fadeTo") {
                      $("#photo").fadeTo(3000, 0.5);
                }
                else {
                      $("#photo")[lsEffet]();
                }
```

```
            }

            $(document).ready(init);
        </script>

</body>
</html>
```

5.2.11 - La méthode animate()

Permet de faire des animations, ie un ensemble d'effets (Déplacement, dimensionnement, fondu, …).

Syntaxe

```
$(élément).animate({ option(s) }[, durée[, easing[, complete]]])
```

easing (la "courbe" du mouvement de l'animation) : swing, linear, ...
complete (fonction à exécuter à la suite) : fonction ou fonction anonyme à exécuter.
Pour le positionnement les références sont seulement top et left (pas de bottom ni de right).

Exemples

Les éléments 1 et 3 sont en positionnement relatif et l'élément 2 est en positionnement absolu.

1 - Déplacer l'élément identifié par [texte] à 200px du haut du document.

```
$("#texte").animate({top: 200}, "slow");
```

2 - Agrandir et déplacer un texte sur un fond de couleur.

```
$("#texteCouleur").animate({left:400, width:"30%", height:"30px"}, 2000);
```

3- Déplacer une image de 200 px vers la droite en la rendant légèrement opaque, le tout en l'espace d'une seconde.

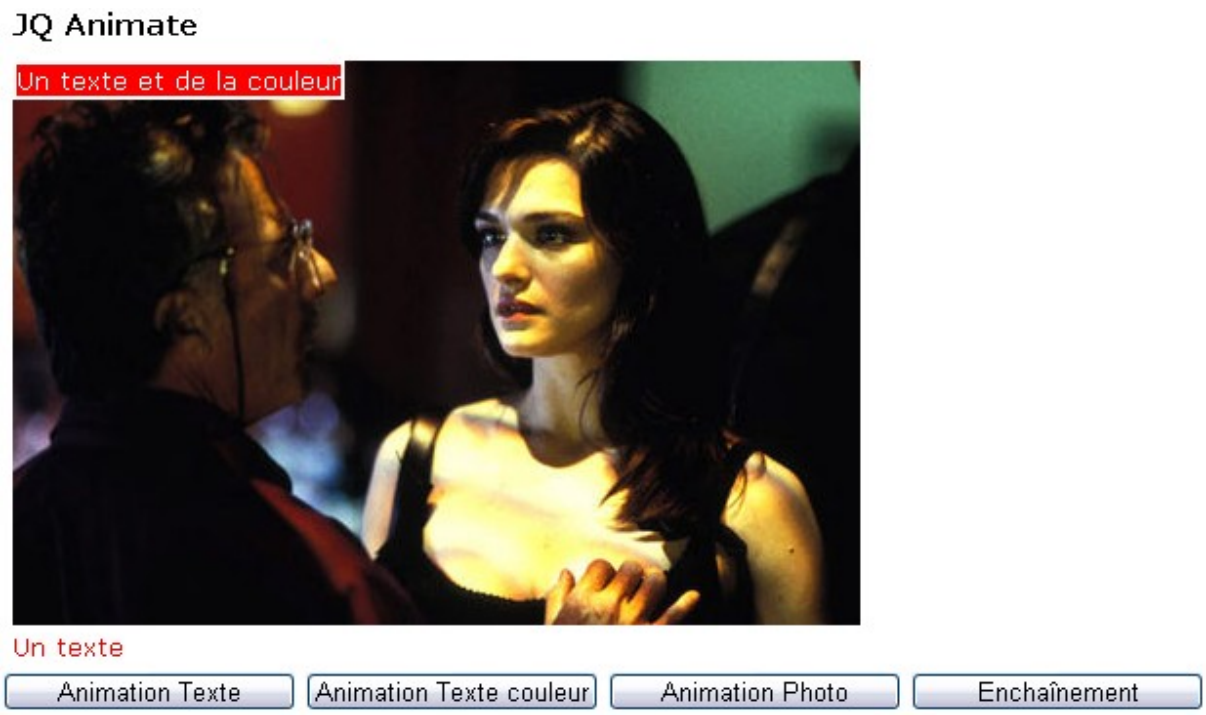
```
$("#photo").animate({left:200, opacity:0.5}, 1000);
```

Pour enchaîner :

$("#photo").animate({left:100, opacity:0.5}, 3000, "swing", animateTexte);

Exemple complet

Avant



Après

Le script complet

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>jqAnimate</title>

    <style type="text/css">
        #photo{position:relative;}
        #texte{position:relative; color:red;}
        #texteCouleur{position:absolute; background-color:red;
color:white; border:2px white solid; z-index:9}
    </style>

    <script src="../jquery/jquery.js"></script>
    <script>
        // ------------
        function init() {
            $("#btTexte").click(animateTexte);
            $("#btTexteCouleur").click(animateTexteCouleur);
            $("#btPhoto").click(animatePhoto);
            $("#btEnchainement").click(enchainement);
        }
        // --------------------
        function animateTexte() {
            $("#texte").animate({top:-200}, 1000);
        }
        // --------------------
        function animateTexteCouleur() {
            $("#texteCouleur").animate({left:400, width:"30%",
height:"30px"}, 2000);
        }
        // --------------------
        function animatePhoto() {
            $("#photo").animate({left:100, opacity:0.5}, 1000);
        }
        // --------------------
        function enchainement() {
            $("#photo").animate({left:100, opacity:0.5}, 3000, "swing",
animateTexte);
        }

        // Démarrage façon JQuery
        $(document).ready(init);
    </script>
</head>

<body>
    <div id="centre">
        <h3>JQ Animate</h3>
        <p>
            <label id="texteCouleur">Un texte et de la couleur</label>
            <img id="photo" src="../images/3a.jpg" alt="Photo 3a du film
Confidences" />
            <br>
            <label id="texte">Un texte</label>
        </p>
        <input name="btTexte" id="btTexte" type="button" value="Animation
Texte" />
```

```
        <input name="btTexteCouleur" id="btTexteCouleur" type="button"
value="Animation Texte couleur" />
        <input name="btPhoto" id="btPhoto" type="button" value="Animation
Photo" />
        <input name="btEnchainement" id="btEnchainement" type="button"
value="Enchaînement" />
    </div>
</body>
</html>
```

Enchaînement multi-niveaux

```
<!DOCTYPE html>
<!--
Animate3Niveaux.html
-->
<html>
    <head>
        <title>Animate3Niveaux.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <style>
            #photo{position: absolute; top: 200; left:0; opacity: 0; }
        </style>
    </head>

    <body>
        <div>
            <h3>Animate3Niveaux.html</h3>
            <img id="photo" src="../images/amira_casar_2.jpg" alt="Photo
d'accueil" title="Photo d'accueil" />
            <label id="lblMessage"></label>
        </div>

        <script src="../jquery/jquery.js"></script>
        <script>
            /**
             *
             * @returns {undefined}
             */
            function deplacerAGauche() {
                $("#photo").animate({left: 200}, 3000);
            } /// deplacerAGauche

            /**
             *
             * @returns {undefined}
             */
            function deplacerADroite() {
                $("#photo").animate({left: 500}, 3000, deplacerAGauche);
            } /// deplacerADroite

            /**
             *
             * @returns {undefined}
             */
            function faireApparaitre() {
                $("#photo").animate({opacity: 1}, 3000);
            } /// faireApparaitre

            /**
             *
             * @returns {undefined}
             */
            function init() {
                // Version 1
                faireApparaitre();
                deplacerADroite();
                deplacerAGauche();
                // Version 2
```

```
                //$("#photo").animate({opacity: 1}, 3000,
deplacerADroite);
                // Version 3
                //$("#photo").animate({opacity: 1}, 3000).animate({left:
500}, 3000).animate({left: 200}, 3000);
            } /// init

            $(document).ready(init);
        </script>

    </body>
</html>
```

5.2.12 - La méthode stop()

Arrête toutes les animations.

Syntaxe

```
$(élément).stop()
```

CHAPITRE 6 – GESTION DU DOM

6.1 - Présentation

Le DOM (Document Object Model) représente un document XML ou HTML sous forme d'arborescence.

Les actions sur le DOM, donc sur les éléments du document, s'apparentent au CRUD.
Je rappelle que JavaScript, et donc jQuery, permet de modifier dynamiquement le DOM, donc dynamiquement le code HTML dans le navigateur et donc le rendu pour l'utilisateur.

Il est donc possible de parcourir l'arbre et de récupérer un ou plusieurs éléments et leur valeur.
Il est aussi possible de créer un nouvel élément, de supprimer un élément et de modifier un élément.

http://openclassrooms.com/courses/simplifiez-vos-developpements-javascript-avec-jquery/selection-d-elements-3

et surtout

http://openclassrooms.com/courses/simplifiez-vos-developpements-javascript-avec-jquery/plus-loin-dans-la-selection-d-elements

Rappel important :

A partir du moment où vous utilisez jQuery vous ne devez plus gérer le DOM avec JavaScript (donc plus de document.getElementById(), …, de window.onload, …).

6.1.1 - Sélectionner un élément

Note : déjà vu avec les sélecteurs.

- **Fonctionnalité**

Récupère un objet jQuery.

Note : c'est l'équivalent de document.getElementById("id") en JavaScript.

- **Syntaxe**

```
let oJQ = $("#id");
```

- **Exemple**

```
let lblMessage = $("#lblMessage");
```

6.2 - Récupérer une valeur saisie

Déjà vu !

L'objectif est d'afficher dans un label la valeur saisie dans une zone de saisie.
La méthode val() permet de récupérer ou d'affecter l'attribut value d'un élément de formulaire qui possède l'attribut value. C'est un getter/setter. Sans paramètre c'est un getter, avec un paramètre c'est un setter.
La méthode html() permet de récupérer ou d'affecter le texte d'un élément HTML. C'est un getter/setter. Sans paramètre c'est un getter, avec un paramètre c'est un setter.


```
<!DOCTYPE html>
<!--
Afficher une valeur saisie
-->
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>GetValeur.html</title>
    </head>

    <body>
        <article id="article">
            <h1>GetValeur</h1>
            <label for="nom">Nom : </label>
            <input type="text" id="nom" value="Dupont" />
            <input type="button" value="Valider" id="btValider" />

            <label id="lblMessage"></label>
        </article>


        <script src="../jquery/jquery.js"></script>
        <script>
            $("#btValider").on("click", recupererSaisie);

            function recupererSaisie() {
                $("#lblMessage").html($("#nom").val());
            }
        </script>

    </body>
</html>
```

6.3 - Activer/Désactiver un élément

L'activation ou la désactivation dynamique d'un élément est réalisé via l'attribut "disabled" et une valeur booléenne ou la String "disabled".

Syntaxes :

```
$("#element").attr("disabled", booléen);
```

```
$("#element").prop("disabled", booléen);
```

et aussi ...

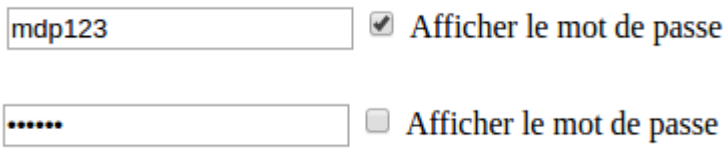
```
$("#element").attr("disabled", "disabled");
```

Note : aucune des 2 méthodes ne fonctionne sur une image.

6.4 - Afficher/Masquer un mot de passe

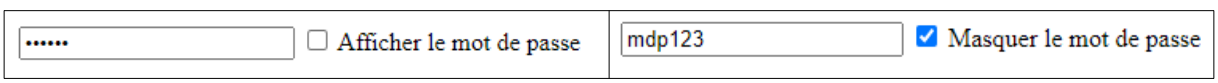
6.4.1 - Exemple

Lorsque l'utilisateur clique sur la case à cocher le mot de passe est affiché ou masqué.




```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>MotDePasseVisible.html</title>
    </head>

    <body>
        <article id="article">
            <input type="password" id="mdp" value="mdp123" />
            <input type="checkbox" id="chkAfficherMasquer" />
            Afficher le mot de passe
        </article>

        <script src="../jquery/jquery.js"></script>
        <script>
            $("#chkAfficherMasquer").on("click", afficherMasquer);

            function afficherMasquer() {
                let coche = $("#chkAfficherMasquer").prop("checked");
                console.log(coche);
                if (coche) {
                    $("#mdp").attr("type", "text");
                }
                else {
                    $("#mdp").attr("type", "password");
                }
            }
        </script>
    </body>
</html>
```

6.4.2 - Exercice : changer aussi le texte



6.5 - Récupérer le parent, les enfants, ...

Une méthode

.parent()

Un attribut

.parentNode

Un attribut

childNodes

Exemple :

```
function updateData(e) {

    let id = e.currentTarget.id;

    // Le parent : la TD
    let parent = e.currentTarget.parentNode;

    // Le grand parent : la TR
    let grandParent = parent.parentNode;

    // Les inputs et leur valeur
    let enfants = grandParent.childNodes;

    let nomFormule = enfants[0].childNodes[0].value; // val()
```

6.6 - Sélectionner un ou plusieurs éléments

6.6.1 - Sélectionner un élément

Note : déjà vu avec les sélecteurs.

- **Fonctionnalité**

Récupère un objet jQuery.

Note : c'est l'équivalent de document.getElementById("id") en JavaScript.

- **Syntaxe**

```
let oJQ = $("#id");
```

- **Exemple**

```
let lblMessage = $("#lblMessage");
```

6.6.2 - Sélectionner plusieurs éléments

Note : déjà vu avec les sélecteurs.

- **Fonctionnalité**

Récupère une liste d'objets jQuery.

Note : l'équivalent de document.getElementsByTagName("balise") en JavaScript.

- **Syntaxe**

```
let toJQ = $("balise");
```

- **Exemple : les paragraphes du document**

```
let tpJQ = $("p");
```

- **Exemple : les <tr> du body d'une <table>**

```
let tTR = $("#tBody tr");
```

Note : le body est identifié par tBody.

6.6.3 - Récupérer la valeur d'un attribut d'un élément

Note : déjà vu avec les sélecteurs.

- **Fonctionnalité**

Récupère la valeur d'un attribut d'un élément.

- **Syntaxe**

```
let valeur = $("id").attr("nom d'attribut");
```

- **Exemple : récupère les valeurs de certains attributs d'un <input type="text" />**

```
// Recupere la valeur d'un attribut
let id   = $("#itNom").attr("id");
let type = $("#itNom").attr("type");
```

- **Syntaxe pour la value**

Note : déjà vu avec les sélecteurs.

```
let valeur = $("id").val();
```

- **Exemple : récupère la valeur d'un <input type="text" />**

```
let valeur = $("#itNom").val());
```

- **Syntaxe pour l'innerHTML**

Note : déjà vu avec les sélecteurs.

Si c'est un élément HTML sans value (div, p, span, label, …).

```
let valeur = $("id").html();
```

ou

```
let valeur = $("id").text();
```

- **Exemple : récupère le texte d'un <label>**

```
// Recupere un innerHTML
let valeurInnerHTML = $("#lblTest").html();
```

Notes :
De façon générale .html() récupère le texte de l'élément transmis en paramètre ($("#pResultat").html()).
De façon générale .text() peut récupérer le texte de tous les éléments transmis en paramètre ($("p").text()).
text() permettra aussi de récupérer des nœuds #text de fichiers XML.

6.6.4 - Affecter une valeur à un attribut d'un élément ou à un texte

Note : déjà vu avec les sélecteurs.

- **Syntaxe pour la value**

```
$(objet).val("texte");
```

- **Exemple : affecte une valeur à un <input type="text" />**

```
$("#itElement").val("Nouvelle valeur pour l'IT");
```

- **Syntaxe pour n'importe quel autre attribut**

```
$(objet).attr("attribut", "Nouvelle valeur");
```

- **Exemple : modifie la taille d'un <input type="text" />**

```
// Affecte une valeur a un attribut d'un IT
$("#itElement").attr("size", "50");
```

- **Syntaxe pour l'innerHTML**

Si c'est un élément HTML sans value (div, p, span, label, …).

```
$("#id").html("Nouvelle valeur");

ou

$("#id").text("Nouvelle valeur");
```

Note : mais text("Message
") affichera ce texte donc il n'y aura de RC dans la page.

- **Exemple : affecte un texte à un **

```
// Affecte une valeur innerHTML a un span
$("#spTest").html("Nouvelle valeur pour le span");
```

Note : la méthode **prepend("valeur")** permet d'ajouter une valeur en début de texte.

6.6.5 - Le cas de l'attribut classe

- **Affectation**

```
$(element).attr("class", "nomDeClasse")
```

- **Ajout d'une nouvelle classe**

```
$(element).addClass = "nomDeClasse"
```

- **Récupération**

```
let nomDeClasse = $(element).attr("class")
```

6.6.6 - Travailler avec l'élément <select>

Savoir si un élément est sélectionné.

```
var selectedIndex = $("#nomDeLaListe").prop('selectedIndex');
```

Si la valeur renvoyée est -1 alors aucun élément dans la liste n'est sélectionné.

```
if ($("#liste option:lt(0)")) {
```

Sélectionner une option dans un élément <select> via un index

Ceci permet de sélectionner visuellement un item dans une liste.

```
$("#liste").prop('selectedIndex', indice);
```

```
$("#liste option:eq(indice)").prop("selected", true);
```

Note : l'indice commence à 0.

et pour désélectionner tous les items :

```
$("#liste").prop('selectedIndex', -1);
```

Sélectionner plusieurs options dans un <select>

Même syntaxe ...

Sélectionner une option dans un élément <select> via une valeur (value de l'<option>)

```
$("#liste").val("valeur");
```

Récupérer la valeur de l'option sélectionnée dans une liste en mono-sélection

```
$("#liste").val();
```

```
// UNIQUE
let lsSelection = "";
lsSelection = $("#lbPays").val();
$("#lblMessageUnique").html(lsSelection);
```

Récupérer l'étiquette de l'option sélectionnée dans une liste en mono-sélection

```
$("#liste option:selected").text()
```

```
// UNIQUE
console.log($("#lbPays option:selected").text());
```

Récupérer les valeurs des options sélectionnées dans une liste en multi-sélections

cf la syntaxe détaillée du each au paragraphe 6.7.

```
// MULTIPLE
var lsSelections = "";
$("select[id='lbVilles'] option:selected").each(function() {
    lsSelections += $(this).val() + '|';
} ) ;
$("#lblMessageMultiple").html(lsSelections);
```

Récupérer les étiquettes des options sélectionnées dans une liste en multi-sélections

```
// MULTIPLE
var lsSelections = "";
$("select[id='lbHobbies'] option:selected").each(function() {
    lsSelections += $(this).text() + '|';
} ) ;
$("#lblMessageMultiple").html(lsSelections);
```

6.6.7 - Travailler avec une case à cocher

La case à cocher est-elle cochée ? Renvoie true ou false.

```
let booleen = $("#caseACocher").prop("checked")
```

Coche la case à cocher.

```
$("#caseACocher").prop("checked", true);
```

6.6.8 - Travailler avec un bouton radio

Le bouton radioi est-il coché ? Renvoie true ou false.

```
let booleen = $("#boutonRadio").prop("checked")
```

Sélectionne le bouton radio.

```
$("#boutonRadio").prop("checked", true);
```

Note : à la place de prop() il existe aussi attr() qui fonctionne très imparfaitement.

6.7 – Ajouts

6.7.1 - Ajouter un élément

- **Syntaxe**

```
let objet = $("#id").append("élément à ajouter [+ valeur]");
```

append() ajoute un élément à la fin d'un élément parent.

Cf aussi **prepend()** qui ajoute un élément au début d'un autre.

- **Exemples**

```
// Ajoute une option à une liste avec une valeur
$("#lbVilles").append("<option value='75011'>Paris 11</option>");
```

ou ceci et c'est nettement mieux

```
let opt = $("<option>");
opt.val("75011");
opt.html("Paris 11");
$("#lbVilles").append(opt);
```

cf aussi $(objet).attr("attribut", "Nouvelle valeur");

Ajouter un élément au body :

```
$(document.body).append("<select>");
```

Et ajouter un élément dans le <head> :

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>appendHead.html</title>
        <meta charset="UTF-8">
        <!-- Vos CSS -->
        <link rel="stylesheet" type="text/css" href="../css/style.css" />
    </head>

    <body>
        <!-- Votre code HTML-->
        <h3>Ajout dans le Head</h3>
        <input type="button" value="Ajouter CSS" name="btAjouterCSS"
id="btAjouterCSS" />
        <br>Eh oui !
        <!-- Chargement de la bibliotheque jQuery -->
        <script src="../jquery/jquery.js"></script>

        <!-- Vos scripts internes -->
        <script>
            // ------------
            function init() {
                $("#btAjouterCSS").click(function() {
                    $(document.head).append("<link rel='stylesheet'
type='text/css' href='../css/sup.css' />");
                });
            } /// init

            // Au chargement
            $(document).ready(init);
        </script>
    </body>
</html>
```

6.7.2 - Ajouter une série d'éléments

Objectif : à partir d'un tableau ajouter des éléments dans une liste


```
<!DOCTYPE html>
<!--
AjoutsDansListes.html
-->
<html>
    <head>
        <title>AjoutsDansListes.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>
            <h3>Ajouts Dans Listes</h3>

            <div id="divListeDeroulante">
                Liste déroulante<br>
            </div>

            <div id="divListe">
                Liste<br>
            </div>

        </div>
        <script src="../jquery/jquery.js"></script>
        <script src="AjoutsDansListes.js"></script>
    </body>
</html>
```

```
/*
 * AjoutsDansListes.js
 */

let tItems = new Array();
tItems["Accueil"] = "Accueil.html";
tItems["Inscription"] = "Inscription.html";
tItems["Authentification"] = "Authentification.html";
tItems["Catalogue"] = "Catalogue.html";
tItems["Panier"] = "Panier.html";
tItems["Boutiques"] = "Boutiques.html";
tItems["S&eacute;lections"] = "Selections.html";

var ul;
var li;
var a;
ul = $("<ul>");
for (let cle in tItems) {
    li = $("<li>");
    a = $("<a>");
    a.html(cle);
    a.attr("href", tItems[cle]);
    li.append(a);
    ul.append(li);
}
$("#divListe").append(ul);

select = $("<select>");
for (let cle in tItems) {
    let option = $("<option>");
    option.val(cle);
    option.html(tItems[cle]);
    select.append(option);
}
$("#divListeDeroulante").append(select);
```

6.7.3 - Insérer un élément

Il existe deux méthodes pour insérer un élément; une pour l'insérer avant un autre élément et une autre pour l'insérer après un autre.

- **Syntaxes**

```
$("#id").before("élément à ajouter [+ valeur]");
```

```
$("#id").after("élément à ajouter [+ valeur]");
```

- **Exemple : insérer une image avant une balise <select>**

```
$("#lbVilles").before("<img src='/images/cinema/0.jpg' alt='Photo' />");
```

Une autre idée pratique : insérer une ligne de commande avant un bouton submit.

Note : il existe aussi les méthodes insertBefore() et insertAfter() qui produisent les mêmes résultats.
La syntaxe est "inversée" : $("élément à ajouter").insertAfter("cible");

6.7.4 - Ajouter un élément qui en enveloppe un autre (Wrap)

- **Syntaxe**

```
$("élément enveloppé").wrap("Code HTML enveloppant");
```

- **Exemple**

Le bouton identifié par btContenu existe dans le <body>. Le paragraphe sera ajouté dynamiquement et enveloppera le bouton.

```
$("#btContenu").wrap("<p class='enveloppe'></p>");
```

Cf aussi wrapAll() : enveloppe tous les éléments mentionnés par une seule enveloppe, wrapInner() : enveloppe les éléments enfants.

- **Exemple d'ajouts**


```
<script src="../jquery/jquery.js"></script>
<script>
	// ---------------
	function ajouter() {
		// Ajoute un element avec la valeur de l'IT
		$("#lbVilles").append("<option value='" + $("#itElement").val()
+ "'>" + $("#itElement").val() + "</option>");
	}

	$(document).ready(function() {
		$("#btAjouter").click(ajouter);
	});
</script>
```

```
<div id="centre">
    <h3>JQ DOM Ajout</h3>

    <select id="lbVilles" size="3"></select><br>

    <label>Ajouter?</label>
    <input id="itElement" type="text" value="Paris 1" /><br>
    <input id="btAjouter" type="button" value="Ajouter" />
</div>
```

6.7.5 - Ajouter un attribut

Ajouter un attribut, cela n'existe pas.
Il suffit d'utiliser attr('attribut' ,'valeur') pour modifier la valeur ou la valeur par défaut.

6.7.6 - Exercice : créer dynamiquement des éléments de formulaire

Créer un formulaire dans lequel on puisse ajouter des lignes de saisie.


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

    <body>
        <div>
            <form action="" method="GET" id="formProduits">
                <label>Produit</label><br>
                <input type="text" name="produit1" id="produit1"
value="Badoit" />
                <br>
                <input type="button" value="+Ligne" id="btPlusLigne" />
                <br>
                <input type="button" id="btSubmit" name="btSubmit" />
            </form>
        </div>
    </body>
</html>
```

Autre idée (mais il faut attendre un peu) : Drag & Drop et visualisation des éléments du panier sous forme d'image.

6.8 - Suppressions

6.8.1 - Supprimer un élément

- **Syntaxe**

```
$("#id").remove();
```

- **Exemple : supprime une liste**

```
$("#lbVilles").remove();
```

6.8.2 - Supprimer tous les éléments enfants

- **Syntaxe**

```
$("#id").empty();
```

- **Exemple : vide une liste**

```
$("#lbVilles").empty();
```

6.8.3 - Supprimer un élément enfant

- **Syntaxes**

```
$("balise").remove("expression");
```

```
$("#id").remove("expression");
```

- **Exemples**

```
$("p").remove(".paire"); // Supprime les paragraphes de classe .paire.
```

```
$("#lbVilles option").remove(".paire"); // Supprime les options paires
```

```
// Supprime les éléments sélectionnés
$("#lbVilles :selected").remove();
```

ou

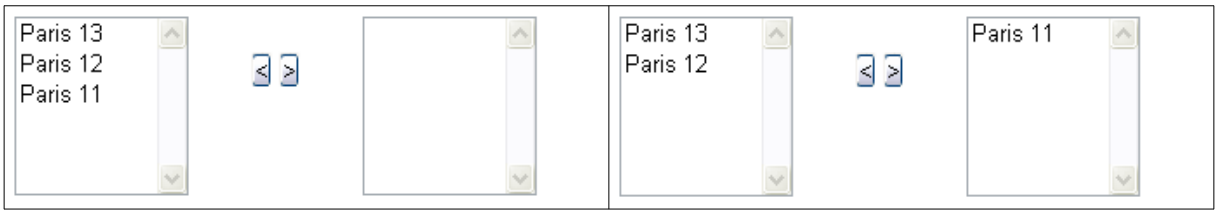
```
// Supprime les éléments (les options) sélectionnés
$("#lbVilles option").remove(":selected ");
```

```
$("#lbVilles option[value='75021']").remove();
```

6.8.4 - Supprimer un attribut

6.8.5 - Exercice : transférer des items d'une liste vers une autre

Transférer des items d'une liste vers une autre.




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

    <style type="text/css">
       *{margin:0; padding:0;}
       div{ width: 100px; height: 100px; text-align: center;}
       select{width: 100px; height: 100px;}
       #gauche, #milieu, #droite{float:left;}
    </style>
  </head>

  <body>
        <div id="gauche">
            <select id='lbVilles' name='lbVilles' size='5'>
                <option value='75011'>Paris 11</option>
                <option value='75012'>Paris 12</option>
                <option value='75013'>Paris 13</option>
            </select>
        </div>
        <div id="milieu">
            <br>
            <input type="button" value="<" id="btMoins" name="btMoins" />
            <input type="button" value=">" id="btPlus"  name="btPlus" />
        </div>
        <div id="droite">
            <select id='lbVillesChoisies' name='lbVillesChoisies'
size='5'></select>
        </div>
  </body>
</html>
```

Autre idée d'exercice : reprendre l'exercice de l'ajout de lignes de commandes dans un formulaire et coder le bouton « Supprimer une ligne ».

6.9 - CLONAGE

- **Objectif**

Copier un élément par clonage.

- **Syntaxes**

```
.clone([withDataAndEvents ])
```

```
.clone([withDataAndEvents ] [, deepWithDataAndEvents ])
```

- **Exemple**

```
<!DOCTYPE html>
<!--
Clonage.html
-->
<html>
    <head>
        <title>Clonage</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>
            <form action="" method="GET" id="formProduits">
                <div id="grille">
                    <label>Produit</label><br>
                    <p id="ligne">
                        <input type="text" name="produit[]" value="Badoit"
/>
                    </p>
                </div>
                <input type="button" value="+Ligne" id="btPlusLigne" />
                <br>
                <input type="submit" id="btSubmit" name="btSubmit" />
            </form>
        </div>
        <script src="../jquery/jquery.js"></script>
        <script src="Clonage.js"></script>
    </body>
</html>
```

```
/*
 * Clonage.js
 */
$("#btPlusLigne").click(cloner);

// ------------
function cloner() {
    // On clone la ligne
    let jqObjet = $("#ligne").clone(false).appendTo("#grille");
    // On met a blanc le dernier input
    $("#grille #ligne input").last().val("");
} /// cloner
```

6.10 - Itération

6.10.1 - Présentation

- **Objectif**

Boucler sur une liste ou une collection (tableau ordinal, tableau associatif, éléments multiples, liste d'éléments, etc).

- **Syntaxes**

http://api.jquery.com/each/

Récupération d'un tableau d'objets.

```
let objets = $("sélecteur")
```

La méthode size() renvoie la dimension d'une collection.

```
objets.size()
```

La méthode each() appliquée à une collection permet de la parcourir.

Elle accepte comme argument une fonction anonyme.

Cette fonction anonyme possède un premier argument index ou clé. Dans ce cas, dans le script de la boucle, l'élément est récupéré via l'objet [$(this)].
Cette fonction anonyme possède éventuellement un deuxième argument [element] qui correspond à l'élément courant dans la boucle.

```
objets.each(function([index [, element]]) { … } )
```

ou

```
$.each(objets, function([index[, element]]) { … } )
```

Note : pour les différences avec les boucles sur les tableaux cf jquery_et_les_tableaux.odt.

- **Exemple**

Cet exemple affiche les valeurs des éléments <li> de la liste identifiée par "listeVilles".

- Paris
- Lyon
- Marseille

Nombre de puces : 3
0:Paris
1:Lyon
2:Marseille

jqDomIterer.html

```
<body>
  <ul id="listeVilles">
    <li>Paris</li>
    <li>Lyon</li>
    <li>Marseille</li>
  </ul>
  <br>
  <p id="pResultat"></p>

  <script src="../jquery/jquery.js"></script>
  <script>
    // ------------------
    function iterations() {
       let lsResultat = "";
       let listeVilles = $("#listeVilles li");

       lsResultat += "Nombre de puces : " + listeVilles.size() + "<br>";

       listeVilles.each(function(index) {
          lsResultat += index + ":" + $(this).text() + "<br>";
       });

       $("#pResultat").html(lsResultat);
    }
    $(document).ready(iterations);
  </script>
</body>
```

listeVilles.each(function(index) ... et $.each(listeVilles , function(index) ... sont identiques.

- **Autre exemple**

Récupérer la liste des images d'une liste <ul>. Utile pour un carrousel.

```
<ul id="listeImages">
     <li><img src="http://localhost/images/cinema/100.jpg" /></li>
     <li><img src="http://localhost/images/cinema/101.jpg" /></li>
     <li><img src="http://localhost/images/cinema/102.jpg" /></li>
     <li><img src="http://localhost/images/cinema/103.jpg" /></li>
     <li><img src="http://localhost/images/cinema/104.jpg" /></li>
</ul>
```

```
// Recupere les URLS des images dans un tableau
let tImages = new Array();

let listeImages = $("#listeImages li img");

listeImages.each(function() {
    tImages.push($(this).attr("src"));
    console.log($(this).attr("src"));
});
```

Autre exemple :

forEach.html

```
<!DOCTYPE html>
<!--
forEach.html
-->
<html>
    <head>
        <title>forEach.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>

    <body>
        <div>
            <h3>forEach.html</h3>
            <select name="lbHobbies" id="lbHobbies" size="3"
multiple="multiple">
                <option value="1">Un</option>
                <option value="2">Deux</option>
                <option value="3">Trois</option>
                <option value="4">Quatre</option>
                <option value="5">Cinq</option>
            </select>
            <br><br>
            <input type="button" value="ForEach1" id="btForEach1" />
            <input type="button" value="ForEach2" id="btForEach2" />
            <input type="button" value="getTextes" id="btGetTextes" />
            <br><br>
            <label id="lblSelections">Sélections</label>
            <br><hr><br>
            <select name="lbNums" id="lbNums">
                <option value="1">Un</option>
                <option value="2">Deux</option>
                <option value="3">Trois</option>
                <option value="4">Quatre</option>
                <option value="5">Cinq</option>
            </select>
            <br>
            <input type="button" value="Texte et Value"
id="btGetSelection" />
            <br><br>
            <label id="lblSelection">Sélection</label>
        </div>
        <script src="../jquery/jquery.js"></script>
        <script src="../js/forEach.js"></script>
    </body>
</html>
```

```
/*
 * forEach.js
 */

// ----------
function init() {
    $("#btForEach1").click(forEach1);
    $("#btForEach2").click(forEach2);
    $("#btGetTextes").click(getTextes);
    $("#btGetSelection").click(getSelection);
    console.log("INIT");
} /// init


// --------------
function getTextes() {
    var lsSelections = "";
    $("select[id='lbHobbies'] option:selected").each(function () {
        lsSelections += $(this).text() + '|';
    });
    $("#lblSelections").html(lsSelections);
} /// getTextes

// ------------------
function getSelection() {
    $("#lblSelection").html($("#lbNums").val() + ":" + $("#lbNums
option:selected").text());
} /// getSelection

// --------------
function forEach1() {
    var lsResultat = "";

    let listeOptions = $("#lbHobbies option");
    listeOptions.each(function (index) {
        lsResultat += index + ":" + $(this).text() + "<br>";
    });
    $("#lblSelections").html(lsResultat);
} /// forEach1

// --------------
function forEach2() {
    var lsResultat = "";

//    $("select[id='lbHobbies'] option").each(function() {
//    let listeOptions = $("#lbHobbies option");
//    listeOptions.each(function() {
//        lsResultat += $(this).val() + ":" + $(this).text() + "<br>";
//    });

    let listeOptions = $("#lbHobbies option");
    $.each(listeOptions, function (cle, valeur) {
        lsResultat += $(this).val() + ":" + $(this).text() + "<br>";
    });


    $("#lblSelections").html(lsResultat);
} /// forEach2

// --------------------
```

```
$(document).ready(init);
```

6.10.2 - Exercice : récupérer les valeurs saisies dans un formulaire

Récupérez les valeurs saisies dans ce formulaire ainsi que les "name" des éléments (Pensez que cela pourrait servir pour générer un ordre SQL INSERT pour de l'AJAX).


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

    </head>

    <body>
        <form name="formSaisie" id="formSaisie" action="" method="GET">
            <label>Pseudo : </label>
            <input type="text" name="pseudo" value="Tintin" />
            <label>MDP : </label>
            <input type="text" name="mdp" value="mdp" />

            <input type="button" name="btValider" id="btValider"
value="Valider"/>
        </form>
        <p id="pResultat"></p>
    </body>
</html>
```

Corrigé :

cf jquery_exercices_corriges.odt

6.10.3 - Exercice : récupérer les éléments multiples d'un formulaire

Reprenons le formulaire d'ajout de lignes de commande … amélioré.
Il doit permettre d'ajouter des lignes de commande et de faire des calculs (montant de chaque ligne de commande, total de la commande).

Produit	Prix	Quantité	Montant
Badoit	1.20	1	1.2
Vittel	1.10	2	2.2
Vichy	1.30	10	13

+Ligne

Calculer Valider

Total : 16.4

```
<!DOCTYPE html>
<html>
    <head>
        <title>ExoJqDomItererForm4</title>
        <meta charset="UTF-8">
    </head>

    <body>
        <div>
            <form action="" method="GET" id="formProduits">
                <table border="0" id="tableLigcdes">
                    <thead>
                        <tr>
                            <th>Produit</th>
                            <th>Prix</th>
                            <th>Quantité</th>
                            <th>Montant</th>
                        </tr>
                    </thead>
                    <tbody id="bodyTable">
                        <tr>
                            <td><input type="text" name="produit[]"
/></td>
                            <td><input type="text" name="prix[]"
class="prix" size="5" /></td>
                            <td><input type="text" name="qte[]"
class="qte" size="5" /></td>
                            <td><label class="montant"></label></td>
                        </tr>
                    </tbody>
                </table>

                <br>
                <input type="button" value="+Ligne" id="btPlusLigne" />
                <input type="button" name="btCalculer" id="btCalculer"
value="Calculer"/>
                <input type="submit" />
            </form>
            Total : <span id="spTotal"></span>

            <p id="pResultat"></p>

        </div>
    </body>
</html>
```

Corrigé :

cf jquery_exercices_corriges.odt

6.10.4 - Exercice : CRUD panier

crudPanier.html

+Ligne

Produit	Quantité	Suppression
a	1	-
b	2	-
c	3	-

Valider

INSERT INTO lignes_de_commande(produit,quantite) VALUES('a','1')
INSERT INTO lignes_de_commande(produit,quantite) VALUES('b','2')
INSERT INTO lignes_de_commande(produit,quantite) VALUES('c','3')

```
<!DOCTYPE html>
<!--
crudPanier.html
-->
<html>
    <head>
        <title>crudPanier.html</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
    </head>
    <body>
        <div>
            <h3>crudPanier.html</h3>

            <form action="" method="GET" id="formProduits">
                <input type="button" value="+Ligne" id="btPlusLigne" />
                <br><br>

                <table border="1">
                    <thead>
                        <tr>
                            <th>Produit</th>
                            <th>Quantité</th>
                            <th>Suppression</th>
                        </tr>
                    </thead>

                    <tbody id="lignes">
                        <!-- Un peu limite d'avoir un tr identifiée
d'autant plus qu'elle va etre clonee -->
                        <tr id="ligne" class="uneLigne">
                            <td>
                                <input type="text" name="produit"
id="produit1" class="saisie" value="" />
                            </td>
                            <td>
                                <input type="text" name="quantite"
id="quantite1" class="saisie" size="2" value="" />
                            </td>
                            <td>
                                <input type="button" value="-"
class="btMoinsLigne" id = "btMoinsLigne1" title="1"/>
                            </td>
                        </tr>
                    </tbody>
                </table>
                <br>
                <input type="button" id="btValider" name="btValider"
value="Valider" />
            </form>
```

Corrigé

```
/*
 * crudPanier.js
 *
 * L'ajout de lignes est artificielle car le panier vient des selections
de l'internaute
 *
 * Quand le panier est vide on affiche pas de SQL
 *
 * bogues :
 *
 *
 */

/*
 * Variables globales
 */
var i = 1;
var numBoutonMoins = 1;
var giNombreDeChamps = 0;

// ----------
function init() {
    // Pour ajouter une ligne par clonage
    $("#btPlusLigne").click(ajouterLigneParClonage);
    // Pour supprimer la première et seulement la première ligne
    $("#btMoinsLigne1").click(supprimerLigneUne);
    // Pour valider et generer le SQL
    $("#btValider").click(valider);

    // Calcul du nombre de champs de saisie
    giNombreDeChamps = $(":text.saisie").size();
    console.log(giNombreDeChamps);
} /// init



// ----------------------------
function ajouterLigneParClonage() {

    // La ligne
    // Un peu limite d'avoir une tr identifiée d'autant plus qu'elle va
etre clonee
    //var ligne = $("#ligne");
    // C'est mieux
    let ligne = $("#lignes").children().first();

    // Clonage donc copie sans data ??? ni event
    $(ligne).clone(false).appendTo("#lignes");


    // Remise a "zero" des Input texte de la derniere (donc nouvelle)
ligne
    // La nouvelle ligne
    // Ce serait mieux avec le parcours de l'arbre et des childnodes ...
    // car si l'on rajoute un input text il faut modifier le script
    //let nouvelleLigne = $(".uneLigne").last();
    // C'est mieux
    let nouvelleLigne = $("#lignes").children().last();
```

```
    // KifKif
//    let nouvelleLigne = $("#ligne").last();

//console.log("nième enfant" + $(nouvelleLigne, ":nth-child(2)"));

    /*
     * je voudrais les :text de la derniere ligne ???
     */

    // Les input texte de la <table>
//    let textes = $("#lignes :text");
//    console.log("Nombre de :text dans la <table> : " + textes.size());

//    console.log(nouvelleLigne);
//    var its = $(nouvelleLigne, "tr td :text");
//    $(nouvelleLigne).each(function () {
//        console.log($(this).val());
//        $(this).val("");
//    });



//    var nbTextesInTable = textes.size();

    // ça (pas tres bon)
//    $(textes).each(function (index) {
//        if (index >= nbTextesInTable - 2) {
//            $(this).val("");
//        }
//    });

    // ou ça (pas tres bon non plus)
//    $(":text:gt(" + (nbTextesInTable - 3) + ")").val("");

    /*
     * IL FAUT TRAVAILLER AVEC childNodes
     * pour effacer les donnees des zones de saisie
     * Au passage FALSE dans le clone ne devrait pas prendre les datas
     */

    console.log("Type ligne : " + ligne.nodeName);
    console.log("Type nouvelleLigne : " + nouvelleLigne.nodeName);

    // Les <td>
    let tds = nouvelleLigne.children();

    console.log("Nombre de <td> de la nouvelle ligne : " + tds.size());

    tds.each(function (index, element) {
        console.log("nodeName : " + element.nodeName);
        // nodeType == 1 --> ELEMENT_NODE
        // nodeType == 3 --> TEXT_NODE
        console.log("nodeType : " + element.nodeType);
        if (element.nodeType === 1) {
            let inputs = $(this).children();
            console.log("Nombre d' <input> de la nouvelle TD : " +
inputs.size());
            console.log("Type du input : " + inputs[0].nodeName);
            // La TD ne contient qu'un seul INPUT
            let input = inputs[0];
            console.log("ID de l'input : " + input.id);
```

```
            console.log("className de l'input : " + input.className);
            //if (input.id !== "btMoinsLigne1") {
            if (input.className === "saisie") {
                inputs[0].value = "";
            }
        }
    });

    /*
     * Le cas du bouton (-);
     * Il faut changer le title
     * il faut lui ajouter un event
     */

    // ++
    numBoutonMoins++;
    // Recup du dernier bouton moins
    let dernierBoutonMois = $(".btMoinsLigne").last();
    // Modif du title
    dernierBoutonMois.attr("title", numBoutonMoins);
    dernierBoutonMois.attr("id", "btMoinsLigne" + numBoutonMoins);
    //console.log("Nombre de boutons moins : " + $
(".btMoinsLigne").size());
    //console.log("Title du dernier bouton moins : " +
dernierBoutonMois.attr("title"));

    // Ajout de l'event
    dernierBoutonMois.on("click", function (e) {
//        console.log("e : " + e);
//        console.log("e : " + e.target);
        console.log("e : " + e);
        let objet = e.target;
        console.log("e.target : " + objet.nodeName);
//        console.log("id : " + objet.id);
//        console.log("e : " + objet.nodeType);
        supprimerLigne(objet);
        console.log("this : " + $(this));
    });
} /// ajouterLigneParClonage


// --------------------
function supprimerLigne(cible) {
    console.log("supprimerLigne");

// cible = input button
// getParent : la TD
// getGrandParent : TR

//let parent = cible.
// let grandParent = parent.;
// grandParent.remove ....


//    console.log("cible.nodeName : " + cible.nodeName);
//    console.log("cible.nodeType : " + cible.nodeType);
//    console.log("cible.className : " + cible.className);
//    console.log("cible.id : " + cible.id);

    let boutonClicke = $("#" + cible.id);
    let pere = boutonClicke.parent().parent();
```

```
    pere.remove();

    // OK
    // l'ID du bouton clicke
//    let idBoutonMoins = cible.id;
//    console.log("ID bouton : " + idBoutonMoins);

    // Le numero du bouton clique
//    let numBoutonMoins = idBoutonMoins.substr(12);
//    console.log("Num bouton : " + numBoutonMoins);

    /*
     * KO car si on supprime au milieu avant
     * le num bouton n'est pas egal au rang dans les lignes
     */
//    $(".uneLigne").eq(numBoutonMoins - 1).remove();

    /*
     * Le rang du bouton clique dans les lignes ?
     */


    // KO
//    console.log($(".btMoinsLigne").attr("title"));
//    // KO avec passage de parametre
//    console.log("numBoutonMoins : " + numBoutonMoins);

//    console.log($(".btMoinsLigne").size());
//    console.log($(".btMoinsLigne").last().attr("title"));
} /// supprimerLigne



// ------------------------
function supprimerLigneUne() {
    // Elle a un statut a part
    //console.log("supprimerLigneUne");
    // Un peu limite avec #ligne mais c'est OK
    // $("#ligne").first().remove();
    // Avec la classe c'est mieux
    $(".uneLigne").first().remove();
} /// supprimerLigneUne



// -------------
function valider() {
    // Les variables
    var lsColonnes = "";
    var lsValeurs = "";
    var lsInsert = "";
    var lsInserts = "";
    var lsNomdelaTable = "lignes_de_commande";
    // Tous les input texte "classes" saisie
    var liste = $(":text.saisie");
    // Compteur pour le modulo
    var i = 1;

    //
    liste.each(function () {
```

```
        lsColonnes += $(this).attr("name") + ",";
        lsValeurs += "'" + $(this).val() + "',";

        if ((i % giNombreDeChamps) === 0) {
            lsInsert = "INSERT INTO ";
            lsInsert += lsNomdelaTable;
            lsColonnes = lsColonnes.substr(0, lsColonnes.length - 1);
            lsValeurs = lsValeurs.substr(0, lsValeurs.length - 1);
            lsInsert += "(" + lsColonnes + ") VALUES";
            lsInsert += "(" + lsValeurs + ")";

            lsInserts += lsInsert + "<br>";

            lsColonnes = "";
            lsValeurs = "";
            lsInsert = "";
        }
        i++;
    });

    $("#lblMessage").html(lsInserts);
} /// valider


// --------------------
$(document).ready(init);
```

6.11 - Gestion générique d'un événement sur les items d'une liste

6.11.1 - Objectif

Générer dynamiquement une liste d'éléments et lier à chaque élément une gestion événementielle.
Dans cet exemple à chaque <li> est lié un événement « click ». Lorsque l'on cliquera une puce on affichera le texte de la puce.

6.11.2 - Principe et démarche

Remplissage dynamique de la liste.
Liaison dynamique de chaque élément de la liste à un événement click.

6.11.3 - Codes

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <title>ElementsListeClicableDynamiquement.html</title>
    </head>

    <body>

        <article id="article">
            <h1>ElementsListeClicableDynamiquement</h1>
            <ul id="listeAPuces"></ul>
            <p>
                <label id="lblMessage"></label>
            </p>
        </article>

        <script src="../jquery/jquery.js"></script>
        <script src="ElementsListeClicableDynamiquement.js"></script>
    </body>
</html>
```

```
/*
 * ElementsListeClicableDynamiquement.js
 */

/**
 *
 * @returns {undefined}
 */
function init() {
    var ul = $("#listeAPuces");

    var t = ["Un", "Deux", "Trois", "Quatre", "Cinq"];
    for (let i = 0; i < t.length; i++) {
        var li = $("<li>");
        li.html(t[i]);
        ul.append(li);
    }

    $("#listeAPuces li").on("click", function() {
        $("#lblMessage").html($(this).html());
    });

} /// init

$(document).ready(init);
```

CHAPITRE 7 - ANNEXES

7.1 - Le CSS du site du cours

```
@charset "utf-8";
/* _coursJQuery.css */

*{margin:0; padding:0;}
html{background-color:#FFF; font-family:Verdana, Arial, Helvetica, sans-
serif; font-size:small;}
body{margin:20px auto; width:920px;} /* Pour center le cadre general */
div{padding:5px;}
/* gris : gray, dimgray, silver, gray */
/* rose : #FF00FF, #CC99FF, pink, #FF00FF */
/* kaki : khaki, darkkhaki, palegoldenrod, khaki */
#entete{
     background-color:darkblue;
     height:30px;
     width:900px;
     border:0px dotted dimgray;
     color:white;
     font-family:Geneva, Arial, Helvetica, sans-serif;
     font-weight:bold;
     background-image:url(../../images/litterature/livres_4.jpg);
}
#sommaire{
     background-color:#DDD;
     height: 400px;
     width: 250px;
     border:0px dotted dimgray;
     float:left;
     overflow:auto;
     font-size:x-small;
}
#centre{
     background-color:white;
     height: 400px;
     width: 640px;
     border:0px dotted dimgray;
     float:left;
     overflow:auto;
     color:black;
}
#pied{
     clear:both;
     background-color:black;
     height: 30px;
     width: 900px;
     border:0px dotted dimgray;
     color:white;
     font-weight:bold;
}

/* W3C CSS2.1: The border properties specify the width, style and color
border:5px solid red; */
table{border-collapse:collapse; border:1px solid black;}
caption, th{border:1px solid black; padding:3px;}
td{border:1px dotted black; padding:3px;}

#centre p{padding:5px;}
```

```
.flottantGauche{float:left;}
#centre h3{padding:5px; color:black;}
#centre select{color:black; background-color:white; width:150px;}
input[type="button"]{ width:150px;}
input[type="reset"]{ width:150px;}
input[type="submit"]{ width:150px;}
/* option{color:darkkhaki;} */

#nettoyeur{clear:both;}

#menu{padding:5px;}
ul#menu{ list-style-type:none;}
ul#menu li a{color:black; text-decoration:none;}
ul#menu li a:hover{text-decoration:underline; color:white;}
```

7.2 - Le modèle HTML pour le site du cours

_MODELEJQuery.html

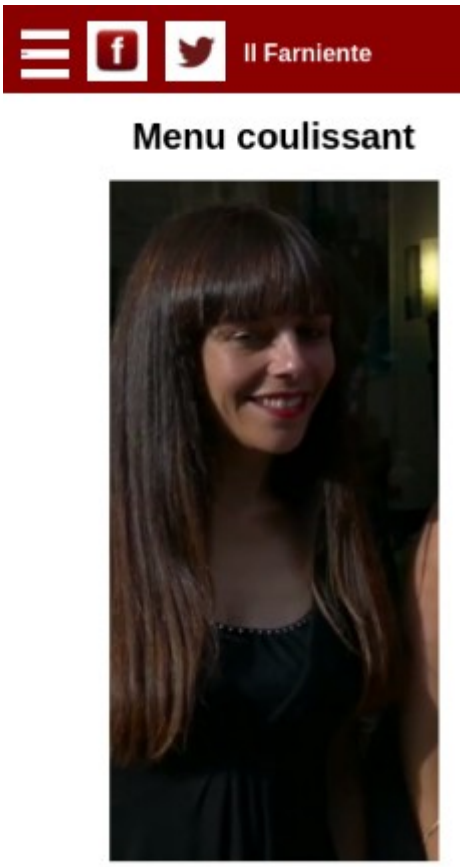
```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8" />
    <!-- Le lien CSS du site -->
    <link href="../css/_coursJQuery.css" rel="stylesheet"
type="text/css" />
    <title>_MODELEJQuery</title>
</head>

<body>
    <div id="entete"></div>
    <div id="sommaire"></div>
<div id="centre">
    <h3>TITRE</h3>
    <!-- Copiez votre code HTML perso ici !!! -->
    <p id="pResultat"></p>
    <label id="lblMessage"></label>
</div>
    <div id="pied"></div>

    <!-- Le lien JQ -->
    <script src="../jquery/jquery.js"></script>
    <!-- Le lien "MODELE" -->
    <script src="../js/_sommaireEntetePiedJQuery.js"></script>

    <!-- Le JavaScript specifique -->
    <script>
        // ------------
        function init() {
            $("#entete").html(afficherEntete());
            $("#sommaire").html(afficherSommaire());
            $("#pied").html(afficherPied());
        }

        // -------------
        function autre() {
        }
        /* Copiez votre code perso JavaScript ici !!! */

        $(document).ready(init);
    </script>

</body>

</html>
```

7.3 - Les inclusions pour le site du cours

_sommaireEntetePiedJQuery.js

```
// _sommaireEntetePiedJquery.js

// ----------------------
function afficherEntete() {
    let lsEntete = "";

    lsEntete += "<label id='lblEntete'>Le site du cours de
jQuery</label>";

    return lsEntete;
} /// afficherEntete

// ------------------------
function afficherSommaire() {
    let lsSommaire = "";

    // DEBUT DE LISTE
    lsSommaire += "<ul id='menu'>";

    // Basics ...
    lsSommaire += "<li><strong> --- Basics et + --- </strong></li>";
    lsSommaire += "<li><a
href='../html/_accueilCoursJQuery.html'>Accueil</a></li>";



    // FIN DE LISTE
    lsSommaire += "</ul>";

    return lsSommaire;
} /// afficherSommaire

// --------------------
function afficherPied() {
    let lsPied = "";

    lsPied += "<label id='lblPied'>&copy; Le site du cours de
jQuery</label>";

    return lsPied;
} /// afficherPied
```

7.4 - Menu coulissant avec icône Hamburger



Code HTML

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <link rel="stylesheet" type="text/css" href="MenuCoulissant.css">
        <title>Menu coulissant.html</title>
    </head>

    <body>

        <header class="entete">
            <table>
                <tr>
                    <td>
                        <a href="#">
                            <span id="hamburger" class="gradient-
icon"> </span>
                        </a>
                    </td>
                    <td>
                        <a href="" target="_blank">
                            <img src="../icones/facebook_rouge_fonce.jpg"
alt="menu" width="40"/>
                        </a>
                    </td>
                    <td>
                        <a href="" target="_blank">
                            <img src="../icones/twitter_rouge_fonce.jpg"
alt="menu" width="40"/>
                        </a>
                    </td>

                    <td>
                        Il Farniente
                    </td>
                </tr>
            </table>
        </header>

        <nav class="sommaire">
            <ul>
                <li>
                    <a href="Accueil.php">
                        Accueil
                    </a>
                </li>
                <li>
                    <a href="Antipasti.php">
                        Antipasti
                    </a>
                </li>
                <li>
                    <a href="PiattiCaldi.php">
                        Piatti Caldi
                    </a>
                </li>
                <li>
```

```
                    <a href="Desserts.php">
                        Desserts
                    </a>
                </li>
                <li>
                    <a href="Vins.php">
                        Vins
                    </a>
                </li>
                <li>
                    <a href="Boissons.php">
                        Boissons
                    </a>
                </li>
                <li>
                    <a href="Formules.php">
                        Formules !
                    </a>
                </li>
                <li>
                    <a href="Acces.php">
                        Plan d'accès
                    </a>
                </li>
                <li>
                    <a href="Contact.php">
                        Je contacte le restaurant !
                    </a>
                </li>
            </ul>
        </nav>

        <article>
            <h1>Menu coulissant</h1>
            <p class="pCentre">
                <img src="../images/brune_il_farniente.png" alt="Brushing"
/>
            </p>
        </article>

        <script src="../jquery/jquery.js"></script>
        <script src="MenuCoulissant.js"></script>
    </body>
</html>
```

Code CSS

```
/*
   MenuCoulissant.css
*/

*{margin: 0; padding: 0;}

html{height: 100%;}
body{
    height: 100%;
    font-family: sans-serif;
}

h1{
    text-align: center;
    margin: auto;
    padding-top: 15px;
    padding-bottom: 15px;
}

td{
    padding: 5px;
}

.entete{
    width: 100%;
    padding: 5px;
    background-color: darkred;
    color: white;
    position: fixed;
    top: 0px;
    height: 50px;
    font-weight: bold;
}

.gradient-icon {
    display: block;
    height: 32px;
    width: 32px;
    background:linear-gradient(to bottom, #fff 0%, #fff 20%, transparent
20%, transparent 40%, #fff 40%, #fff 60%, transparent 60%, transparent
80%, #fff 80%, #fff 100%);
}

.sommaire {
    display: none;
    margin-top: 60px;
    width: 100%;
    background-color: darkred;
}

.sommaire ul li{
    background-color: darkred;
    padding: 5%;
    border-bottom: 1px silver solid;
}

.sommaire ul li a{
```

```
    color: white;
    font-weight: bold;
    font-family: sans-serif;
    text-decoration: none;
}

article{
    margin-top: 60px !important;
    width: 100%;
}

.pCentre{
    text-align: center !important;
    margin: 0;
    padding: 0;
}
```

Code JavaScript

```
/*
 * MenuCoulissant.js
 * cf aussi en pur CSS : http://www.alsacreations.com/tuto/lire/1234-
creer-volet-coulissant-CSS3-target-transition.html
 *
 */

var lbMenuAffiche = false;

$("#hamburger").click(afficherMasquerSommaire);

/**
 *
 * @returns {undefined}
 */
function afficherSommaire() {

    $(".sommaire").css("position", "absolute");
    $(".sommaire").css("left", "-300px");
    $(".sommaire").css("top", "0px");
    $(".sommaire").css("width", "200px");
    $(".sommaire").css("opacity", "0");
    $(".sommaire").css("height", $(window).height() + "px");
    $(".sommaire").css("display", "block");
    $(".sommaire").animate({left: 0, opacity: 1}, 1000);
}

function masquerSommaire() {
    console.log("masquerSommaire");
    $(".sommaire").fadeOut(1000);
}


function afficherMasquerSommaire() {
    if (lbMenuAffiche) {
        masquerSommaire();
    }
    else {
        afficherSommaire();
    }
    lbMenuAffiche = !lbMenuAffiche;
}
```

7.5 - Détecter l'orientation

Si la largeur est > à la hauteur : paysage.
Si la hauteur est > à la largeur : portrait.

```
/**
 *
 * @returns {String}
 */
function getOrientation() {

    var lsOrientation = "";

    if ($(window).height() > $(window).width()) {
        lsOrientation = "Portrait";
        console.log("Orientation : Portrait");
    }
    else {
        lsOrientation = "Paysage";
        console.log("Orientation : Paysage");
    }
    return lsOrientation;

} /// getOrientation
```

7.6 - Storages

Pour les cookies et le storage cf jquerypi.odt.
Ou encore JavaScript.odt.

7.7 - Corrigés des exercices

Cf jquery_exercices_corriges.odt.

7.8 - Liens, bibliographies

Liens

Le site officiel : http://jquery.com/

UI : http://jqueryui.com/

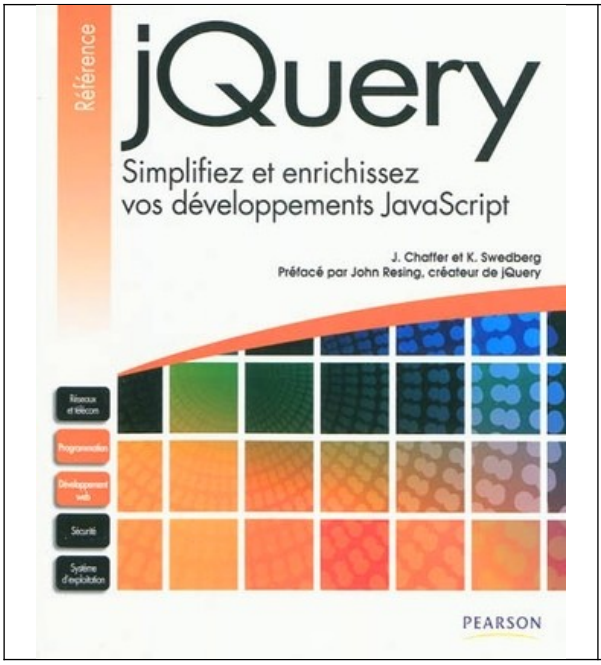
Documentation : http://docs.jquery.com/Main_Page

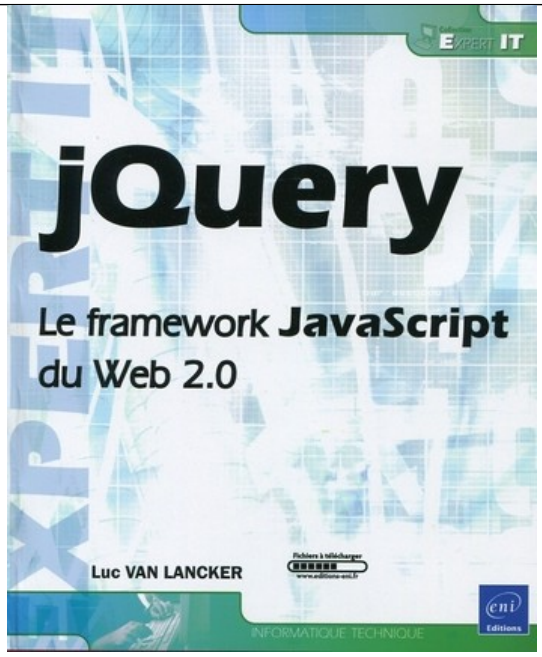
Les plugins : http://plugins.jquery.com/

Tutoriels :
http://docs.jquery.com/Tutorials
http://www.webcssdesign.com/ajax/jquery-plugins-240-tutoriels-a-decouvrir/

Téléchargement : http://docs.jquery.com/Downloading_jQuery

Bibliographie





7.9 - TODO

Récupérer une <tr>, les valeurs des <td> d'une <table> sur clic …