



# JQUERY

---



# SOMMAIRE

- Objectif du jour
  - Appréhender les possibilités offertes par jQuery pour parcourir le DOM
- Plan
  - La fonction magique \$
  - Les sélecteurs CSS
  - Parcourir le DOM
  - Manipuler les éléments
  - Gérer les évènements
  - Les plugins

# PRÉSENTATION DE JQUERY

---

- Trois grosses versions JQuery
  - JQuery 3.1.0 → 2016 Juillet 07 // Démarrée en 2016
  - JQuery 2.2.3 → 2016 Avril 05 // Démarrée en 2013
  - JQuery 1.12.3 → 2016 Avril 05 // Démarrée en 2011
- Fondateur John Resig (@jeresig sur Twitter)
- Communauté de plusieurs centaines de développeurs qui participent à l'amélioration du noyau et au développement de plugins tiers



# PRÉSENTATION DE JQUERY

---

- Différences entre les versions
  - Entre la 2 et la 3
    - Correction de bug,
    - API deprecated enlevée
    - Compatibilité entre les navigateurs, pour la 3 :
      - Internet Explorer: 9+
      - Chrome, Edge, Firefox, Safari: Current and Current - 1
      - Opera: Current
      - Safari Mobile iOS: 7+
      - Android 4.0+
  - Entre la 1 et la 2
    - Correction de bug,
    - API deprecated enlevée
    - <https://jquery.com/upgrade-guide/1.9/#overview>
- Possibilité d'utiliser la version récente de JQuery sur un vieux navigateur à l'aide du plugin de migration
  - Attention : cette solution n'est pas magique, tout ne fonctionnera pas

# INSTALLATION DE JQUERY

---

- Deux versions disponibles en téléchargement sur le site
  - Production ⇔ *compressed* : le code est compressé, le fichier est beaucoup moins gros (diminue la consommation de bande passante du site) mais illisible
    - 85 ko pour la version 3
  - Développement ⇔ *uncompressed* : code source lisible, pratique en cas de debug pour savoir sur quelle fonction le script a planté
    - 258 ko pour la version 3

# INSTALLATION DE JQUERY

---

- Pour installer jQuery il suffit d'ajouter le fichier JS dans la partie <head> du document HTML
  - <http://jquery.com/download/>

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Nom de ma page</title>
  <script src="chemin/vers/mon/fichier/jquery-3.1.0.min.js"></script>
</head>

<body>
  <h1>Ma page</h1>
</body>
</html>
```

- Il est aussi conseillé de télécharger le fichier map et de le placer au même niveau que le fichier js

# INSTALLATION DE JQUERY

---

- Vous pouvez aussi faire une référence sur une adresse externe (utilisation du CDN)
  - <https://code.jquery.com/>

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Nom de ma page</title>
  <script src="https://code.jquery.com/jquery-3.1.0.min.js"
    integrity="sha256-cCueBR6CsyA4/9szpPfrX3s49M9vUU5BgtiJj06wt/s="
    crossorigin="anonymous"></script>
</head>

<body>
  <h1>Ma page</h1>
</body>
</html>
```

- Attention : pas toujours conseillé selon vos contraintes d'infrastructure

# LA FONCTION MAGIQUE \$()

---

- Tout script jQuery repose sur la fonction magique `$()`
  - Cette fonction permet de récupérer n'importe quel élément ou ensemble d'éléments dans le DOM et de leur appliquer des traitements
  - L'alias `$` est utilisé par de nombreuses librairies et framework (Prototype, Scriptaculous, Mootools, etc.)
  - L'appel à la fonction `jQuery.noConflict()` permet d'assurer la compatibilité avec d'autres librairies, dans ce cas il faudra utiliser `jQuery()` et non plus `$()`



# LA FONCTION MAGIQUE \$()

---

- Avant de manipuler des éléments dans votre page, vous devez attendre que cette dernière soit correctement chargée (⇔ le DOM est bien analysé)
- Cette opération est réalisée par la méthode `$(document).ready(...)` de JQuery

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Ma première page avec JQuery</title>
    <script src="../../lib/jquery-3.1.0.js"></script>
    <script>
      //<![CDATA[
      $(document).ready(function() {
        // Mon code JQuery
      });
      //]]>
    </script>
  </head>
  <body>
    <div id="message"></div>
  </body>
</html>
```

# MANIPULER LE DOM AVEC JQUERY

---

- Important ! Il faut attendre que le DOM soit « ready » pour le manipuler
  - Equivalent du `window.onload()` en JavaScript pur, permet de s'assurer que l'accès à un élément sera possible
  - Tout code ajouté dans la fonction `jQuery.ready()` est exécuté après le chargement de la page
  - Il est possible d'appeler plusieurs fois la fonction `jQuery.ready()`, le code inséré dedans est ajouté dans une pile et non pas écrasé
- Pas d'obligation sur le positionnement du script dans la page
  - Yahoo! recommande tout de même de placer les JS en bas de page

# LA FONCTION MAGIQUE \$()

---

- Rechercher avec la fonction \$
  - Paramètres de la fonction \$(selector, context)
    - Une chaîne de caractères qui correspond au sélecteur permettant d'atteindre les éléments recherchés
    - Le contexte dans lequel la recherche est effectuée, par défaut la recherche s'effectue dans tout le document
  - Valeur de retour
    - L'élément renvoyé par cette fonction est un objet jQuery, c'est-à-dire que l'on peut utiliser toutes les fonctions de l'API sur cet élément renvoyé

# LES SÉLECTEURS

---

- Sélecteurs JQuery

```
// recherche d'élément sur son ID
var login = $("#login");
// recherche tous les éléments ayant la class CSS maClasse
var elms = $(".maClasse");
// le champ de saisi ayant pour id login dans le premier
// formulaire de la page
var login = $("input[name='login']", document.form[0]);
// Tous les éléments de type div
var elmsDiv = $("div");
// Tous les éléments de type div qui ont la classe msg
var elmsDiv = $("div.msg");
// Tous les éléments de type div qui ont l'id d01
var elmsDiv = $("div#d01");
// Tous les éléments de type div et form
var elms = $("div, form");
```

+ <https://learn.jquery.com/using-jquery-core/selecting-elements/>

# CHAÎNAGE DES MÉTHODES

---

- jQuery a été pensé de telle sorte que l'on puisse chaîner les méthodes appelées sur un objet
- Ce qui a pour avantage de
  - Ne pas recalculer le chemin jusqu'à l'objet DOM à chaque fois
  - Gagner en lisibilité
- Syntaxe :

```
$('#input[name=phoneNumber]')  
  .after('<div class="error"></div>')  
  .addClass('control phoneNumber')  
  .attr('maxlength', 10)  
  .focus(someFunction);
```

# EXERCICE 1

---

- Dans le code suivant :

```
<ul id="menu">
  <li><a href="#">lien 1</a></li>
  <li><a href="#">lien 2</a></li>
</ul>
<h1>Titre</h1>
<p class="about">
  Lorem ipsum dolor sit amet,
  <strong>consectetur</strong> adipiscing elit.
</p>
<p>
  Nulla pellentesque <em>molestie</em> tempus.
  <strong>Curabitur</strong> turpis est.
</p>
```

- Quels sélecteurs permettent de retrouver les éléments suivants ?
  - Le titre de niveau 1
  - Le menu
  - Le strong qui se trouve dans le premier paragraphe
  - Les éléments de la liste du menu

# FONCTIONS ET PROPRIÉTÉS CLASSIQUES

---

- Fonction **.html()** ou **.text()**
  - Donne le contenu de l'élément
- Fonction **.html(valeur)** ou **.text(valeur)**
  - Change le contenu de l'élément
- Fonction **.eq(index)**
  - Sélectionne l'élément à l'index donné
- Fonction **.length**
  - Donne la taille de l'élément si c'est un tableau

# FONCTIONS ET PROPRIÉTÉS CLASSIQUES

---

- Fonction **.attr(nom)**
  - Donne la valeur de l'attribut ciblé
- Fonction **.attr(nom, valeur)**
  - Change la valeur de l'attribut ciblé
- Fonction **.removeAttr(nom)**
  - Retire l'attribut ciblé
- Fonction **.val()**
  - Donne la valeur de l'attribut value
- Fonction **.val(valeur)**
  - Modifie la valeur de l'attribut value
- Fonction **.css(propriétéCss)**
  - Récupère la valeur associée à la propriété CSS
- Fonction **.css(propriétéCss, valeur)**
  - Modifie la valeur associée à la propriété CSS



# FONCTIONS ET PROPRIÉTÉS CLASSIQUES

---

- Fonction **.prop(nom)**
  - Donne la valeur de la propriété ciblée
    - ex: checked et selected
- Fonction **.prop(nom, valeur)**
  - Change la valeur de la propriété ciblée
- Fonction **.addClass(valeur)**
  - Ajoute une classe CSS à l'élément
- Fonction **.hasClass(valeur)**
  - Indique si l'élément possède la classe CSS
- Fonction **.removeClass(valeur)**
  - Retire une classe CSS à l'élément
- Fonction **.toggleClass(valeur)**
  - Retire ou ajoute une classe CSS à l'élément

# FONCTIONS ET PROPRIÉTÉS CLASSIQUES

---

- Fonction **.each(function(index, elem) { ... });**
  - admet comme paramètre une fonction anonyme qui contiendra le traitement à exécuter sur chaque objet
  - **\$( "p" ).each( function( i, item ){console.log( i, item );});**
  - pour récupérer l'objet en cours de traitement dans le **.each()**, il suffit d'appeler la fonction magique **\$** sur l'objet **this**
- Permet de parcourir un ensemble de valeurs
  - elem : le contenu de l'élément courant
  - index : l'index de l'élément courant
  - \${this}** : l'élément courant
  - \$( "p" ).each( function( i, item ){**  
**\$(this).css('background','red')}**  
**);**

# FONCTIONS ET PROPRIÉTÉS CLASSIQUES

---

- Fonction **.first()**
  - Le premier élément
- Fonction **.last()**
  - Le dernier élément
- Fonction **.filter(selector)**
  - Retourne un ensemble d'élément filtré en fonction du sélecteur
- Fonction **.has(selector)**
  - Retourne un ensemble d'élément filtré en fonction du sélecteur, ne prend que les enfants associés à l'élément.
- Fonction **.not(selector)**
  - Retourne un ensemble d'élément filtré en fonction du sélecteur, retire tous les éléments qui respecte le sélecteur

# RECHERCHER LES FILS (1/2)

---

- Fonction **.find()**

- L'appel à `$(element).find('div');` donne le même résultat que `$('div', element);`
- Pour remonter d'un niveau dans le chaînage de fonctions après un `.find()`, il faut utiliser la fonction `.end()`
- Exemple :  

```
$('#my-form')  
  .find('input[name=login]').addClass('required').end()  
  .find('input[name=pass]').addClass('required');
```

- Fonction **.children()**

- Recherche parmi les enfants directs de l'élément

## RECHERCHER LES FILS (2/2)

---

- Sélecteur \$("parent child") → identique à .find()
  - \$(' #foobar ul ') => Tous les éléments ul de #foobar **quelque soit le niveau de profondeur**
- Sélecteur \$("parent > child") → identique à .children()
  - Tous les éléments **directement** fils du parent
  - Code jQuery : \$(' #foobar > ul ') sélectionne le ul directement descendant
  - Code HTML :

```
<div id="foobar">
  <ul>...</ul>
  <div>
    <ul>...</ul>
  </div>
</div>
```

# EXERCICE 2

---

- Dans le code suivant :

```
<h1>Titre</h1>
<div class="about">
  <strong>Adipiscing
elit.</strong>
<div>
  Lorem ipsum dolor sit
amet,
  <strong>consectetur</stron
g>
</div>
</div>
<p>
  Nulla pellentesque
  <em>molestie</em> tempus.
  <strong>Curabitur</strong>
  turpis est.
</p>
```

- Quels sélecteurs permettent de retrouver les éléments suivants
  - Tous les strongs de la page
  - Tous les strongs fils de la première div
  - Le strong directement fils de la première div
  - Le strong de plus bas niveau dans la première div

# RECHERCHER LES PARENTS

---

- Fonction **.parent()**

- Récupère le parent **direct** de l'élément

- Code HTML :

```
<div>
  <ul>
    <li>jQuery</li>
    <li>Mootools</li>
  </ul>
</div>
```

- Code jQuery : `$('li').parent()` renverra ul

- Fonction **.parents(selector)**

- Remonte dans les parents jusqu'à trouver le père recherché

- Code jQuery : `$('li').parents('div')` renverra la div parente du ul

# RECHERCHER LES FRÈRES (1/2)

---

- Récupérer les frères suivants
  - `.next()` → le frère suivant immédiat
  - `.nextAll()` → tous les éléments suivants
- Récupérer les frères précédents
  - `.prev()` → le frère précédent immédiat
  - `.prevAll()` → tous les éléments précédents



## RECHERCHER LES FRÈRES (2/2)

---

- Fonction `.siblings()`
  - Récupère tous les frères (précédents et suivants) de l'élément
  - Code jQuery : `$("#container").siblings("div");`
  - Code HTML :

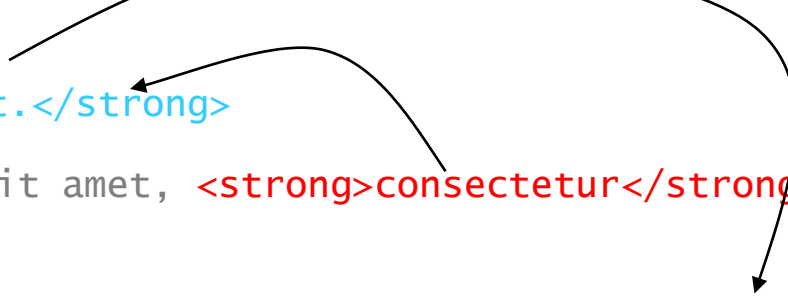
```
<p>Lorem ipsum dolor sit amet.</p>
<div id="container">
  <ul>
    <li>item 1</li>
    <li>item 2</li>
  </ul>
</div>
<div></div>
```
- que ferait `$('#container').siblings();` (sans le sélecteur "div") ?

# EXERCICE 3

---

- Dans le code suivant :

```
<h1>Titre</h1>
<div class="about">
  <strong>Adipiscing elit.</strong>
</div>
  Lorem ipsum dolor sit amet, <strong>consectetur</strong>
</div>
<p>
  Nulla pellentesque <em>molestie</em> tempus. <strong>Curabitur</strong>
  turpis est.
</p>
```



- En une seule requête et en utilisant le chaînage de méthodes, partir du **strong** et se déplacer jusqu'au **strong** puis jusqu'au **strong**
- Grace à la fonction `siblings` faire un encadré de 2 pixels en rouge sur le dernier paragraphe et le titre. Les encadrés devront faire la largeur du contenu et pas la largeur de la page (*display* sera donc *inline*).
  - `border-style: solid;border-color: red;border-width: 2px;`
  - `border: 2px solid red;`

# RECHERCHER GRÂCE AUX ATTRIBUTS (1/2)

---

- Sélecteur `$("element[attribut=value]")`
  - Trouve les éléments dont l'attribut spécifié a la valeur recherchée
  - Code jQuery : `$('input[name=login]')`
  - Code HTML :  
`<input type="text" name="login" />`  
`<input type="password" name="password" />`
- Pour les inputs il y a même plus efficace
  - `:text`, `:radio`, `:reset`, `:password`, `:submit`, `:checkbox`, `:button`
- Pour ne récupérer que les éléments qui contiennent un attribut particulier on utilise `$('input[alt]')`, pour forcer les inputs ayant un attribut alt

## RECHERCHER GRÂCE AUX ATTRIBUTS (2/2)

---

- Pour la valeur des attributs il est possible d'effectuer des comparaisons plus poussées :
  - ^= la valeur de l'attribut commence par ...
  - \$= la valeur de l'attribut fini par ...
  - != la valeur de l'attribut ne contient pas ...
  - \*= la valeur de l'attribut contient ...
  - ...

# RECHERCHER GRÂCE À L'INDEX (1/2)

---

- Sélecteur `$(":eq(index)")`
  - Renvoie le nœud à l'index indiqué dans une liste de nœuds
  - **Remarque** : les index commencent à 0
  - Code HTML :

```
<ul>  
  <li>PHP</li>  
  <li>HTML</li>  
  <li>JavaScript</li>  
</ul>
```
  - Code jQuery: `$('ul li:eq(1)')`

## RECHERCHER GRÂCE À L'INDEX (2/2)

---

- D'autres sélecteurs basés sur l'index du nœud
  - :lt(index) : les index inférieurs strictement
  - :gt(index) : les index supérieurs strictement
  - :even et :odd : un index sur deux (permet de créer une sélection par alternance)
  - :first → alias de :eq(0)
  - :last → alias de :eq(dernier index)

# EXERCICE 4

---

## ● Dans le code suivant :

```
<div id="menu">
  <ul>
    <li><a href="#">Lien 1</a></li>
    <li><a href="#">Lien 2</a></li>
  </ul>
</div>
<div class="contact">
  <strong>Adipiscing elit.</strong>
  <div>
    Lorem ipsum dolor sit amet, <strong>consectetur</strong>
  </div>
  <form method="POST">
    <input type="text" name="email" />
    <input type="submit" value="s'inscrire" />
  </form>
</div>
```

## ● Que fait l'instruction suivante ?

```
$('#menu')
  .find('li:eq(1)').siblings('li').css('border', '1px solid
blue').end().end()
  .siblings('.contact').find(':text[name=email]').parent('form').css('bord
er', '1px solid red');
```

# AJOUTER ET SUPPRIMER UN NOEUD

---

- Ajouter
  - Fonction **.append(contenu)**
    - Insère le contenu fourni en paramètre à la fin des éléments sélectionnés
  - Fonction **\$("contenu").appendTo(selector)**
    - Insère le contenu fourni à la fin des éléments ciblés via le sélecteur
  - Fonction **.prepend(contenu)**
    - Insère le contenu fourni en paramètre au début des éléments sélectionnés
  - Fonction **\$("contenu").prependTo(selector)**
    - Insère le contenu fourni au début des éléments ciblés via le sélecteur
  - Fonction **.after(contenu)**
    - Insère après chaque élément sélectionné
  - Fonction **.before(contenu)**
    - Insère avant chaque élément sélectionné



# AJOUTER ET SUPPRIMER UN NOEUD

---

- Supprimer
  - Fonction **.remove()** ou **.remove(selector)**
    - Supprime les éléments contenus dans l'ensemble d'éléments sélectionné
  - Fonction **.detach()** ou **.detach(selector)**
    - Supprimer les éléments du contenu sans le détruire (conserve les données associées à l'objet)

## EXERCICE 5

---

- Expérimentez dans le code html fourni l'ensemble des fonctions de manipulation :
  - .append(), .appendTo()
  - .prepend(), .prependTo()
  - .after(), .before()
  - .remove() , detach()
- Information :
  - Pour créer un élément il suffit d'écrire \$('<div/>');

# ATTACHER DES ÉVÈNEMENTS

---

- Pour associer des événements sur des nœuds du DOM en jQuery, il ne faut pas utiliser les attributs « onclick », etc.
  - Ne pas faire :  
`<input type="button" onclick="maFonction();" />`
- Il faut utiliser les fonctions :
  - `.click()`, `.focus()`, `.blur()`, `.change()`, `.hover()`, etc.
- Syntaxe :  
`$(selecteur).click(function() {  
 // comportement devant être exécuté sur le click  
});`

# ATTACHER DES ÉVÈNEMENTS

---

- Fonctions `.live()`, `.bind()` et `.one()`
  - Elles reçoivent en paramètre le nom de l'évènement (format texte)
  - Une fonction à exécuter lorsque l'évènement est déclenché
- Syntaxe :

```
$(element).bind('click', function() {});  
$(element).bind({ click: function() {},  
                  focus: function() {}});
```
- Cas particulier de la fonction `.live()`
  - Permet d'attacher **immédiatement et dans le futur** un évènement à tous les objets correspondant au sélecteur
  - Ne pas utiliser cette méthode, faire usage de **on** ou **delegate** selon votre version de JQuery
- Cas particulier de la fonction `.one()`
  - L'évènement n'est déclenché qu'une seule fois

# ATTACHER DES ÉVÈNEMENTS

---

- Ne pas utiliser `live` si vous êtes dans une version récente de JQuery.
- Utilisez une des équivalences suivantes

```
// En jQuery 1.3+
$( "#elm" ).live( "click", function() { alert( "Goodbye!" );});

// jQuery 1.4.3+
$( 'elm1' ).delegate( 'elm2', "click", function() { alert( "Goodbye!" );});
$( 'p' ).delegate( 'span', 'mouseenter', function() { alert( 'Goodbye!' );});

// jQuery 1.7+
$( document ).on( "click", "elm", function() { alert( "Goodbye!" );});
$( 'elm' ).on( 'click', function() { $(this).css('background','red')});
```

- **IMPORTANT** : Quand vous modifiez l'arborescence du DOM, il est fortement recommandé de le faire dans un événement via `on`. Sinon, votre DOM ne sera pas à jour.

# DÉTACHER LES ÉVÈNEMENTS

---

- Détacher un évènement se fait avec la fonction `.unbind()`
  - Syntaxe : `$(element).unbind('click');`
- Pour détacher un évènement attaché par la fonction `.live()` il faut utiliser la fonction `.die()`
- Depuis la version 1.4.3 de jquery
  - il existe aussi `undelegate ()`
- Depuis la version 1.7 de jquery
  - il existe aussi `off()`

# DÉCLENCHER UN ÉVÈNEMENT

---

- Pour déclencher manuellement un évènement, plusieurs syntaxes sont possibles :
  - Soit en appelant directement l'évènement :
    - `$(element).click();`
  - Soit en utilisant la fonction `.trigger()` :
    - `$(element).trigger('click');`
- Le choix entre une méthode ou l'autre est à la discrétion du développeur

# EXERCICE 6

---

- Complétez le HTML fourni :
  - Un champ texte accompagné d'un bouton « + » permettant d'ajouter une nouvelle tâche
  - Mettre en place un « placeholder » dans le champ texte : « Nouvelle tâche »
  - Chaque tâche dispose d'un bouton de suppression « x » à côté
  - Au clic sur la checkbox à côté de chaque tâche, celle-ci est marquée comme résolue ⇔ elle est barrée

## Ma liste de tâches

<input checked="" type="checkbox"/> Acheter du pain.	X
<input type="checkbox"/> Acheter du vin.	X
<input type="text" value="Nouvelle tâche"/>	+



# ANIMATIONS

---

- La partie des animations dans JQuery est délégué à un plugin JQuery = JQuery-ui
- <https://jqueryui.com/>
- Il faut l'installer pour en faire usage
  - Ajout du JS associé à JQuery UI **APRES** celui de JQuery
  - Ajout du style (CSS) sélectionné
- Vous pouvez fabriquer votre propre version de JQuery UI en ne prenant que les éléments qui vous intéressent.
  - Fait un JS plus petit,
  - mais ne contiendra pas tout, ce qui peut poser problème pour d'autres plugins

# ANIMATIONS ET COMPOSANTS

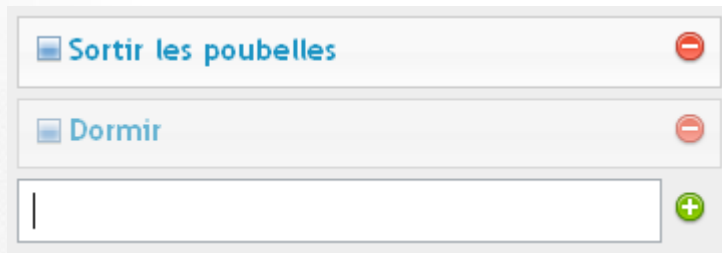
---

- Rendez vous sur le site de JQuery UI afin de consulter toutes les animations disponibles
  - <https://jqueryui.com/effect/>
- JQuery ui sait gérer
  - Le drag and drop
  - Le trie
  - Le redimensionnement
- JQuery ui sait aussi rendre des composants visuels comme
  - Un calendrier
  - Des accordéons
  - Des boutons
  - Des checkbox / radio
  - Des menus
  - Des fenêtres (modales ou non) pour remplacer les alert
  - Des slider, spinner
  - Des selects avec auto complétion
  - Une barre de progression

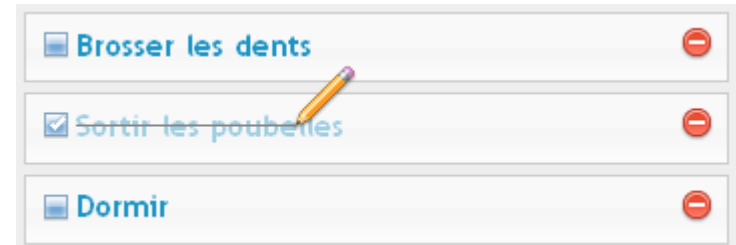
# EXERCICE 7

---

- Améliorer la Todo List en ajoutant :
  - Un effet de fondu à la création d'une tâche
    - Utilisez la méthode toggle avec un effet clip
    - Pensez à cacher (méthode hide()) votre bloque de div avant de l'ajouter
  - Un effet de fondu à la suppression d'une tâche
    - Utilisez la méthode toggle avec un effet blind
  - Une animation avec un crayon qui vient « rayer » une tâche réalisée
    - <http://api.jquery.com/animate/>



A screenshot of a todo list application. It features a light gray background with rounded corners. The list contains two items: "Sortir les poubelles" and "Dormir", each with a blue square icon on the left and a red circle with a minus sign on the right. Below the list is a white input field with a green plus icon on the right.

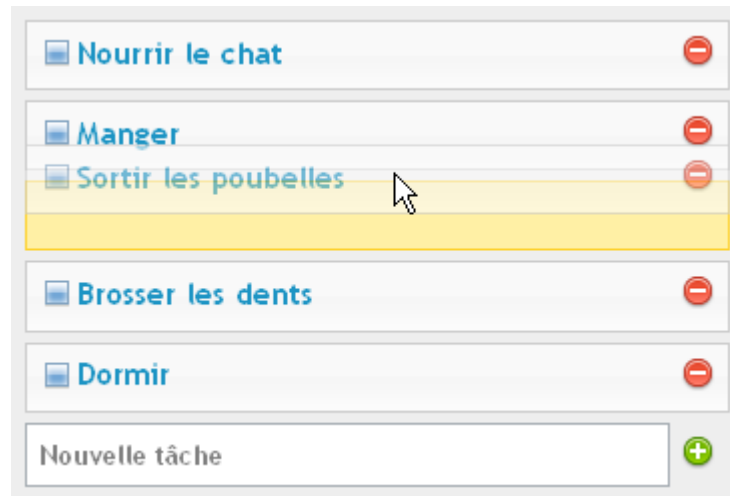


A screenshot of the same todo list application, but with an additional item, "Sortir les poubelles", which is now checked with a blue checkmark and crossed out with a yellow pencil. The list now contains three items: "Brosser les dents", "Sortir les poubelles", and "Dormir". Each item has a blue square icon on the left and a red circle with a minus sign on the right.

## EXERCICE 8

---

- Ajouter la possibilité de réordonner les tâches à l'aide de « Sortable » de jQuery UI
  - Vous pouvez transformer les div en li pour plus de simplicité



# JQUERY DATA

---

## + Association de données avec data()

- `$("div").data("price", { value: 16, currency: "euro" });`

## + Data HTML

- `<div data-role="page" data-last-value="43" data-hidden="true" data-options='{ "name": "John" }'></div>`
- `$("div").data("role") === "page";`
- `$("div").data("lastValue") === 43;`
- `$("div").data("hidden") === true;`
- `$("div").data("options").name === "John";`

## + Rappel :

- `===` : égalité stricte (typage inclus)
- `==` : égalité faible

# PLUGINS

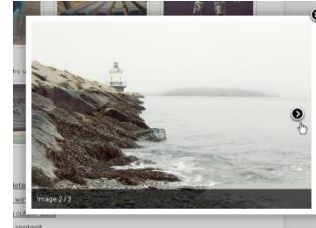
---

- Un plugin est un ensemble de fichiers
  - Code JavaScript
  - Code CSS
  - Images / ressources Web
- Il ajoute une ou plusieurs fonctionnalités
- JQuery possède un très grand nombre de plugins
  - Rechercher un plugin via <http://plugins.jquery.com/>
  - Il faut choisir en fonction
    - Du besoin fonctionnel
    - De la contrainte d'internationalisation
    - De la version de JQuery
    - Du degrés de qualité de réalisation

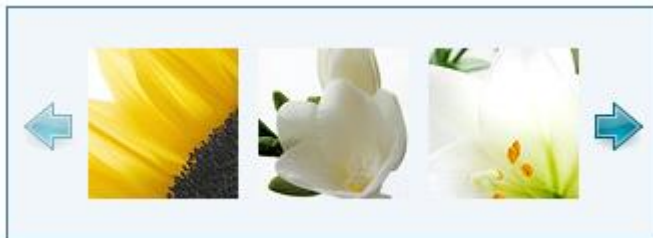
# PLUGINS GRAPHIQUES (1/2)

---

- FancyBox
  - Site : <http://fancybox.net/>
  - Permet de faire surgir des images



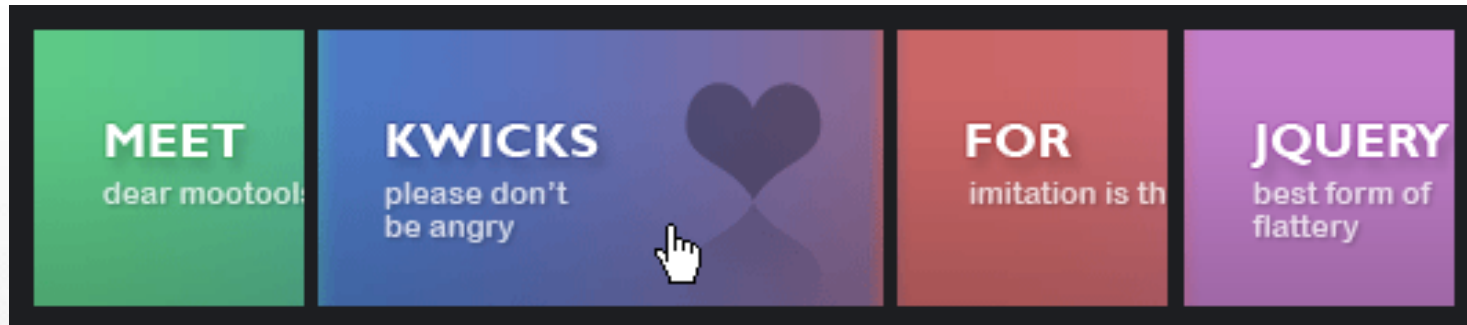
- jCarousel
  - Site : <http://sorgalla.com/jcarousel/>
  - Galerie d'images



# PLUGINS GRAPHIQUES (2/2)

---

- Kwick's
  - Site : <http://devsmash.com/projects/kwicks>
  - Ex : mettre en avant des produits avec des infos complémentaires cachées

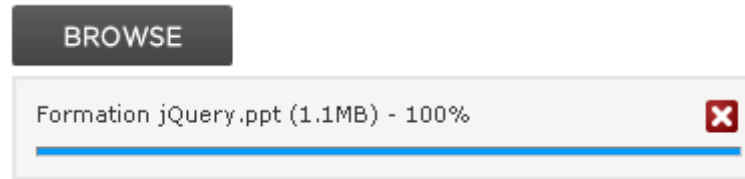




# GESTION DE FICHIERS ET DE NOTIFICATIONS

---

- Uploadify
  - Site : <http://www.uploadify.com>
  - Formulaire d'upload cross-browser avec indicateur de progression



- + Noty
  - Site : <http://ned.im/noty/#/about>
  - Popup de notifications

# VALIDATION DE FORMULAIRE

---

- JQuery Validator
  - Site : <https://jqueryvalidation.org/>
  - Permet de valider simplement un formulaire web en indiquant des règles de validation
  - Affiche un message automatiquement lors des erreurs

# AFFICHAGE DE DONNÉES

## + TableSorter

- Site : <http://tablesorter.com/docs/#Demo>
- Affichage et tri de données complexes

First Name ▲	Last Name ◆	Age ▼	Total ◆	Discount ◆	Difference ◆	Date ◆
Bruce	Almighty	45	\$153.19	44.7%	+77	Jan 18, 2001 9:12 AM
Bruce	Evans	22	\$13.19	11%	-100.9	Jan 18, 2007 9:12 AM
Bruce	Evans	22	\$13.19	11%	0	Jan 18, 2007 9:12 AM
Clark	Kent	18	\$15.89	44%	-26	Jan 12, 2003 11:14 AM
John	Hood	33	\$19.99	25%	+12	Dec 10, 2002 5:14 AM
Peter	Parker	28	\$9.99	20.9%	+12.1	Jul 6, 2006 8:14 AM

**TIP!** Sort multiple columns simultaneously by holding down the shift key and clicking a second, third or even fourth column header!

## + DataTables

- Site : <https://datatables.net/>
- Affichage et tri de données complexes

Show 10 entries

Search:

Name ▲	Position ◆	Office ◆	Age ◆	Start date ◆	Salary ◆
Airi Satou	Accountant	Tokyo	33	2008/11/28	\$162,700
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009/10/09	\$1,200,000
Ashton Cox	Junior Technical Author	San Francisco	66	2009/01/12	\$86,000
Bradley Greer	Software Engineer	London	41	2012/10/13	\$132,000
Brenden Wagner	Software Engineer	San Francisco	28	2011/06/07	\$206,850