JQuery – Introduction

- Le JavaScript présente quelques inconvénients
  - le DOM n'est pas exactement le même d'un navigateur à l'autre (noms de certains attributs, méthodes présentes dans document vs. window,...)
  - le JavaScript n'est pas exactement le même d'un navigateur à l'autre (p.ex. l'interface de connexion web asynchrone)
  - le parcours du DOM est pénible, sauf si on peut aller directement au but (cf. getElementById())
- JQuery est *une* bibliothèque écrite en JavaScript qui fournit une sur-couche d'abstraction pour 1) ajouter des fonctionnalités et 2) cacher les différences entre les navigateurs
- http://jquery.com/, http://api.jquery.com/

JQuery – Le $ « magique »

- Après avoir chargé JQuery (<script src="jquery.min.js"> </script>), on dispose d'une nouvelle constante : $
- $ est un objet JavaScript qui contient les propriétés et les fonctions de JQuery. C'est le point d'entrée de toute déclaration ou expression en JQuery.
- Dans une fonction JavaScript, on peut donc librement mixer des déclarations en JavaScript classique avec celles en JQuery.
  Ex. :
```
function verifierFormulaire() {
        var n = $('input[type="text"][value=""]').size()
        if (n > 0)
            alert ('Vous devez encore remplir '+n+'champs')
    }
```

JQuery – Parcours du DOM

- $(elem) : transforme un élément du DOM en objet JQuery (on peut retrouver l'élément par .get())
- $('sel') : renvoie un objet JQuery représentant les éléments du DOM qui correspondent au sélecteur CSS
  *ex : tous les paragraphes $('p'), tous les premiers items de listes $('ul li:first, ol li:first'), fils directs de toto $('#toto > *')*
  - En plus des sélecteurs CSS, JQuery offre de nouveaux, cf. http://api.jquery.com/category/selectors/
- Fonctions JQuery pour le parcours d'arbre : .children(), .first(), .next(), .parent(),...
  ex : fils directs de toto $('#toto').children(),
  les tables filles directes de toto $('#toto').children('table')
  - cf. http://api.jquery.com/category/traversing/

JQuery – Chaînage des appels

- En général, une fonction JQuery renvoie un objet JQuery, ce qui permet d'enchaîner les appels.
  ex : *$('input[type="text"]').parent().children('label')*
  tous les labels partageant leur parent direct avec un input de type text
- Le chaînage est possible indépendamment du nombre d'objets concernés :
  $('#toto').parent().next() - le résultat est 0 ou 1 objet
  $('p').parent().next() - les résultat est une collection d'objets

- Exceptions : .get() renvoie un élément du DOM, .size() donne un entier, .hasClass() fournit un booléen,...

JQuery – Manipulation du DOM

•http://api.jquery.com/category/manipulation/

- Création de contenu : .clone(), .html(), .text()
- Insertion dans : .append(), .prepend(), .appendTo(), .prependTo()
- Insertion autour : .after(), .before(), .insertAfter(), .insertBefore()
- Suppression : .remove(), .detach()
- Remplacement : .replaceAll(), .replaceWith()
- Styles : .css(), .position(),...
- Attributs : .attr(), .prop(), .val(), .removeAttr()
- Classes : .hasClass(), .addClass(), .removeClass(), .toggleClass()

JQuery – Manipulation du DOM (ex.)

- Si l'élément toto est vide, modifier sa valeur
  if ($('#toto').text() == '') $('#toto').text('non spécifié')
- Supprimer tous les tableaux qui contiennent des cellules de classe « xxx »
  $('table:has(td.xxx)').remove()
- Insérer l'élément « dernier » à la fin de chaque liste numérotée
  *$('ol > li:last').after('
dernier
')*
- Mettre en rouge tous les paragraphes de classe « coco » et supprimer la classe
  $('p.coco').css('color', 'red').removeClass('coco')

JQuery – Fonctions de collections

- .eq(index) : fournit l'objet JQuery à l'indexe spécifié
- .get() ou .get(index) : fournit l'élément du DOM spécifié
- .size() ou .length : taille de la collection
- .each(f) : itérer une fonction sur chaque élément
  ex : $('tr').each(function (index, elem) {
  if ($(elem).text() == '')
  $(elem).addClass('vide')
  else
  $(elem).removeClass('vide')
  })
- .add(), .slice(), .not() : opérations ensemblistes
- .first(), .last()

JQuery – Événements

- JQuery permet dynamiquement d'attacher/détacher des écouteurs d'événements
  - fonctions spécifiques : .keypress(), .focus(), .click(),...
  - fonctions génériques : .on(), .off(), .one()
- .on(events [, selector] [, data], handler(eventObject)) permet de gérer tous les cas de figure
  ex : ajouter une coloration en bleu du paragraphe survolé
  $(document).on('mouseover','p',function(e) {
  $(this).css('color', 'blue')})
- .on() permet d'attacher l'écouteur même aux éléments qui seront insérés dans la page plus tard, alors que .click() etc. ne s'appliquent qu'aux éléments présents
- par contre .click() etc. permettent aussi de générer l'événement correspondant

JQuery – Animations et effets

- JQuery permet d'animer les modifications appliquées à la page : .animate(), .show(), .hide(), .delay(),... cf. http://api.jquery.com/category/effects/
- Ex : *$('#toto').css('opacity', '0.25')* rend l'élément toto transparent à 75% immédiatement, alors que *$('#toto').animate('{opacity : 0.25}', 5000)* réalise l'opération progressivement sur 5 secondes

JQuery – Ajax

- JQuery offre des fonctions pratiques pour Ajax
  cf. http://api.jquery.com/category/ajax/
- $.ajax() : fonction générique qui englobe toutes les autres
- $.get() et $.post() : fonctions spécifiques pour les requêtes GET et POST
- .fail(), .done(), .always() : gestionnaires d'erreur, succès
- *Ex : dès que l'utilisateur choisit un login, vérifier en arrière-plan si ce login n'est pas déjà utilisé*

```
$(document).on('change','input[name="login"]',function() {
    var login = $(this).value()
    $.get('./check-unused?login='+login, function(data) {
        if (data == 'taken') $('#warning1').show()
    })
})
```