

# Paris Web 2007 - Ateliers

## Les bibliothèques JS

### jQuery

Orange Labs

Julien Wajsberg, Recherche & Développement  
17/11/2007, présentation à Paris Web 2007



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

- jQuery: une simple bibliothèque à importer
- Son but: rendre JavaScript plus "sympa" à utiliser.
  - *write less, do more*
- Assez des incompatibilités entre navigateurs ! jQuery nous cache tout ça
- ... et le fait mieux que vous : il sait utiliser des spécificités natives de certains navigateurs pour aller plus vite ! (ex: XPath chez Mozilla, Javascript 1.6+)

# \$('.jquery')

- jQuery: une fonction
  - des sélecteurs à passer en argument
  - renvoie un objet qui représente les éléments correspondants
- Prototype l'a plébiscité: \$ est la fonction-raccourci pour créer les objets

## \$('#anything')

- \$ accepte un sélecteur CSS en argument
- \$ accepte des ID :
  - \$('#jquery') retourne **un** élément <-> document.getElementById
- \$ accepte des classes :
  - \$('.jquery') retourne tous les éléments qui correspondent
- \$ accepte **plusieurs** sélecteurs (comme en CSS, en fait)
  - \$('.article, .nouvelles, .edito')

## \$(anything)

- \$ accepte des sélecteurs complexes :
  - \$(' .article + p a')
- \$ accepte des sélecteurs spécifiques :
  - \$(':radio'), \$(':header'), \$(':first-child')
- des sélecteurs en forme de filtres :
  - \$(':checked'), \$(':odd'), \$(':visible')
  - plus fort: \$(':contains(du texte)')
- des attributs
  - \$('a[href]'), \$('a[href^=http://]'), \$('img[src\$=.png]')

# `$('#jquery').show()`

- jQuery: un objet
  - les méthodes s'appliquent généralement à tous les éléments sélectionnés
    - `$('#jquery').hide();`
- plein de méthodes utilitaires
  - parcourir le DOM: `.parent()`, `.next()`, `.children()`, `.parents()` ...
  - ajouter ou retirer des classes CSS: `addClass`, `removeClass`
  - manipuler: `append`, `wrap`, `prepend`
  - effets: `show`, `hide`...

```
$(...).parent().next().find(...).show();
```

- Intérêt fondamental: la plupart des méthodes de l'objet retournent l'objet lui-même
  - on peut chaîner les appels !
    - `$('anything').parent().find('still anything').show();`
- Cette propriété est extrêmement puissante !

## javascript avancé : les callbacks

- une fonction prend en argument une autre fonction, qu'elle pourra appeler ensuite (événements, complétion...)

```
function traitementFini() {  
    alert('Mon traitement est fini');  
}  
traitementLong(parametres, traitementFini);
```

- jQuery utilise des callbacks pour tous les événements

## function() { .. } : les closures

- closures, ou fonctions anonymes
    - peut être utilisée partout où on attend une fonction
- ```
var mafonction = function() { ... };  
traitementLong(parametres, function() { ... });
```
- On utilise énormément ces constructions dans un développement JavaScript "moderne"...

## \$(anything).click(function)

- Javascript non intrusif: bref rappel
  - code HTML propre : des balises, des classes, des id, **et c'est tout**
  - le javascript exploite les sélecteurs CSS, pour utiliser des événements, ou ajoutant des éléments
- `$('#a.popup').click(function) {  
    alert(this.href);  
}`
- Imaginez le nombre de lignes en DOM classique pour faire ça... (ou encore `$('#a.popup > span + span[attr=$toto]')`).

## \$.ajax

- une méthode utilitaire pour encapsuler une requête ajax
- on reçoit la réponse dans une callback
- on peut traiter cette réponse à la mode jQuery...

## \$.ajax : exemple

```
var callback = function(data) {  
    var $data = $(data);  
    var nb = $data.find("reponse").length;  
    ...  
    var nb =  
        $data.find("choix:contains('Bonne)').parent().length;  
};  
  
$.ajax({  
    cache: false,  
    success: callback,  
    url: file  
});
```

## les plugins

- on peut étendre facilement jQuery en utilisant des plugins
- les méthodes ajoutées sont au même niveau que les méthodes natives...
- il faut tâcher de conserver les mêmes sémantiques que les méthodes natives: retourner l'objet jQuery, appliquer la méthode à tous les éléments représentés
- à user et abuser pour factoriser du code, ou donner un nom métier à du traitement techniques (ex: `cacheContenu()` pour factoriser `$('.classe').children('.contenu').hide()`)
- beaucoup beaucoup beaucoup de plugins existent d'ores et déjà, à la qualité variable; certains sont mis en avant par l'équipe de développement

# les mains dans le cambouis

- exemple live

merci

