

# Javascript jQuery

Guy Lapalme

# Javascript

- syntaxe à la Java
- pas de typage strict
- pas de déclaration de type
- modèle à objet particulier (Prototype)
  - objet: liste de couples (attribut:valeur)

# Frameworks Javascript

- Ensemble de fonctions JS pour la programmation client
  - manipulation DOM
  - extensions diverses aux tableaux, chaînes
  - gestion uniforme des événements
  - widgets pour interface usager
- Permet de s'affranchir des particularités des browsers (y compris diverses versions)
- Utilise à fond les aspects fonctionnels de JS
  - jQuery
  - Prototype+Scriptaculo.us
  - Dojo...

liste des evenements  
pour le TP2

angular.js  
node.js  
react.js  
les frameworks plus demandes dans le marche de travail

Écrire moins du code et avoir plus de  
fonctionnalité  
c aussi la difference entre jQuery et DOM

# jQuery

<http://jquery.com>

fortement inspiré de : <http://web.stanford.edu/class/cs98si/slides/jquery.html>

- Orienté requête pour chercher et manipuler le DOM
- Sélecteurs CSS pour identifier des éléments du DOM
- Exécution de fonctions sur tous les éléments identifiés
- `$(document).ready(code de lancement)`
- Doit charger le code dans l'entête  
`<script src="//code.jquery.com/jquery-latest.min.js"></script>`

# \$

- désigne **jQuery** : seule variable globale exportée
- aussi une fonction surchargée selon son paramètre
  - sélecteur CSS : collection d'éléments DOM
  - chaîne HTML : crée un nouvel élément DOM
  - élément DOM : retourne cet élément comme une collection jQuery
  - vide : collection jQuery vide
  - collection jQuery : retourne cette collection

# Sélection

$\$(\textit{sélecteur CSS}, \textit{contexte})$

<i>*</i>	tous les éléments
<i>.class</i>	tous les éléments de cette classe
<i>élément</i>	tous les éléments de ce nom
<i>#id</i>	l'élément avec cet id
<i>sel<sub>1</sub>, sel<sub>2</sub>, ...</i>	combine tous les résultats des sélecteurs
<i>parent enfant</i>	tous les enfants d'un parent

Si *contexte* est présent, la sélection est limitée aux enfants de cet élément

# Collection jQuery

- résultat d'un appel à `$ (...)`
- semblable un `array`
  - attribut `length`
  - accès par `[n]`
    - retourne un élément DOM
    - pas un objet jQuery
- donne accès à une foule de fonctions
  - `.css()`, `.text()`, `.html()`, `.each()`, ...
  - `.add()`, `.filter()`, `.not()` // comme des ensembles
- retourne la collection modifiée
- *convention*: débiter nom de variable par `$`

# objet jQuery vs élément DOM

```
var $toto = $( "#toto" )
```

```
var toto = document.getElementById( "toto" )
```

```
var $bleus = $( ".bleu" )
```

```
var bleu = document.getElementsByClassName( "bleu" )
```

```
var $toto = $( toto )
```

```
var toto = $toto.get( 0 ) // ou $toto[ 0 ]
```



Key	valeur
-----	--------

--	--

# Manipulation éléments DOM



- `.attr(nom attribut [, valeur])`
  - ▀ obtenir la valeur ou changer la valeur d'un attribut
- `.css(nom propriété, valeur)`
  - ▀ obtenir la valeur ou changer la valeur d'une propriété de style CSS
- `.addClass/.removeClass/.toggleClass(noms de classes)`
  - ▀ ajouter/enlever ou basculer la présence des noms de classes séparés par des blancs dans une chaîne
- `.hide()/ .show()`
  - ▀ cacher ou afficher des éléments
- `.html(...)` / `.text(...)`
  - ▀ changer le contenu HTML ou texte d'un élément

# Manipulation d'éléments DOM

- `.append(élément)`, `.prepend(élément)`
  - ▀ ajouter l'élément comme dernier/premier enfant
- `.after(élément)`, `.before(élément)`
  - ▀ ajouter l'élément après / avant comme frère
- `.remove()`
  - ▀ enlever un élément pour le remettre ailleurs

Comme pour le DOM, un élément existant ajouté est enlevé de son endroit précédent

# Traitement des événements

- `.on( "...", f(ev) )` ajoute un listener
- `.off( "...", f(ev) )` enlève un listener
- `.click ( f(ev) )` ajoute un click listener
- `.keypress ( f(ev) )` ajoute un key listener
- `.fadeIn(...)` / `.fadeOut(...)`  
apparition graduelle
- `.slideDown(...)` / `.slideUp(...)`  
apparition par déplacement