jQuery

Mohamed DERKAOUI

Objectifs

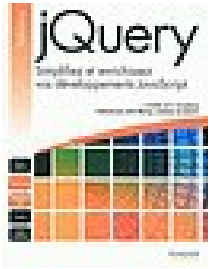
- Construire des interfaces performantes avec jQuery et développer des plugins additionnels

Bibliographie

Premières applications Web avec Ajax, jQuery et PHP
40 ateliers pour réaliser des moteurs Ajax-PHP et les plug-ins jQuery
Jean-Marie Defrance – Eyrolles - Janvier 2010

jQuery - Le framework JavaScript du Web 2.0
Luc Van Lancker - Eni
Décembre 2009

jQuery
Simplifiez et enrichissez vos développements Javascript
Jonathan Chaffer et Jonathan Swedberg
Pearson Education - Octobre 2009



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

Plan

- Introduction
- Rappel CSS : mise en forme et positionnement
- Rappel JavaScript / DOM
- Interfaces utilisateurs grâce à jQuery
- AJAX avec jQuery
- Plugins et augmentation de jQuery

Introduction

Introduction

- Standards du web : XHTML / CSS / XML
- JavaScript / DOM
- Frameworks JavaScript
- jQuery

Rappel CSS

Introduction

- **CSS =** Feuilles de styles en cascade

Langage permettant de définir la mise en forme/page d'éléments

- **Intérêts :**

- Séparer l'interface de l'information
- Simplifier la maintenance des pages web
- Alléger le poids de la page HTML
- Réduction du trafic (chargement unique)

- **Utilisation :**

- Feuille de style externe (fichier.css)
- Feuille de style interne <style> (dans l'entête <head>)
- Style « en ligne » (dans l'élément HTML)

Utilisation : attribut style

```
<!-- ... -->
 <body style="color: green;">
  <p>
   <a href="index.html" style="color: red;">Home</a>
   <a href="index.html">Home</a>
   Home
  </p>
 </body>
</html>
```

Utilisation: feuille de styles interne

```
<!-- ... -->
 <head>
  <!-- ... -->
   <style type="text/css">
    body { color: green; }
    a { color: red; }
   </style>
 </head>
 <body>
  <p>
   <a href="index.html" style="color: blue;">Home</a>
   <a href="index.html">Home</a>
   Home
  </p>
 </body>
</html>
```

Utilisation: feuille de styles externe

- Un fichier .css séparé
- Le lien dans la page HTML avec la balise <link> :

```
<!-- ... -->
  <head>
    <!-- ... -->
    <link type="text/css" rel="stylesheet" href="myStyle.css"
      [title="default style"]
      [media="support"]  />
  </head>
  <body>
    <!- -
      Code HTML affecté par les règles de styles CSS
    -->
  </body>
</html>
```

Règles de styles

- Une feuille de style est un ensemble de règle de styles.
- Syntaxe : sélecteur {
  propriété: valeur;
  …
  }
- Exemple : (/* ceci est un commentaire css */)

```
p {
    background-color: white;
    color: red;
    font-family: verdana, "sans-serif";
}
```

- On peut regrouper plusieurs règles css : sel1,sel2
- Les styles sont appliqués dans l'ordre de leur définition.
- En cas de conflits, les propriétés définies plus bas « écrasent » les précédentes.

Types de règles CSS

Il existe plusieurs types de règles de styles :

- Règles de type balise
- Règles de type classe
- Règles de type id
- Règles de type pseudo-classes
- …

N.B. : on présente uniquement les principaux types de règles CSS dans ce document.

Vous pouvez cependant jeter un coup d'oeil sur les autres à cette adresse : http://www.yoyodesign.org/doc/w3c/css2/selector.html

Règles de type balise

- Permet de redéfinir le style d'une balise :

```
nomBalise {
...
}
```

- Exemple :

```
h1 {
        background-color: white;
        color: red;
}
```

N.B. :

E : Tous les éléments E

E F : Tout élément F descendant de E

Règles de type classe

- Définir un style applicable à un ensemble défini de balises : .nomClasse{
  ...
  }
- CSS :

```
.section {
    color: red;
}
```

- HTML : appliquer le style à l'élément (attribut class)

```
<!-- ... -->
 <body>
  <h1 class="section">un titre</h1>
  <h1>un autre titre</h1>
 <body>
</html>
```

Seul le premier h1 est en rouge

Règle de type id

- Permet d'appliquer un style à UN SEUL et UNIQUE objet.
- Un id permet d'identifier UN objet

```
#unId{
   …
}
```

- CSS :

```
#MyObjectId {
    background-color: red;
}
```

- HTML : appliquer le style à l'élément (attribut id)

```
<!-- … -->
 <body>
  <div id="MyObjectId">un contenu</div>
 <body>
</html>
```

Règles de type pseudo-classes

- Elles permettent de définir un style applicable, suite à un événement ou bien, à la position relative de la balise parmi d'autres balises.
- Parmi elles, on retrouve notamment les pseudo-classes applicables à des liens :

:hover définit le style d'un lien survolé

a:hover {text-decoration: underline;}

:active définit le style d'un lien cliqué

:link définit le style d'un lien non encore visité

:visited définit le style d'un lien déjà visité

Supports

- @media : pouvoir spécifier comment représenter un document pour différents médias : un écran, une feuille de papier, un synthétiseur de parole, un appareil braille, etc.
- Exemple :

```
@media print {
    p {
        font-family: serif;
        font-size: 12pt;
    }
}
```

- Attribut *media* de la balise <link>
- Supports possibles : all, aural, braille, embossed, handheld, print, projection, screen, tty, tv

Positionnement CSS

- Il existe plusieurs types de positionnement :

- positionnement relatif (en flux)
- positionnement absolu
- positionnement fixe
- positionnement flottant

Positionnement relatif

Les balises sont positionnées sur la page selon :

- leur ordre dans le code
- le type d'élément (en-ligne ou bloc)
- les marges internes et externes
- la propriété css **position:relative** associée aux valeurs au choix top, left, bottom & right

Positionnement fixe

- Semblable au positionnement absolu
- Fixe même si vous utilisez les barres de défilement.
- Ce positionnement n'est pas géré par IE6 et n'est pas recommandé.
- Propriétés css :

position:fixed;

top:valeur;

left:valeur;

Positionnement flottant

- Permet de faire flotter des éléments pour les positionner côte à côte
- Flottement à gauche ou à droite
- Propriétés CSS :

float : left; ou float:right;

- Effacer le flottement : propriété clear

left, right ou both

Hiérarchie des styles

- Ordre de priorité :

  style en ligne > style de document > @import > link

- Différence entre @import et link :

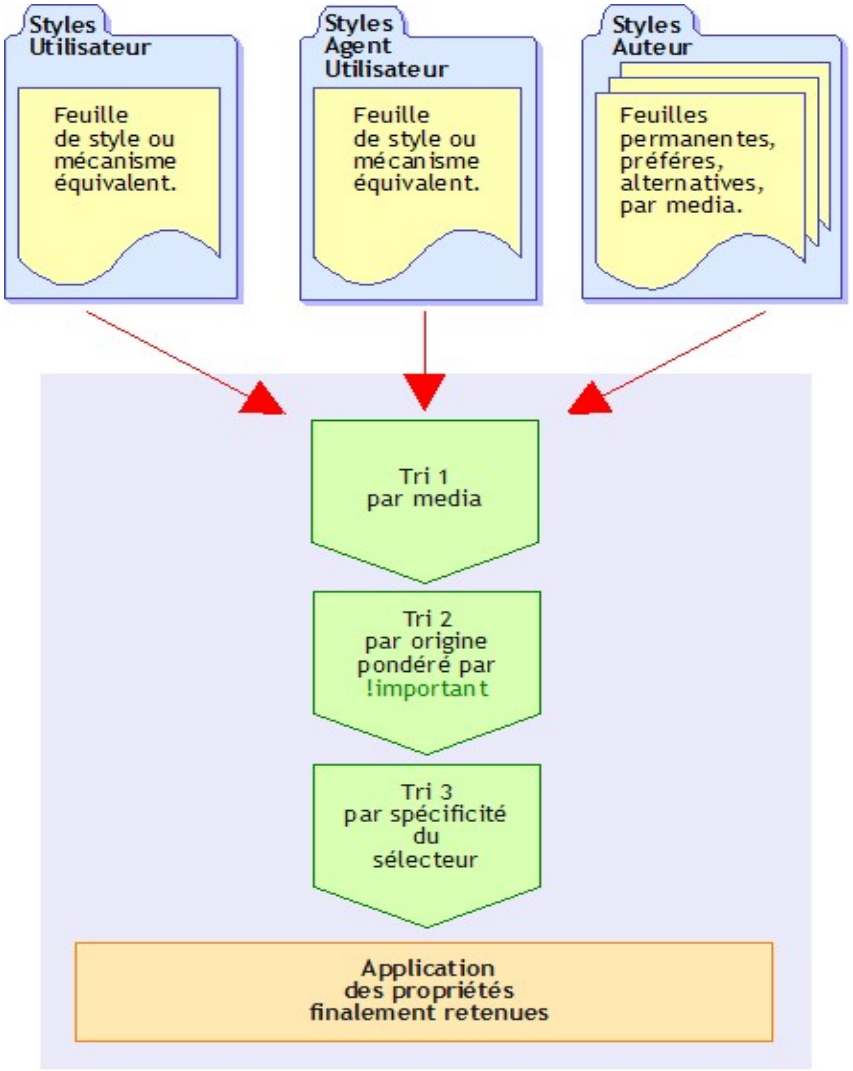
  - Équivalence lorsqu'une seule balise <link>

  - Lorsqu'il y a plusieurs feuilles externes, @import est préférable, car il y a fusion des styles. La feuille du dernier import est prioritaire

Cascade des styles

- Les feuilles de style peuvent provenir de : l'auteur, l'utilisateur et l'agent utilisateur (le navigateur)
- La cascade de CSS définit un ordre de priorité, ou poids, pour chaque règle de style. Quand plusieurs règles sont mises en œuvre, celle avec le plus grand poids a la préséance

Ordre de priorité



Ordre de priorité (2)

- **Tri par media**
- **Tri par origine** :

  1 - styles par défaut de l'agent utilisateur, qui seront écrasés par tous les styles suivants

  2 - styles normaux de l'utilisateur (c'est à dire qui n'ont pas été marqués !important)

  3 - styles normaux de l'auteur (idem)

  4 - styles de l'auteur marqués !important

  5 - styles de l'utilisateur (navigateur) marqués !important, qui écraseront tous les styles précédents

Ordre de priorité (3)

- **Tri par priorité calculée des sélecteurs :** Les styles restant en concurrence ont un degré de priorité variable, dépendant du sélecteur CSS utilisé et de sa syntaxe. Degré = nombre à 4 chiffres **ABCD** :

  A : A=1 s'il ne s'agit pas d'un sélecteur CSS mais d'un attribut style. Sinon, A=0. De cette manière, un style placé dans l'attribut style d'un élément HTML aura systématiquement un poids plus élevé qu'un style auteur placé dans une feuille de style (ABCD = 1000, la plus haute valeur possible) ;

  B : si A vaut 0, B est le nombre d'ids présents dans le sélecteur. Par exemple, B=2 pour le style #conteneur p#special {...} ;

  C : si A vaut 0, C est le nombre de classes (du type .ma_classe) et de pseudo-classes (du type [hreflang=en]) dans le sélecteur. Par exemple, C=2 pour le style .article p a[hreflang=en] {...} ;

  D : si A vaut 0, D est le nombre d'éléments HTML et de pseudo-éléments (du type :before et :after, :first-line, etc) dans le sélecteur. Par exemple, : D=2 pour le style a:after {...}.

Ordre de priorité (4)

- *** {...}** : 0000 (aucun identifiant, aucune classe, aucun élément) ;
- **p {...}** : 0001 (aucun identifiant, aucune classe, un élément) ;
- blockquote p {...} : 0002 (aucun identifiant, aucune classe, deux éléments) ;
- **.class {...}** : 0010 (aucun identifiant, une classe, aucun élément) ;
- **#id {...}** : 0100 (un identifiant, aucune classe, aucun élément) ;
- **blockquote.class p#id {...}** : 0112 (un identifiant, une classe, deux éléments) ;
- **<p style="...">** : 1000 (attribut HTML style qui ne sera supplanté que par un style utilisateur normal) ;
- **<p style="... !important">** : 1000 (attribut HTML style marqué !important qui ne sera supplanté que par un style utilisateur lui-même marqué !important).

- Si après le tri par priorité calculée deux styles de même degré de priorité restent en concurrence, le dernier apparu dans l'ordre linéaire CSS-HTML l'emporte.

Rappel JavaScript

Présentation du langage

- JavaScript = langage de scripting interprété côté client
- Créé par Netscape en 1995
- Implémentation de la norme ECMAScript (comme JScript, ActionScript,...)
- Permet d'ajouter de l'interactivité utilisateur/pages web

Intégration du code

- Le code JavaScript peut être placé :

- dans des balises <script></script>

```
<script type="text/JavaScript">
    code
</script>
```

- dans un fichier externe :

```
<script type="text/JavaScript" src="nomFichier.js"></script>
```

- Des fonctions peuvent être déclenchées grâce aux évènements associés à chaque élément HTML

Syntaxe de base

- Commentaires :

```
// une ligne
/* plusieurs lignes */
```

- Variables : zones mémoire stockant des valeurs

```
maVariable = 5;
Var MaVariable;
```

- Sensible à la casse (respect des majuscules et minuscules)

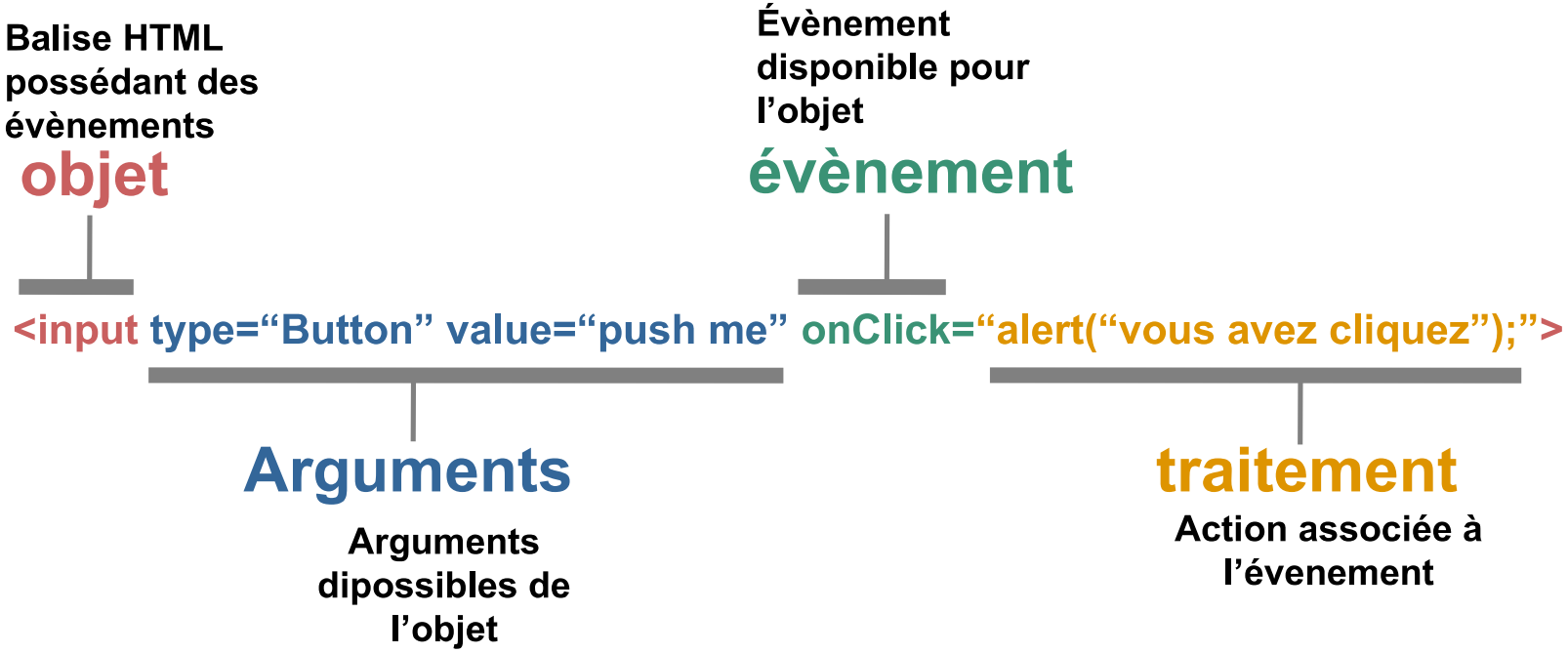
```
maVariable ≠ MaVariable;
```

- Typage faible

- Opérateurs

Évènements

- Déclenchent un traitement en fonction des actions de l'utilisateur (click, mouvement de souris, etc.)
- Traités par des gestionnaires d'événements (event Handler)
- Utilisation :



Structures de contrôle

Structures conditionnelles

- if
- switch

Structures itératives

- while
- do ... while
- for
- for ... in

Fonctions

- Ensemble d'instructions réalisant une certaine tâche
- Peuvent prendre 0 ou plusieurs paramètres
- Peuvent renvoyer un résultat
- Permettent la création de code réutilisable
- Permettent de structurer le code et gagner en clarté

```
function nomFonction(parametres){
        //return valeur;
}
```

- Appel : `nomFonction(arguments);`
- Appel dans un événement :

```
<input type="button" onclick="nomFonction();" >
```

DOM

Document Object Model

Standard W3C

- Ensemble standardisé d'objets pour HTML
- Moyens standardisés pour accéder et manipuler des documents HTML

- Indépendant du langage et de la plateforme
- Permet la création, la suppression et la modification de :
  - tous les éléments HTML
  - leurs attributs
  - le texte qu'ils contiennent

Manipulation du document

- **Accès via l'ID de l'élément**
  Retourne un élément unique
  document.getElementById(*id*);

- **Accès via le nom des éléments**
  Retourne une collection d'objets avec le même nom
  document.getElementsByName(*name*);

- **Accès via un nom de balise**
  Retourne une collection d'objets de la même balise
  document.getElementsByTagName(*name*);

Manipulation du document

- **Accès à tous les fils d'un élément**

  element.childNodes;

- **Accès à l'élément parent**

  element.parentNode;

- **Parcourir l'arborescence**

  document.childNodes[0].childNodes[1];

Accès aux attributs

- Accéder aux attributs d'un élément
  element.getAttribute("attribut");
- Modifier les attributs d'un élément:
  element.setAttribute("attribut", "valeur");
- Accéder au texte d'un élément
  element.firstChild.nodeValue
- Modifier le texte d'un élément
  element.firstChild.nodeValue="texte";

Manipulation des éléments

- Création d'un élément

```
var e = document.createElement('p');
```

- Modifier un élément

```
e.style.textAlign = 'center';
```

- Ajouter l'élément au parent

```
parent.appendChild(e);
```

- Ajouter l'élément avant

```
element.parentNode.insertBefore(new_element,this);
```

- Supprimer un élément

```
var e = document.getElementById('deleteMe');
e.parentNode.removeChild(e);
```

L'Objet en JavaScript

Création d'un objet

Notion d'Object :

```
var p1 = new Object();
p1.prenom="Jacques";
p1.nom="DUPONT";
```

JSON (JavaScript Object Notation) :

```
var p2 = {
    prenom:"Jacques",
    nom:"DUPONT"
}
```

N.B. : p1.nom est équivalent à :

```
p1["nom"]
```

Associer une méthode

Notion d'Object :

```
var p1 = new Object();
.....
p1.affich = function(){
  return this.nom+" "+this.prenom;
}
```

JSON :

```
var p3 ={
  prenom:"Jacques",
  nom:"DUPONT",
  afficher : function(){
    return this.nom+" "+this.prenom;
  }
}
```

Support de l'objet en Javascript

Classe	Support partiel, aucun élément élément de langage dédié. Mais possibilité d'utilisation via les fonctions/prototypes
Composition/ agrégation	Ces deux mécanismes sont supportés
Encapsulation/ visibilité	Supporté (JavaScript 1.2)
Héritage	JS permet de faire de l'héritage mais il n'existe pas de mot-clé extends prévu à cet effet.
Mot-clé super	Non supporté
Mot-clé this	*this* est supporté il fait référence à l'objet en cours sur lequel la méthode s'applique
Polymorphisme	Concept non supporté
Typage	JS possédant un typage dynamique, le type d'un objet n'est connu que lors de son exécution et peut également varier.

Agrégation

Notion d'Object :

```
var p1 = new Object();

.....

p1.adresse = new Object();

p1.adresse.rue="Arrivee";

p1.adresse.cp=75015;

p1.affich = function(){

   return this.prenom+" "+this.nom;
}
```

JSON :

```
var p2 ={
   prenom:"Jacques",
   nom:"DUPONT",
  adresse:{
    rue:"arrivee",
    cp:75015
  },
  afficher : function(){
    return this.prenom+" "+this.nom;
  }
}
```

Constructeur

```
function Personne(prenom,nom){
   this.prenom=prenom;
   this.nom=nom;
   this.toString=function(){
      return this.prenom+" "+this.nom;
   }
}

var p4 = new Personne("Mohamed","DERKAOUI");
```

N.B.: destruction avec : `delete p4;`

Propriété : prototype

Propriété associée à un objet pour modifier sa structure interne

- Ajout d'une méthode

```
Personne.prototype.afficher = function(){
        return (this.prenom + "  " + this.nom);
};
```

- Ajout d'une constante :

```
Personne.prototype.coef = 150;
```

NB. : on peut afficher les propriétés et méthodes d'un objet grâce à une boucle « for/in »

Encapsulation

- Possibilité de définir des propriétés privées

```
function monObjet(param1,param2){
   var prop1 = null; //propriété privée
   setProp1(param1); //Pointeurs vers des méthodes
publiques (Getters/Setters)

   this.getProp1 = getProp1;
   this.setProp1 = setProp1;
   function getProp1(){ //getter
      return prop1;
   }
   function setProp1(parameter){ //setter
      prop1 = parameter;
   }
}
```

Héritage

L'héritage se fait par affectation du prototype :

- L'objet "dérive" d'un objet créateur
- Les attributs et méthodes du créateur sont ajoutés au prototype de l'objet

```
function Employe(prenom, nom, poste){
        this.base=Personne;
        this.base(prenom, nom);
        this.poste=poste;
}
Employe.prototype=new Personne;

emp1 = new Employe("jacques","DUPONT","développeur-Dawan");
```

Héritage (2)

- Appel d'un constructeur de la classe mère : apply()
- Redéfinition de méthode

```
function Cadre(prenom,nom,fonction){
  Personne.apply(this,[prenom,nom]);
  this.fonction=fonction;

  this.toString=function(){
    return this.nom+" "+this.prenom+ " "+this.fonction;
  }
}
```

/* Instanciation */

```
var cadre1 = new
Cadre("prenomCadre1","nomCadre1","Formateur");
```

Objets JavaScript

Il existe trois classes d'objets JavaScript :

- les objets intrinsèques

  (Object, Array, Date, Math, String, ...)
- les objets du navigateur

  (window, document, forms, ...)
- les objets créés par script (Personne, Cadre,...)

Frameworks JavaScript

- Prototype
- JQuery
- Dojo
- Ext
- MooTools
- Rico
- ScriptAculous
- ...

jQuery

jQuery

- Bibliothèque de fonctions JavaScript
  (John Resig - 2006)
- S'affranchit des tâches rébarbatives et répétitives de façon uniforme sur les navigateurs les plus courants
- Sous licence GPL et MIT
- Légère
- Compatible avec les autres frameworks JS

Exemple de sites l'utilisant : Dell, Google Code, Digg, NBC, Amazon, Mozilla, Wordpress, SPIP, ...

Fonctionnalités

- Parcours et modification du DOM (y compris le support des CSS 1 à 3 et un support basique de XPath)
- Événements
- Effets et animations
- Manipulations des feuilles de style en cascade (ajout/suppression des classes, d'attributs…)
- AJAX
- Plugins

Navigateurs supportés :
Firefox 1.5+ - Internet Explorer 6+ - Safari 2.0.2+ - Opera 9+ - Google Chrome ...

Utilisations

- Sélecteurs (DOM, CSS, Complexes, Filtres,...)
- Associer des évènements
- Requêtes Ajax
- Parser du XML
- ...

Inconvénients et Alternatives

Inconvénients :

- moins de contrôle sur le code produit
- projet porté par une communauté
- incompatibilités avec les plugins

Alternatives :

- Nombre de bibliothèques : Ext JS, Yahoo! User Interface, Prototype, Dojo, MooTools, Mochikit, base2,...

Mise en place

Téléchargement : http://jquery.com/

PRODUCTION (24KB, Minified « JSMin » and Gzipped)

DEVELOPMENT (155KB, Uncompressed Code)

```
<script type="text/javascript" src="js/jquery.js">
</script>
<script type="text/javascript">
   $(document).ready(function(){
      //Appel de vos fonctions
   });
</script>
```

Syntaxe raccourcie :

```
$(function(){
      //Appel de vos fonctions
});
```

Sélecteurs

```
$("id/class/element")
```

- $('*') : sélectionne tous les éléments
- $('#monDiv') : sélectionne l'élément ayant l'ID "monDiv"
- $('p.first') : sélectionne les éléments <p> ayant la classe "first"
- $('ul, ol, dl') : sélectionne les éléments <ul>, <ol> et <dl>
- $('div .desc') : sélectionne les éléments ayant la classe "desc" descendants (au sens CSS) d'éléments
- $('div > .enfant') : sélectionne les éléments ayant la classe "enfant" enfants d'éléments

Sélecteurs (2)

- $('label + input') : sélectionne les <input /> dont l'élément précédent (dans le DOM) est <label>
- $('#debut ~ div') : sélectionne les frères se situant après l'élément dont l'id est "debut"
- $('p[title]') : sélectionne les <p> ayant un attribut "title"
- $('p[title="Bonjour"]') : sélectionne les <p> dont l'attribut title est "Bonjour"
- $('p[title!="Bonjour"]') : sélectionne les <p> dont l'attribut title n'est pas "Bonjour"
- $('p[title^="H"]') : sélectionne les éléments dont l'attribut title commence par "H"
- $('p[title$="H"]') : sélectionne les éléments dont l'attribut title fini par "H"
- $('p[title*="H"]') : sélectionne les éléments dont l'attribut title contient "H".

Filtres

- $('div:first') : sélectionne le premier
- $('div:last') : sélectionne le dernier
- $('div:not(.ok)') : sélectionne les n'ayant pas la classe "ok"
- $('div:[even|odd]') : sélectionne les éléments de rang [pair|impair] (le premier rang est 0)
- $('div:[eq|lt|gt](n)') : sélectionne le ou les éléments de rang [égal|inférieur|supérieur] à n
- $(':header') : sélectionne les éléments (h1, h2, ... h6)
- $(':animated') : sélectionne les éléments actuellement animés
- $("div:contains('dvp')") : sélectionne les contenant le texte "dvp" (sensible à la casse !)

Filtres (2)

- $('div:parent') : sélectionne les ayant des enfants (y compris les nœuds texte)
- $('div:nth-child([n|even|odd|equation])') :

  les enfants de [de rang n|pairs|impairs|résultat de]
- $('div:[first-child|last-child]') // Les éléments [premier|dernier] enfants d'un
- $('div:only-child') : les éléments qui sont les seuls enfants d'un élément
- $('div:empty') : sélectionne les vides
- $('div:has(p)') : sélectionne les ayant un descendant <p>

Sélecteurs et évènements

- $("id/class/element").fonction("options");

```
$(".boite").fadeOut("slow");
```

- $("id/class/element").fonction("options", callback);

```
$("#boite2").fadeOut("slow", function(){
        $("#boite2").fadeIn("30");
});
```

Fonctions utilitaires

- length et size : retournent le nombre d'éléments d'un objet
- each : boucler sur une liste d'éléments
- Browser : infos sur le navigateur (nom, version)
- Trim : suppression des espaces

Effets - animations

- show / hide
- slideUp / slideDown / slideToggle
- fadeTo
- animate

AJAX

Asynchronous Javascript and Xml

Terme représentant un ensemble de technologies combinées pour le développement d'applications web client riche.

- Inventé en 2005 : Jesse J. Garrett
- Mise à jour d'une partie de la page
- Communication asynchrone avec le serveur

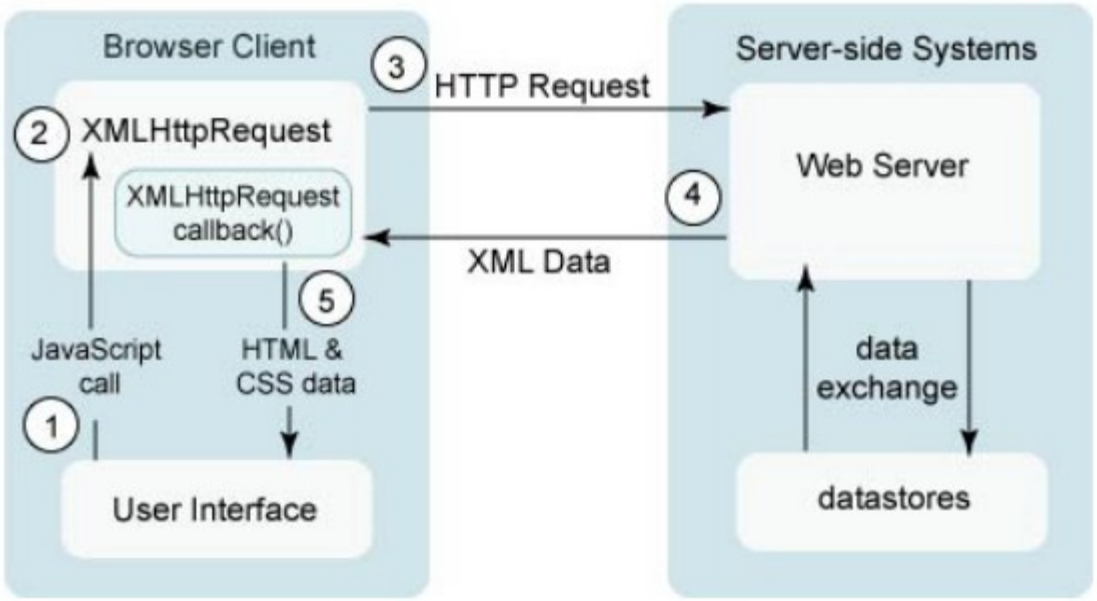
Utilisation :

- Rafraichissement partiel de pages
- Suggestions de saisies
- applications documentaires
- ...

L'objet XmlHttpRequest

Objet JavaScript qui permet d'émettre une requête, de l'annuler (c'est parfois utile) et de spécifier le traitement à effectuer à la réception de sa réponse

Disponible depuis 1998 dans Microsoft Internet Explorer, il l'est désormais dans tous les navigateurs récents



Instancier un objet XmlHttpRequest

```
function getRequest(){

 var xhr;
 if(window.XMLHttpRequest || window.ActiveXObject) {
        if(window.XMLHttpRequest) {
                xhr = new XMLHttpRequest();
        }
        else { // Internet Explorer <7
                try {
                        xhr = new ActiveXObject("Msxml2.XMLHTTP");
                } catch(e) {
                        xhr = new ActiveXObject("Microsoft.XMLHTTP");
                }
        }
 }
 else {
         alert("Votre navigateur ne supporte pas l'objet XMLHTTPRequest...");
         return;
 }
return xhr;
}
```

Envoi de la requête

La propriété onreadystatechange de l'objet XmlHttpRequest permet de définir quelle action sera exécutée quand le serveur aura répondu à cette requête :

```
xhr.onreadystatechange = fonctionDeCallback;
```

Envoi de la requête :

```
xhr.open("GET", "pageDeTraitement.jsp", true);
xhr.send(null);
```

Si vous utilisez la méthode POST, vous devez absolument changer le type MIME de la requête avec la méthode setRequestHeader, sinon le serveur ignorera la requête :

```
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send("sennomparam1=valeurparam1&nomparam2=valeurparam2");
```

Réponse du serveur

Etat de la requête : propriété readyState. 5 états différents :

0 : L'objet XHR a été créé mais pas encore initialisé (la méthode open n'a pas encore été appelée)

1 : L'objet XHR a été créé, mais pas encore envoyé (avec la méthode send)

2 : La méthode send vient d'être appelée

3 : Le serveur traite les informations et a commencé à renvoyer des données

4 : Le serveur a fini son travail, et toutes les données sont réceptionnées

```
xhr.onreadystatechange = function() {
        if(xhr.readyState == 4 && (xhr.status == 200 || xhr.status == 0)) {
                alert("OK"); // On va s'occuper des données reçues
        }
}
```

jQuery - Ajax

$.ajax()

```
$.ajax({
    type: "POST",
    url: "test.html",
    success: function(retour){
        alert("Données retournées : " + retour );
    }
});
```

Paramètres de $.ajax() :

datatype – url – type – dataType – ifModified – timeout – global – error – success - complete – data – contentType – processData – async - beforeSend

Fonctions Ajax

- **$.post("test.php");**
  Effectue un requête en POST sur la page "test.php"
- **$.get("test.php");**
  Effectue un requête en GET sur la page "test.php"
- **$.getIfModified("test.php");**
  Effectue un requête en GET sur la page "test.php" si elle a changé
- **$.getJSON("test.js", function(json){alert(json);});**
  Effectue un requête en JSON sur le script "test.js"
- **$.getScript("test.js");**
  Va cherche le script "test.js"

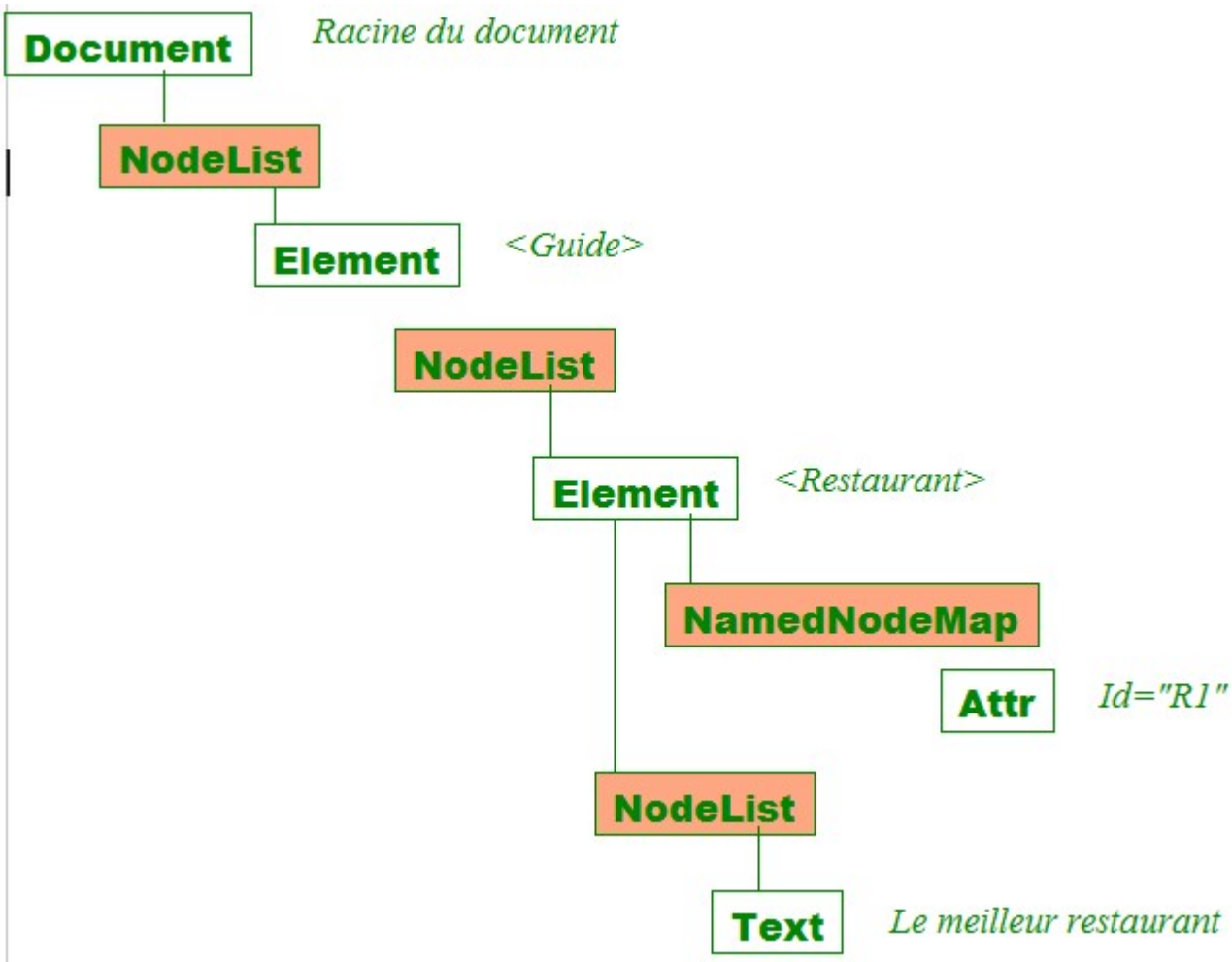
Fonctions Ajax solitaires

- ajaxComplete()
- ajaxError()
- ajaxSend()
- ajaxStart()
- ajaxStop()
- ajaxSuccess()
- load()
- loadIfModified()

- multiples cas d'utilisation

Arbre DOM / Parsing XML

```
<Guide>
    <Restaurant id="R1">Le meilleur restaurant</Restaurant>
</Guide>
```



Plugins

- Extensions de jQuery
- Liste des plugins : http://docs.jquery.com/Plugins
- Intégration
- Exemples d'utilisation de plugins
- Création

Plus d'informations sur http://www.dawan.fr
Contactez notre service commercial au **0810.001.917** (prix d'un appel local)

DAWAN Paris, Tour CIT Montparnasse - 3,rue de l'Arrivée, 75015 PARIS
DAWAN Nantes, Le Sillon de Bretagne - 26e étage - 8, avenue des Thébaudières, 44800 ST-HERBLAIN
DAWAN Lyon, Le Britannia, 4ème étage - 20, boulevard Eugène Deruelle, 69003 LYON
DAWAN Lille, Parc du Chateau Rouge - 4ème étage, 276 avenue de la Marne, 59700 MARCQ-EN-BAROEUL
formation@dawan.fr