

Introduction

Ajax
JSON
jQuery

SOMMAIRE

- 1 Introduction à AJAX, JSON et jQuery
- 2 AJAX Asynchronous JavaScript And XML
- 3 Fonctionnement du Web «Classique»
- 5 XMLHttpRequest
- 6 Application Web utilisant AJAX
- 7 Les différences
- 9 Exemples : Google Search et Google Map
- 10 Exemple : une simple somme
- 13 Exemple : une simple somme «classique»
- 14 Exemple : une simple somme AJAX
- 17 AJAX : Avantages et inconvénients
- 18 AJAX : Le bouton précédent
- 19 AJAX : Fonctionnement
- 20 AJAX : Objet XMLHttpRequest
- 21 AJAX : Requête synchrone
- 22 AJAX : Les types de réponse
- 23 AJAX : Réponse DOM/XML
- 29 AJAX : Réponse XML/XSLT
- 30 AJAX : Réponse XML/XSLT
- 33 AJAX : Réponse Texte/HTML
- 36 AJAX : Pourquoi «Ne jamais faire ainsi» ?
- 37 AJAX : XMLHttpRequest Référence
- 39 JSON JavaScript Object Notation
- 40 JSON : Objets et tableaux littéraux
- 41 JSON : La fonction eval()
- 43 JSON : L'objet JSON
- 46 JSON : L'objet JSON
- 47 JSON : format texte pour AJAX
- 48 JSON : PHP
- 49 JSON : les autres langages
- 52 jQuery Framework JavaScript
- 53 Frameworks JavaScript : introduction
- 55 jQuery
- 56 jQuery : sélection d'objets DOM
- 71 jQuery : gestion des événements
- 78 jQuery : modifications CSS et effets
- 85 jQuery : manipulations DOM
- 95 jQuery : AJAX
- 107 jQuery : Formulaires

Introduction à AJAX, JSON et jQuery



Introduction à Ajax, JSON et aux frameworks JavaScript - 3iL - William Ruchaud - Version 2.3

AJAX Asynchronous JavaScript And XML

Introduction à Ajax, JSON et aux frameworks JavaScript - 3iL - William Ruchaud - Version 2.3

Fonctionnement du Web «Classique»

Principe de non conservation d'état :

- Un principe fondamental du Web depuis l'origine.
- Le navigateur ne conserve aucune donnée des pages consultées hormis l'historique (en théorie).
- Toute requête au serveur se traduit par l'**effacement** document actuel et le **chargement** d'un nouveau document.

Applications Web, interfaces et données :

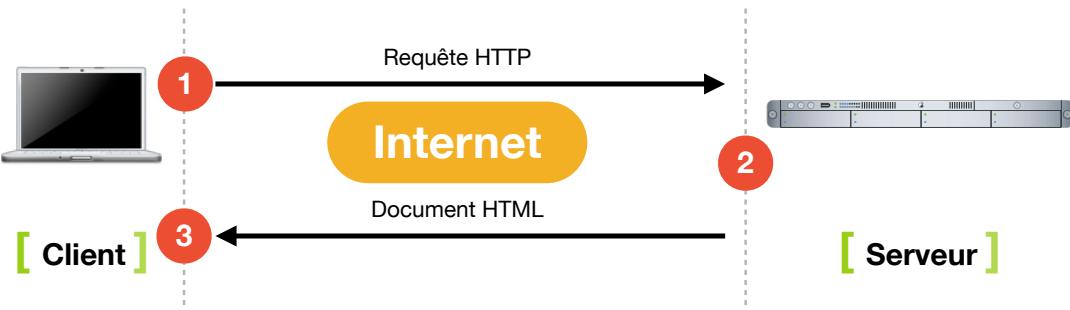
- Pour une application Web (exemple un vieux webmail), l'interface fait partie du document HTML.
- Une action simple (exemple : la suppression d'un mail) entraîne le chargement **integral** de la page actualisée donc de l'interface **qui n'a pas changé**, seules les données ont changé.
- Remarque : le cache du navigateur travaille à l'échelle des fichiers, mais pas au niveau du contenu de ceux-ci.

Requêtes HTTP et navigation synchrone :

- Requête HTTP principalement déclenchées par :
 - L'interface de navigation (boutons précédents et suivant, l'historique)
 - La saisie d'une URL dans la barre d'adresse.
 - Le clic sur un lien.
 - L'envoie d'un formulaire.
- C'est le navigateur qui prend en charge les requêtes HTTP.
- On parle de **navigation synchrone** : l'utilisateur est mis en attente durant le chargement.

Fonctionnement du Web «Classique»

Fonctionnement du Web “classique” :



1 Événement navigateur

- Déclenché par :
 - Clic sur un lien
 - Envoi de formulaire
 - Saisie d'URL
- Mise en suspend de l'utilisateur en attendant la réponse du serveur

2 Traitement serveur

- Recherche du fichier demandé.
- Si nécessaire exécution du script «côté serveur»
- **Retourne un document HTML**

3 Interprétation document HTML

- Destinataire : le navigateur
- Effacement du document précédent à la réception du nouveau.
- Mise en place du HTML.
- Lancement des «sous requêtes» pour les autres fichiers (CSS, images, JavaScript)

XMLHttpRequest

Le cadeau de Microsoft :

- 2001 : Microsoft sort Internet Explorer 5 avec MSXML et un objet ActiveX (extension native pour Internet Explorer) nommé XMLHttpRequest capable d'effectuer une requête HTTP en JScript ou VBScript.
- Dans un premier temps XMLHttpRequest est passé pratiquement inaperçu.
- Comportement particulier d'Outlook Web Access (composant webmail d'Exchange Server) qui ne rechargeait pas l'intégralité des pages à chaque action utilisateur.

Le décollage :

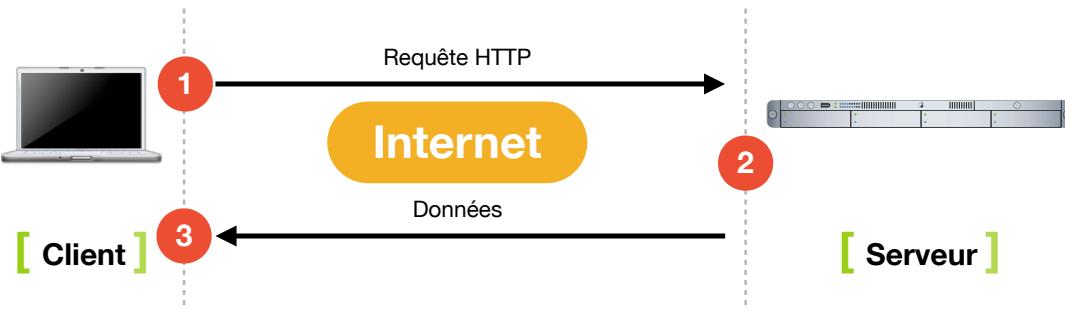
- Intégration dans les autres navigateurs d'un objet JavaScript XMLHttpRequest copiant les fonctionnalités de l'objet XMLHttpRequest.
- En 2004 Google utilise XMLHttpRequest dans GMail, puis en 2005 dans Google Maps (les images des cartes se chargent suivant les déplacements).
- En 2005 : Jesse James Garrett lance le terme AJAX dans son article «AJAX : une nouvelle approche pour les applications Web».
- AJAX : Asynchronous JavaScript And XML
- En 2006 le W3C propose un brouillon pour normaliser XMLHttpRequest.
- À partir de la version 7, Internet Explorer adopte XMLHttpRequest.

Pourtant :

- AJAX ce n'est pas une technologie, ni un langage.
- C'est juste façon différente de concevoir une application Web, surtout au niveau des échanges avec le serveur.

Application Web utilisant AJAX

Fonctionnement d'AJAX :



1 Événement d'application

- Pratiquement n'importe quel type d'événement
 - Clic sur n'importe quel élément
 - Mouvement de la souris
 - Clavier
- Création d'un objet XMLHttpRequest
- Envoi de la requête

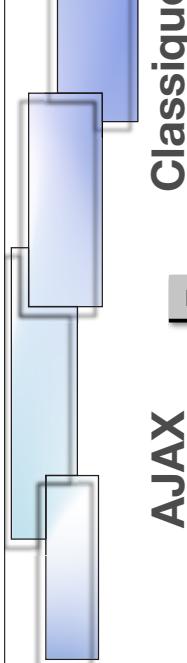
2 Traitement serveur

- Recherche du fichier demandé.
- Si nécessaire exécution du script «côté serveur»
- **Ne retourne que des données.**
- Peut retourner :
 - Un document XML
 - Du texte

3 Réception des données

- Destinataire : une application qui s'exécute sur le navigateur.
- Traitement à la réception des données.
- Mise à jour du contenu de la page.

Les différences



Subtiles mais réelles :

Événement navigateur



1

Requête HTTP

2

Internet

Classique

3

[Client]

[Serveur]

Document HTML

Le navigateur interprète le document HTML

3

Document HTML VS

Données XML ou texte

Événement d'application



1

Requête HTTP

2

Internet

AJAX

[Client]

[Serveur]

Données

L'application réceptionne et modifie l'affichage actuel

Les différences

Plus concrètement :

	Application Web «Classique»	Application Web AJAX
Localisation de l'application	Sur le serveur	1 partie serveur + 1 partie client (JavaScript)
Prise en charge des requêtes HTTP	Le navigateur	Partie cliente (principalement) + Navigateur
Résultat d'une requête HTTP	Document HTML complet (et ses dépendances)	Document HTML complet ou données XML ou texte

Principe :

- Dissociation entre la réponse à une action utilisateur et les interactions avec le serveur.
 - L'application peut répondre visuellement à une action sans attendre la réponse du serveur.
 - Les échanges avec le serveur n'ont pas nécessairement d'implication visuelle.
- Rôle central de JavaScript (via XMLHttpRequest)
 - Échanges **asynchrones** avec le serveur de l'application.
 - Analyse des données reçues.
 - Modification du document Web vu par l'utilisateur (via DOM HTML).

Exemples : Google Search et Google Map

The screenshot shows two browser windows side-by-side. The left window is a Google search results page for the query "3il". It displays various links related to the institution, including its website (www.3il.fr), extranet, admissions, and contact information. The right window is a Google Maps search for "3il" in Limoges. It shows a map of the area around 3IL Entreprise, located at 43 Rue de Sainte-Anne. A callout box provides the address and a link to leave a review. The map also shows the surrounding streets and landmarks.

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 9

Exemple : une simple somme

Principe des exemples :

- Notre premier exemple est une application qui calcule la somme entre deux entiers.
- Cette application sera déclinée en version «classique» et AJAX.
- Nous nous intéresserons surtout aux données échangées dans les deux cas (utilisation de Firebug).
- L'interface de l'application sera identique dans les deux cas.
- La somme calculée doit s'afficher dans la même interface, en gardant les valeurs dont elle est issue.

The screenshot shows a web browser window titled "01-Somme Classique". It contains a form with two input fields labeled "Valeur A :" and "Valeur B :" and a button labeled "Somme". Below the form, there is a section labeled "Résultat :" which displays the result of the addition. The browser's status bar indicates the URL "cours3il/3ilA2_Wel".

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 10

Exemple : une simple somme

Données échangées :

	Somme Web «Classique»	Somme Web AJAX
Réponse première requête (saisie de l'URL)	455 octets (740 octets avec les en-têtes)	HTML : 481 octets (713 octets avec les en-têtes) JavaScript : 646 octets (877 octets avec les en-têtes)
Réponse deuxième requête (clic sur le bouton «Somme»)	464 octets (749 octets avec les en-têtes)	3 octets (286 octets avec les en-têtes)

Explications :

- La version AJAX est plus lourde en premier chargement à cause de la partie JavaScript inexistant dans la version classique.
- La version AJAX est en revanche beaucoup plus légère pour le retour du résultat de la somme, car elle ne retourne que le résultat, sans aucune interface HTML, contrairement à la version classique.

Réseau					
URL	Statut	Domaine	Poids	Remote IP	
▶ GET 02-SommeAJAX.html	304 Not Modified	cours3il	481 B	127.0.0.1:80	
▶ GET 02-SommeAJAX.js	304 Not Modified	cours3il	646 B	127.0.0.1:80	
▶ POST 02-scriptSomme.php	200 OK	cours3il	3 B	127.0.0.1:80	
3 requests			1.1 KB	(1.1 KB à parti	

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 11

Exemple : une simple somme

Structure de l'application :

	Somme Web «Classique»	Somme Web AJAX
Fichier de départ	script PHP avec une balise <form>	document HTML et JavaScript sans balise <form>
Au clic sur le bouton «Somme»	envoi classique du formulaire (requête HTTP)	déclenchement en JavaScript de l'envoi d'une requête HTTP
Destinataire de la requête	le script PHP de départ	un script PHP dédié
Au retour de la réponse	interprétation classique du document HTML	traitement JavaScript d'insertion de la réponse

Remarques :

- Pour faire apparaître les données dans le formulaire d'origine, le script de départ de la version «classique» est son propre destinataire.
- Dans la version AJAX, il n'y a plus d'envoi du formulaire, seulement une requête HTTP contenant les valeurs dont il faut faire la somme, la balise <form> n'est plus nécessaire.
- Le nombre de fichiers au total dans les deux versions n'est pas un élément déterminant.

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 12

Exemple : une simple somme «classique»

Exemple :

Script PHP «01-SommeClassique.php»

```
01 <?php
02     $somme = '';
03     $valeurA = '';
04     $valeurB = '';
05     if(isset($_POST['valeurA']) && isset($_POST['valeurB'])){
06         $valeurA = intval($_POST['valeurA']);
07         $valeurB = intval($_POST['valeurB']);
08         $somme = $valeurA + $valeurB;
09     }
10 ?>
11 <html>
12     <head>
13         <title>01-Somme Classique</title>
14     </head>
15     <body>
16         <form action="01-SommeClassique.php" method="POST">
17             <label>Valeur A : </label>
18             <input name="valeurA" type="text" value=<?php echo $valeurA;?>"/>
19             <label>Valeur B : </label>
20             <input name="valeurB" type="text" value=<?php echo $valeurB;?>"/>
21             <input type="submit" value="Somme" />
22         </form>
23         <p>Résultat : <?php echo $somme;?></p>
24     </body>
25 </html>
```

Si des données sont envoyées, déclenche le calcul de la somme

Ce script est le destinataire du formulaire

Pour faire apparaître les valeurs saisies après calcul

Exemple : une simple somme AJAX

Exemple :

Script PHP «02-SommeAJAX.html»

```
01 <html>
02     <head>
03         <title>01-Somme Classique</title>
04         <script type="text/javascript" src="02-SommeAJAX.js"></script>
05     </head>
06     <body>
07         <label>Valeur A : </label>
08         <input name="valeurA" type="text" id="inputA"/>
09         <label>Valeur B : </label>
10         <input name="valeurB" type="text" id="inputB"/>
11         <input type="button" value="Somme" onclick="somme();"/>
12         <p>Résultat : <span id="resultat"></span></p>
13     </body>
14 </html>
```

Partie JavaScript de l'application

Emplacement pour l'affichage du résultat

Bouton déclenchant la fonction JavaScript somme()

Remarque :

Observer les identifiants des balises <input>.

Exemple : une simple somme AJAX

Exemple :

NE JAMAIS FAIRE ANSI

Script JavaScript «02-SommeAJAX.js»

```
01 function somme(){
02     var XHR;
03     var valA = parseInt(document.getElementById('inputA').value);
04     var valB = parseInt(document.getElementById('inputB').value);
05     if(window.XMLHttpRequest){
06         XHR = new XMLHttpRequest(); ← Création de l'objet XMLHttpRequest
07     } else {
08         XHR = new ActiveXObject("Microsoft.XMLHTTP");
09     }
10     XHR.onreadystatechange = function(){ ← Fonction appelée à chaque étape de l'échange avec le serveur, 4 = réponse reçue
11         if(XHR.readyState == 4) && XHR.status == 200){
12             document.getElementById('resultat').innerHTML = XHR.responseText;
13         }
14     }
15     XHR.open("POST", "02-scriptSomme.php", true);
16     XHR.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
17     XHR.send("valeurA="+valA+"&valeurB="+valB); ← Lignes 15 à 17, ouverture d'une requête et envoi avec les données
18 }
```

Récupère les valeurs des <input>

← Création de l'objet XMLHttpRequest

Fonction appelée à chaque étape de l'échange avec le serveur, 4 = réponse reçue

Texte retourné par la requête

← Lignes 15 à 17, ouverture d'une requête et envoi avec les données

Explications :

- Lignes 03 et 04 : récupération des valeurs des balises <input>
- Lignes 05 à 09 : création de l'objet XMLHttpRequest (XHR pour les intimes).
- Lignes 10 à 14 : fonction de traitement des étapes de la requête. L'état 4 signifie réponse reçue.
- Lignes 15 à 17 : ouverture d'une requête, paramétrage et envoie avec les données.

Exemple : une simple somme AJAX

Exemple :

Script PHP «02-scriptSomme.php»

```
01 <?php
02     $somme = '';
03     $valeurA = '';
04     $valeurB = '';
05     if(isset($_POST['valeurA']) && isset($_POST['valeurB'])){
06         $valeurA = intval($_POST['valeurA']);
07         $valeurB = intval($_POST['valeurB']);
08         $somme = $valeurA + $valeurB;
09     }
10     echo $somme; ← Se contente d'une simple sortie texte, sans balise HTML
11 ?>
```

Traitement d'un formulaire classique

← Se contente d'une simple sortie texte, sans balise HTML

Remarque :

La partie PHP se limite à une simple sortie texte, sans aucun élément HTML.

AJAX : Avantages et inconvénients

Avantages :

- Applications Web AJAX plus proches des applications de "bureau traditionnelles".
- Réponses aux actions utilisateur plus rapides.
- Réduction des échanges réseau (modifications DOM VS chargement complet de document).
- Déploiement facile : repose sur des technologies existantes et répandues.

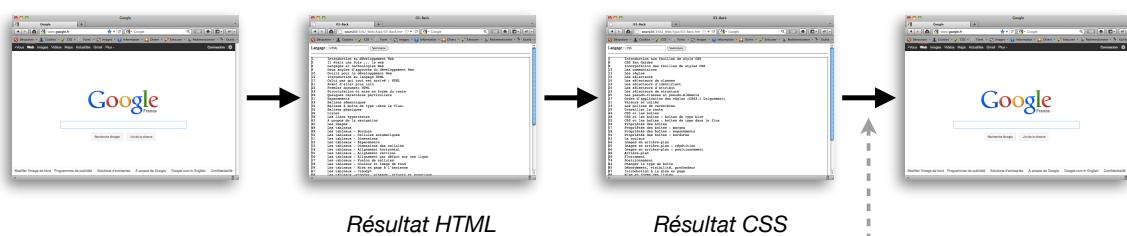
Inconvénients :

- JavaScript peut être désactivé (de plus en plus rare).
- XMLHttpRequest n'est pas standard, juste très répandu (norme du W3C en cours).
- Problème de compatibilité entre les navigateurs (souvent IE VS le reste du monde).
- Référencement par les moteurs de recherche (contenus générés dynamiquement non pris en compte).
- Changement dans le fonctionnement de l'historique du navigateur :
 - Modification du contenu sans changer de document HTML
 - Comportement du bouton précédent du navigateur
 - ▶ Bouton précédent : retour au document précédent
 - ▶ Web classique : association état de l'application / document consulté
 - ▶ AJAX : état de l'application dissocié du document
- État de l'application pouvant être difficilement mis en marque-page (bookmark).
- Écriture de l'application à l'aide de nombreux gestionnaires d'événements.
- Possibilités d'injection de code JavaScript.
- Interactions client/serveur invisibles à l'utilisateur (espionnage de mots de passe).
- Ne change pas un des principes du Web : le client est à l'initiative des échanges.
- En principe, **ne permet d'interagir qu'avec des scripts serveurs issus du même domaine.**

AJAX : Le bouton précédent

Exemple :

- Cet exemple permet d'obtenir le sommaire des supports de cours de Web en utilisant AJAX.
- Les données sont stockées dans une base MySQL, un script PHP sert à leur récupération.
- L'application se compose d'un formulaire pour saisir le langage. Au clic sur le bouton «sommaire», la requête AJAX est déclenchée et le résultat est affiché dans une balise <pre>.
- Scénario de l'exemple :
 - On ouvre le navigateur sur www.google.com (ou n'importe quelle autre page)
 - On ouvre sur l'application AJAX de sommaire
 - On recherche le sommaire HTML
 - On recherche le sommaire CSS
 - On appuie sur le bouton «précédent» du navigateur.



Observations :

- Lors de l'appui sur le bouton précédent, on s'attendait à revenir sur le sommaire de HTML, le navigateur revient sur la page de Google.
- Il existe des techniques pour gérer cela, elles demandent un effort supplémentaire.

AJAX : Fonctionnement

Principe :

- Création d'un objet de type XMLHttpRequest, objet à états.
 - Le mode de création dépend du navigateur (IE 6 ou le reste du monde).
 - Cet objet est souvent désigné XHR dans la littérature.
- Positionnement d'un callback pour les changement d'états de XMLHttpRequest via la propriété onreadystatechange.
- Préparation d'une requête HTTP via la méthode open.
- Constitution des données de la requête.
- Appel de la méthode send avec les données de la requête.
- A chaque étape de la requête (changement d'état) appel du callback.
- Les informations sont reçus par les propriétés responseText ou responseXML.

Valeurs de la propriété readyState :

Valeur	Description
0 (<i>uninitialized</i>)	non initialisé
1 (<i>loading</i>)	début du transfert des données
2 (<i>loaded</i>)	données transférées
3 (<i>interactive</i>)	réponse en cours de réception
4 (<i>complete</i>)	réponse arrivée

AJAX : Objet XMLHttpRequest

Création de l'objet XMLHttpRequest :

- Tenir compte des différents types de navigateurs.
- Tenir compte des différentes version d'Internet Explorer.

Exemple :

NE JAMAIS FAIRE AINSI

Script PHP «02-SommeAJAX.js»

```
01 function somme(){
02     var XHR;
03     var valA = parseInt(document.getElementById('inputA').value);
04     var valB = parseInt(document.getElementById('inputB').value);
05     if(window.XMLHttpRequest){
06         XHR = new XMLHttpRequest(); ← Pour tous les navigateurs y compris IE 7+
07     } else {
08         XHR = new ActiveXObject("Microsoft.XMLHTTP");
09     }
10     XHR.onreadystatechange = function(){
11         if(XHR.readyState == 4 && XHR.status==200){ ← Pour IE 6
12             document.getElementById('resultat').innerHTML = XHR.responseText;
13         }
14     }
15     XHR.open("POST", "02-scriptSomme.php", true);
16     XHR.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
17     XHR.send("valeurA="+valA+"&valeurB="+valB);
18 }
```

AJAX : Requête synchrone

Plus de callback :

- Il est aussi possible de faire des requêtes synchrones.
- L'exécution de JavaScript est suspendue pendant toute la durée de l'échange avec le serveur.
- Plus besoin de callback pour la réception des données.
- Le troisième paramètre de open() doit être à false.
- Attention : les temps de réponse peuvent être très longs !

Exemple :

NE JAMAIS FAIRE AINSI

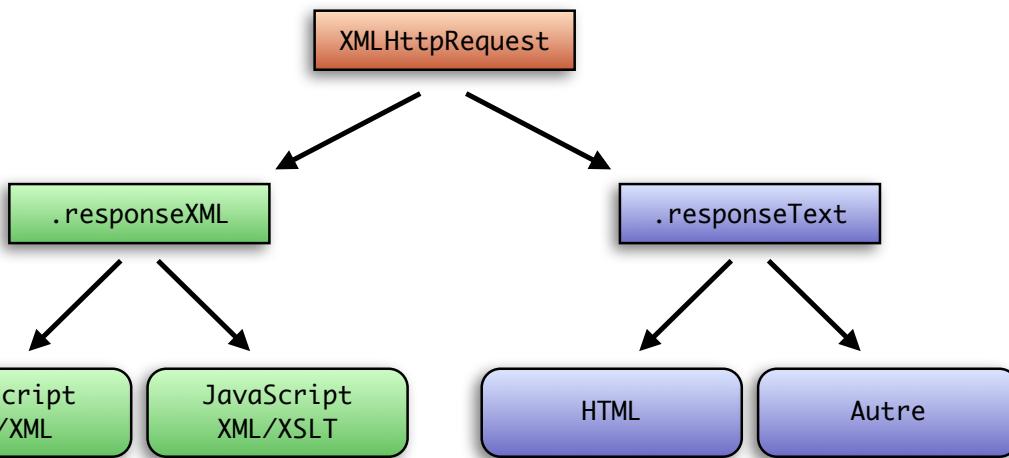
Script JavaScript «04-Synchrone.js»

```
01 function somme(){  
02     var XHR;  
03     var valA = parseInt(document.getElementById('inputA').value);  
04     var valB = parseInt(document.getElementById('inputB').value);  
05     if(window.XMLHttpRequest){  
06         XHR = new XMLHttpRequest();  
07     } else {  
08         XHR = new ActiveXObject("Microsoft.XMLHTTP");  
09     }  
10     XHR.open("POST","04-scriptSomme.php",false);  
11     XHR.setRequestHeader("Content-type","application/x-www-form-urlencoded");  
12     XHR.send("valeurA="+valA+"&valeurB="+valB);  
13     document.getElementById('resultat').innerHTML = XHR.responseText;  
14 }
```

Résultat disponible
sans callback

AJAX : Les types de réponse

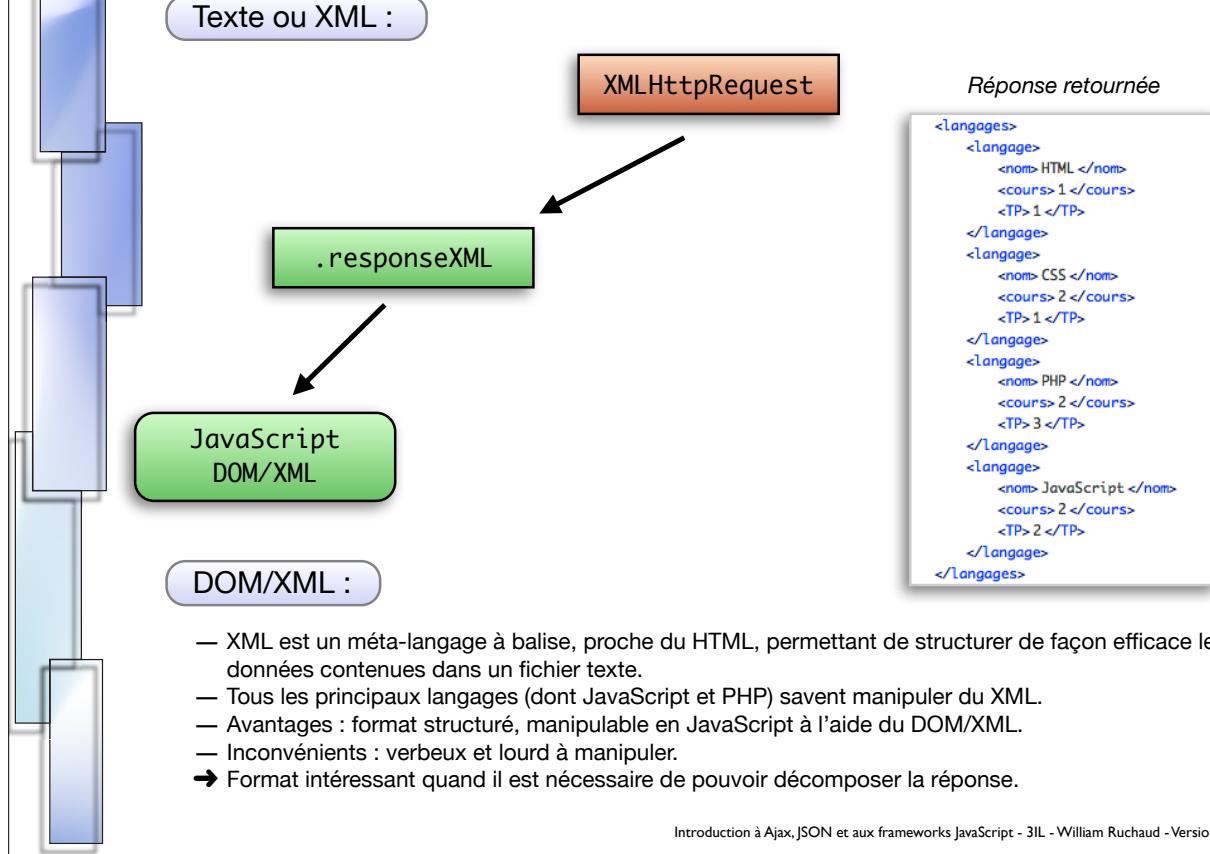
Texte ou XML :



Selon l'usage :

- Le choix entre XML et texte dépend de l'exploitation des données qui sont récupérées.
- Principaux usages :
 - Utilisation directe, sans transformation
 - Utilisation totale avec transformation
 - Utilisation partielle avec ou sans transformation.
- Il est parfois nécessaire de pouvoir extraire des valeurs de ce qui est reçu, parfois non.
- **Toutes les possibilités ne sont pas listées !**

AJAX : Réponse DOM/XML



AJAX : Réponse DOM/XML

The screenshot shows a browser window with the title "AJAX : Réponse DOM/XML". Inside the window, there is a heading "Exemple :" and a code editor-like area titled "Document HTML «05-XMLDOM.html»".

Document HTML «05-XMLDOM.html»

```
01 <html>
02   <head>
03     <title>05-XMLDOM</title>
04     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
05     <script type="text/javascript" src="05-XMLDOM.js"></script>
06     <style type="text/css">
07       table {
08         border-collapse: collapse;
09       }
10       td {
11         min-width: 80px;
12         border: 1px solid black;
13       }
14     </style>
15   </head>
16   <body>
17     <input type="button" value="Langages" onclick="langages()"/>
18     <div id="resultat"></div>
19   </body>
20 </html>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 24

AJAX : Réponse DOM/XML

Exemple :

NE JAMAIS FAIRE AINSI

Script JavaScript «05-XMLDOM.js» 1/2

```
01 function langages(){
02     var XHR;
03     if(window.XMLHttpRequest){
04         XHR = new XMLHttpRequest();
05     } else {
06         XHR = new ActiveXObject("Microsoft.XMLHTTP");
07     }
08     XHR.onreadystatechange = readyStateChange; ← diapo suivante
09     XHR.open("POST", "05-LangagesXML.php", true);
10     XHR.send("");
11 }
```

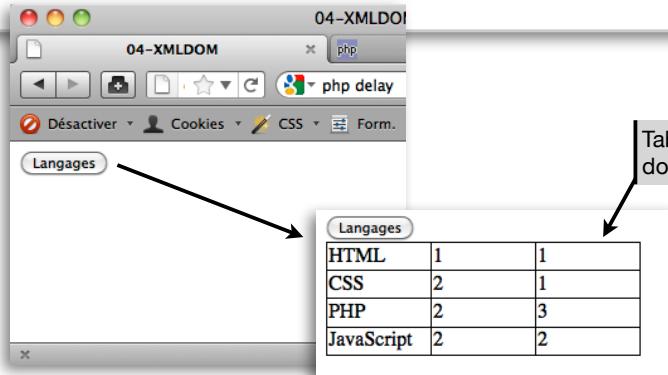


Tableau HTML créé à partir des données XML reçues en AJAX

AJAX : Réponse DOM/XML

Exemple :

NE JAMAIS FAIRE AINSI

Script JavaScript «05-XMLDOM.js» 2/2

```
13 function readyStateChange(){
14     var XML, tab, tbody, tr, td, langs, lang, cours, tp, i, j;
15     if(XHR.readyState == 4 && XHR.status == 200){
16         XML = XHR.responseXML;
17         langs = XML.getElementsByTagName('langage'); ← DOM/XML pour la liste des langages
18         tab = document.createElement('table');
19         tbody = document.createElement('tbody');
20         tab.appendChild(tbody);
21         for(i=0;i<langs.length;i++){
22             lang = langs[i];
23             tr = document.createElement('tr');
24             for(j=0;j<lang.childNodes.length;j++){
25                 td = document.createElement('td');
26                 td.appendChild(document.createTextNode(
27                     lang.childNodes[j].textContent));
28                 tr.appendChild(td);
29             }
29             tbody.appendChild(tr);
30         }
31         document.getElementById('resultat').appendChild(tab);
32     }
33 }
34 }
```

```
<langages>
    <langage>
        <nom> HTML </nom>
        <cours> 1 </cours>
        <TP> 1 </TP>
    </langage>
    <langage>
        <nom> CSS </nom>
        <cours> 2 </cours>
```

AJAX : Réponse DOM/XML

Exemple :

NE JAMAIS FAIRE AINSI

Script PHP «05-LangagesXML.php» 1/2

```
01 <?php
02     try {
03         $db = new PDO('mysql:host=localhost;dbname=3il2web','root','');
04     } catch(PDOException $err) {
05         die('Erreur connexion BD');
06     }
07     $sql = 'SELECT * FROM ajax_web'; ← Récupération des données
08     $req = $db->prepare($sql);
09     if(!$req->execute()){
10         die('Erreur SQL');
11     }
12     $data = $req->fetchAll(PDO::FETCH_ASSOC);
13     header('Content-Type: text/xml'); ← En-tête pour XML
14     $doc = new DOMDocument();
15     $langages = $doc->createElement('langages'); ← Création de la balise racine
16     $doc->appendChild($langages); ← <langages> ajout au document XML
```

```
<langages>
  <langage>
    <nom> HTML </nom>
    <cours> 1 </cours>
    <TP> 1 </TP>
  </langage>
  <langage>
    <nom> CSS </nom>
    <cours> 2 </cours>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 27

AJAX : Réponse DOM/XML

Exemple :

NE JAMAIS FAIRE AINSI

Script PHP «05-LangagesXML.php» 2/2

```
17     foreach($data as $L){
18         $langage = $doc->createElement('langage');
19
20         $nom = $doc->createElement('nom');
21         $nom->appendChild($doc->createTextNode($L['langage']));
22
23         $cours = $doc->createElement('cours');
24         $cours->appendChild($doc->createTextNode($L['cours']));
25
26         $tp = $doc->createElement('TP');
27         $tp->appendChild($doc->createTextNode($L['tp']));
28
29         $langage->appendChild($nom);
30         $langage->appendChild($cours);
31         $langage->appendChild($tp);
32         $langages->appendChild($langage);
33     }
34     echo $doc->saveXML(); ← Effectue la sortie du XML
```

\$L est l'itérateur du foreach sur l'ensemble des langages, contient les informations d'un langage

```
<langages>
  <langage>
    <nom> HTML </nom>
    <cours> 1 </cours>
    <TP> 1 </TP>
  </langage>
  <langage>
    <nom> CSS </nom>
    <cours> 2 </cours>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 28

AJAX : Réponse XML/XSLT

Texte ou XML :

XMLHttpRequest

.responseXML

JavaScript
XML/XSLT

Réponse retournée

```
<langages>
<language>
<nom> HTML </nom>
<cours> 1 </cours>
<TP> 1 </TP>
</language>
<language>
<nom> CSS </nom>
<cours> 2 </cours>
<TP> 1 </TP>
</language>
<language>
<nom> PHP </nom>
<cours> 2 </cours>
<TP> 3 </TP>
</language>
<language>
<nom> JavaScript </nom>
<cours> 2 </cours>
<TP> 2 </TP>
</language>
</langages>
```

XML/XSLT :

- XSLT est un langage (dialecte XML, norme W3C) qui permet de décrire les transformations à opérer sur un document XML pour le convertir en un autre format.
- Ici XSLT est utilisé pour transformer les données XML en un fragment HTML (la balise <table>) pour l'insérer dans le document HTML de départ.
- Le script PHP est identique à l'exemple précédent, le document HTML est quasi identique, ils ne seront pas détaillés ici.
- L'apparence visuelle est identique à l'exemple précédent.
- N'a d'intérêt que si le résultat doit être transformé en HTML.

AJAX : Réponse XML/XSLT

Exemple :

NE JAMAIS FAIRE AINSI

Script JavaScript «06-XMLXSL.js» 1/2

```
01 function langages(){
02     var XHR;
03     if(window.XMLHttpRequest){
04         XHR = new XMLHttpRequest();
05     } else {
06         XHR = new ActiveXObject("Microsoft.XMLHTTP");
07     }
08     XHR.onreadystatechange = function(){
09         var XML,XSL,XSLTProc,tab;
10         if(XHR.readyState == 4 && XHR.status == 200){
11             XML = XHR.responseXML;
12             XSL = xsl();           ← Fonction xsl() sur la diapo suivante
13             XSLTProc = new XSLTProcessor(); ← Objet permettant la transformation
14             XSLTProc.importStylesheet(XSL);
15             tab = XSLTProc.transformToFragment(XML,document);
16             document.getElementById('resultat').appendChild(tab);
17         }
18     }
19     XHR.open("POST", "05-LangagesXML.php",true);
20     XHR.send("");
21 }
```

Remarque :

Seule la version Firefox/Chrome/Safari/Opéra est présente pour des raisons de place.

AJAX : Réponse XML/XSLT

Exemple :

NE JAMAIS FAIRE AINSI

Script JavaScript «06-XMLXSL.js» 2/2

```
23 function xsl(){  
24     var XHR;  
25     if(window.XMLHttpRequest){  
26         XHR = new XMLHttpRequest();  
27     } else {  
28         XHR = new ActiveXObject("Microsoft.XMLHTTP");  
29     }  
30     XHR.open("GET", "06-XMLXSL.xsl", false); ← Récupération synchrone du  
31     XHR.send("");  
32     return XHR.responseXML;  
33 }
```

AJAX : Réponse XML/XSLT

Exemple :

NE JAMAIS FAIRE AINSI

Fichier XSL «06-XMLXSL.js»

```
01 <?xml version="1.0" encoding="UTF-8"?>  
02 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">  
03     <xsl:output method="html"/>  
04     <xsl:template match="/"> ← Pour traiter la racine du XML reçu  
05         <table>  
06             <tbody>  
07                 <xsl:for-each select="langages/langage"> ← Pour chaque balise <langage>  
08                     <tr>  
09                         <td><xsl:value-of select="nom"/></td>  
10                         <td><xsl:value-of select="cours"/></td>  
11                         <td><xsl:value-of select="TP"/></td>  
12                     </tr>  
13                 </xsl:for-each>  
14             </tbody>  
15         </table>  
16     </xsl:template>  
17 </xsl:stylesheet>
```

value-of permet de récupérer
le texte dans la balise indiquée
par select

```
<langages>  
    <langage>  
        <nom> HTML </nom>  
        <cours> 1 </cours>  
        <TP> 1 </TP>  
    </langage>  
    <langage>  
        <nom> CSS </nom>  
        <cours> 2 </cours>
```

AJAX : Réponse Texte/HTML

Texte ou XML :

Réponse retournée

```
<table>
  <tbody>
    <tr>
      <td>HTML</td>
      <td></td>
      <td>1</td>
    </tr>
    <tr>
      <td>CSS</td>
      <td>2</td>
      <td>1</td>
    </tr>
    <tr>
      <td>PHP</td>
      <td>2</td>
      <td>3</td>
    </tr>
    <tr>
      <td>JavaScript</td>
      <td>2</td>
      <td>2</td>
    </tr>
  </tbody>
```

XMLHttpRequest

.responseText

HTML

Texte/HTML :

- Principe : l'assemblage du tableau HTML à intégrer est assuré par le serveur.
- N'a d'intérêt que les le tableau doit être intégré tel quel.
- Ne permet pas de sélectionner des données précisément.
- Rend plus délicat l'usage du même service de fourniture de données par plusieurs applications (pouvant être de diverses natures).
- Le document HTML d'origine est quasi identique aux exemples précédents.

AJAX : Réponse Texte/HTML

Exemple :

NE JAMAIS FAIRE AINSI

Script JavaScript «07-TexteHTML.js»

```
01 function langages(){
02   var XHR;
03   if(window.XMLHttpRequest){
04     XHR = new XMLHttpRequest();
05   } else {
06     XHR = new ActiveXObject("Microsoft.XMLHTTP");
07   }
08   XHR.onreadystatechange = function(){
09     var XML,XSL,XSLTProc,tab;
10     if(XHR.readyState == 4 && XHR.status == 200){
11       document.getElementById('resultat').innerHTML = XHR.responseText;
12     }
13   }
14   XHR.open("POST", "05-LangagesHTML.php", true);
15   XHR.send("");
16 }
```

AJAX : Réponse Texte/HTML

Exemple :

NE JAMAIS FAIRE AINSI

Script PHP «05-LangagesHTML.js»

```
01 <?php
02     try {
03         $db = new PDO('mysql:host=localhost;dbname=3il2web','root','');
04     } catch(PDOException $err) {
05         die('Erreur connexion BD');
06     }
07     $sql = 'SELECT * FROM ajax_web';
08     $req = $db->prepare($sql);
09     if(!$req->execute()){
10         die('Erreur SQL');
11     }
12     $data = $req->fetchAll(PDO::FETCH_ASSOC);
13 ?>
14 <table>
15     <tbody>
16         <?php foreach($data as $L): ?>
17             <tr>
18                 <td><?php echo $L['langage'];?></td>
19                 <td><?php echo $L['cours'];?></td>
20                 <td><?php echo $L['tp'];?></td>
21             </tr>
22         <?php endforeach; ?>
23     </tbody>
24 </table>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 35

AJAX : Pourquoi «Ne jamais faire ainsi» ?

AJAX à la main :

- Préparation des données à l'envoi :
 - Utilisation du format URL.
 - Peu pratique pour des données structurées (tableaux, objets).
- Récupération des données :
 - Choix entre format texte ou XML.
 - Texte valable pour une réponse simple (problème d'identification et séparation des données).
 - XML peut rapidement devenir lourd à exploiter (en DOM/XML JavaScript).
- Exploitation des données :
 - Manipulation d'un document HTML en DOM peut rapidement devenir lourde (création de tableaux avec possibilités d'interactions).
 - Problèmes récurrents des comportements spécifiques à chaque navigateur.
 - Problèmes récurrents dans la façon de gérer les données (vérifications, formatage, etc.)
- Sécurité :
 - Limitation aux scripts de même domaine que l'URL appelante (contournable).
 - Difficile de maîtriser toutes les injections de code possibles.
- Nécessité de se doter d'outils aptes à rendre plus simple la création d'applications Web avec Ajax.

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 36

AJAX : XMLHttpRequest Référence

Propriétés :

Propriété	Description
readyState	état de l'échange avec le serveur vaut 0, 1, 2, 3 ou 4
responseText	données renvoyées par le serveur au format texte
responseXML	données renvoyées par le serveur au format XML
status	code de statut HTTP (404 : page non trouvée, 200 page OK, etc.)
statusText	version "textuelle" du statut HTTP

AJAX : XMLHttpRequest Référence

Méthodes :

Propriété	Description
onreadystatechange	méthode appelée à chaque changement d'état
abort()	arrête la requête Ajax en cours
open(method, url, [[async], [username], [password]])	indique l'URL pour la prochaine requête. Méthode peut valoir "GET/POST". Async peut valoir true (asynchrone), false (synchrone). Possibilité d'identification HTTP.
send(data)	envoie la requête. data doit être à null si aucun paramètre, ou être une chaîne de paramètres type URL (pour un POST).
getResponseHeader(headerelt)	retourne l'élément headerelt contenu dans l'en-tête de la réponse
getAllResponseHeaders()	retourne l'intégralité de l'en-tête de la réponse.
setRequestHeader(headerelt, value)	règle la valeur du paramètre headerelt de l'en-tête de la réponse. Doit être utilisé entre open et send.

JSON

JavaScript Object Notation

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3

JSON : Objets et tableaux littéraux

Rappel :

- JavaScript est extrêmement souple pour la création d'objets et de tableaux.
- Pour l'un comme pour l'autre JavaScript accepte des formes littérales.

Tableau :

Extrait de script JavaScript

```
1 var tab = ['William','Ruchaud','3iLA2 Web',8,7];
```

Objet :

Extrait de script JavaScript

```
01 var obj = {  
02   nom:'William',  
03   prenom:'Ruchaud',  
04   matiere:'3iLA2 Web',  
05   cours:8,  
06   tp:7  
07 };
```

JSON : La fonction eval()

eval() :

- JavaScript est un langage interprété et dynamique.
- La fonction eval() permet d'exécuter du code JavaScript donné sous la forme d'une chaîne de caractères.

Exemple :

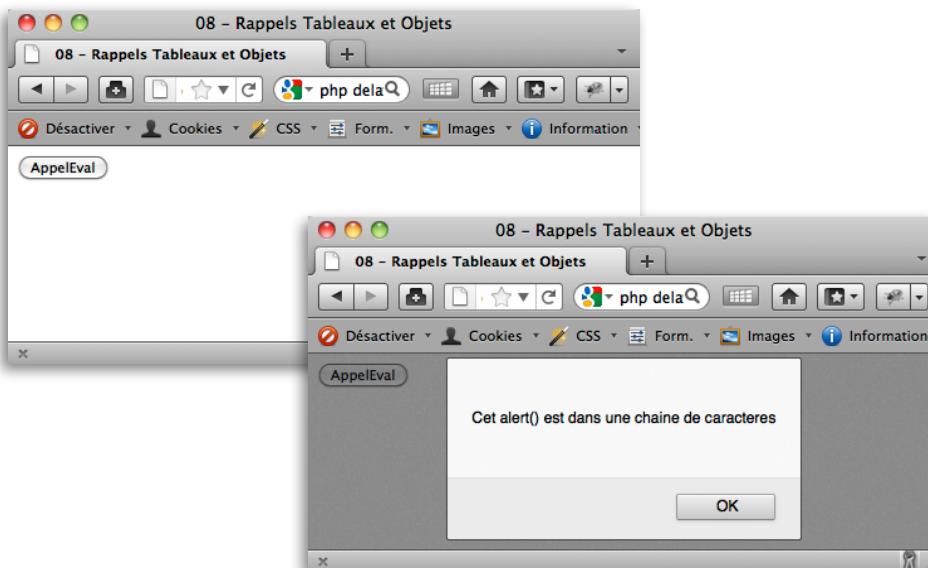
Document HTML/JavaScript

```
01 <html>
02   <head>
03     <title>09 - La fonction eval()</title>
04     <script>
05       function AppelEval(){
06         eval('alert("Cet alert() est dans une chaine de caracteres")');
07       }
08     </script>
09   </head>
10   <body>
11     <input type="button" value="Tab" onclick="AppelEval()" />
12   </body>
13 </html>
```

Évaluation du code JavaScript contenu dans une chaîne.

JSON : La fonction eval()

Exemple :



JSON : L'objet JSON

Encodage JSON en JavaScript :

- Un objet JSON est disponible dans les navigateurs pour assurer l'encodage et le décodage JSON (Firefox 3.5+, IE8+, Opera 10.5+, Webkit).
- Pour les autres versions de navigateur, il existe de nombreuses bibliothèques.
- La méthode `stringify()` permet l'encodage en chaîne de caractères.

Exemple :

Document HTML/JavaScript

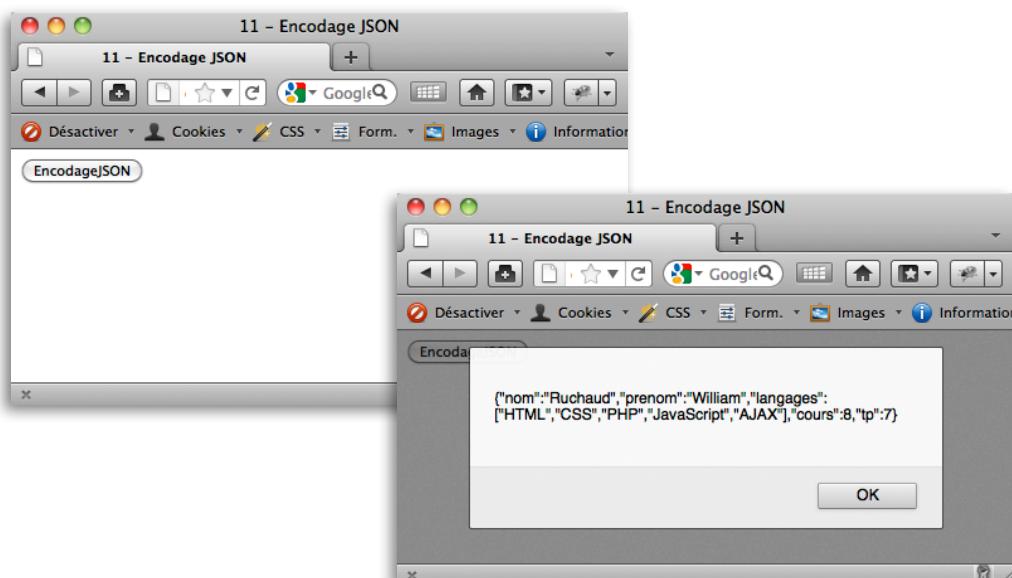
```
01 <html>
02   <head>
03     <title>11 - Encodage JSON</title>
04     <script>
05       function EncodageJSON(){
06         var prof = {
07           nom:"Ruchaud",
08           prenom:"William",
09           langages:['HTML','CSS','PHP','JavaScript','AJAX'],
10           cours:8,
11           tp:7
12         };
13         alert(JSON.stringify(prof));
14       }
15     </script>
16   </head>
17   <body>
18     <input type="button" value="EncodageJSON" onclick="EncodageJSON()" />
19   </body></html>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 43

Pour obtenir la chaîne JSON correspondant à l'objet prof

JSON : L'objet JSON

Exemple :



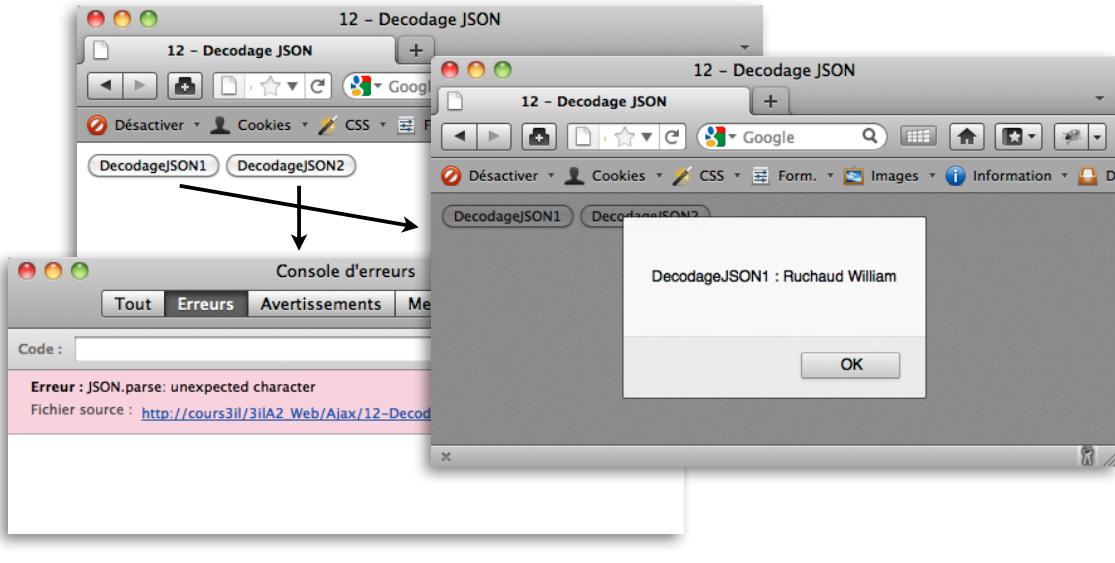
Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 44

JSON : L'objet JSON

Décodage JSON en JavaScript :

- Utiliser la fonction eval() pour décoder une chaîne JSON est dangereux car aucun contrôle n'est effectué en dehors de la syntaxe. Il y a un risque d'injection JavaScript.
- L'objet JSON dispose de la méthode parse() pour effectuer le décodage en rejetant tout ce qui n'est pas au format JSON.

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 45

JSON : L'objet JSON

Exemple :

Document HTML/JavaScript

```
01 <html>
02   <head>
03     <title>12 - Décodage JSON</title>
04     <script>
05       function DecodageJSON1(){
06         var data = '{"nom":"Ruchaud","prenom":"William"}';
07         var prof = JSON.parse(data);
08         alert("DecodageJSON1 : "+prof.nom+ ' '+prof.prenom);
09     }
10     function DecodageJSON2(){
11       var data = "alert('coucou')";
12       var res = JSON.parse(data);
13       alert("DecodageJSON2 : "+res);
14     }
15   </script>
16 </head>
17 <body>
18   <input type="button" value="DecodageJSON1" onclick="DecodageJSON1()" />
19   <input type="button" value="DecodageJSON2" onclick="DecodageJSON2()" />
20 </body>
21 </html>
```

Pour obtenir un objet prof à partir de la chaîne JSON data

Provoque une erreur car data n'est pas une chaîne JSON

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 46

JSON : format texte pour AJAX

JSON :

- Format léger.
- Accès direct ou quasi direct des données.
- Support multi-langage.
- Ne peut pas être rendu visuellement directement (JSON → HTML par exemple).

XML :

- Format lourd et verbeux.
- Utilisation indirecte des données (utilisation de DOM).
- Support multi-langage.
- Peut être rendu visuellement directement via XLST.

JSON : PHP

JSON pour PHP :

- PHP dispose des fonctions `json_encode()` et `json_decode()` depuis la version 5.2
- JSON peut être utilisé à la fois pour transférer des données :
 - JavaScript → PHP
 - PHP → JavaScript

Exemple :

Script PHP

```
01 <?php
02     $objPHP = new stdClass(); ← Déclaration d'un objet en PHP
03     $objPHP->nom = "Ruchaud"; ← L'opérateur -> permet d'accéder aux champs de l'objet
04     $objPHP->prenom = "William";
05     $objPHP->langage = array("HTML", "CSS", "PHP", "JavaScript", "AJAX");
06     $objPHPJSON = json_encode($objPHP);
07
08     $objJSON = '{"ecole":"3iL", "matiere":"3iLA2-Web", "prof":"WR"}';
09     $obj = json_decode($objJSON);
10
11     echo 'objPHPJSON : '.$objPHPJSON;
12     echo '<br/>';
13     echo 'obj->ecole '.$obj->ecole;
14     echo '<br/>';
15     echo 'obj->matiere '.$obj->matiere;
16 ?>
```

objPHPJSON :{"nom":"Ruchaud","prenom":"William","langage":["HTML", "CSS", "PHP", "JavaScript", "AJAX"]}
obj->ecole 3iL
obj->matiere 3iLA2-Web

JSON : les autres langages

- ASP:
 - JSON for ASP.
 - JSON ASP utility class.
- ActionScript:
 - ActionScript3.
 - JSONConnector.
- Bash:
 - Jshon.
 - JSON.sh.
- BlitzMax:
 - bmx-rjson.
- C:
 - JSON_checker.
 - M's JSON parser.
 - YAJL.
 - cJSON.
 - Jansson.
 - jsOn.
 - LibU.
 - jsmn.
 - cson.
 - json-c.
- C++:
 - jsoncpp.
 - zoolib.
 - JOST.
 - CAJUN.
 - libjson.
- nosjob.
- JSONKit.
- JsonBox.
- jsonme--.
- C#:
 - fastJSON.
 - JSON_checker.
 - Jayrock.
 - Json.NET - LINQ to JSON.
 - JSONSharp.
 - LitJSON.
 - JSON for .NET.
 - JsonFx.
 - JsonExSerializer.
 - JSON@CodeTitans
 - fluent-json
 - How do I write my own parser?
- Clojure:
 - clojure-json.
 - API for json.
- Cobol:
 - XML Thunder.
- ColdFusion:
 - ColdFusion 8.
 - toJSON.
- D:
 - Cashew.
 - Libdjson.
- Delphi:
 - Delphi Web Utils.
 - JSON Delphi Library.
 - JSON Toolkit.
- E:
 - JSON in TermL.
- Eiffel:
 - eJSON.
- Erlang:
 - ejson.
 - mochijson2.
- Fantom:
 - Json.
- Go:
 - package json.
- Haskell:
 - RJson package.
 - json package.
- haXe:
 - hxJSON.
- Java:
 - org.json.
 - org.json.me.
 - Jackson JSON Processor.

JSON : les autres langages

- Java:
 - Json-lib.
 - JSON Tools.
 - json-simple.
 - Stringtree.
 - SOJO.
 - Jettison.
 - json-taglib.
 - XStream.
 - JsonMarshaller.
 - Flexjson.
 - JON tools.
 - google-gson.
 - Argo.
 - jsonjj.
 - fastjson.
 - Json-smart.
 - mjson.
 - jjson.
- JavaScript:
 - json2.js.
 - json_sans_eval.
- Lasso:
 - JSON.
- Lisp:
 - Common Lisp JSON.
 - Yason.
 - Emacs Lisp.
- LotusScript:
 - JSON LS.
- Lua:
 - Json4Lua.
 - LuaJSON.
 - LuaJSON C Library.
 - Fleece.
 - Lua CJSON.
 - dkjson.
- Matlab:
 - JSONlab.
- Objective C:
 - json-framework.
 - MTJSON.
 - JSONKit.
 - yajl-objc.
 - TouchJSON.
- Objective CAML:
 - Yojson.
- OpenLaszlo:
 - JSON.
- Perl:
 - CPAN.
- PHP:
 - PHP 5.2.
 - json.
 - Services_JSON.
 - Zend_JSON.
- Solar_Json.
- Comparison of php json libraries.
- Pike:
 - Public.Parser.JSON.
 - Public.Parser.JSON2.
- PL/SQL:
 - pljson.
 - Librairie-JSON.
- PowerShell:
 - PowerShell.
- Prolog:
 - SWI-Prolog HTTP support
- Puredata:
 - PuRestJson
- Python:
 - The Python Standard Library.
 - simplejson.
 - pyson.
 - Yajl-Py.
 - ultrajson.
- Qt:
 - QJson.
- R:
 - rjson.

JSON : les autres langages

- Racket:
 - [json-parsing](#).
- Rebol:
 - [json.r](#).
- RPG:
 - [JSON Utilities](#).
- Ruby:
 - [json](#).
 - [yajl-ruby](#).
 - [json-stream](#).
- Scheme:
 - [MZScheme](#).
 - [PLT Scheme](#).
- Squeak:
 - [Squeak](#).
- Symbian:
 - [s60-json-library](#).
- Tcl:
 - [JSON](#).
- Visual Basic:
 - [VB-JSON](#).
 - [PW.JSON](#).
- Visual FoxPro:
 - [fwJSON](#).
 - [JSON](#).

source : [www.json.org](#)

jQuery

Framework JavaScript

Frameworks JavaScript : introduction

Principes d'un framework : (tout langage)

- Constats :
 - Aspect répétitif de l'écriture des applications (fonctionnalités similaires).
 - Éléments propres aux langages sont souvent de trop bas niveau.
 - Les bibliothèques ne fournissent la plupart du temps que des outils.
 - De nombreux problèmes récurrents (portabilité, extensibilité, sécurité, etc.).
- Avantage d'un framework :
 - Ensemble de composants prévus pour fonctionner ensembles.
 - Automatise certaines fonctionnalités courantes.
 - Peuvent aller jusqu'à proposer un cadre pour la création d'une application.
 - Le plus souvent extensible (ajout de composants personnalisés)
 - Répond d'une "philosophie" du développement d'une application : une fois la philosophie comprise apprentissage de plus en plus aisés.
- Inconvénients :
 - Apprentissage loin d'être immédiat (surtout le côté "philosophique").
 - Rajoute une difficulté pour le débutant en plus du langage sur lequel il repose.
 - Extensibilité demandant un certain degré de maîtrise du framework.
 - Forte ressemblance entre applications qui dépendent d'un même framework.

Dans le cas des frameworks JavaScript :

- Simplification des manipulations DOM.
- Simplification des accès AJAX.
- Résolution de nombreux problèmes de compatibilité de navigateurs.
- Composants d'interface évolués (liste à proposition, zone de saisie filtrante, etc.).
- Composants d'effets graphiques (lightbox, smartbox, graybox, carousel, accordéon, etc.).

Frameworks JavaScript : introduction

JavaScript en mode non intrusif :

- À l'origine JavaScript est un langage pour l'interactivité des documents HTML :
 - Effet de rollover sur les images.
 - Vérification de formulaires avant envoi.
 - Affichage de messages pop-up.
- Ni JavaScript ni HTML n'ont été pensés pour construire des applications :
 - Manipulations DOM puissantes mais lourdes.
 - JavaScript utilisé souvent en mode intrusif (écriture de gestionnaires d'événement onClick).

Frameworks JavaScript :

- Sont le plus souvent prévus pour fonctionner en mode non intrusif.
- Étendent et simplifient le fonctionnement JavaScript (concision d'écriture).

Principaux Framework :

- Prototype
- JQuery
- Dojo
- Mochikit
- Ext JS
- Mootools
- Yahoo UI
- SproutCore



jQuery

jQuery :

- Bibliothèque multi-navigateur conçue pour simplifier les manipulation HTML / JavaScript.
- Créé par John Resig.
- Première version en Janvier 2006.
- Utilisé par près d'1/3 des 10000 sites les plus visités.
- Libre, open-source, sous licence MIT et GNU GPL 2.
- Actuellement en version : 1.7.2 (02 juillet 2012)
- Taille : 98Ko en version compressée, 254Ko non compressé.
- Extensible par le biais de plugins.
- Version 2.0 en cours de préparation.



Principales caractéristiques :

- Simplification des opérations sur le DOM.
- Gestion des événements.
- Manipulations CSS.
- Effets et animations (de nombreux plugins).
- Ajax.
- Fonctions utilitaires.

jQuery et l'objet \$:

- jQuery utilise l'objet \$ pour mettre à disposition la plupart de ses fonctionnalités (\$ est un nom de variable autorisé en JavaScript).
- C'est une pratique courante dans de nombreuses bibliothèques JavaScript.
- jQuery peut-être paramétré pour changer le nom de cet objet afin d'éviter les conflits.

jQuery : sélection d'objets DOM

\$() :

- La construction \$() est l'opération fondamentale de jQuery.
- \$() sert à sélectionner des objets DOM.
- \$() prend en paramètre un sélecteur CSS.
- Beaucoup plus puissant et précis que les getElementById(), getElementByName() et getElementsByTagName() standards.
- Retourne un objet de type jQuery (collection d'objets DOM).

.each() :

- .each() permet d'itérer sur un tableau (d'éléments DOM ou pas).
- S'applique aussi au résultat d'une sélection \$().
- .each() prend en paramètre la fonction à appliquer à chaque élément, elle-même pouvant avoir pour paramètre l'index de l'élément.

.get() :

- .get() la récupération d'un objet DOM par son index dans une sélection.
- S'applique à un objet jQuery.

jQuery : sélection d'objets DOM

Exemple :

Document JavaScript

```
01 <html>
02   <head>
03     <title>01 - DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="01-DOM.js"></script>
06   </head>
07   <body>
08     <h1 class="fondGris">jQuery</h1>
09     <p>Un framework JavaScript</p>
10     <p id="celui_la">Pour faire de l'Ajax</p>
11     <p class="fondGris">Et bien plus encore !!!</p>
12     <input type="button" value="Balise" onclick="ParBalise();"/>
13     <input type="button" value="id" onclick="ParId();"/>
14     <input type="button" value="Classe CSS" onclick="ParClasse();"/>
15   </body>
16 </html>
```

Inclusion jQuery

jQuery : sélection d'objets DOM

Exemple :

Document JavaScript

```
01 function ParBalise(){
02   $('p').each(function() { ← Pour toutes les balises <p>.
03     this.innerHTML = '-' + this.innerHTML;
04   }
05 );
06 }
07 function ParId(){ ← Remplace document.getElementById().
08   $('#celui_la').get(0).style.backgroundColor = "#AAAAAA";
09 }
10
11 function ParClasse(){
12   $('.fondGris').each(function(){ ← Pour toutes les balises de classe CSS
13     this.style.fontStyle = "italic"; ← «fondGris».
14   }
15 );
16 }
17 }
```

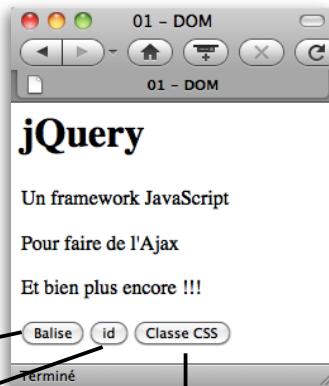
jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
08 <h1 class="fondGris">jQuery</h1>
09 <p>Un framework JavaScript</p>
10 <p id="celui_la">Pour faire de l'Ajax</p>
11 <p class="fondGris">Et bien plus encore !!!</p>
```

Le document HTML est rechargeé avant chaque capture.



jQuery

- Un framework JavaScript
- Pour faire de l'Ajax
- Et bien plus encore !!!

Balise id Classe CSS

Terminé

jQuery

- Un framework JavaScript
- Pour faire de l'Ajax
- Et bien plus encore !!!

Balise id Classe CSS

Terminé

jQuery ←

- Un framework JavaScript
- Pour faire de l'Ajax
- Et bien plus encore !!! ←**

Balise id Classe CSS

Terminé

jQuery : sélection d'objets DOM

Presque tous les sélecteurs CSS :

- \$() autorise pratiquement l'usage de tous les sélecteur CSS.
- L'exemple suivant utilise les sélecteurs de structure et par attribut.

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>02 - DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="02-DOM.js"></script>
06   </head>
07   <body>
08     <p>Paragraphe 1</p>
09     <div>
10       <p>Paragraphe 2</p>
11     </div>
12     <div id="div01">
13       <p>Paragraphe 3</p>
14     </div>
15     <p id="p01">Paragraphe 4</p>
16     <input type="button" value="Structure" onclick="Structure();"/>
17     <input type="button" value="Attribut" onclick="Attribut();"/>
18   </body>
19 </html>
```

jQuery : sélection d'objets DOM

Exemple :

Document JavaScript

```
01 function Structure(){
02     $('div p').each(function(){
03         this.style.fontStyle="italic"; ← Pour toutes les balises <p> contenues dans
04     });
05 }
06
07 function Attribut(){
08     $('[id]').each(function(){
09         this.style.backgroundColor = "#DDDDDD"; ← Toutes les balises ayant un attribut «id»
10     });
11 }
```

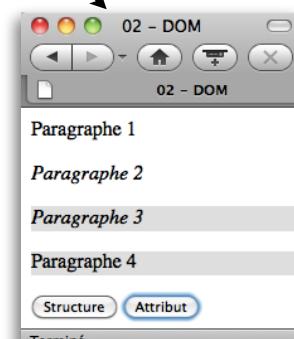
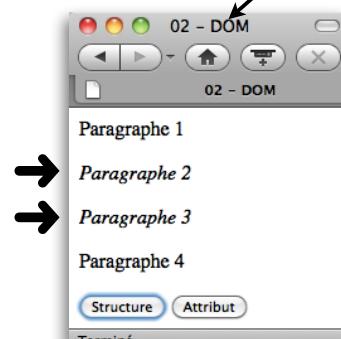
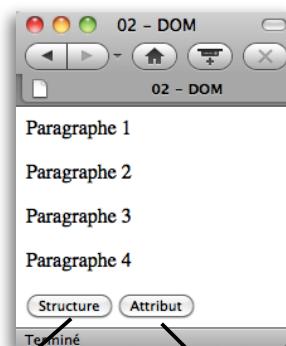
jQuery : sélection d'objets DOM

Exemple :

Fragment HTML

```
08 <p>Paragraphe 1</p>
09 <div>
10     <p>Paragraphe 2</p>
11 </div>
12 <div id="div01">
13     <p>Paragraphe 3</p>
14 </div>
15 <p id="p01">Paragraphe 4</p>
```

Appel des fonctions sans recharger le document HTML.



jQuery : sélection d'objets DOM

Au delà du CSS :

- \$() va au delà du CSS en ajoutant des pseudo-selecteurs très pratiques.
- :even permet de sélectionner les éléments d'index pair (numérotation partant à 0).

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>03 - DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="03-DOM.js"></script>
06   </head>
07   <body>
08     <p>Paragraphe 0</p>
09     <div>
10       <div>
11         <p>Paragraphe 1</p>
12       </div>
13     </div>
14     <div>
15       <p>Paragraphe 2</p>
16     </div>
17     <p>Paragraphe 3</p>
18     <p>Paragraphe 4</p>
19     <input type="button" value="P / 2" onclick="PseudoSelecteur1();"/>
20     <input type="button" value="P / 2 bis" onclick="PseudoSelecteur2();"/>
21   </body></html>
```

jQuery : sélection d'objets DOM

Exemple :

Document JavaScript

```
01 function PseudoSelecteur1(){
02   $('p:even').each(function(){
03     this.style.backgroundColor="#DDDDDD"; | Pour toutes les balises <p> d'index
04   });
05 }
06
07 function PseudoSelecteur2(){
08   $('body > p:even').each(function(){
09     this.style.fontStyle='italic';
10   });
11 }
```

jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
08 <p>Paragraphe 0</p>
09 <div>
10   <div>
11     <p>Paragraphe 1</p>
12   </div>
13 </div>
14 <div>
15   <p>Paragraphe 2</p>
16 </div>
17 <p>Paragraphe 3</p>
18 <p>Paragraphe 4</p>
```

Appel des fonctions sans recharger le document HTML.

```
03 - DOM
Paragraphe 0
Paragraphe 1
Paragraphe 2
Paragraphe 3
Paragraphe 4
P / 2 P / 2 bis
Terminé
```

```
03 - DOM
Paragraphe 0
Paragraphe 1
Paragraphe 2
Paragraphe 3
Paragraphe 4
P / 2 P / 2 bis
Terminé
```

```
03 - DOM
Paragraphe 0
Paragraphe 1
Paragraphe 2
Paragraphe 3
Paragraphe 4
P / 2 P / 2 bis
Terminé
```

jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>04 - DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       function Contenu(){
07         $("body *:contains('jQuery')").each(function(){
08           this.style.backgroundColor="#DDDDDD";
09         });
10       }
11     </script>
12   </head>
13   <body>
14     <h1>jQuery</h1>
15     <p>Petites d&acute;monstration</p>
16     <div>
17       <p>des s&acute;lecteurs de jQuery</p>
18       <p>Mais attention</p>
19     </div>
20     <h1>Fin de l'exemple</h1>
21     <input type="button" value="Contenu" onclick="Contenu();"/>
22   </body>
23 </html>
```

Toutes les balises descendantes de <body> avec 'jQuery' dans le contenu.

jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
14 <h1>jQuery</h1>
15 <p>Petites démonstration</p>
16 <div> ←
17   <p>des sélecteurs de jQuery</p>
18   <p>Mais attention</p>
19 </div>
20 <h1>Fin de l'exemple</h1>
```

```
$(“body *:contains(‘jQuery’)”)
```

Sélectionné car un de ses enfants contient 'jQuery'.



jQuery

Petites démonstration
des sélecteurs de jQuery
Mais attention

Fin de l'exemple

Contenu

Terminé



jQuery

Petites démonstration
des sélecteurs de jQuery
Mais attention

Fin de l'exemple

Contenu

Terminé



jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>05 - DOM</title>
04     <script type="text/javascript" src="jquery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="05-DOM.js"></script>
06   </head>
07   <body>
08     <h1>jQuery</h1>
09     <p>Paragraphe 1</p>
10     <div>
11       <p>Paragraphe 2</p>
12       <p id="celui_la">Paragraphe 3</p>
13       <p>Paragraphe 4</p>
14       <p>Paragraphe 5</p>
15     </div>
16     <p>Paragraphe 6</p>
17     <input type="button" value="Parent" onclick="Parent();"/>
18     <input type="button" value="NextAll" onclick="NextAll();"/>
19   </body>
20 </html>
```

jQuery : sélection d'objets DOM

Exemple :

Document JavaScript

```
01 function Parent(){
02     $("#celui_la").parent().each(function(){
03         this.style.backgroundColor="#AAAAAA";
04     });
05 }
06
07 function NextAll(){
08     $("#celui_la").nextAll().each(function(){
09         this.style.fontStyle="italic";
10     });
11 }
```

Retourne le parent de l'élément courant.

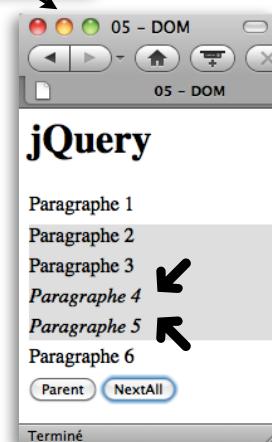
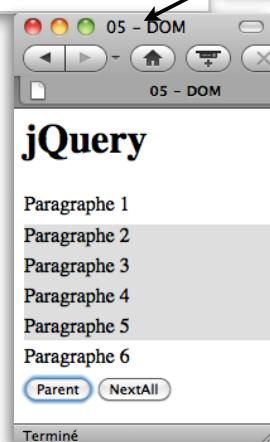
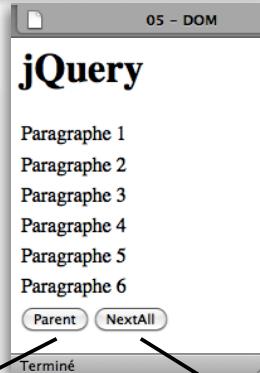
Tous les noeuds frères suivants.

jQuery : sélection d'objets DOM

Exemple :

Document HTML

```
08 <h1>jQuery</h1>
09 <p>Paragraphe 1</p>
10 <div>
11     <p>Paragraphe 2</p>
12     <p id="celui_la">Paragraphe 3</p>
13     <p>Paragraphe 4</p>
14     <p>Paragraphe 5</p>
15 </div>
16 <p>Paragraphe 6</p>
```



Appel des fonctions sans recharger le document HTML.

jQuery : gestion des événements

`$(document).ready()` :

- Équivalent à `window.onload`.
- Permet de déclencher un traitement une fois la construction DOM achevée.

`.bind()` :

- Permet d'associer à un événement une fonction de traitement.
- Dans sa forme la plus simple, `.bind()` doit recevoir une chaîne de caractères contenant le nom de l'événement ainsi que la fonction de traitement.

Les méthodes gestionnaires :

Méthodes

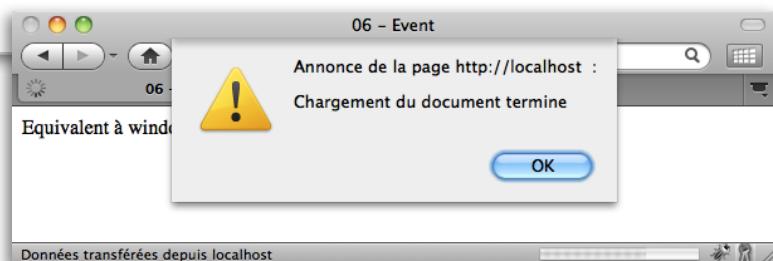
<code>.blur()</code>	<code>.hover()</code>	<code>.mouseout()</code>
<code>.change()</code>	<code>.keydown()</code>	<code>.mouseover()</code>
<code>.click()</code>	<code>.keypress()</code>	<code>.mouseup()</code>
<code>.dblclick()</code>	<code>.keyup()</code>	<code>.resize()</code>
<code>.erro()</code>	<code>.load()</code>	<code>.scroll()</code>
<code>.focus()</code>	<code>.mousedown()</code>	<code>.select()</code>
<code>.focusin()</code>	<code>.mouseenter()</code>	<code>.submit()</code>
<code>.focusout()</code>	<code>.mouseleave()</code>	<code>.unload()</code>
<code>.hover()</code>	<code>.mousemove()</code>	

jQuery : gestion des événements

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>06 - Event</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       $(document).ready(function(){ ← Équivalent à window.onload
07         alert("Chargement du document terminé");
08       });
09     </script>
10   </head>
11   <body>
12     <p>Équivalent à window.onload</p>
13   </body>
14 </html>
```



jQuery : gestion des événements

Exemple :

Document HTML

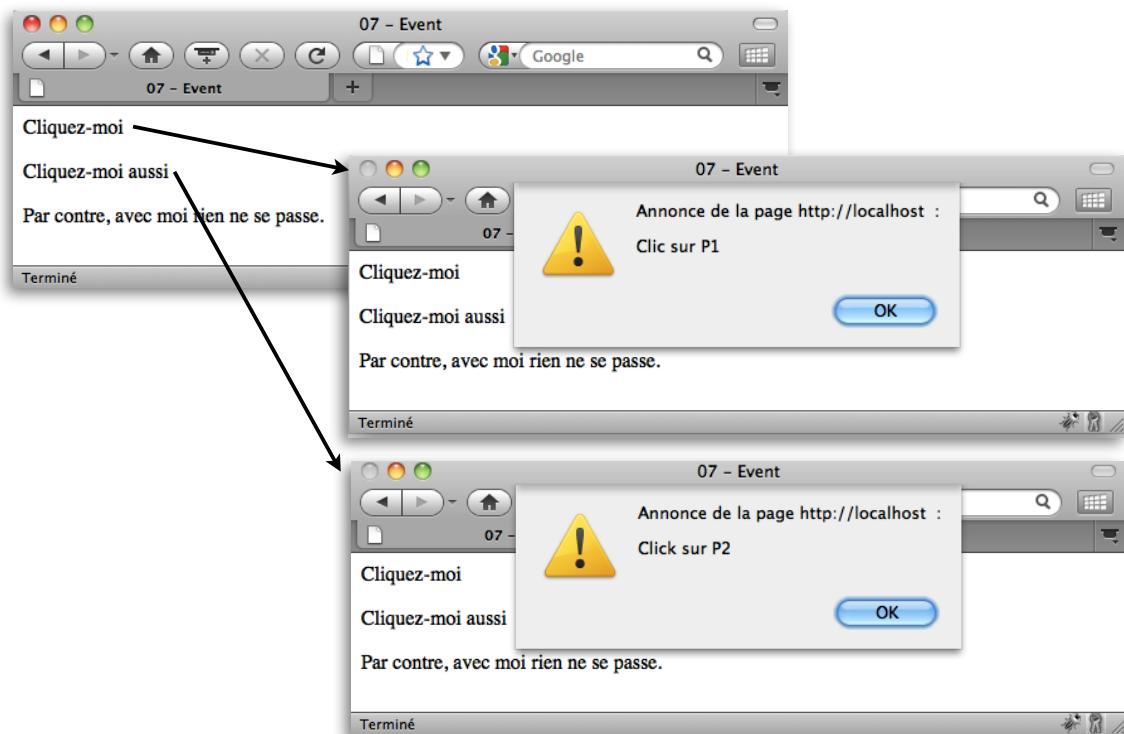
```
01 <html>
02   <head>
03     <title>07 - Event</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       $(document).ready(function(){
07         $('#p1').click(function(){
08           alert("Clic sur P1");
09         });
10         $('#p2').bind('click',function(){
11           alert('Click sur P2');
12         });
13       });
14     </script>
15   </head>
16   <body>
17     <p id="p1">Cliquez-moi</p>
18     <p id="p2">Cliquez-moi aussi</p>
19     <p>Par contre, avec moi rien ne se passe.</p>
20   </body>
21 </html>
```

Pour gérer le clic sur #p1 et
#p2

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 73

jQuery : gestion des événements

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 74

jQuery : gestion des événements

L'objet Event :

- Lors d'un appel de fonction de traitement d'événement un objet de type Event est généré.
- L'objet Event contient toutes les données relatives à l'événement.
- Il contient principalement :
 - event.type : contient la nature de l'événement.
 - event.target : l'objet DOM qui a initié l'événement.
 - event.which : indique le bouton de la souris ou la touche du clavier qui a initié l'événement.
- D'autres propriétés sont disponibles en fonction du type d'événement.

.trigger() :

- Permet de simuler une action de l'utilisateur en déclenchant l'émission d'un événement à partir de l'objet DOM servant à l'appeler.
- Prend pour paramètre le nom de l'événement à générer.

jQuery : gestion des événements

Exemple :

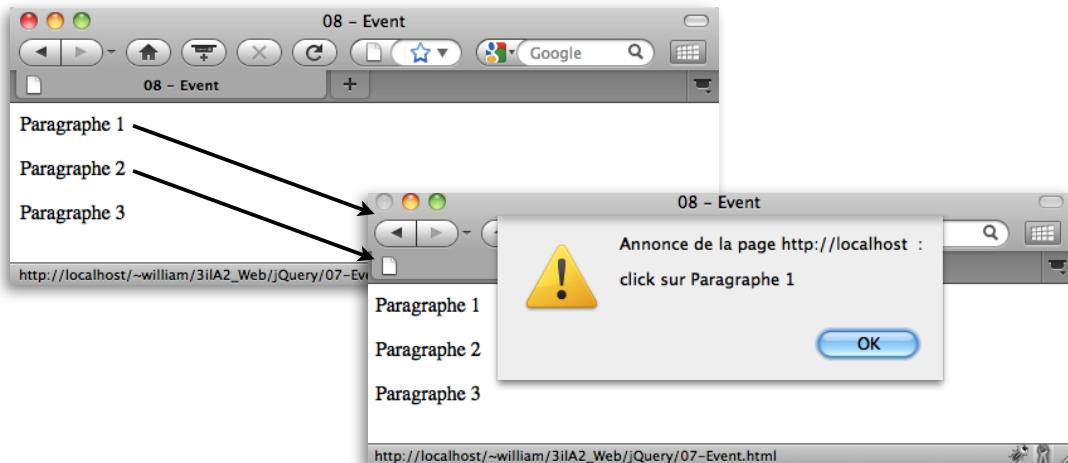
Document HTML

```
01 <html>
02   <head>
03     <title>08 - Event</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       $(document).ready(function(){
07         $('#p1').click(function(event){
08           alert('click sur '+event.target.innerHTML);
09         });
10         $('#p2').click(function(){
11           $('#p1').trigger('click');
12         });
13       });
14     </script>
15   </head>
16   <body>
17     <p id="p1">Paragraphe 1</p>
18     <p id="p2">Paragraphe 2</p>
19     <p>Paragraphe 3</p>
20   </body>
21 </html>
```

Objet sur lequel on a cliqué
Simule un clic sur #p1 à partir d'un clic sur #p2

jQuery : gestion des événements

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 77

jQuery : modifications CSS et effets

.css() :

- Permet de modifier le style CSS d'un ou plusieurs objets DOM sélectionné(s).
- Accepte en paramètres :
 - Un objet détaillant les propriétés CSS à appliquer.
 - Une propriété CSS et sa valeur, les deux sous forme de chaînes de caractères.

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>09 - Effets</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       function Effet(){
07         $("#p1").css({border:"1px solid black",fontStyle:"italic"});
08         $("#p1").css('backgroundColor',"#DDDDDD");
09       }
10     </script>
11   </head>
12   <body>
13     <p id="p1">Paragraphe au d&eacute;part normal</p>
14     <input type="button" value="Effet" onclick="Effet();"/>
15   </body>
16 </html>
```

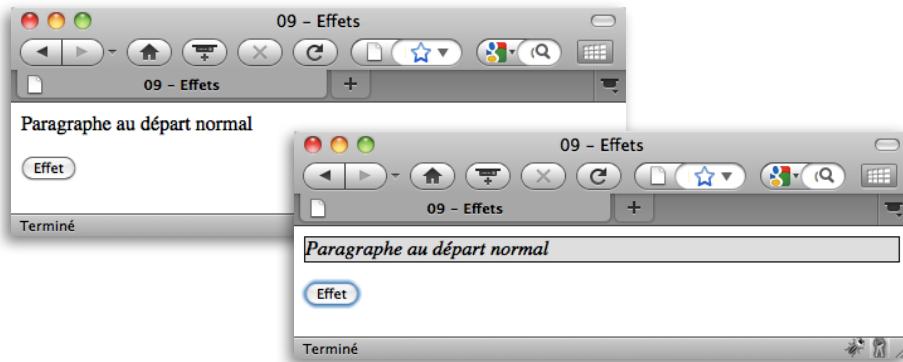
Pour une propriété

Façon objet

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 78

jQuery : modifications CSS et effets

Exemple :



.addClass() et .removeClass() :

- Ces fonctions permettent d'ajouter ou supprimer une classe CSS aux objets DOM sélectionnés.
- Les deux gèrent les classes multiples.

jQuery : modifications CSS et effets

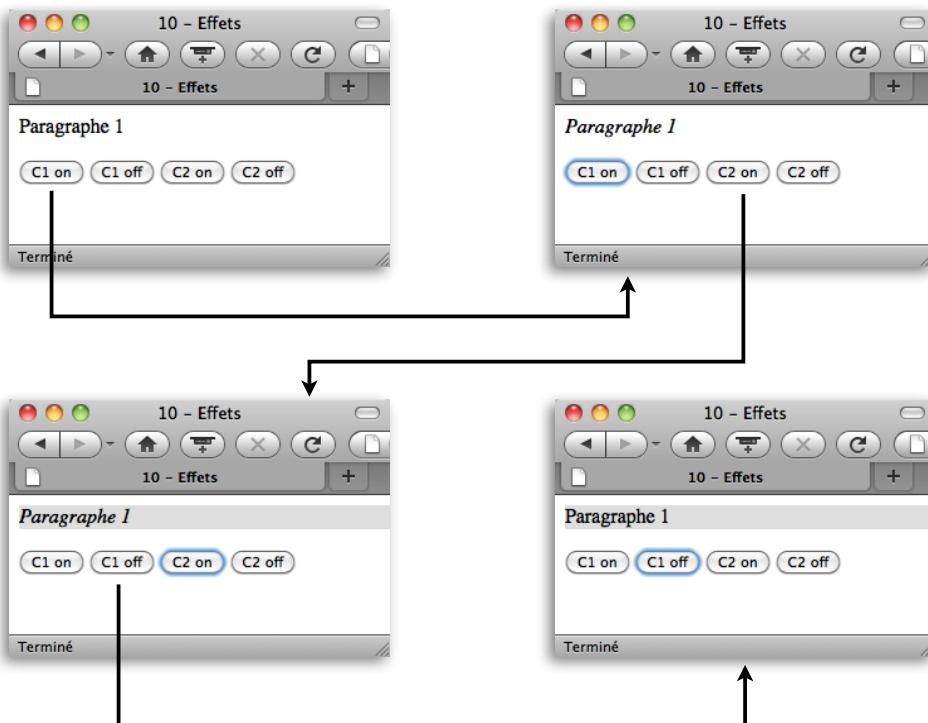
Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>10 - Effets</title>
04     <style type="text/css">
05       .C1 { font-style:italic;}
06       .C2 { background-color:#DDDDDD;}
07     </style>
08     <script type="text/javascript" src="jQuery/jquery-1.4.3.js"></script>
09     <script type="text/javascript">
10       function C1on(){ $('#p1').addClass('C1');}
11       function C1off(){ $('#p1').removeClass('C1');}
12       function C2on(){ $('#p1').addClass('C2');}
13       function C2off(){ $('#p1').removeClass('C2');}
14     </script>
15   </head>
16   <body>
17     <p id="p1">Paragraphe 1</p>
18     <input type="button" value="C1 on" onclick="C1on();"/>
19     <input type="button" value="C1 off" onclick="C1off();"/>
20     <input type="button" value="C2 on" onclick="C2on();"/>
21     <input type="button" value="C2 off" onclick="C2off();"/>
22   </body>
23 </html>
```

jQuery : modifications CSS et effets

Exemple :

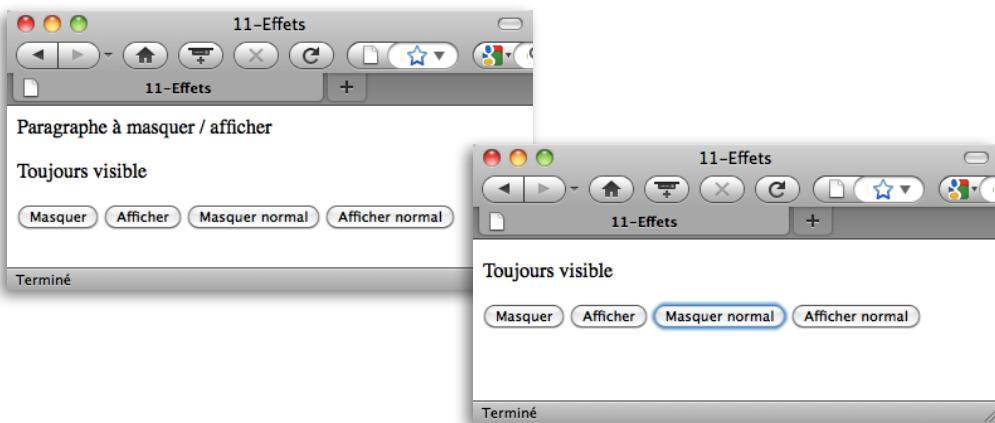


jQuery : modifications CSS et effets

.hide() et .show() :

- jQuery est surtout connu pour ses effets visuels.
- Les fonctions .hide() et .show() permettent de masquer et faire apparaître des éléments avec un effet de balayage ou non.
- Les deux s'appliquent à une sélection d'objets DOM.
- Elles acceptent en paramètre la durée de l'animation soit sous la forme d'une durée prédéfinie (chaîne), soit un nombre en millisecondes.
- Il est aussi possible de spécifier une fonction à appeler à la fin de l'animation.
- Les durées prédéfinies sont 'normal', 'fast' et 'slow'.

Exemple :



jQuery : modifications CSS et effets

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>11-Effets</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       function Masquer(vitesse){
07         $('#p1').hide(vitesse);
08       }
09       function Afficher(vitesse){
10         $('#p1').show(vitesse);
11       }
12     </script>
13   </head>
14   <body>
15     <p id="p1">Paragraphe &agrave; masquer / afficher</p>
16     <p>Toujours visible</p>
17     <input type="button" value="Masquer" onclick="Masquer('');"/>
18     <input type="button" value="Afficher" onclick="Afficher('');"/>
19     <input type="button" value="Masquer normal" onclick="Masquer('normal');"/>
20     <input type="button" value="Afficher normal" onclick="Afficher('normal');"/>
21   </body>
22 </html>
```

jQuery : modifications CSS et effets

.fadeIn() et .fadeOut() :

- .fadeIn() et .fadeOut() fonctionnent de la même manière que .hide() et .show().
- Elle permettent de mettre en place des fondus entrants ou sortants avec divers vitesses.

Exemple :

Document HTML

```
01 <html>
02   <head><title>11-Effets</title>
03   <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
04   <script type="text/javascript">
05     function Masquer(vitesse){
06       $('#p1').fadeOut(vitesse);
07     }
08     function Afficher(vitesse){
09       $('#p1').fadeIn(vitesse);
10     }
11   </script>
12 </head>
13 <body>
14   <p id="p1">Paragraphe &agrave; masquer / afficher</p>
15   <p>Toujours visible</p>
16   <input type="button" value="fadeOut" onclick="Masquer('');"/>
17   <input type="button" value="fadeIn" onclick="Afficher('');"/>
18   <input type="button" value="fadeOut slow" onclick="Masquer('slow');"/>
19   <input type="button" value="fadeIn slow" onclick="Afficher('slow');"/>
20 </body></html>
```

jQuery : manipulations DOM

Création d'éléments avec \$() :

- En passant une «chaîne de caractères HTML» en paramètre à \$(), jQuery crée le fragment DOM/HTML correspondant.
- jQuery dispose d'un jeu de fonctions beaucoup plus pratiques que celles du DOM HTML pour opérer les ajouts suppression d'objets DOM.

.appendTo() :

- Ajoute l'objet DOM **appelant** la méthode à chaque élément désigné par les paramètres.
- L'objet DOM est ajouté en tant que dernier enfant.
- Le paramètre peut-être
 - Un sélecteur.
 - Un objet DOM.
 - Une chaîne de caractères HTML.
 - Un objet jQuery.

.prependTo() :

- Fonction similaire à .appendTo().
- L'objet DOM est ajouté en tant que premier enfant.

jQuery : manipulations DOM

Exemple :

Document HTML

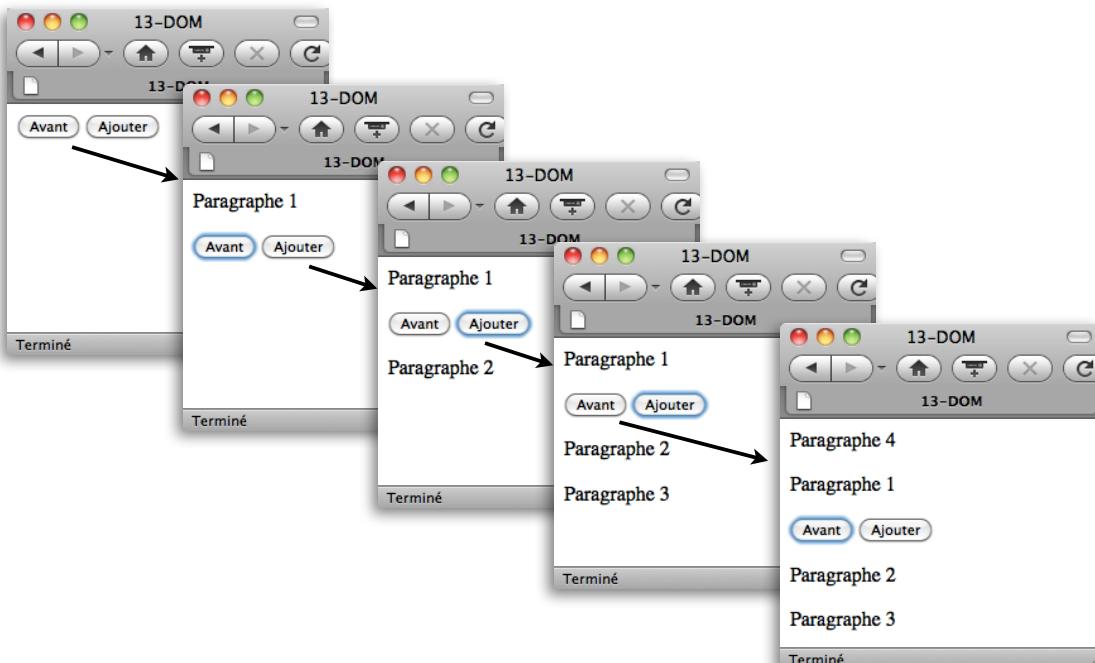
```
01 <html>
02   <head>
03     <title>13-DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       var num = 0;
07       function Apres(){
08         num++;
09         $('<p>Paragraphe '+num+'</p>').appendTo('body');
10       }
11       function Avant(){
12         num++;
13         $('<p>Paragraphe '+num+'</p>').prependTo('body');
14       }
15     </script>
16   </head>
17   <body>
18     <input type="button" value="Avant" onclick="Avant();"/>
19     <input type="button" value="Ajouter" onclick="Apres();"/>
20   </body>
21 </html>
```

Ajoute en dernier enfant

Ajoute en premier enfant

jQuery : manipulations DOM

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 87

jQuery : manipulations DOM

Exemple :

Document HTML

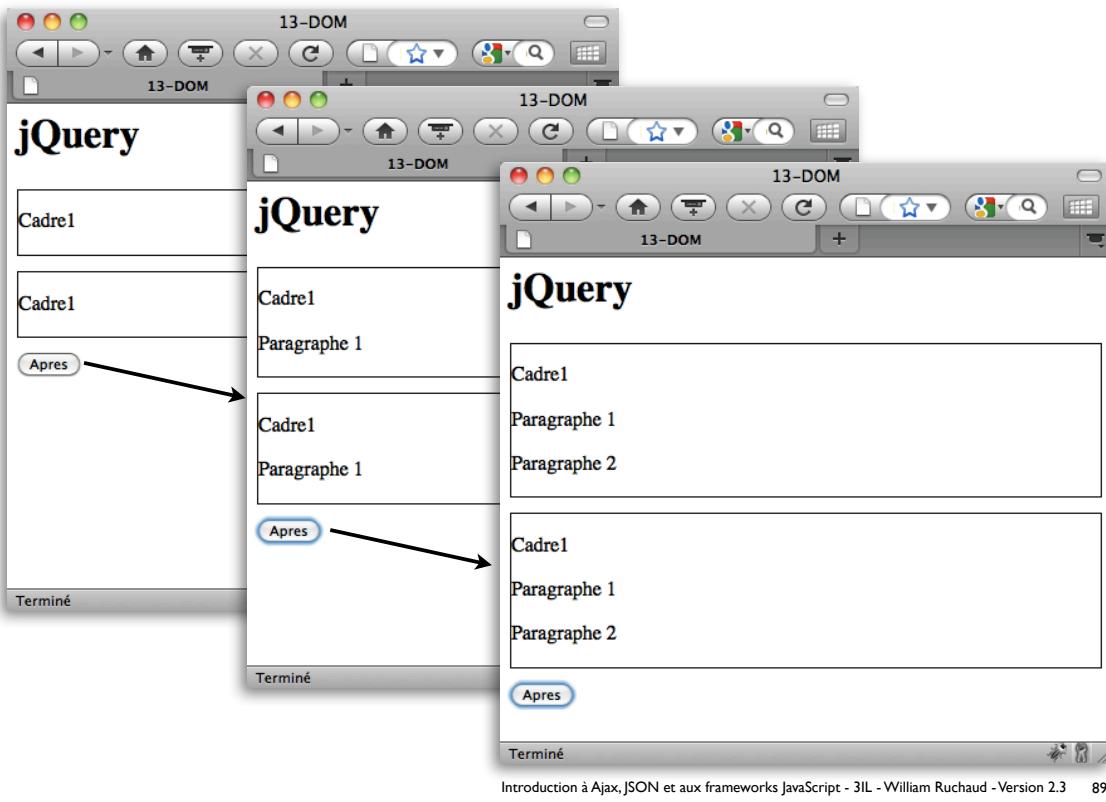
```
01 <html>
02   <head>
03     <title>14-DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.4.3.js"></script>
05     <style type="text/css">
06       .cadre { border:1px solid black; margin-bottom:12px; }
07     </style>
08     <script type="text/javascript">
09       var num = 0;
10       function Apres(){
11         num++;
12         $('<p>Paragraphe '+num+'</p>').appendTo('.cadre');
13       }
14     </script>
15   </head>
16   <body>
17     <h1>jQuery</h1>
18     <div class="cadre">
19       <p>Cadre1</p>
20     </div>
21     <div class="cadre">
22       <p>Cadre1</p>
23     </div>
24     <input type="button" value="Apres" onclick="Apres();"/>
25   </body>
26 </html>
```

Ajoute à tout ce qui porte la classe .cadre

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 88

jQuery : manipulations DOM

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 89

jQuery : manipulations DOM

Autres fonctions d'ajout :

Méthodes

.after()	Ajoute le paramètre après l'appelant
.append()	Ajoute le paramètre en tant que dernier enfant de l'appelant
.appendTo()	Ajoute l'appelant en tant que dernier enfant de chaque élément sélectionné en paramètre
.before()	Ajoute le contenu avant l'appelant
.insertAfter()	Ajoute l'appelant après chaque élément sélectionné en paramètre
.insertBefore()	Ajoute l'appelant avant chaque élément sélectionné en paramètre
.prepend()	Ajoute le paramètre avant l'appelant

.remove() :

- Sans paramètre : supprime l'élément appelant sélectionné.
- Avec paramètre : filtre pour l'appelant.

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 90

jQuery : manipulations DOM

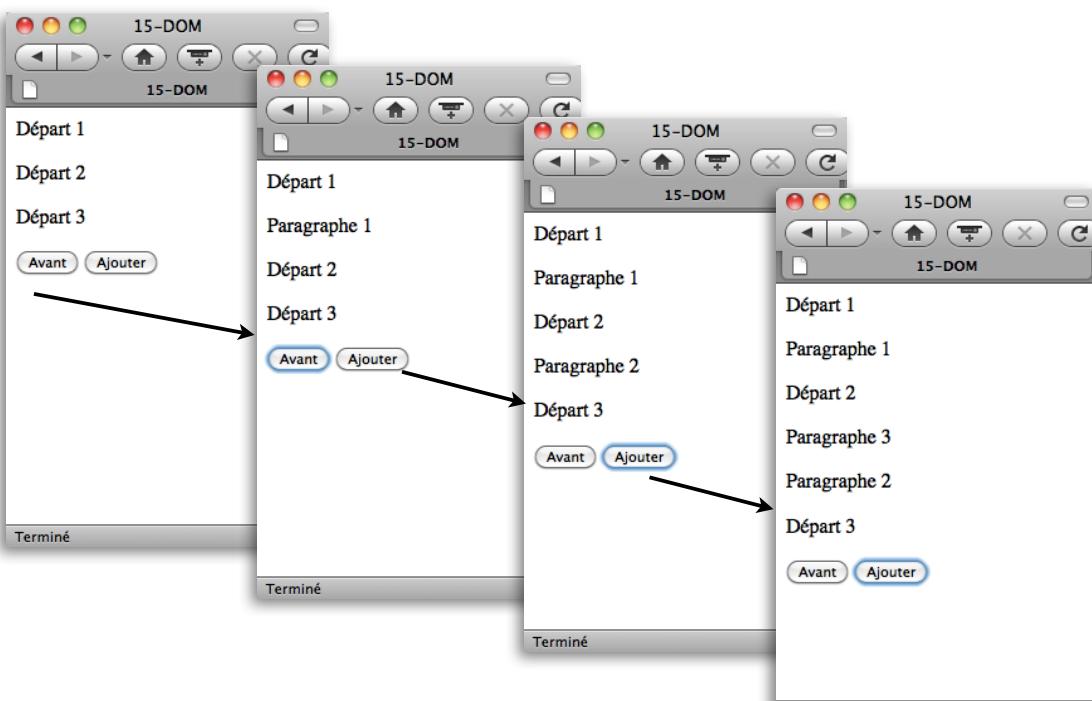
Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>15-DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       var num = 0;
07       function Apres(){
08         num++;
09         $('<p>Paragraphe '+num+'</p>').insertAfter('#repere');
10      }
11      function Avant(){
12        num++;
13        $('<p>Paragraphe '+num+'</p>').insertBefore('#repere');
14      }
15    </script>
16  </head>
17  <body>
18    <p>D&eacute;part 1</p>
19    <p id="repere">D&eacute;part 2</p> ← Élément ciblé par les sélections
20    <p>D&eacute;part 3</p>
21    <input type="button" value="Avant" onclick="Avant();"/>
22    <input type="button" value="Ajouter" onclick="Apres();"/>
23  </body>
24 </html>
```

jQuery : manipulations DOM

Exemple :



jQuery : manipulations DOM

Exemple :

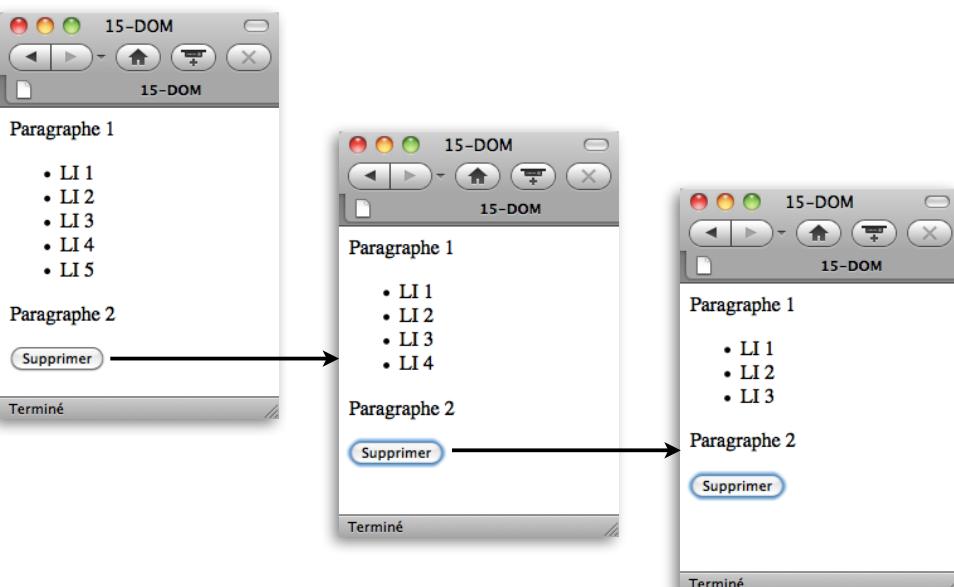
Document HTML

```
01 <html>
02   <head>
03     <title>15-DOM</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       function Supprimer(){
07         $('#liste li:last').remove();
08       }
09     </script>
10   </head>
11   <body>
12     <p>Paragraphe 1</p>
13     <ul id="liste">
14       <li>LI 1</li>
15       <li>LI 2</li>
16       <li>LI 3</li>
17       <li>LI 4</li>
18       <li>LI 5</li>
19     </ul>
20     <p>Paragraphe 2</p>
21     <input type="button" value="Supprimer" onclick="Supprimer();"/>
22   </body>
23 </html>
```

Le dernier de #liste

jQuery : manipulations DOM

Exemple :



jQuery : AJAX

\$.get() et \$.post() :

- Effectue une requête Ajax respectivement en mode GET ou POST.
- Paramètres :
 - URL du script serveur : obligatoire
 - Objet codant les paramètres de la requête : facultatif
 - Fonction callback à appeler lorsque la requête réussie : facultatif
- La fonction callback est appelée avec les paramètres suivants :
 - Les données reçues.
 - Le statut de la requête.
 - L'objet XMLHttpRequest.

\$.getJSON() :

- Charge des données encodées en JSON en mode GET uniquement.
- Les paramètres sont identiques à \$.get() et \$.post()

Exemple :

Script PHP «heure.php»

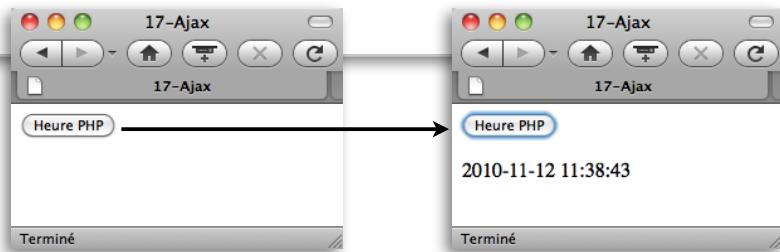
```
01 <?php  
02     echo date('Y-m-d H:i:s');  
03 ?>
```

jQuery : AJAX

Exemple :

Document HTML

```
01 <html>  
02     <head>  
03         <title>17-Ajax</title>  
04         <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>  
05         <script type="text/javascript">  
06             function HeurePHP(){  
07                 $.get("heure.php",function(data){  
08                     $('body').append('<p>' + data + '</p>');  
09                 });  
10             }  
11         </script>  
12     </head>  
13     <body>  
14         <input type="button" value="Heure PHP" onclick="HeurePHP();"/>  
15     </body>  
16 </html>
```



jQuery : AJAX

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>18-Ajax</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript">
06       function SommePHP(){
07         var a,b;
08         var i = $('#valeurA');
09         a = parseInt($('#valeurA').val());
10         b = parseInt($('#valeurB').val());
11         $.get('somme.php',{valA:a,valB:b},function(data){
12           $('#resultat').val(data);
13         });
14       }
15     </script>
16   </head>
17   <body>
18     <input type="text" id="valeurA" /> +
19     <input type="text" id="valeurB" /> =
20     <input type="text" id="resultat" /><br/>
21     <input type="button" value="Somme PHP" onclick="SommePHP();"/>
22   </body>
23 </html>
```

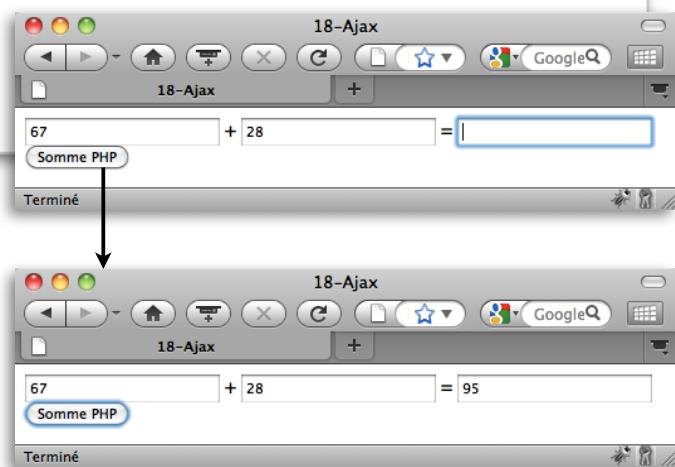
Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 97

jQuery : AJAX

Exemple :

Script PHP «somme.php»

```
01 <?php
02   $nombreA = 0;
03   if(isset($_GET['valA'])){
04     $nombreA = $_GET['valA'];
05   }
06   $nombreB = 0;
07   if(isset($_GET['valB'])){
08     $nombreB = $_GET['valB'];
09   }
10   echo $nombreA + $nombreB;
11 ?>
```



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 98

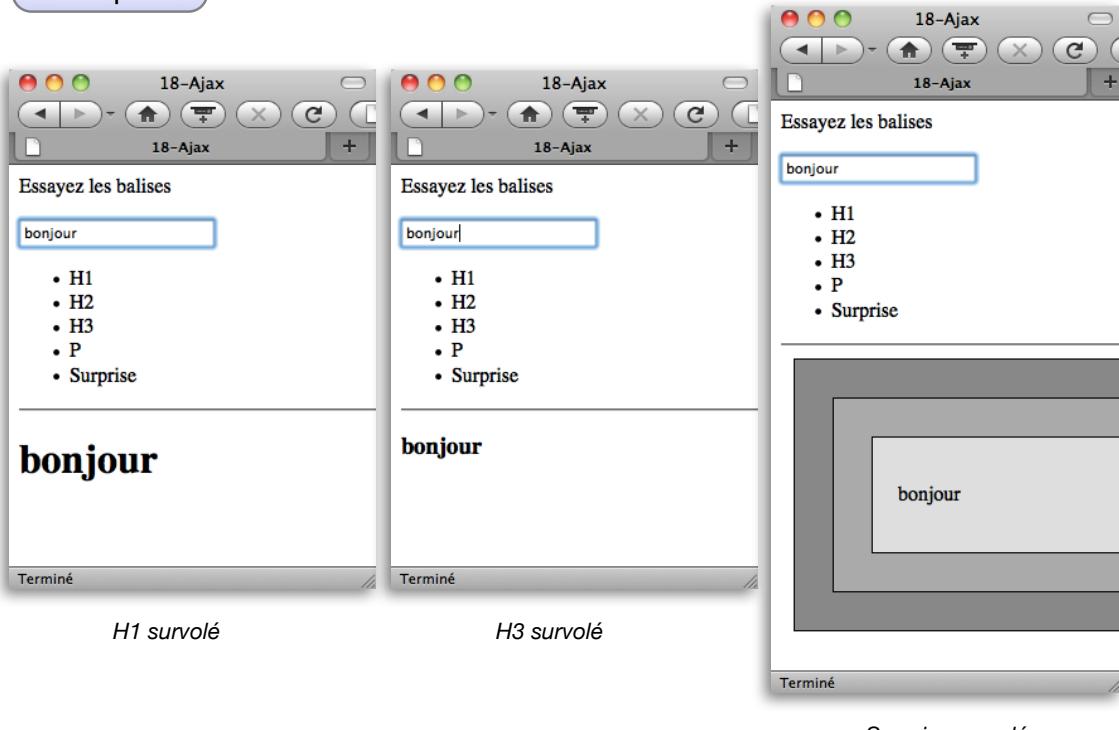
jQuery : AJAX

Possibilité de bloc HTML :

- Avec JQuery il est très facile d'ajouter un bloc HTML avec `$(...).html()`.
- Principe de l'exemple suivant :
 - Une zone de texte permet de saisir un contenu à insérer
 - Une liste `/` présente les différentes formes d'insertion possibles (H1/H2/...)
 - En survolant un ``, une requête Ajax est envoyée au serveur qui retourne un bloc HTML comprenant le contenu saisi dans la forme indiquée par le survol
 - Ce qui est reçu est ajouté en tant que contenu d'un `<div>`, le contenu de ce `<div>` est systématiquement remplacé.

jQuery : AJAX

Exemple :



jQuery : AJAX

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>19-Ajax</title>
04     <script type="text/javascript" src="jquery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="19-Ajax.js"></script>
06   </head>
07   <body>
08     <p>Essayez les balises</p>
09     <input type="text" id="contenu" />
10     <ul>
11       <li>H1</li>
12       <li>H2</li>
13       <li>H3</li>
14       <li>P</li>
15       <li>Surprise</li>
16     </ul>
17     <hr/>
18     <div id="visu"></div>
19   </body>
20 </html>
```

jQuery : AJAX

Exemple :

Document JavaScript

```
01 $(document).ready(function(){
02   $('#li').mouseover(function(event){
03     var balise = event.target.innerHTML;
04     var contenu = $('#contenu').val();
05     $.get('balise.php',{balise:balise,contenu:contenu},function(data){
06       $('#visu').html(data);
07     });
08   });
09 });
```

Les données reçues sont ajoutées en tant que fragment
HTML comme contenu de #visu

jQuery : AJAX

Exemple :

Script PHP

```
01 <?php
02     if(!isset($_GET['balise'])||!isset($_GET['contenu'])){
03         die('Erreur');
04     }
05     $balise = $_GET['balise'];
06     $contenu = htmlentities($_GET['contenu']);
07     if($balise!='Surprise'){
08         echo '<'.$balise.'>'.$contenu.'</'.$balise.'>';
09     } else {
10    ?>
11     <div style="border:1px solid black;margin:10px;padding:20px;
12             background-color:#888888;">
13         <div style="border:1px solid black;margin:10px;padding:20px;
14             background-color:#AAAAAA;">
15             <div style="border:1px solid black;margin:10px;padding:20px
16                 ;background-color:#DDDDDD;">
17                 <p><?php echo $contenu;?></p>
18             </div>
19         </div>
20     </div>
21 <?php
22     }
23 ?>
```

jQuery : AJAX

jQuery et JSON :

- De manière surprenante, jQuery n'a jamais été bien pourvu pour dialoguer en JSON.
- Une extension (non utilisée ici), rajoute ces fonctionnalités.
- Principe de l'exemple suivant :
 - On construit en JavaScript un objet contenant :
 - Un tableau d'objets détaillant les langages du cours de Web
 - L'indice du langage actuellement affiché
 - Le code HTML de ce qui est à afficher
 - À chaque clic sur le bouton suivant :
 - L'objet est encodé pour être transmis en Ajax à PHP.
 - PHP incrémente l'indice du langage à afficher et fabrique le code HTML à afficher
 - PHP retourne le code JSON décodé en JavaScript pour récupérer le code HTML.

Exemple :

Document HTML

```
01 <html>
02     <head>
03         <title>20-Ajax</title>
04         <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05         <script type="text/javascript" src="20-Ajax.js"></script>
06     </head>
07     <body>
08         <input type="button" value="Suivant" onclick="LangagePHP();"/>
09         <div id="visu"></div>
10     </body>
11 </html>
```

jQuery : AJAX

Exemple :

Document JavaScript

```
01 var coursDeWeb = {  
02     langages:[  
03         {nom:'HTML',cours:1,TP:1},  
04         {nom:'CSS',cours:2,TP:1},  
05         {nom:'PHP',cours:2,TP:3},  
06         {nom:'JavaScript',cours:2,TP:2},  
07         {nom:'Ajax',cours:2,TP:0}],  
08     langageCourant:-1, ← Champ modifié par PHP  
09     html:''  
10 };  
11  
12 function LangagePHP(){  
13     $.post('langages.php',{data:JSON.stringify(coursDeWeb)},function(retour){  
14         coursDeWeb = JSON.parse(retour);  
15         $('#visu').html(coursDeWeb.html); ← Envoie tout l'objet coursDeWeb  
16     });  
17 }
```

Remarques :

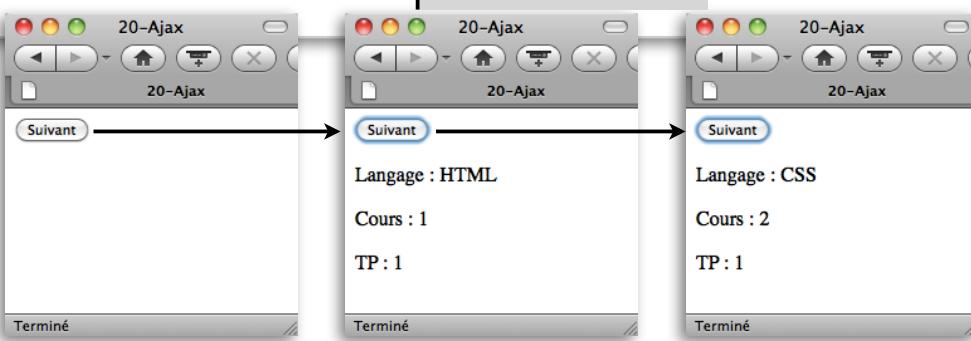
- La variable `coursDeWeb` est envoyée à PHP qui la modifie, et JavaScript récupère cette modification (ligne 14).
- Les champs `langageCourant` et `html` sont modifiés par PHP. `html` contient le fragment à présenter.

jQuery : AJAX

Exemple :

Script PHP

```
01 <?php  
02     if(!isset($_POST['data'])){  
03         die("Erreur");  
04     }  
05     $data = $_POST['data'];  
06     $objet = json_decode($data);  
07     $objet->langageCourant = ($objet->langageCourant+1)%count($objet->langages);  
08     $L = $objet->langages[$objet->langageCourant];  
09     $html = '<p>Langage : '.$L->nom.'</p>'; ← Fabrique le fragment HTML à  
10     $html .= '<p>Cours : '.$L->cours.'</p>'; ← retourner via le champ html  
11     $html .= '<p>TP : '.$L->TP.'</p>';  
12     $objet->html = $html;  
13     echo json_encode($objet); ← Encode le retour en JSON  
14 ?>
```



jQuery : Formulaires

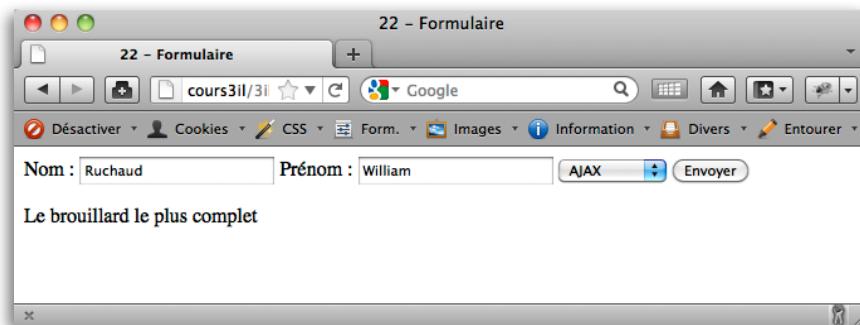
\$().serialize() :

- `$.serialize()` appelée à partir d'un formulaire retourne l'ensemble de ses champs sous la forme de paramètres d'URL (`champ1=valChamp1&champ2=valChamp2&...`).
- Très pratique pour effectuer une requête AJAX sans avoir à concaténer tous les champs à la main !

Principe de l'exemple :

- Un formulaire HTML demande un nom, un prénom et un langage de ce cours.
- JavaScript/jQuery utilise `$.serialize()` pour envoyer toutes les données du formulaire à PHP.
- Si le nom vaut «Ruchaud» et le prénom «William» PHP retourne une phrase, «Inconnu au bataillon» sinon.

Exemple :



Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 107

jQuery : Formulaires

Exemple :

Document HTML

```
01 <html>
02   <head>
03     <title>22 - Formulaire</title>
04     <script type="text/javascript" src="jQuery/jquery-1.7.2.js"></script>
05     <script type="text/javascript" src="22-Formulaire.js"></script>
06   </head>
07   <body>
08     <form id="form1">
09       <label>Nom : </label>
10       <input type="text" name="nom" />
11       <label>Prénom : </label>
12       <input type="text" name="prenom" />
13       <select name="langage">
14         <option>HTML</option>
15         <option>CSS</option>
16         <option>PHP</option>
17         <option>JavaScript</option>
18         <option>AJAX</option>
19       </select>
20       <input type="button" value="Envoyer" onclick="Envoyer()" />
21     </form>
22     <div id="texte"></div>
23   </body>
24 </html>
```

Introduction à Ajax, JSON et aux frameworks JavaScript - 3IL - William Ruchaud - Version 2.3 108

jQuery : Formulaires

Exemple :

Script JavaScript

```
01 function Envoyer(){
02     $.post('22-Formulaire.php', $('#form1').serialize(), function(retour){
03         $('#texte').html(retour);
04     });
05 }
```

Prend en compte tous les champs de form1

Script PHP

```
01 <?php
02     if(!isset($_POST['nom'])||!isset($_POST['prenom'])||!isset($_POST['langage'])){
03         die("Veuillez remplir votre formulaire");
04     }
05     if($_POST['nom']=='Ruchaud' && $_POST['prenom']=='William'){
06         switch($_POST['langage']){
07             case 'HTML' : echo 'Simple pour commencer'; break;
08             case 'CSS' : echo 'Moins facile'; break;
09             case 'PHP' : echo 'Il en a beaucoup d\'autres comme ca ?'; break;
10             case 'JavaScript' : echo 'Pourquoi se faire autant mal ?'; break;
11             case 'AJAX' : echo 'Le brouillard le plus complet'; break;
12         }
13     } else {
14         echo 'Inconnu au bataillon';
15     }
16 ?>
```