

Licence CRRW – IUT de Marne-la-Vallée  
02/11/2016  
Cours de jQuery

# *Cours 4*

## *AJAX avec jQuery*

# Sources

---

- Cours de Jean-Loup Guillaume

[http://jlguillaume.free.fr/www/documents/teaching/ntw1213/LI385\\_C5\\_Jquery.pdf](http://jlguillaume.free.fr/www/documents/teaching/ntw1213/LI385_C5_Jquery.pdf)

- Cours de programmation web avancée de Thierry Hamon

<https://perso.limsi.fr/hamon/PWA-20122013/Cours/JQuery.pdf>

- *jQuery, Le guide complet*, de Guillaume Allain et Timothy Stubbs

- *Javascript & Ajax pour les nuls*, d'Andy Harris

# Introduction à AJAX



= architecture informatique pour mettre à jour une page web côté client (navigateur) à partir d'informations du serveur sans la recharger

**asynchrone** = on n'attend pas le résultat des requêtes précédentes pour lancer les requêtes suivantes et la suite de l'exécution du code Javascript

# Introduction à AJAX



= architecture informatique pour mettre à jour une page web côté client (navigateur) à partir d'informations du serveur sans la recharger

**asynchrone** = on n'attend pas le résultat des requêtes précédentes pour lancer les requêtes suivantes et la suite de l'exécution du code Javascript

## **Atouts d'AJAX :**

- gérer les flux d'informations en temps réel
- permettre les outils web collaboratifs
- optimiser le temps de chargement, la bande passante

# Charger du contenu

Pour mettre du contenu dans un objet :

- une possibilité plus simple qu'avec les fonctions AJAX
- possibilité de ne charger qu'une partie du fichier (même si tout le fichier est récupéré dans ce cas, puis traité pour en extraire la partie voulue)

```
// version sans arguments :  
// appelle fichier.html et met le contenu dans div  
  
$("div").load("fichier.html");  
  
// version avec arguments : appelle fichier.php  
// en transmettant nom=philippe en POST  
  
$("div#content").load("fichier.php",  
{"nom":"philippe"});  
  
// version ne récupérant que l'objet d'id monid  
  
$("div").load('test.html #monid');
```

# Avec GET et POST

```
// récupère le fichier fichier.php  
// puis exécute une fonction
```

```
// GET : paramètres dans l'URL :
```

```
$.get(  
    "fichier.php",  
    {"nom": "philippe"},  
    function(data) {  
        alert(data);  
    });
```

```
// POST : paramètres dans l'entête du message  
// (caractères spéciaux, taille importante, etc.)
```

```
$.post(  
    "fichier.php",  
    {"nom": "philippe"},  
    function(data) {  
        alert(data);  
    });
```

# Les fonctions AJAX

```
// Fonction à appeler à la fin de la requête
.ajaxComplete()

// Fonction à appeler en cas de fin sur erreur
.ajaxError()

// Fonction à appeler au lancement de la requête
.ajaxStart()

// Fonction à appeler avant l'envoi
.ajaxSend()

// Fonction à appeler quand toutes les requêtes
// Ajax sont terminées
.ajaxStop()

// Fonction à appeler quand la requête termine
// avec succès
.ajaxSuccess()
```

# Contraintes de l'exécution asynchrone

Les événements AJAX ne sont pas liés à un appel en particulier :

→ il faut se souvenir des appels et retrouver d'où est venu l'appel

```
$( '.log' ).ajaxComplete(function(e, xhr, settings) {  
    if (settings.url == 'ajax/test.html') {  
        $(this).text('ok.');    }  
});
```

# Test des fonctions AJAX

## Code jQuery :

```
$('.log').ajaxStart(function() {$(this).append('Début.')});  
$('.log').ajaxSend(function() {$(this).append('Envoi.')});  
$('.log').ajaxComplete(function() {$(this).append('Fini.')});  
$('.log').ajaxStop(function() {$(this).append('Stop.')});  
$('.log').ajaxSuccess(function() {$(this).append('Succès.')});  
$('.trigger').click(function() {  
    $('.result').load('fichier.json');  
});
```

## Code HTML :

```
<div class="trigger">Trigger</div>  
  
<div class="result"></div>  
  
<div class="log"></div>
```

# Test des fonctions AJAX : alternatives

`[requete].done(function(data, textStatus, jqXHR){ }) :`  
fonction exécutée lorsque la requête AJAX [requete] réussit

`[requete].fail(function(jqXHR, textStatus, errorThrown){ }) :`  
fonction exécutée lorsque la requête AJAX [requete] échoue

`[requete].always(function(jqXHR, textStatus){ }) :`  
fonction exécutée dans tous les cas après done ou fail.

`jqXHR` : objet Javascript utilisé pour faire la requête AJAX

`textStatus` : chaîne de caractères qui décrit le statut de la requête  
(success, timeout, error, notmodified, parsererror)

`errorThrown` : l'erreur renvoyée s'il y en a une

`data` : la donnée renvoyée par le serveur

# Récupérer du JSON / du code Javascript

```
// Chargement et extraction d'informations  
// d'un fichier JSON
```

```
$.getJSON(  
    "fichier.json",  
    {id:1},  
    function(users) {  
        alert(users[0].name);  
    });
```

```
// Chargement et exécution d'un code Javascript
```

```
$.getScript(  
    "script.js",  
    function() {  
        ...;  
    });
```

# Fonction AJAX générique

Les fonctions `get`, `post`, `getJavascript`, `getJSON`, etc. sont des cas particuliers d'utilisation de la fonction `ajax` :

```
$.ajax({  
    async: false,  
    type: "POST",  
    url: "test.html",  
    data: "nom=JL",  
    success: function(msg) {  
        alert( "Data Saved: " + msg );  
    }  
});
```

# Fonction AJAX générique

```
$.ajax({  
    async: false,  
    type: "POST",  
    url: "test.html",  
    data: "nom=JL",  
    success: function(msg) {  
        alert( "Data Saved: " + msg );  
    }  
});
```

success équivalent à done

error équivalent à fail

complete équivalent à always