JavaScript

CM4 : XMLHTTPRequest

Mickaël Martin Nevot

V4.2.1

Cette œuvre de Mickaël Martin Nevot est mise à disposition sous licence Creative Commons Attribution - Utilisation non commerciale - Partage dans les mêmes conditions.

BY-NC-SA

JavaScript

I. Présentation
II. JS
III. Types/opérateurs
IV. Avancé
V. Ajax
VI. DOM
VII. XHR
VIII. JSON
IX. jQuery
X. JS/Web 2.0

 (cc) BY-NC-SA

HTTP : rappels

- HTTP (*hypertext transfer protocol*) :
  - Protocole de communication (client-serveur)
  - Méthodes (commandes) :

Utilisent très souvent une URL

    - **GET** : construction d'URL dynamique

http://localhost/login.php?login=login&pwd=test

    - **POST** : passage « caché » des valeurs

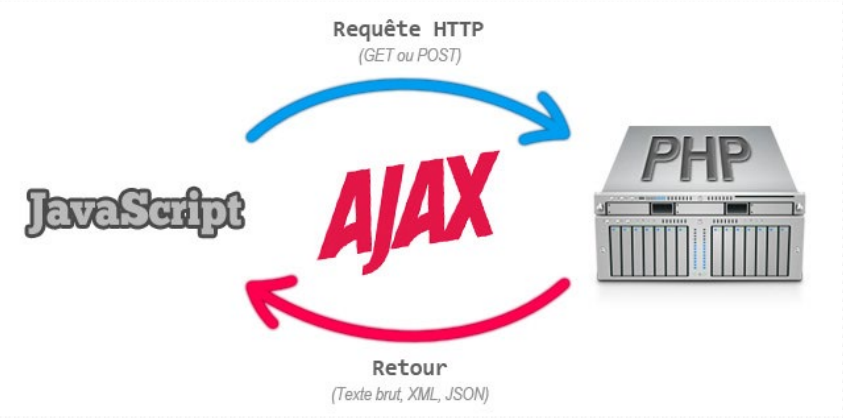
http://localhost/login.php

URL : de l'anglais *uniform resource locator*, littéralement « localisateur uniforme de ressource », est une chaîne de caractères utilisée pour adresser les ressources du Web, aussi appelée **adresse Web**

 BY-NC-SA

Synchrone/asynchrone

- **Synchrone** :
  - La fonction qui envoie une requête au serveur est la même que celle qui en recevra la réponse
  - Exécution suspendue en attendant la réponse du serveur
- **Asynchrone** : ← Attention : penser à avertir l'utilisateur !
  - La fonction qui envoie une requête au serveur n'est pas la même que celle qui en recevra la réponse
  - Non bloquant
  - Fonction de *callback*



 BY-NC-SA

XMLHTTPRequest

- Création du moteur Ajax

```
function createXHR() {
    // Mozilla, Safari, Opera,...
    try { return new XMLHttpRequest(); } catch(e) {}
    // Internet Explorer.
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0"); } catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0"); } catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP"); } catch (e) {}
    try { return new ActiveXObject("Microsoft.XMLHTTP"); } catch (e) {}
    alert("XMLHttpRequest non supporté");
    // Non supporté.
    return null;
}
```

 BY-NC-SA

XMLHTTPRequest

- Utilisation synchrone :

```
…
let anticache = new Date().getTime();
// Oblige le navigateur à mettre à jour.
// Le paramètre URL « anticache » peut être appelé autrement.
let param = "p1=foo&p2=bar" + "&anticache=" + anticache;
// Création de l'objet XMLHTTPRequest.
let XHRObj = createXHR();
// Ouvre une connexion synchrone avec le serveur en GET.
XHRObj.open("get", "server.php?" + param, false);
// Envoi de la « requête » au serveur en GET (toujours null).
XHRObj.send(null);
// Récupération de la réponse du serveur.
let res = XHRObj.responseText; // Ou XHRObj.responseXML…
```

Attention : Penser à protéger les variables (caractères spéciaux, espaces, etc.) : `encodeURIComponent()`

 BY-NC-SA

XMLHTTPRequest

- Utilisation asynchrone : Le plus souvent

```
function callback() {
    // 0 non init., 1 en chargement, 2 chargé, 3 en traitement, 4 terminé.
    if (XHRObj.readyState == 4) {
        // 404 page non trouvée, 403 accès refusé, 200 requête réussie, etc.
        if (XHRObj.status == 200) {
            alert("Réponse : " + XHRObj.responseText);
        }
    }
}
…
let param = "p1=foo&p2=bar"; // Pas besoin d'anti-cache.
// Création de l'objet XMLHTTPRequest.
let XHRObj = createXHR();
// Ouvre une connexion asynchrone avec le serveur en POST.
XHRObj.open("post", "server.php", true);
// Déclaration de la fonction de rappel.
XHRObj.onreadystatechange = callback;
// Internet media type: application/x-www-form-urlencoded, text/xml, etc.
XHRObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
// Envoi de la « requête » au serveur en POST (toujours un paramètre).
XHRObj.send(param);
```

 BY-NC-SA

XMLHTTPRequest

- Utilisation asynchrone : Le plus souvent

```
function callback() {
    // 0 non init., 1 en chargement, 2 chargé, 3 en traitement, 4 terminé.
    if (XHRObj.readyState == 4) {
        // 404 page non trouvée, 403 accès refusé, 200 requête réussie, etc.
        if (XHRObj.status == 200) {
            alert("Réponse : " + XHRObj.responseText);
        }
    }
}
…
let param = "p1=foo&p2=bar"; // Pas besoin d'anti-cache.
// Création de l'objet XMLHTTPRequest.
let XHRObj = createXHR();
// Ouvre une connexion asynchrone avec le serveur en POST.
XHRObj.open("post", "server.php", true);
// Déclaration de la fonction de rappel.
XHRObj.onreadystatechange = callback;
// Internet media type: application/x-www-form-urlencoded, text/xml, etc.
XHRObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
// Envoi de la « requête » au serveur en POST (toujours un paramètre).
XHRObj.send(param);
```

 BY-NC-SA

XMLHTTPRequest

- Côté serveur (PHP) :

```
// Type de la réponse : HTML, encodage : UTF-8.
header('Content-Type: text/html ; charsert=utf-8');
// Anti-cache pour HTTP/1.1.
header('Cache-Control: no-cache, private');
// Anti-cache pour HTTP/1.0.
header('Pragma: no-cache');

if (isset($_REQUEST['p1']) {
    $p1 = $_REQUEST['p1']);
}
if (isset($_REQUEST['p2']) {
    $p2 = $_REQUEST['p2']);
}

// Traitement PHP qui affecte le résultat à renvoyer à la variable $res.
// Formats possibles de $res : XML, JSON ou textuel (chaîne de caractères).

// Renvoi au client.
echo $res;
```

 (cc) BY-NC-SA

Promesses

- **Promesse** (ou *promise*) : objet renvoyé par une fonction asynchrone représentant l'état courant de l'opération
- `async` : rend la fonction **asynchrone**
- `await` : rend la fonction **synchrone** (contexte asynchrone)

```
fetch("http://mysite.com/my-ressource").then(function () {
    // Après avoir reçu la réponse.
    …
});

…

await fetch("http://mysite.com/my-ressource");
// Après avoir reçu la réponse.
…
```

Crédits

Auteur

Mickaël Martin Nevot
mmartin.nevot@gmail.com

Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com