Récupérer et traiter du texte

C'est assurément la façon la plus simple d'appréhender le concept AJAX. Elle reste néanmoins très riche au niveau des possibilités offertes aux développeurs.

Affichons, de façon asynchrone, une réponse sous forme de texte dans une zone de la page par la propriété `responseText`.

Soit une page HTML. Au clic de l'utilisateur sur le bouton de formulaire, la réponse est affichée dans l'élément `<div id=" affichage "> ... </div>`.

Le fichier texte (encodé en UTF-8) est :

```
César du meilleur film français : Guillaume Gallienne pour "Les
garçons et Guillaume, à table !"
```

Il porte le nom de cesar2014.txt.

Le fichier HTML se présente comme suit :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
</head>
<body>
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```

Élaborons le script pas à pas.

Au clic sur le bouton, la fonction `extraire()` est appelée.

```
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
```

Après avoir défini la variable `xhr` comme une variable globale, le script définit la requête HTTP vers le serveur de façon compatible avec les différents navigateurs. Ces lignes de script ont été largement abordées au chapitre L'objet XMLHttpRequest.

```
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var txtdocument = xhr.responseText;
afficher(txtdocument);
}
}
xhr.open("GET", "cesar2006.txt", true);
xhr.send(null);
}
}
```

La requête doit aller rechercher, selon la méthode GET, le fichier cesar2014.txt de façon asynchrone.

Au changement d'état de la requête (`onreadystatechange`), on s'assure tout d'abord qu'elle a bien abouti (`readyState == 4` et `status == 200`). Le fichier est renvoyé simplement comme un fichier texte par `responseText`. Le script passe alors la main à la fonction `affiche()` avec le fichier texte en argument (`afficher(txtdocument)`).

```
function afficher(txtdocument) {
var target = document.getElementById("affichage");
target.innerHTML = txtdocument;
}
```

La fonction `afficher()` repère tout d'abord la zone d'affichage pour la réponse, par son identifiant `getElementById("affichage");`. La cible ainsi déterminée, la réponse est alors affichée par la propriété `innerHTML`.

Le script complet devient donc :

```
<script>
function afficher(txtdocument) {
var target = document.getElementById("affichage");
target.innerHTML = txtdocument;
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n’est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var txtdocument = xhr.responseText;
afficher(txtdocument);
}
}
xhr.open("GET", "cesar2014.txt", true);
xhr.send(null);
}
}
</script>
```

Le document HTML complet devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<script>
function afficher(txtdocument) {
var target = document.getElementById("affichage");
target.innerHTML = txtdocument;
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n’est pas compatible avec AJAX...");
}
if(xhr) {
```

```
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var txtdocument = xhr.responseText;
afficher(txtdocument);
}
}
xhr.open("GET", "cesar2014.txt", true);
xhr.send(null);
}
}
</script>
<style>
#affichage { margin-top: 12px;}
</style>
</head>
<body>
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```

Le fichier se présente dès lors comme suit à l'écran :

Diverses sources recommandent de bien mettre en évidence la réponse fournie par l'approche AJAX, à l'intention des internautes débutants qui pourraient être pertubés dans leurs habitudes de navigation, ce qui peut se réaliser aisément par une feuille de style CSS.

```
<style>
#affichage { width: 250px;
             font-family: sans-serif;
             font-weight: bold;
             font-size: 16px;
```

```
            margin-top: 12px;
            padding: 5px;
            background-color: rgb(195,215,235);}
</style>
```

Ainsi, la zone d'affichage possède un arrière-plan de couleur et est délimitée par une bordure. En outre, la police de caractères est différente.

Le fichier HTML final prend la forme suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<script>
function afficher(txtdocument) {
var target = document.getElementById("affichage");
target.innerHTML = txtdocument;
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var txtdocument = xhr.responseText;
afficher(txtdocument);
}
}
xhr.open("GET", "cesar2014.txt", true);
xhr.send(null);
}
}
</script>
<style>
#affichage { width: 250px;
            font-family: sans-serif;
            font-weight: bold;
            font-size: 16px;
            margin-top: 12px;
            padding: 5px;
            background-color: rgb(195,215,235);}
</style>
</head>
<body>
```

```
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !

</div>
</body>
</html>
```

Il ne faut pas s'arrêter à une interprétation limitative de la propriété `responseText`. En effet, cela signifie que la réponse est renvoyée sous forme de texte, au sens informatique du terme. Tous les types de format sont acceptés : des fichiers HTML, des données traitées par du PHP en passant par du code JavaScript. Le tout dépend de l'utilisation ultérieure de ces données textuelles.

Soit un fichier js.txt qui contient le JavaScript : `alert("Message d'alerte provenant du serveur.")`.

Prenons un fichier HTML (js.htm). Ce fichier ne contient aucune instruction `alert()`. Pourtant, une boîte d'alerte peut être créée en exécutant le code JavaScript à partir de la chaîne de caractères contenue dans le fichier js.txt. Ce qui est effectué par la méthode JavaScript `eval()`.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<script>
function getxhr(){
var xhr = null;
if(window.XMLHttpRequest){
var xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
var xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

```
else {
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200) {
eval(xhr.responseText);
}
}
xhr.open("GET", "js.txt", true);
xhr.send(null);
}
</script>
</head>
<body>
<form>
<input type="button" value="Test" onclick="getxhr()">
</form>
</body>
</html>
```

Une boîte d'alerte s'affiche donc, alors qu'elle n'était pas présente dans le code de la page HTML.