JavaScript - QCM 10 (AJAX GET)

Adrien JOLY

Bonjour, Adrien JOLY

Date/heure de rendu: Wed Mar 29 2017 10:42:06 GMT+0200 (CEST)

Sign out

QCM

Question 1

Pour effectuer une requête AJAX GET, et afficher la réponse du serveur dans un `alert()`, il faut instancier la classe `XMLHttpRequest` puis...

- (●) ... appeler 2 méthodes, et définir 1 fonction
- ( ) ... appeler 1 méthode, et définir 2 fonctions
- ( ) ... appeler 3 méthodes
- ( ) ... définir 3 fonctions

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://jsonplaceholder.typicode.com/users/1');
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4) {
    alert(xhr.responseText);
  }
};
xhr.send();
```

Dans le code de cette requête, on:

- instancie la classe `XMLHttpRequest` dans la variable `xhr`,
- **définit 1 fonction** qu'on affecte à la propriété `onreadystatechange` de `xhr`,
- et on **appelle 2 méthodes** de `xhr` : `open()` et `send()`.

À noter que `alert()` est un appel de fonction, et non un appel de méthode, car cette fonction n'est pas rattachée à une instance de classe.

Question 2

J'ai écrit le code permettant d'envoyer une requête HTTP GET, mais rien ne se passe, et rien n'apparait dans la console.

Cela pourrait être dû à:

- ○ une erreur de syntaxe
- ○ une URL erronée
- ◉ l'oubli de l'appel à send()
- ○ l'oubli de l'usage de JSON.parse()

Dans ces trois cas, on aurait obtenu une erreur dans la console:

- *une erreur de syntaxe*
- *une URL erronée*
- *l'oubli de l'usage de* `JSON.parse()`

La bonne réponse est donc: *l'oubli de l'appel à* `send()`.

En effet, c'est cette méthode qui permet l'envoi de la requête. Sans cela, notre code n'aura eu aucun effet, sauf l'initialisation de la requête dans l'instance `xhr`.

Question 3

Quel est le format le plus couramment utilisé de nos jours pour échanger des informations en AJAX avec une API ?

- ○ HTML
- ○ XML
- ◉ JSON
- ○ texte brut

AJAX (*Asynchronous Javascript And XML*) a été initialement conçu pour échanger des informations au format XML, très en vogue dans les années 90, mais il permet d'utiliser n'importe quel format sérialisable sous forme d'une chaine de caractères.

Aujourd'hui, on utilise majoritairement le format *JSON* dans les requêtes AJAX, car il a l'avantage d'être concis (et peu consommateur de bande passante), facilement lisible à l'oeil nu, et directement manipulable en JavaScript.

Question 4

```
var xhr = new XMLHttpRequest();
xhr.open('GET', 'https://jsonplaceholder.typicode.com/users/1');
xhr.onreadystatechange = function() {
  if (xhr.readyState === 4) {
    var reponse = JSON.parse(xhr.responseText);
    alert(/* A SAISIR */);
  }
};
xhr.send();
```

Si on veut afficher la propriété `email` de l'objet contenu dans la réponse à notre requête, par quoi faut-il replacer `/* A SAISIR */` ?

- ○ JSON.parse(xhr.responseText.email)
- ○ xhr.responseText.email
- ◉ reponse.email
- ○ responseText['email']

- la propriété `xhr.responseText` est de type `string` (c'est la forme *sérialisée* de l'objet contenu dans la réponse à la requête), donc on ne peut pas utiliser directement la notation pointée pour accéder à la propriété `email` demandée.
- `JSON.parse()` a été appelé sur `xhr.responseText`, et l'objet résultant est stocké dans la variable `reponse`. Il n'est donc pas nécéssaire d'appeler `JSON.parse()` à nouveau.
- `responseText['email']` causerait une erreur car `responseText` est une propriété de l'objet `xhr`, et non une variable existante.

Il va donc falloir extraire la propriété `email` de l'objet contenu dans la variable `reponse` (qui a été désérialisé par `JSON.parse()`), en utilisant la notation pointée: `reponse.email`.