Développement d'Applications Web

Cours 6 : Exercises

Vincent Simonet, 2014-2015

Exercise 1: AJAX

Question 1.1

Écrivez une fonction `ajax` ayant le prototype suivant :

```
/** @param {string} url
    @param {function(Object)} fn
*/
function ajax(url, fn)
```

Cette fonction effectuera une requête AJAX GET vers l'URL `url`. La réponse est supposée avoir le format JSON. Si la requête réussit, la fonction `fn` sera appelée avec l'objet JSON en argument.

Question 1.2

Écrivez une fonction `ajax_2_parallel` ayant le prototype suivant :

```
/** @param {string} url1
    @param {string} url2
    @param {fn(Object, Object)} fn
 */
function ajax_2_parallel(url1, url2, fn)
```

Cette fonction effectuera deux requêtes AJAX GET vers les URLs `url1` et `url2`, en parallèle. Les réponses sont supposées avoir le format JSON. Si les deux requêtes réussissent, la fonction `fn` sera appelée avec les deux objets JSON en arguments (dans l'ordre).

Question 1.3

Écrivez une fonction `ajax_2_series` ayant le prototype suivant :

```
/** @param {string} url1
    @param {string} url2
    @param {fn(Object, Object)} fn
 */
function ajax_2_series(url1, url2, fn)
```

Cette fonction effectuera deux requêtes AJAX GET vers les URLs `url1` et `url2`, en série. Les réponses sont supposées avoir le format JSON. Si les deux requêtes réussissent, la fonction `fn` sera appelée avec les deux objets JSON en arguments (dans l'ordre).

Question 1.4

Écrivez une fonction `ajax_n_parallel` ayant le prototype suivant :

```
/** @param {Array.<string>} urls
    @param {fn(Array.<Object>)} fn
 */
function ajax_n_parallel(urls, fn)
```

Cette fonction généralise la fonction `ajax_2_parallel` en effectuant N appels AJAX en parallèle (un par URL contenue dans la liste `urls`). La fonction `fn` sera appelée avec la liste (ordonnée) des objets reçus en réponse.

Question 1.5

Écrivez une fonction `ajax_n_series` ayant le prototype suivant :

```
/** @param {Array.<string>} urls
    @param {fn(Array.<Object>)} fn
 */
function ajax_n_series(urls, fn)
```

Cette fonction généralise la fonction `ajax_2_series` en effectuant N appels AJAX en série (un par URL contenue dans la liste `urls`). La fonction `fn` sera appelée avec la liste (ordonnée) des objets reçus en réponse.

Exercise 2: JSONP

Question 2.1

Écrivez une fonction json ayant le prototype suivant :

```
/** @param {string} url
    @param {function(Object)} fn
*/
function jsonp(url, fn)
```

Cette fonction effectuera une requête JSONP vers l'URL `url`. Le serveur est supposé attendre un argument supplémentaire dans l'URL nommé `callback` et contenant le nom de la fonction de continuation (le "padding"). Cet argument n'est pas inclu dans l'URL passée en argument. Si la requête réussit, la fonction `fn` sera appelée avec l'objet JSON en argument.

Question 2.2

La fonction que vous avez écrite à la question 2.1 traite-t-elle des appels parallèles correctement? Si non, comment peut-on la modifier ?

Question 2.3

Comment peut-on implémenter des fonctions similaires à celles réalisées dans les questions 1.2 à 1.5 mais utilisant le protocole JSONP ?