TP 11: Récupération de données avec AJAX

Objectifs:

- Interroger une API du Web en JavaScript (ex: météo)
- Effectuer une requête `GET` avec `XMLHttpRequest`

Slides du TP

Effectuer une requête `GET` avec `XMLHttpRequest`

Introduction

Créé en 1995, le procédé AJAX (**A**synchronous **J**avascript **a**nd **X**ML) a été intégré aux navigateurs pour permettre aux sites Web d'être plus dynamiques.

Cette technique permet à une application JavaScript de:

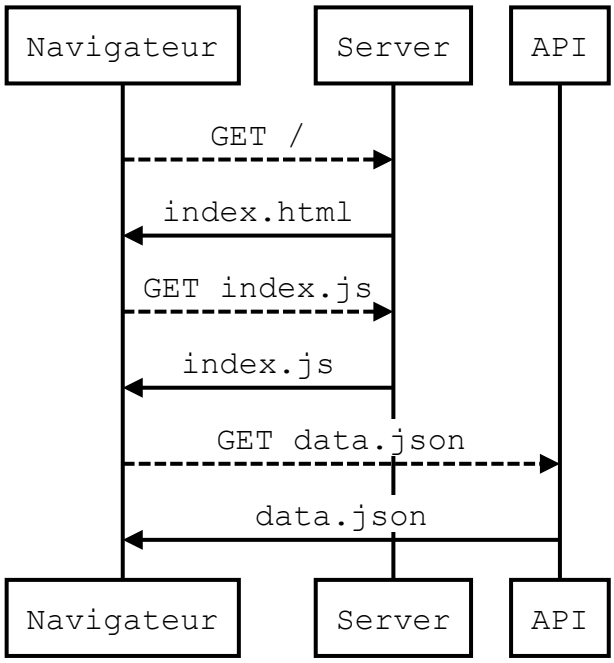
- **échanger** des données avec des serveurs Web;
- **charger** des données sur une page sans la recharger;
- accéder à des **API** (**A**pplication **P**rogramming **I**nterface) du Web.

Une requête AJAX consiste à effectuer une requête HTTP depuis du code JavaScript.

Pour rappel, HTTP est le protocole employé par les navigateurs pour communiquer avec les serveurs Web.

Par exemple, quand nous allons sur Google, nous tapons `http://google.com` dans la barre du navigateur, ce qui a pour effet de demander la page HTML de recherche au serveur HTTP de `google.com`. Ceci est une requête de type `GET`, car elle permet seulement de récupérer des données depuis ce serveur, contrairement aux requêtes `POST` permettant d'envoyer des données à un serveur.

A noter que le chargement d'une page HTML déclanche généralement d'autres requêtes HTTP, afin de récupérer toutes les ressources associées à cette page Web: scripts, feuilles de style CSS, images, etc...



Formats de données

Une requête HTTP permet de récupérer des données de formats variés.

Par exemple:

- le format HTML, pour les pages Web,
- le format CSS, pour les feuilles de styles,
- le format JS, pour les scripts en JavaScript,
- le format JPG, pour les images, etc...

Nous allons nous intéresser en particulier aux formats représentant des données structurées.

Initialement, AJAX a été créé pour récupérer des données au format XML. Un format proche du HTML, et très en vogue à l'époque.

Exemple de document XML:

```
<inbox>
  <email from="amazon">votre colis a été envoyé</email>
</inbox>
```

Comme en HTML, un document XML est composé d'éléments, d'attributs, et de noeuds textuels.

Désormais, la majorité des sites Web emploient le format JSON pour échanger des données structurées.

Exemple de document JSON:

```
[
  {
    "from": "amazon",
    "subject": "votre colis a été envoyé"
  }
]
```

Pour rappel, JSON (**J**ava**S**cript **O**bject **N**otation) est en fait la syntaxe utilisé pour définir des objets en JavaScript.

En supposant que cet objet JSON soit stocké dans une variable `objet`, le langage JavaScript nous permet d'accéder facilement aux données qui y sont stockées.

Par exemple, pour afficher la valeur de la propriété `subject` de la propriété `email`, il suffit de saisir:

```
console.log(objet.email.subject);
```

Émettre une requête HTTP GET avec AJAX

Pour récupérer des données depuis une URL HTTP, le navigateur met à notre disposition la classe `XMLHttpRequest`. Malgré son nom, il est possible de l'utiliser pour récupérer des données exprimées dans d'autres formats que XML.

Voici un exemple de requête HTTP GET simple, en JavaScript:

```
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();
```

Cette requête permet de récupérer le texte d'un document disponible à l'URL `https://jsonplaceholder.typicode.com/users/1`, puis de l'afficher dans un `alert()` une fois qu'il a été téléchargé intégralement.

Effectuer une requête avec `XMLHttpRequest` consiste en 4 étapes:

1. instancier la classe, avec `new`;
2. spécifier la méthode (`GET`) et l'URL du document à récupérer, en appelant la méthode `open()`;
3. définir ce qu'on souhaite faire de la réponse, une fois reçue intégralement, en affectant une fonction à la propriété `onreadystatechange` de notre instance;
4. puis envoyer la requête, en appelant la méthode `send()` de notre instance.

La propriété `readyState` permet de savoir où en est le téléchargement du document. Lorsque celui-ci est complet, `readyState` vaut `4`, et la fonction affectée à `onreadystatechange` est appelée à nouveau par le navigateur. À ce moment là, nous pouvons accéder au contenu du document téléchargé, stocké dans la propriété `responseText`.

À noter que, même si le document en question est au format JSON (par exemple), `responseText` contiendra la version *sérialisée* du document, c'est à dire qu'il sera de type `string`, et non `object`. À ce stade, il n'est donc pas possible d'accéder à une propriété spécifique de cet objet.

> Conseil: avant d'effectuer une requête AJAX GET, coller l'URL de destination dans un onglet de votre navigateur, il vous affichera la réponse à cette requête. (c.a.d. la valeur de `responseText`)

Dans la prochaine partie, nous allons voir comment convertir cette chaîne de caractères en véritable objet.

Récupérer un document JSON avec AJAX

Dans le cas où notre requête retourne un document JSON, il est possible de convertir sa représentation textuelle (fournie dans la propriété `responseText`) en véritable objet JavaScript.

Pour cela, le navigateur fournit la fonction `JSON.parse()`.

Cette fonction prend en paramètre une chaîne de caractères (type: `string`) contenant la version sérialisée d'un objet, et retourne l'objet correspondant (type: `object`).

Par exemple, imaginons l'objet sérialisé suivant:

```
var chaine = '{"message":"bonjour!"}';
console.log(chaine.message); // => undefined
```

Comme cet objet est sérialisé sous forme de chaine de caractères, il n'est pas possible d'accéder directement à ses propriétés.

Pour permettre cela, nous le passer en paramètre de la fonction `JSON.parse()`:

```
var chaine = '{"message":"bonjour!"}';
var objet = JSON.parse(chaine);
console.log(objet.message); // => "bonjour!"
```

Donc, pour récupérer un objet JavaScript depuis la réponse JSON de notre requête AJAX, il faut utiliser le code suivant:

```
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(reponse.name);
  }
};
xhr.send();
```

Bien entendu, n'oubliez pas de remplacer l'URL de cet exemple par celle à laquelle se trouve le document que vous souhaitez récupérer !

N'oubliez pas non plus de consulter la console de votre navigateur ainsi que la partie "Réseau" (ou "Network", en anglais) de Chrome Dev Tools, afin de diagnostiquer les éventuelles erreurs.

Exemples d'API interrogeables via AJAX

APIs simples:

- Adresse IP: httpbin.org/ip
- Heure: time.jsontest.com (*HTTP Only*)
- Utilisateur: jsonplaceholder.typicode.com/users/1

APIs plus complexes:

- Animations: Giphy
- Photos: Flickr
- Météo: openweathermap (doc) (*HTTP Only*)

- Chatons: puppygifs.tumblr.com/api/read/json (JSON var with CORS)

Annuaire d'APIs: Programmable Web

> Note: Certaines APIs nécessitent l'utilisation d'une clé API. Donc:
>
> - lisez la documentation de l'API,
> - créez un compte développeur sur le site de l'API, si besoin,
> - lisez les parties suivantes pour intégrer l'éventuelle clé API dans l'URL de votre requête AJAX.

Exercice 1: Récupérer l'adresse IP de l'utilisateur

Effectuer une requête AJAX vers l'API https://httpbin.org/ip puis afficher dans un `alert()` l'adresse IP fournie en réponse de cette requête.

Exercice 2: Météo de la ville de l'utilisateur

Réaliser une page Web permettant d'afficher la météo d'une ville saisie par l'utilisateur. Pour récupérer la météo, utiliser une requête AJAX vers l'API de openweathermap.

Étapes:

0. Effectuer une requête vers l'API fournie, après en avoir consulté la documentation
1. réaliser une page HTML contenant un champ de saisie et un bouton.
2. ajouter un code JavaScript qui affiche le contenu du champ quand on clique sur le bouton.
3. intégrer la requête AJAX correspondante à la première étape de cette exercice, de manière à ce que la réponse de requête soit affichée dans la console.
4. faire en sorte que l'appel AJAX s'adapte en fonction de la saisie, à chaque clic sur le bouton.

> Notes:
>
> - Pour accéder à cette API, vous aurez besoin de vous inscrire et de demander une clé API (*API Key*, en anglais).
> - Pour pouvoir exécuter votre requête AJAX vers cette API, vous aurez peut-être besoin de changer son URL de manière à utiliser le protocole `HTTPS` au lieu du protocole `HTTP`. (cf explication plus bas)

Exercice 3: Afficher les images demandées

Réaliser une page Web permettant d'afficher des images ou animations en rapport avec les mots-clés saisis par l'utilisateur. Pour récupérer l'URL des images, utiliser une requête AJAX vers l'API de Flickr ou de Giphy.

Les images doivent être affichées directement dans la page à l'aide d'éléments `<img>`, après la saisie de l'utilisateur. Pour cela, vous pouvez utiliser `innerHTML`, ou les méthodes présentées dans le chapitre sur la manipulation avancée du DOM.

<!--

> Solution: flickr, giphy
> < !-- ajax-get-images-flickr.html -- >
> < !-- ajax-get-images-giphy.html -- >
> -->

Restrictions de protocole HTTP / HTTPS

Comme vous le savez probablement, le protocole HTTP existe aussi en version sécurisée (**HTTPS**).

Pour des raisons de sécurité, les navigateurs empêchent le protocole HTTP d'être employé depuis une page Web affichée via HTTPS. Cette contrainte a des implications importantes sur le fonctionnement des requêtes AJAX.

Notamment:

- Une page ouverte en HTTPS (ex: jsfiddle) ne peut pas effectuer de requête AJAX vers des URLs HTTP (non sécurisée), seulement vers des URLs HTTPS.
- Si vous exécutez une requête AJAX depuis la console d'une page HTTPS, les mêmes constraintes s'appliquent.

Donc, dans la mesure du possible, assurez-vous d'utiliser le protocole HTTPS dans l'URL de vos requêtes AJAX.

Restrictions de domaines

En plus des restrictions de sécurité HTTPS, certaines APIs ne permettent pas l'exécution de requêtes AJAX depuis certains domaines. (ex: jsfiddle)

D'autres refusent carrément de répondre aux requêtes émises depuis d'autres domaines que le leur.

Dans ce cas, vous pourrez seulement exécuter des requêtes AJAX depuis une page de leur site Web, via la console.

Manipuler les paramètres d'une URL HTTP

Dans les exemples d'APIs "complexes" fournis plus haut, vous remarquerez que certaines URLs contiennent des paramètres, séparés par les caractères `?` puis `&`.

Exemple:

```
https://api.flickr.com/services/rest/?
method=flickr.photos.search&api_key=74a9b070d21072ccac3a7b5f44f09efa&tags=soccer
&format=json&nojsoncallback=1
```

Comme dans la définition de propriétés d'un objet JavaScript, chaque paramètre est composé d'un nom (aussi appelé *clé*) et d'une valeur. Dans une URL, le nom et la valeur de chaque paramètre sont séparés par un `=`.

Dans l'URL fournie ci-dessus en exemple, quatre paramètres sont transmis:

- le paramètre nommé `method` vaut `flickr.photos.search`,
- le paramètre nommé `api_key` vaut `74a9b070d21072ccac3a7b5f44f09efa`,
- le paramètre nommé `tags` vaut `soccer`,
- le paramètre nommé `format` vaut `json`,
- et le paramètre nommé `nojsoncallback` vaut `1`.

De le cas où vous voudriez passer la valeur d'une variable de votre programme JavaScript en paramètre de votre URL, vous pouvez utiliser la concaténation.

Exemple:

```
var tags = 'soccer';
var url = 'https://api.com/?tags=' + tags + '&format=json';
```

Attention: Ceci fonctionne bien quand la valeur de votre variable ne contient aucun caractère spécial. Par contre que se passerait-il si elle contenait un espace, ou un `&` ? L'URL deviendrait alors invalide...

Pour éviter que les caractères spéciaux éventuels d'une chaine de caractères ne puisse invalider votre URL, utilisez la fonction `encodeURIComponent()` de la manière suivante:

```
var tags = 'soccer';
var url = 'https://api.com/?tags=' + encodeURIComponent(tags) + '&format=json';
```

Identification du développeur et/ou de l'application

Vous aurez remarqué que certaines URLs contiennent un paramètre `api_key`, ou `appid`.

Exemple:

```
http://api.openweathermap.org/data/2.5/weather?
q=London,uk&appid=44db6a862fba0b067b1930da0d769e98
```

Ce genre de paramètre permet d'identifier l'application utilisant l'API, ou le développeur de cette application. En effet, les fournisseurs d'APIs souhaitent connaitre cette information, afin de contrôler l'accès et l'usage de leur API.

Pour obtenir une clé, rendez-vous sur le site Web de cette API, puis intégrez la clé fournie dans l'URL de votre/vos requête(s) AJAX.