KIT DE SURVIE JAVASCRIPT - TD #2

Pierre BIASUTTI
IUT Informatique, Licence Pro

1 jQuery et AJAX

1.1 Qu'est ce que l'AJAX?

Traditionnellement, la navigation sur Internet se traduit par le chargement successif de pages sur demande de l'utilisateur. Néanmoins, pour augmenter le dynamisme des sites et par soucis d'économie, il peut être intéressant de ne pas charger l'intégralité d'une page d'un site lors de la navigation. C'est à ce titre qu'intervient l'AJAX. Ce concept est illustré dans la Figure 1.

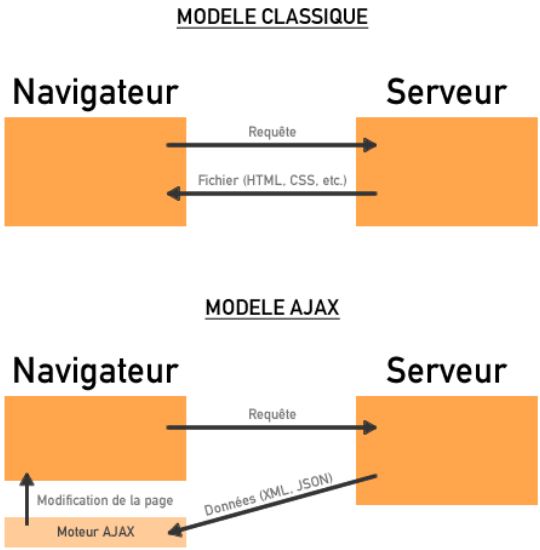

Figure 1: Différence entre le modèle classique de chargement de données web et le modèle AJAX.

L'AJAX (pour Asynchronous Javascript And XML) permet ainsi de ne télécharger qu'une portion d'une page, voire même que le contenu de la page sans la présentation, le plus souvent de manière asynchrone. Dans la plupart des cas, le schéma classique de chargement d'une page est suffisant. Cependant, pour certaines applications particulières, l'utilisation de l'AJAX permet d'économiser des ressources tout en améliorant l'expérience de l'utilisateur. Cette technique est particulièrement utilisée pour la mise à jour d'informations sur une page (comme sur Facebook), pour la saisie prédictive (Google) et dans bien d'autre cas. Enfin, l'AJAX permet de charger différentes sortes de données, de la page web au fichier de données XML en passant par des scripts Javascript.

AJAX et navigateurs Web L'utilisation de l'AJAX a, pendant longtemps, été extrêmement fastidieuse du fait des variations d'implémentations entre les navigateurs. Un code fonctionnel sur un navigateur n'était pas nécessairement compatible avec un autre, rendant l'utilisation de l'AJAX très instable. Bien que les moteurs Web se soient largement améliorés avec le HTML5 et la mise à jour du Javascript, tous les problèmes

de compatibilités n'ont pas été réglés. Par soucis de simplicité, nous utiliserons la bibliothèque jQuery pour la suite de ce TP.

Autorisation et Access-Control Pour des raisons de sécurité, les navigateurs Web empêchent le téléchargement de fichiers en local via AJAX. Ainsi, il n'est pas possible par défaut de charger via l'AJAX un fichier présent sur votre ordinateur. En effet, le navigateur vérifiera nécessairement le protocole via lequel le fichier va être envoyé, et n'acceptera en général que le HTTP ou autre protocoles Web.

D'autre part, le chargement de données d'un serveur à un autre est reglementé et par défaut désactivé. C'est ce que l'on appelle l'entête "Access-Control" d'une requête HTTP. Lorsqu'elle est présente dans un fichier que l'on souhaite télécharger depuis un autre serveur, elle indique au serveur emetteur de la requête que ce contenu est conçu pour être chargé de manière distante, via AJAX par exemple.

Ces notions d'autorisations sont très importantes car elles permettent de cloisonner l'utilisation de l'AJAX. Dans la suite du TP, les pages web présentées sont toutes hébergées sur des serveurs et envoyées avec le protocole HTTP et l'entête Acccess-Control réglée de sorte à ce que les fichiers soient téléchargeables via AJAX.

1.2 Les bases

Chargement simple jQuery comporte de nombreuses méthode liées à l'AJAX. La plus simple est la méthode ***.load***. Celle-ci permet de remplacer le contenu des éléments du selecteur précedent par celui d'un fichier donné en argument.

Exemple

```
$('selecteur').load('fichier'); // Charge le contenu de ''fichier'' dans ''selecteur''

// Exemple
$('div#conteneur').load('fichier.html');
```

Notons que la méthode ***.load*** est la même que la méthode permettant d'effectuer une action une fois le DOM chargé, comme vu dans le TP précédent. De ce fait, on peut effectuer une action un fois le contenu AJAX chargé en définissant un **callback**:

Exemple

```
$('div#conteneur').load('fichier.html', function(){
        alert('Le fichier est correctement chargé');
});
```

Enfin, la méthode ***.load*** permet aussi de selectionner directement la portion du fichier que l'on souhaite charger dans le conteneur HTML. Pour ce faire, il suffit de rajouter le selecteur du segment souhaité après l'url du fichier.

Exemple

```
$('div#conteneur').load('fichier.html #portion1');
```

Ici, seul le contenu de l'élément ayant pour **id** "portion1" sera chargé dans le conteneur html. Notons que peu importe l'appel, le fichier html est intégralement téléchargé, puis découpé pour n'ajouter que la portion choisie. L'AJAX ne permet le chargement partiel d'un fichier.

Exercice 1. A l'aide du fichier: *http://www.labri.fr/perso/pbiasutt/KDJS/TD02/exo1.php.*
a) Ecrire un script permettant de charger ce fichier au sein d'un *div* d'un fichier HTML.
b) A l'aide de l'inspecteur d'élément de votre navigateur (Chrome, Firefox), vous pouvez constater que l'intégralité du fichier a été copié dans le *div*, comme notamment les méta. Modifiez votre commande pour ne charger que le corps de la page.

Exercice 2. Créez un fichier HTML avec deux boutons.
a) En utilisant la page web de l'exercice précédent, écrivez un script qui charge le contenu du premier *div* de la page lorsque l'on clique sur le premier bouton, et du second *div* lorsque l'on clique sur le second bouton.
b) Rajoutez un indicateur de chargement qui ne s'affiche que pendant le chargement AJAX d'un div.

1.3 La méthode AJAX de jQuery

La méthode AJAX La manipulation de données récupérées via AJAX avec jQuery se fait au travers de la méthode ***.ajax***. Cette méthode ne nécessite pas de selecteur, elle s'appelle directement sur l'objet jQuery **$**.

Exemple

```
$.ajax({ parametre : valeur });
```

La méthode ***.ajax*** comporte de nombreux paramètres comme décrit dans la documentation à l'adresse suivante: *http://api.jquery.com/jquery.ajax/*. Nous verrons ici quelques paramètres indispensables au bon fonctionnement de cette méthode.

- **url**: l'url du fichier à charger
- **method**: la méthode HTTP qui permet de passer les arguments (GET ou POST)
- **data**: un tableau représentant les paramètres de la requête
- **dataType**: le type de réponse attendue (JSON, XML, HTML, Script)
- **beforeSend**: fonction anonyme s'executant avant l'envoi de la requête
- **complete**: fonction anonyme s'executant une fois la requête terminée
- **success**: fonction anonyme s'executant si la requête termine avec succès
- **error**: fonction anonyme s'executant si la requête a échouée
- **statusCode**: code HTTP de retour de la requête

Un exemple simple d'utilisation de la méthode ***.ajax***:

Exemple

```
$.ajax({
        url: ‘‘http://www.labri.fr/perso/pbiasutt/KDJS/TD02/exo1.php’’,
        dataType: ‘‘html’’,
        beforeSend: function(){ console.log(’Envoi de la requête ...’); },
        complete: function(data){
                console.log(’Réponse reçue’);
                console.log(data);
        },
        method: ‘‘GET’’
});
```

Note: il existe des méthodes ad-hoc pour chaque type de données de retour ou pour les différentes méthodes HTTP (***.get, .post, .getJSON, .getScript***, *http://api.jquery.com/category/ajax/shorthand-methods/*)

Exercice 3. Etudiez le fichier: *http://www.labri.fr/perso/pbiasutt/KDJS/TD02/functions.php.*
a) Ecrire un script permettant de charger la fonction “isPrime” contenue dans le fichier ci-dessus.
b) Créez un formulaire muni d'un champ de texte et d'un bouton, qui lorsqu'il est soumis, indique si la valeur du champ de texte est un nombre premier ou non.

Exercice 4. Etudiez le fichier json suivant: *http://www.labri.fr/perso/pbiasutt/KDJS/TD02/characters.php.* Celui-ci contient la plupart des personnages de Game Of Thrones. Ecrire un script permettant de charger le contenu de ce fichier. Une fois chargées, les données doivent être affichées dans une liste (ul, li).

A ce stade, vérifiez vos codes avec le chargé de TD

1.4 AJAX et APIs

Les APIs Une API (Application Programming Interface) est un ensemble de scripts permettant un accès sécurisé aux données d'un service (Web, programme, etc.). Les APIs sont très utilisées pour permettre l'échange et l'inclusion de données entre différents services web. Par exemple, Twitter offre une API qui permet de lire, poster et chercher des Tweets. Grâce à l'AJAX, on peut accéder à toutes ces informations de manière dynamique. De manière générale, les APIs nécessitent une authentification. Par soucis de simplicité, les APIs utilisées dans ce TP sont publiques et ne nécessitent aucune connexion/identifiants.

Une API renvoie généralement les données au format XML ou JSON.

Exercice 5. L'Institut Géographique National (IGN) propose une API permettant de chercher des communes en france (*https://api.gouv.fr/api/api-geo.html*, onglet "Technique").
a) Ecrivez un fichier HTML contenant un formulaire muni des éléments suivants:

- Un select avec les options "Nom", "Code postal", "Code département"
- Un champ de texte
- Un bouton

b) Ecrivez un script qui permet d'effectuer une recherche en utilisant l'API de l'IGN en fonction des paramètres du formulaire (par exemple: si "Code postal" est sélectionné, alors le champ de texte contient le code postal et la recherche se fait par code postal) c) Présentez les résultats dans des listes (ul) de manière lisible.

Vérifiez votre code avec le chargé de TD

Exercice 6. MetaWeather (*https://www.metaweather.com/*) possède une API permettant d'accèder aux prévisions météo du monde entier. Etudiez la documentation de l'API (*https://www.metaweather.com/api/*).
Dans le fichier "exo6.html":
a) Ecrivez un script qui, lorsque l'on soumet le formulaire (dans "exo6.html") effectue une recherche via l'API de MetaWeather, et affiche les résultats dans la liste sous le formulaire. L'identifiant de la ville (woeid) sera stocké dans l'attribut "data-id" de la balise *a* de chaque item.
b) Ecrivez un script qui, lorque l'on clique sur une des villes dans la liste sous le formulaire, charge les prévisions météo de la ville en question. On souhaite simplement afficher les informations des 6 prochains jours de la semaine.
Note: Les données retournées par l'API contiennent une entrée "weather_state_abbr" qui est une abréviation de la météo. Vous pouvez récuéprérer une icone correspondant à la météo à l'adresse suivante :
*https://www.metaweather.com/static/img/weather/png/64/**ABBREVIATION**.png.*

Exercice 7. An API of Ice and Fire (*https://anapioficeandfire.com/*) possède une API permettant d'accèder aux informations des différents personnages de la serie Game Of Thrones. Le fichier de l'exercice 4 contient la liste des personnages disponibles sur cette api, au format JSON. Dans le fichier "exo7.html":
a) Ecrivez un script qui charge la liste des personnages, et la stocke dans une structure de donnée apprioriée.
b) Ecrivez maintenant un script qui, lorsque l'on est dans le champ de texte du formulaire et que l'on presse une touche du clavier, affiche les personnages dont le nom contient le contenu du champ de texte (On cherche une à faire un autocomplétion à la manière de Google Search).
c) On souhaite maintenant qu'un clique sur un des résultat entraine le chargement et l'affichage des données du personnage dans le panel de droite. Les informations du personnages sont récupérées via l'API. On souhaite aussi qu'une icone de chargement s'affiche le temps que les données soient téléchargées depuis l'API.

Validez votre code avec le chargé de TD