VueJS

Les Requêtes Web

Qu'est ce qu'AJAX ?

Asynchronous **J**avaScript **A**nd **X**ML... Oui et encore ?

- Ni un nouveau langage, ni une nouvelle technologie.
- C'est une méthode côté client pour créer des applications web interactives.
- Le but est de récupérer des données sans interférer avec l'affichage de la page.
- Initialement (cf le nom) basé sur le XML
- Compatible avec les formats JSON, HTML ..

Dans les années 1990, le Web était basé sur des pages complètement en HTML :

- Toutes les actions rechargeaient la page. Cela cause une mauvaise expérience utilisateur.
- Temps de rechargement de page important
- Disparition de la page puis retour du contenu
- Ces problèmes sont moins fréquents maintenant avec la fibre, mais subsitent sur les connexions 3G des portables

Avec le fait de renvoyer l'intégralité de la page pour chaque changement, même minime, cela cause une consommation excessive de la bande passante des serveurs. Par exemple renvoyer tout le HTML et CSS pour ne modifier qu'une valeur....

- En 1995, Sun Microsystems introduit les Applets Java pour le chargement asynchrone.
- En 1996, Microsoft introduit l'élément iframe pour un chargement asynchrone.
- En 1999, Microsoft crée le composant XMLHTTPActivex.
- Le terme AJAX commence en 2005.
- En 2006, le W3C dévoile le premier draft pour l'objet `XMLHttpRequest`

Quels en sont les usages?

- Auto-complétion
- Auto-save
- Rechargement partiel
- Boites mails
- Les chats
- Réseaux sociaux

XMLHttpRequest

Get

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'https://www.site.com');
xhr.send(null);
```

Post

```
let xhr = new XMLHttpRequest();
xhr.open('POST', 'https://www.site.com');
xhr.send('id=3&name=bob');
```

ReadyState

La valeur `readyState` donne des informations sur la requête. Cinq valeurs sont possibles :

1. Uninitialized
2. Open
3. Send
4. Receiving
5. Finished

Le callback `onreadystatechange` est appelé pour chaque changement d'état de la propriété `readyState`. On peut récupérer la réponse du serveur dans la propriété response de l'objet xhr.

```
xhr.onreadystatechange = () => {
  if(xhr.readyState === 4 && xhr.status === 200){
  console.log(JSON.parse(xhr.response));
  }
};
```

En combinant tout, on obtient un code ressemblant à :

```
let xhr = new XMLHttpRequest();
xhr.onreadystatechange = () => {
  if(xhr.readyState === 4 && xhr.status === 200)
  {
    console.log(JSON.parse(xhr.response));
  }
};
xhr.open('GET', 'https://site.com');
xhr.send(null);
```

Exercice: 20min (step 5)

Effectuer un appel web via `xhr` sur https://api.github.com/users/:pseudo Remplacer :pseudo par un pseudo d'un utilisateur sur GitHub saisissable au sein d'un formulaire

** Exemple > exercices/ajax/xhr.html **

Axios

Axios est une bibliothèque qui fournit un client HTTP pour le navigateur ou pour NodeJS. Elle est conçue pour utiliser les Promises :

- basée sur XMLHttpRequest
- Transforme automatiquement en json
- Permet l'annulation d'une requête
- ...

Mais axios n'est pas de base dans les navigateurs, il faut l'installer. Dans le répertoire du projet faites:

```
npm install axios
```

Puis il faut l'importer juste après la balise script pour s'en servir

```
import axios from 'axios';
```

Une fois ces opérations faites, vous pouvez vous servir d'axios.

GET

```
axios.get('https://www.site.com')
.then( (reponse) => {
  console.log(reponse.data);
});
```

POST

```
axios.post('https://www.site.com', { age: 30 })
.then( (reponse) => {
  console.log(reponse.data);
});
```

Axios possède d'autres fonctionnalités. Vous pouvez aller consulter la documentation pour les découvrir. Axios est la solution recommandée par VueJS pour effectuer des requêtes web et est compatible sur tous les navigateurs.

Mais attention, utiliser Axios veut dire importer une bibliothèque.Un site plus lourd et donc plus long à charger. Un deuxième soucis émerge :

- Vous avez rajouté une dépendance à Axios. Celà peut être un risque si axios n'est plus maintenu / n'a plus de contributeurs.

Exercice : 20min

Afin d'utiliser facilement npm et les modules, il est très pratique d'utiliser webpack. Afin de ne pas s'attarder sur ce module, nous allons travailler dans un projet déjà buildé par le vue-cli.

Reprendre le code HelloWorld et y implémenter Axios.

** Exemple > exercices/ajax/axios-exercice **

Fetch

Enfin, la dernière solution est d'utiliser fetch. Cette fonctionnalité a été inclu dans la spécification ECMAScript de 2016 :

- Elle est valide dans les navigateurs depuis plus de 2 ans.
- C'est un standard
- Mais n'est pas supportée par Internet Explorer !

Get

```
fetch('https://www.site.com')
.then( (data) => data.json())
.then( (user) => {
  console.log(user);
});
```

Post

```
fetch('https://www.site.com', {
  method: "POST",
  body: JSON.stringify(elem),
})
.then( (data) => data.json())
.then( (user) => {
  console.log(user);
});
```

`fetch` c'est l'avenir, une fois que les sociétés auront enfin abandonnées Internet Explorer ! Comme Axios, fetch possède d'autres options pour répondre à tous les besoins. Vous pouvez consulter la documentation sur MDN.

Exercice: 20min

- Remplacer Axios par fetch
- A vous de décider votre solution préférée !

** Exemple > exercices/ajax/axios-exercice **
