JS – 19 - Utiliser Fetch

Fetch est une nouvelle API basée sur des promesses qui nous permet de faire des requêtes Ajax sans tout le tracas associé à **XMLHttpRequest** .

Comme vous le verrez dans cette partie, Fetch est très facile à utiliser et simplifie grandement la récupération des ressources à partir d'une API.
De plus, il est maintenant pris en charge dans tous les navigateurs modernes, donc utiliser Fetch est vraiment une évidence.

Requête `GET`

Démontrons une simple requête GET en récupérant nous-mêmes quelques données factices de l' API [JSONPlaceholder](https://jsonplaceholder.typicode.com/) :

```
fetch('https://jsonplaceholder.typicode.com/users')
  .then(res => res.json())
  .then(res => res.map(user => user.username))
  .then(userNames => console.log(userNames));
```

Et la sortie sera un tableau de noms d'utilisateurs comme celui-ci :

```
["Bret", "Antonette", "Samantha", "Karianne", "Kamren",
"Leopoldo_Corkery", "Elwyn.Skiles", "Maxime_Nienow", "Delphine",
"Moriah.Stanton"]
```

Étant donné que nous attendons une réponse JSON, nous devons d'abord appeler la méthode `json()` pour transformer l'objet `Response` en un objet avec lequel nous pouvons interagir.
Le `text()` pourrait être utilisé si nous attendions une réponse XML à la place.

Requêtes POST , PUT et DELETE

Pour effectuer des requêtes autres que **GET** , passez un objet en second argument à un appel fetch avec la méthode à utiliser ainsi que les en-têtes nécessaires et le corps de la requête :

```
const myPost = {
  title: 'A post about true facts',
  body: '42',
  userId: 2
}

const options = {
  method: 'POST',
  body: JSON.stringify(myPost),
  headers: {
    'Content-Type': 'application/json'
  }
};

fetch('https://jsonplaceholder.typicode.com/posts', options)
  .then(res => res.json())
  .then(res => console.log(res));
```

JSONPlaceholder nous renvoie les données POSTées avec un ID :

```
Object {
  body: 42,
  id: 101,
  title: "A post about true facts",
  userId: 2
}
```

Vous remarquerez que le corps de la requête doit être stringified . Les autres méthodes que vous pouvez utiliser pour récupérer les appels sont
DELETE, PUT, HEAD et OPTIONS

La gestion des erreurs (Error Handling)

Il y a un hic (jeu de mots 😉) en ce qui concerne la gestion des erreurs avec l'API Fetch : si la requête atteint correctement le point de terminaison et revient, aucune erreur ne sera générée.
Cela signifie que la gestion des erreurs n'est pas aussi simple que d'enchaîner avec l'appel d'un catch à la fin de votre chaîne de promesses fetch.

Cependant et bien heureusement, l'objet de réponse d'un appel `fetch()` aura une propriété ok qui sera soit true soit false en fonction du succès de la requête.
Vous pouvez ensuite utiliser la promesse `Promise.reject()` si la propriété ok est `false`:

```
fetch('https://jsonplaceholder.typicode.com/postsZZZ', options)
  .then(res => {
    if (res.ok) {
      return res.json();
    } else {
      return Promise.reject({ status: res.status, statusText:
res.statusText });
    }
  })
  .then(res => console.log(res))
  .catch(err => console.log('Error, with message:', err.statusText));
```

Avec l'exemple ci-dessus, notre promesse sera rejetée car nous appelons un point de terminaison qui n'existe pas.
L'appel catch chaîné sera atteint et les éléments suivants seront affichés :

```
"Error, with message: Not Found"
```

Fetch + Async/Await

Étant donné que Fetch est une API basée sur des promesses, l'utilisation de fonctions asynchrones est une excellente option pour rendre votre code encore plus facile à raisonner et à regarder de manière synchrone.
Voici par exemple une fonction **async/await** qui exécute une simple requête **GET** et extrait les noms d'utilisateur de la réponse JSON renvoyée pour ensuite enregistrer le résultat sur la console :

```
async function fetchUsers(endpoint) {
  const res = await fetch(endpoint);
  let data = await res.json();

  data = data.map(user => user.username);

  console.log(data);
}

fetchUsers('https://jsonplaceholder.typicode.com/users');
```

Ou, vous pouvez simplement renvoyer une promesse de votre fonction async/await et vous aurez alors la possibilité de continuer à enchaîner ensuite avec `then` après avoir appelé la fonction :

```
async function fetchUsers(endpoint) {
  const res = await fetch(endpoint);
  const data = await res.json();

  return data;
}

fetchUsers('https://jsonplaceholder.typicode.com/users')
  .then(data => {
    console.log(data.map(user => user.username));
  });
```

Un appel `json()` renvoie donc une promesse dans l'exemple ci-dessus ; lorsque nous avons `return data` dans la fonction async, nous renvoyons une promesse.

Et encore une fois, vous pouvez également générer une erreur si la réponse ok est false et la « catch » comme d'habitude dans votre chaîne de promesses :

```
async function fetchUsers(endpoint) {
  const res = await fetch(endpoint);

  if (!res.ok) {
    throw new Error(res.status); // 404
  }

  const data = await res.json();
  return data;
}

fetchUsers('https://jsonplaceholder.typicode.com/usersZZZ')
  .then(data => {
    console.log(data.map(user => user.website));
  })
  .catch(err => console.log('Ooops, error', err.message));

Ooops, error 404
```

Polyfills

- Si vous devez prendre en charge des navigateurs plus anciens, comme Internet Explorer 11, vous devrez utiliser un polyfill Fetch comme celui-ci de Github .
- Si vous devez utiliser Fetch dans Node.js, deux des options les plus populaires sont isomorphic-fetch et node-fetch .

Exercice 1

A partir de l'archive ***Pizza Départ.rar*** donné par votre formateur :
Nous allons configurer les informations d'une pizzeria en JSON.
Pour cela vous devrez dans un premier temps créer un fichier JSON avec les informations suivantes :

- Un nom
- Un slogan
- Un tableau de pizzas (minimum 3)
- Chaque pizza devra à son tour contenir: un nom, un prix, une image et une liste de garnitures parmi les suivantes:
  Ananas, Anchois, Bacon, Brocolis, Champignons, Extra fromage, Feta, Fromage bleu, Jambon, Jalapenos, Olives vertes, Olives noires, Oignons, Parmesan, Pepperoni, Piments forts, Poivrons verts, Salami, Saucisse et Tomates.

Lors de l'appui sur le bouton JSON, vous générerez visuellement la liste des pizzas avec les informations énumérés précédemment.
Les pizzas seront contenues dans des cards.

Exercice 2

Réaliser le projet "Gestion des ouvrages informatiques