Javascript et Fetch

Il est possible en javascript d'écrire nous même nos propres requêtes (http) et de recevoir des réponses directement depuis notre code.

Pour cela, une fonction universelle : **fetch** (vas chercher) !

Comment ça fonctionne

Fetch c'est une fonction très simple, qui accépte 1 ou 2 paramètres :

1. L'url qu'on veut allez chercher
2. De possible options

Cette fonction retourne une `Promise` d'un objet `Response`.

```
async function main() {
  // Je viens d'envoyer un requête au serveur du bon  coin !
  const response = await fetch('https://leboncoin.fr')
}
```

Par défaut javascript, envoie une requête `GET` et il spécifie les en-tête http classique d'un navigateur.

Manipuler la `Response`

Lorsqu'on fait requête fetch, on reçoit une réponse. Cette dernière contient les informations du petit fichier text de réponse du serveur :

```
async function main() {
  // Je viens d'envoyer un requête au serveur du bon  coin !
  const response = await fetch('https://leboncoin.fr')

  // On peut obtenir le code de status
  response.status // number 200

  // On peut récupérer l'en-tête HTTP
  response.headers['Content-Type'] // "text/html"

  // On peut récupérer le contenu de la réponse sous forme de stream
  // https://developer.mozilla.org/en-US/docs/Web/API/ReadableStream
  // C'est ç dire qu'on ne charge pas tout le contenue dans la mêmoire,
  // mais on peut piocher parmis des partis de se contenue.
  response.body

  // On peut obtenir le contenue de la requpete sous forme de string
  const content = await response.text()
}
```

Le format `JSON`

Le `JSON` est un format d'affichage des données, c'est le même principe qu'HTML mais en beaucoup beaucoup beaucoup plus simple et léger !

HTML utilise un système de balise pour formater les informations :

```
<article>
  <h2>Titre</h2>
  <p>Contenue ...</p>
</article>
```

Ce sont ce que l'on nomme des langages de **semantic**. C'est la même chose pour JSON, mais en plus léger, il s'inspire de la syntax Javascript

```
{
  "titre": "Titre",
  "contenue": "Contenue ..."
}
```

Aujourd'hui **JSON** est le format le plus utilisé sur le web !

Il a l'avantage d'être plus conscie, donc plus léger, plus rapide et diréctement compréhensible par javascript !

C'est un format inspiré de javascript permettant de transmettre des données, il est très simple :

```
"" // string
125131 // number
true // boolean
["dkhfsdkhf"] // array
{ "foo": "bar" } // objet
```

> Ça ressemble à du **javascript** mais ce n'en n'est pas ! on peut pas faire de boucles, de conditions, function, block de code etc ... et la syntaxe est légérement différente

```
// Exemple en js
{
  nom: "john",
  prenom: "Doe",
  age: 32,
  notes: [128, 9, 17],
}
```

```
// exemple en JSON
{
  // les clefs contiennent des guillemets double
  "nom": "john",
  "prenom": "Doe",
  "age": 32,
  // La virgule finale n'est pas autorisé !
  "notes": [128, 9, 17]
}
```

Manipuler `JSON` en Javascript

Il est possible en javascript de convertir une chaîne de caractère avec du json en objet javascript :

```
const json = `
{
  "nom": "Doe",
  "prenom": "John",
  "age": 24
}
`

// Si je fais
json.nom // Erreur !!! Parce que json est une string

// Je convertie le json en objet javascript :
const eleve = JSON.parse(json)

// Eleve devient un objet JS
eleve.nom // "Doe"

// On peut aussi passer d'un objet JS à une chaîne de caractère JSON :
const prof = {
  nom: 'Doe',
  prenom: 'Jane',
  age: 44,
}

// On convertit l'objet prof en JSON :
const profJson = JSON.stringify(prof)
```

Obtenir du json depuis une `Response` de `fetch`

Dans la plupart des cas, lorsque l'on réalise un requête à un serveur en utilisant `fetch`, le serveur nous renvoit du `JSON`. C'est tellement courant que `fetch` à penser à tout :

```
async function main() {
  // Je viens d'envoyer un requête au serveur du bon  coin !
  const response = await fetch('https://api.leboncoin.fr/finder/search')

  // Je récupére l'objet javascript depuis le contenue json de la requête
  const data = await response.json()

  console.log(data.total)
  console.log(data.aggregations.energy_rate.b)
}
```

Les options de `fetch`

Le fonction `fetch` accépte un deuxième paramètre, ce sont des options. C'est un objet dans lequel nous pouvons spécifier des informations supplémentaire pour notre requête :

```
async function main() {
  // Je viens d'envoyer un requête au serveur du bon  coin !
  const response = await fetch('https://api.leboncoin.fr/finder/search', {
    // Je spécifie des options
    // J'utilise la méthode HTTP `POST`
    method: 'POST',
    // Je rajoute des headers
    headers: {
      // On ajoute l'en-tête http User-Agent
      'User-Agent': 'Pirate Moi',
      // On ajoute le type de contenue, me permettant
      // d'envoyer du json au serveur
      'Content-Type': 'application/json',
    },
    // Je peux aussi envoyer des données au serveur
    // et notament en json :
    body: JSON.stringify({
      filters: {
        category: 10,
      },
    }),
  })

  // Je récupére l'objet javascript depuis le contenue json de la requête
  const data = await response.json()

  console.log(data.total)
  console.log(data.aggregations.energy_rate.b)
}
```

Pour aller plus loin

N'hésitez à consulter la document officiel de fetch :

[Documentation officiel de fetch]