

Javascript - ajax

Auparavant utilisées via l'objet **XmlHttpRequest** des navigateurs, les requêtes partielles se font maintenant en utilisant l'api **fetch**.

Promise

Les promesses permettent d'écrire du code asynchrone, pour les méthodes qui nécessitent un temps d'exécution et ne peuvent retourner immédiatement un résultat.

Promise

Exemple : Chargement d'un script

```
function loadScript(src) {
  return new Promise(function(resolve, reject) {
    let script = document.createElement('script');
    script.src = src;

    script.onload = () => resolve(script);
    script.onerror = () => reject(new Error(`Script load error for ${src}`));

    document.head.append(script);
  });
}
```

- **resolve** permet d'indiquer que la promesse est résolue
- **reject** qu'elle n'a pas abouti

Utilisation

```
let promise =
loadScript("https://cdnjs.cloudflare.com/ajax/libs/lodash.js/4.17.11/lodash.js");

promise.then(
  script => alert(`${script.src} is loaded!`),
  error => alert(`Error: ${error.message}`)
);
```

async/await

L'utilisation de **async** et **await** permet d'utiliser les fonctions asynchrones sans obligation de chaîner les promesses (pas besoin de mettre le code dans le retour **then**).

```
function resolveAfter2Seconds() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // Expected output: "resolved"
}

asyncCall();
```

API Fetch

L'api fetch permet en javascript d'effectuer des requêtes partielles vers le serveur, sans changer de page, selon le mode AJAX (Asynchronous Javascript And XML).

fetch fonctionne selon la syntaxe suivante :

```
let promise = fetch(url, [options])
```

- Requête vers une **url** cible
- en passant un tableau d'**options** (méthode, headers...)

Sans options, **fetch** effectue un simple **GET** téléchargeant le contenu.

Lors de l'utilisation de **fetch**, la requête est envoyée vers le serveur et renvoie une promesse qui pourra être utilisée via un objet **Response**.

La réponse peut être analysée (via ses membres **status** ou **ok**, et on obtient son corps fourni sous la forme d'une seconde promesse :

```
let response = await fetch(url);

if (response.ok) { // Statut dans les 200
  // obtenir le corps de réponse au format JSON
  let json = await response.json();
} else {
  alert("HTTP-Error: " + response.status);
}
```

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/web/js/ajax?rev=1680429856>

Last update: **2023/04/02 12:04**

