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