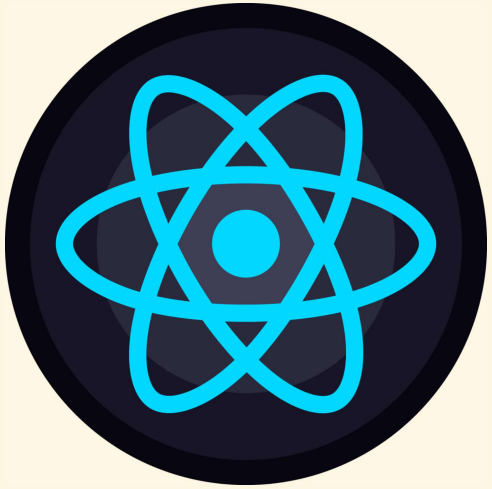
REACT : L'APPEL DE DONNÉE



L'APPEL DE DONNÉE

- On sait récuperer des données depuis un objet TS.
- Mais comment faire pour les récuperer depuis une API ?
- Ou pour modifier, ou enregistrer des données sur une API ?

FETCH

- La méthode fetch() permet de récupérer des données depuis une API.
- Elle retourne une promesse qui contient la réponse de l'API.

DÉFINITION

- Fetch est une API JavaScript qui permet de faire des requêtes HTTP
- Elle est disponible dans les navigateurs modernes
- Elle est basée sur les Promesses
- Elle est disponible dans Node.js

LES TYPES DE REQUÊTES

- **GET** : récupérer des données
- **POST** : envoyer des données
- **PUT** : mettre à jour des données (écrase les données existantes)
- **PATCH** : mettre à jour des données (modifie les données existantes)
- **DELETE**: supprimer des données

SYNTAXE

```
fetch(url, options)
```

UTILISATION

```
fetch('https://api.github.com/users/github')
  .then(response => response.json())
  .then(data => console.log(data));
```

Sans passer d'option, fetch() effectue une requête GET

DÉMONSTRATION

Afficher une liste de superHéros

LES OPTIONS

- **method**: GET, POST, PUT, PATCH, DELETE
- **headers** : objet contenant les en-têtes HTTP (option / configuration)
- **body** : corps de la requête
- **mode** : cors, no-cors, same-origin

HEADERS

Les differentes options possible

- **Accept** : type de contenu attendu
- **Content-Type** : type de contenu envoyé
- **Authorization** : authentification
- **Cache-Control** : cache
- **Cookie** : cookie
- ...

REQUÊTES POST

Pour faire une requête POST, ou une requête avec une autre méthode, nous devons utiliser les options fetch

- **method** : POST, PUT, PATCH ou DELETE
- **headers** : objet contenant les en-têtes HTTP
- **body** : corps de la requête

BODY

- **FormData** : pour envoyer des données au format multipart/form-data
- **Blob** : pour envoyer des données au format blob
- **BufferSource** : pour envoyer des données au format ArrayBuffer, ArrayBufferView ou DataView
- **JSON** : pour envoyer des données au format JSON
- **text** : pour envoyer des données au format texte

Le format Json est le plus utilisé

EXEMPLE

```
fetch('https://jsonplaceholder.typicode.com/users', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    name: 'Hubot',
    login: 'hubot',
  })
})
  .then(response => response.json())
  .then(data => console.log(data));
```

EXEMPLE

- Veuillez noter que si la requête body est une chaîne de caractères, alors l'en-tête Content-Type est défini sur text/plain;charset=UTF-8 par défaut.
- Mais, si nous envoyons du JSON, nous utiliserons à la place l'option headers pour envoyer application/json, le bon Content-Type pour les données encodées en JSON.

DÉMONSTRATION

EXERCICE 1

Créer un composant

LA SUITE

Par ici !