JAVASCRIPT – API REST

BASES ET CONCEPTS

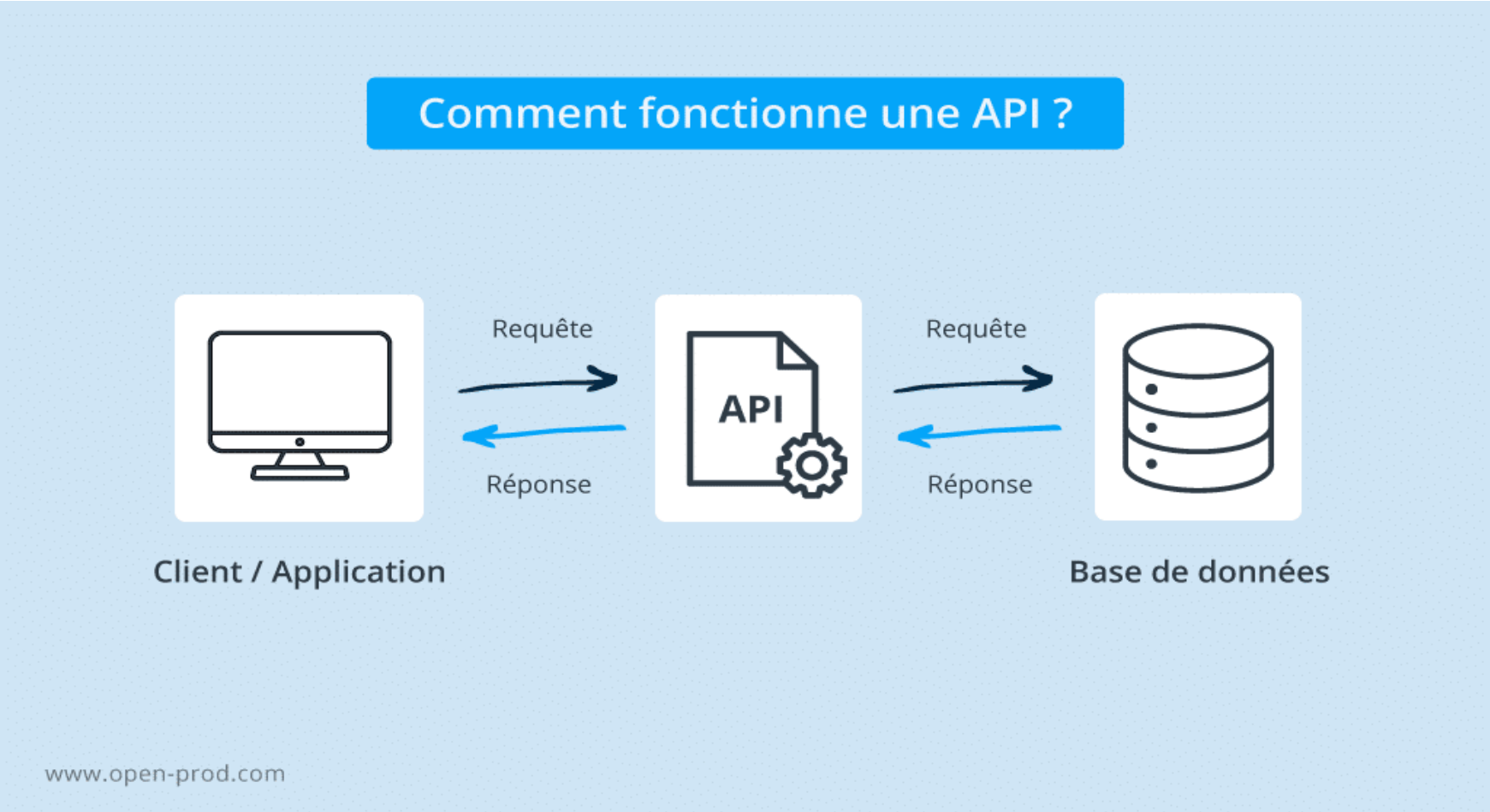
DÉFINITION

UNE API (APPLICATION PROGRAMMING INTERFACE OU « INTERFACE DE PROGRAMMATION D'APPLICATION ») EST UNE INTERFACE LOGICIELLE QUI PERMET DE « CONNECTER » UN LOGICIEL OU UN SERVICE À UN AUTRE LOGICIEL OU SERVICE AFIN D'ÉCHANGER DES DONNÉES ET DES FONCTIONNALITÉS.

UNE API REST (REPRESENTATIONAL STATE TRANSFER) EST UNE API DONT L'ARCHITECTURE SATISFAIT DES PRINCIPES DIRECTEURS SPÉCIFIQUES.

blocdroid

DÉFINITION



blocdroid

EXEMPLES

Si on saisit cette url dans le navigateur :

```
https://api.quotable.io/random
```

On obtient, par exemple :


```
{"_id":"HkgF9bFkMa5","content":"Knowledge is proud that it knows so much; wisdom is humble that it knows no more.","author":"William Cowper","tags":["Wisdom"],"authorSlug":"william-cowper","length":81,"dateAdded":"2019-02-21","dateModified":"2023-04-14"}
```


blocdroid

EXEMPLES

Si on saisit cette url dans le navigateur :

```
https://picsum.photos/1600/1000
```

On obtient, par exemple :

blocdroid

LA MÉTHODE FETCH()

LA MÉTHODE FETCH() PREND UN ARGUMENT OBLIGATOIRE :

LE CHEMIN DE LA RESSOURCE QU'ON SOUHAITE RÉCUPÉRER.

ELLE PREND UN SECOND ARGUMENT FACULTATIF SOUS LA FORME D'UN OBJET PRÉCISANT :

- LA MÉTHODE D'ENVOI
- DES ENTÊTES
- DES DONNÉES D'ENTRÉE, ETC.

ELLE RENVOIE COMME RÉPONSE UN OBJET DE TYPE PROMISE (UNE PROMESSE)

blocdroid

LA MÉTHODE FETCH()

POUR **RÉCUPÉRER UNE PROMESSE**, ON UTILISE LA MÉTHODE « THEN »

```
fetch("https://www.une-url.com")
.then( (response) => response.json())
.then( (response) => alert(JSON.stringify(response)))
.catch( (error) => alert("Erreur : " + error));
```

ON **NE PEUT PAS EXPLOITER** LA RÉPONSE RENVOYÉE DANS CETTE PROMESSE EN L'ÉTAT :

IL FAUT INDIQUER LE FORMAT DE RÉPONSE SOUHAITÉ. ICI, ON CHOISIT JSON AVEC REPONSE.JSON().

reponse.json() envoie également une promesse contenant la réponse à la demande de traduction en JSON

On traite les erreurs avec le bloc catch

blocdroid

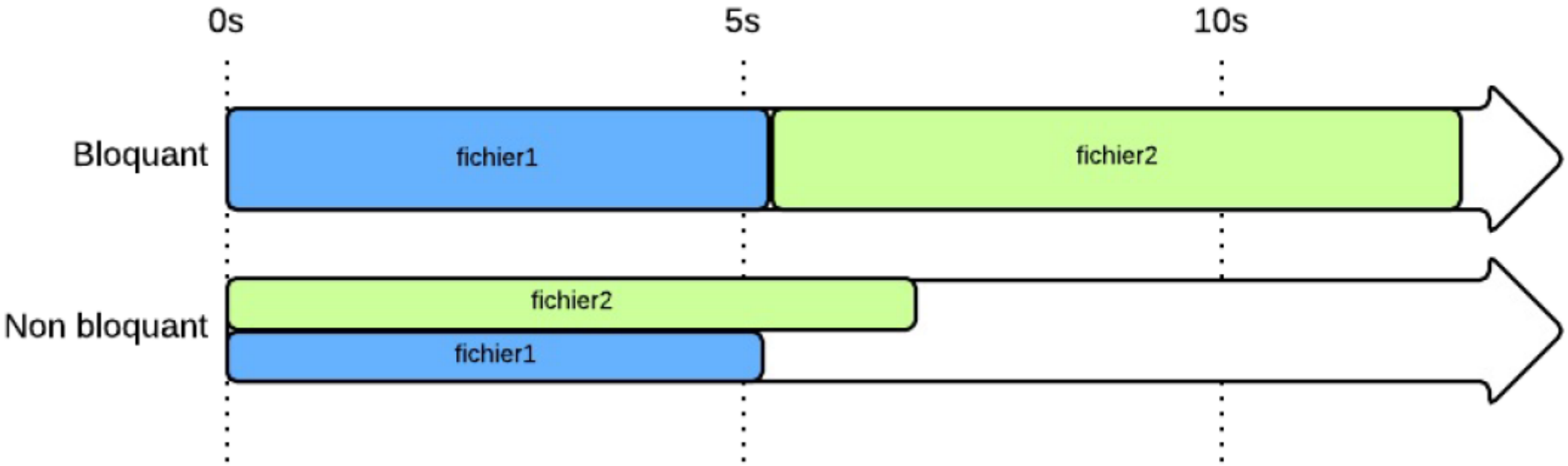
EXEMPLES

```
fetch('https://api.quotable.io/random')
    .then((response) => response.json())
    .then((data) => {
        quote.innerHTML = data.content + "<br>" + data.author;
    });
```

```
fetch('https://picsum.photos/1600/1000')
    .then((res) => {
        pic.innerHTML = `<img src=${res.url}>`;
    });
}
```

blocdroid

FONCTIONNEMENT ASYNCHRONE



blocdroid

ASYNC / AWAIT

LE MOT-CLE ASYNC SE PLACE DEVANT UNE FONCTION

LE MOT-CLE ASYNC DEMANDE LE RENVOIE D'UNE PROMESSE :

```
async function f() {
    return 1;
}

f().then(alert);
```

blocdroid

ASYNC / AWAIT

LE MOT-CLE AWAIT NE FONCTIONNE QUE DANS LES FONCTIONS ASYNCHRONES

LE MOT-CLE AWAIT NE FONCTIONNE PAS DANS LES FONCTIONS RÉGULIÈRES

```
async function MyAsyncFct() {

    // lire notre JSON
    let response = await fetch('url-to-your-json-file.json');
    let user = await response.json();
    // ...
}

MyAsyncFct();
```

blocdroid

GESTION DES ERREURS

```
async function f() {

  try {
    let response = await fetch('/no-user-here');
    let user = await response.json();
  } catch(err) {
    // attrape les erreurs à la fois dans fetch et response.json
    alert(err);
  }
}

f();
```

blocdroid