




# La Plateforme

/ Fetch API

# Sommaire

- Monothread et synchrone
- La boucle d'évènements
- Des promesses, encore des promesses
- Async / Await
- XMLHttpRequest (XHR)
- Fetch



**JS**

## **Javascript :**

- **Monothread**
- **Synchrone**
- **Non bloquant**



### Callstack

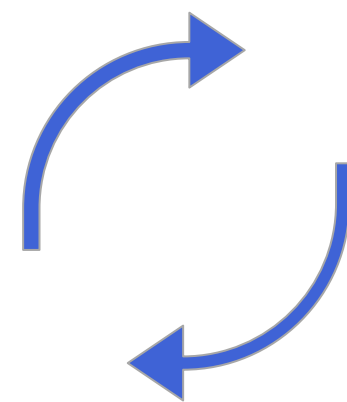
anotherFunction()

functionTimeout()

mySuperFonction()

### WebAPI

```
setTimeout() => {  
  console.log('delayed')  
}, 500);
```



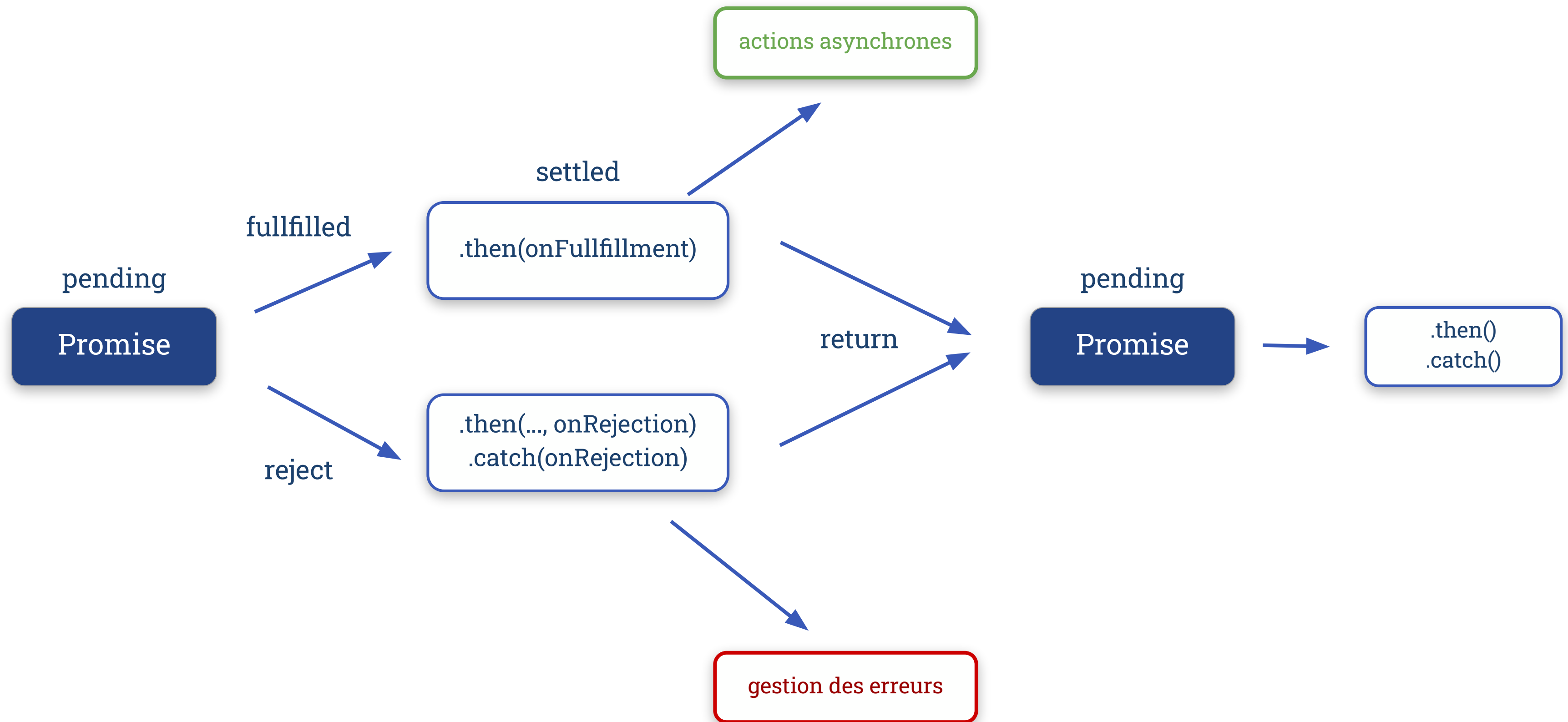
### Queue

console.log('delayed')

# Promise

## Des promesses, encore des promesses

- Permettent de changer le comportement de Javascript
- Rendent le code asynchrone
- Elle s'exécute dans la partie WebAPI
- Retourne une promesse
- 3 états :
  - pending
  - rejected
  - fulfilled





```
const test = async () => {  
  console.log('Async')  
}  
  
console.log('Synchrone 1');  
  
setTimeout(() => console.log('Timeout'), 0)  
  
Promise.resolve().then(() => console.log('Promesse'))  
  
test().then(() => {console.log('Then')})  
  
console.log('Synchrone 2')
```



```
const test = async () => {  
  console.log('Async')  
}  
  
console.log('Synchrone 1');  
  
setTimeout(() => console.log('Timeout'), 0)  
  
Promise.resolve().then(() => console.log('Promesse'))  
  
test().then(() => {console.log('Then')})  
  
console.log('Synchrone 2')
```



#Output

Synchrone 1

Async

Synchrone 2

Promesse

Then

Timeout



# Async

- Permet de rendre l'exécution d'une fonction asynchrone
- Retourne toujours une promesse
- Permet de grandement simplifier l'écriture des promesses

# Await

- Permet d'attendre la résolution d'une promesse avant le reste de l'exécution du code
- **S'utilise uniquement dans une fonction asynchrone**

# XMLHttpRequest

- Permet de récupérer des données serveurs sans rafraichir la page
- Utilisé dans la méthode AJAX (*Asynchronous Javascript and XML*)
- Renvoie une promesse



```
let oReq = new XMLHttpRequest();

oReq.open("GET", url, true);
oReq.responseType = "arraybuffer";
oReq.onload = function(e) {
    let arraybuffer = oReq.response; // n'est pas responseText
    /* ... */
}
oReq.send( );
```

# Fetch API

- Repose sur XHR
- Interface plus simple et plus puissante
- Retourne une promesse





```
const myImage = document.querySelector('img');

fetch('flowers.jpg')
  .then(function (response) {
    return response.blob();
  })
  .then(function (myBlob) {
    const objectURL = URL.createObjectURL(myBlob);
    myImage.src = objectURL;
  });
```

# Démonstration

THE

ASYNC

AWAIT

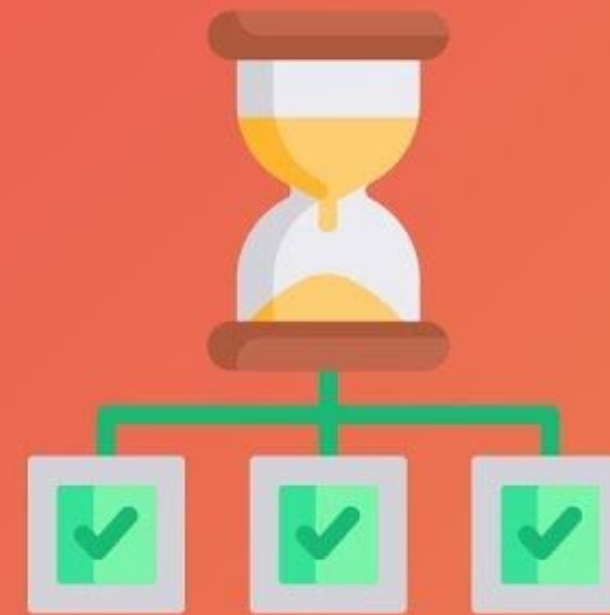
episode

Promise

pro

tips

JS



143







DEV

Search...



Log in

Create account



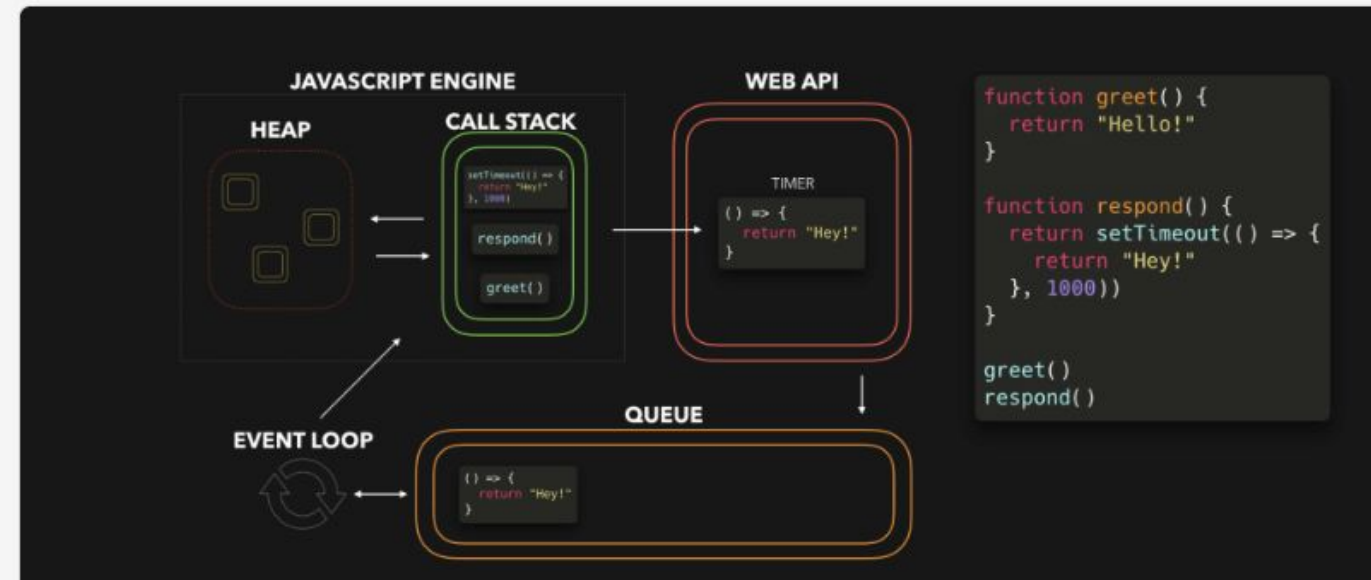
2582



146



4003



Lydia Hallie

Posted on 20 nov. 2019 • Updated on 16 janv. 2020



## JavaScript Visualized: Event Loop

#javascript

### JavaScript Visualized (7 Part Series)

1 🌟♻️ JavaScript Visualized: Event Loop

2 🔥👤 JavaScript Visualized: Hoisting

... 3 more parts...



Lydia Hallie

Follow

Software eng who likes to visualize technical concepts in my free time 🧑‍💻 (I use Keynote..)

JOINED  
21 juil. 2019

### More from Lydia Hallie

🌟👤 JavaScript Visualized: Promises & Async/Await  
#javascript #node #webdev

🌟 Interactive JavaScript Quiz #1  
#javascript

💡👤 JavaScript Visualized: Generators and Iterators  
#javascript #node