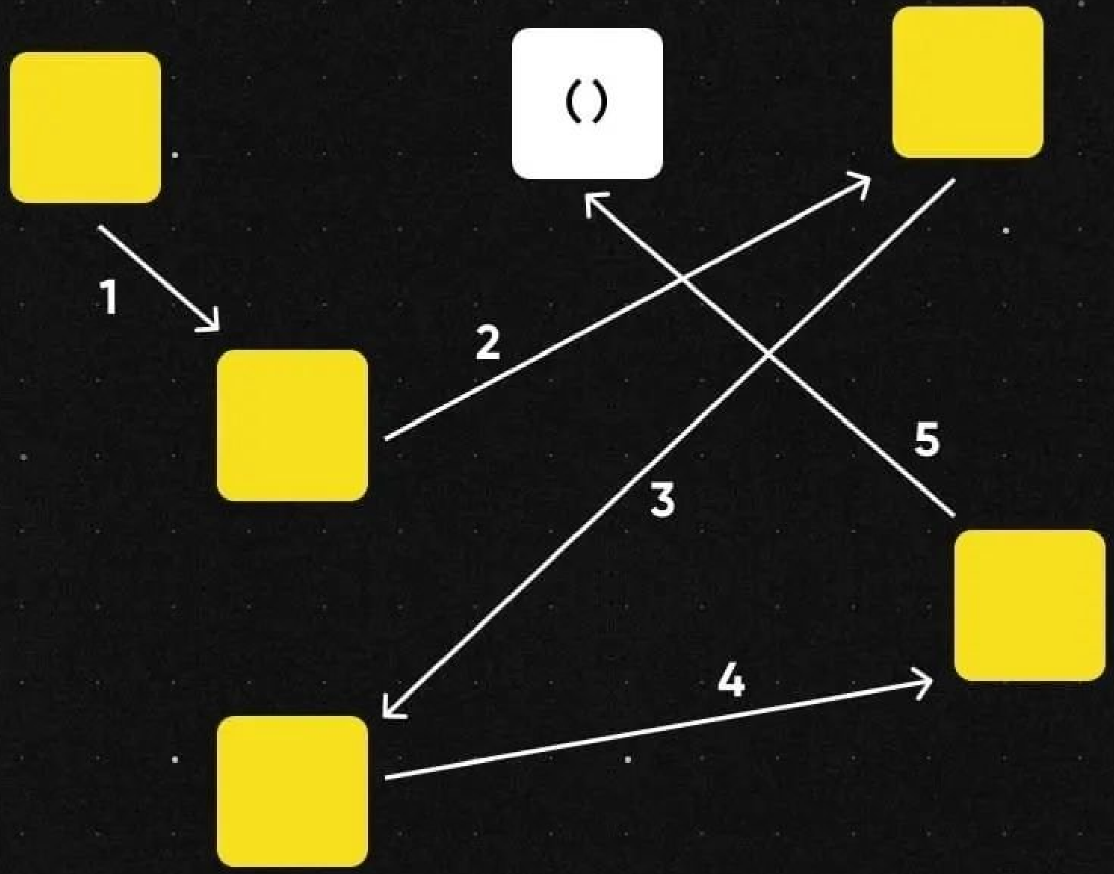
JS

Asynchronous programming

in JavaScript



codewithsloba.com

Swipe ⟶

Asynchronous programming is a way to handle tasks that do not need to be executed in a sequential order.

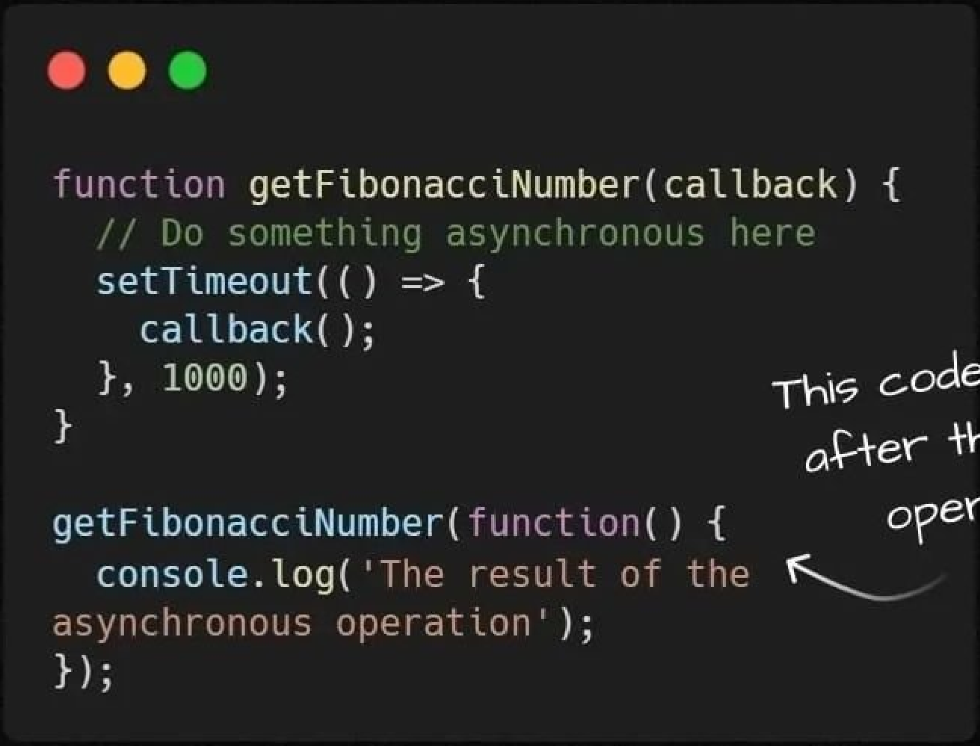
This can be useful for tasks that take a long time to complete, such as making an API request or loading an image.

There are three main ways

Swipe ⟶

Callbacks

Callbacks are functions that are passed as arguments to other functions. They are called when the other function finishes executing.



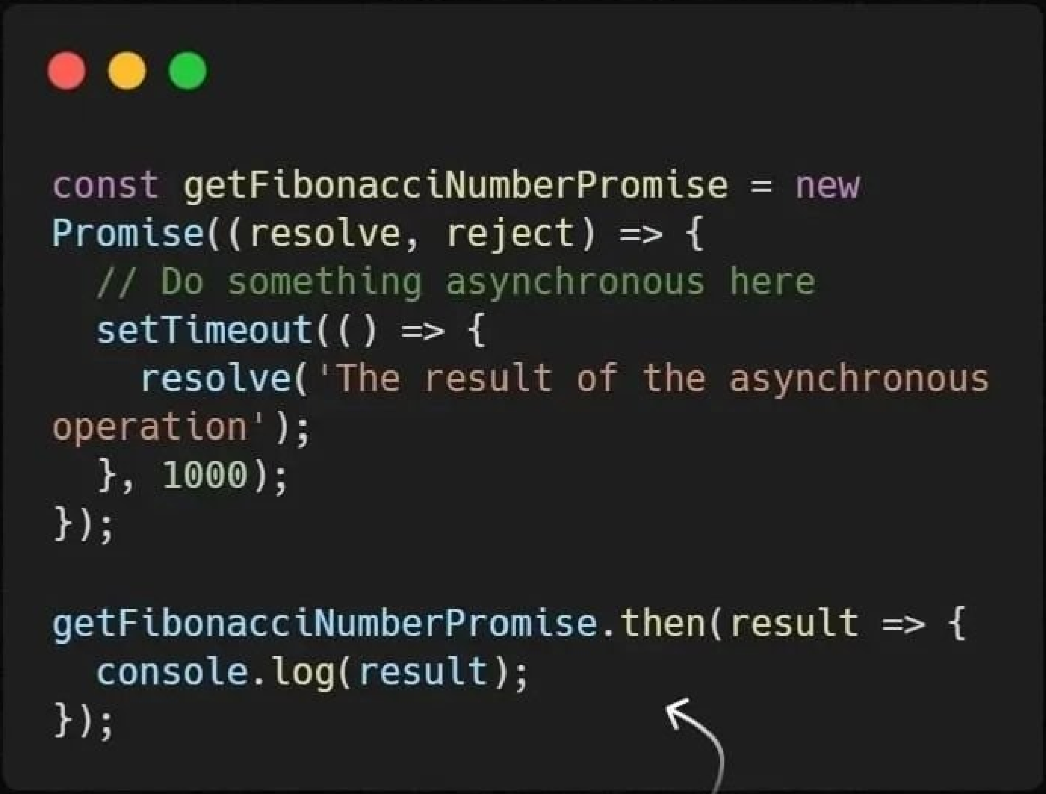
```
function getFibonacciNumber(callback) {
  // Do something asynchronous here
  setTimeout(() => {
    callback();
  }, 1000);
}

getFibonacciNumber(function() {
  console.log('The result of the
asynchronous operation');
});
```

This code will be executed after the asynchronous operation finishes

```
"The result of the asynchronous operation"
```

Promises

Promises are objects that represent the eventual result of an asynchronous operation. They can be used to chain together asynchronous operations.

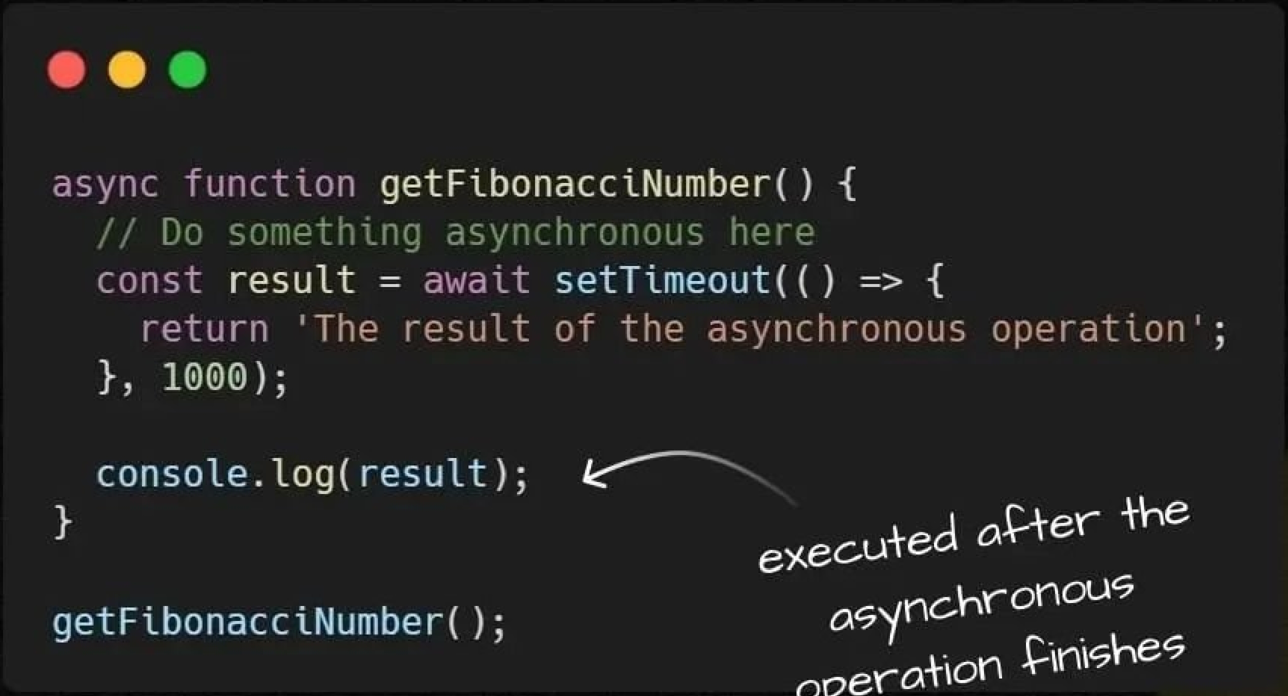


```
const getFibonacciNumberPromise = new
Promise((resolve, reject) => {
  // Do something asynchronous here
  setTimeout(() => {
    resolve('The result of the asynchronous
operation');
  }, 1000);
});

getFibonacciNumberPromise.then(result => {
  console.log(result);
});
```

executed after the asynchronous operation finishes

Async/await

Async/await is a newer feature of JavaScript that makes it easier to write asynchronous code. It allows you to write code that looks like synchronous code, but it actually executes asynchronously.



```
async function getFibonacciNumber() {
  // Do something asynchronous here
  const result = await setTimeout(() => {
    return 'The result of the asynchronous operation';
  }, 1000);

  console.log(result);
}

getFibonacciNumber();
```

executed after the asynchronous operation finishes

Did You Find it Useful?

Alamin CodePapa

@CodePapa360

Follow for more

Like | Comment | Repost

♥ 💬 🔁