JavaScript Async Generators

Summary: in this tutorial, you'll learn about the JavaScript async generators that iterate over data that comes asynchronously.

What is an async generator

An async generator is similar to a regular generator except that its `next()` method returns a Promise. To iterate over an async generator, you use the `for await...of` statement.

Introduction to the JavaScript async generators

A regular generator is a function that can pause midway and continue from where it paused. See the following example:

```
function* sequence(start, end) {
    for (let i = start; i <= end; i++) {
        yield i;
    }
}

let seq = sequence(1, 5);

for (const num of seq) {
    console.log(num);
}
```

The `sequence` is a generator that returns a number from the `start` to the `end` .

An async generator is similar to a regular generator with the following differences:

- The `async` keyword is placed in front of the `function` keyword.

- The `yield` returns a `Promise`, instead of a value. The `Promise` is typically a wrapper of an asynchronous operation.

The following illustrates how to convert the generator `sequence` to the async generator `asyncSequence`:

```
async function* asyncSequence(start, end) {
    for (let i = start; i <= end; i++) {
        yield new Promise((resolve, reject) => {
            setTimeout(() => {
                resolve(i);
            }, 1000);
        });

    }
}
```

Note that we used the `setTimeout()` to simulate an asynchronous operation.

To iterate over the entire async generator, you use the `for await...of` statement.

Since we only can use the `await` keyword inside an `async` function, we wrap the code inside an async IIFE as follows:

```
(async () => {
    let seq = asyncSequence(1, 5);

    for await (let num of seq) {
        console.log(num);
    }
})();
```

The code returns a sequence from 1 to 5 after every second:

```
1
2
3
```

```
4
5
```

The async generators can be very useful when you access a stream of data and want to report the progress like using a progress bar.