Asynchronous JavaScript

Introduction ➖

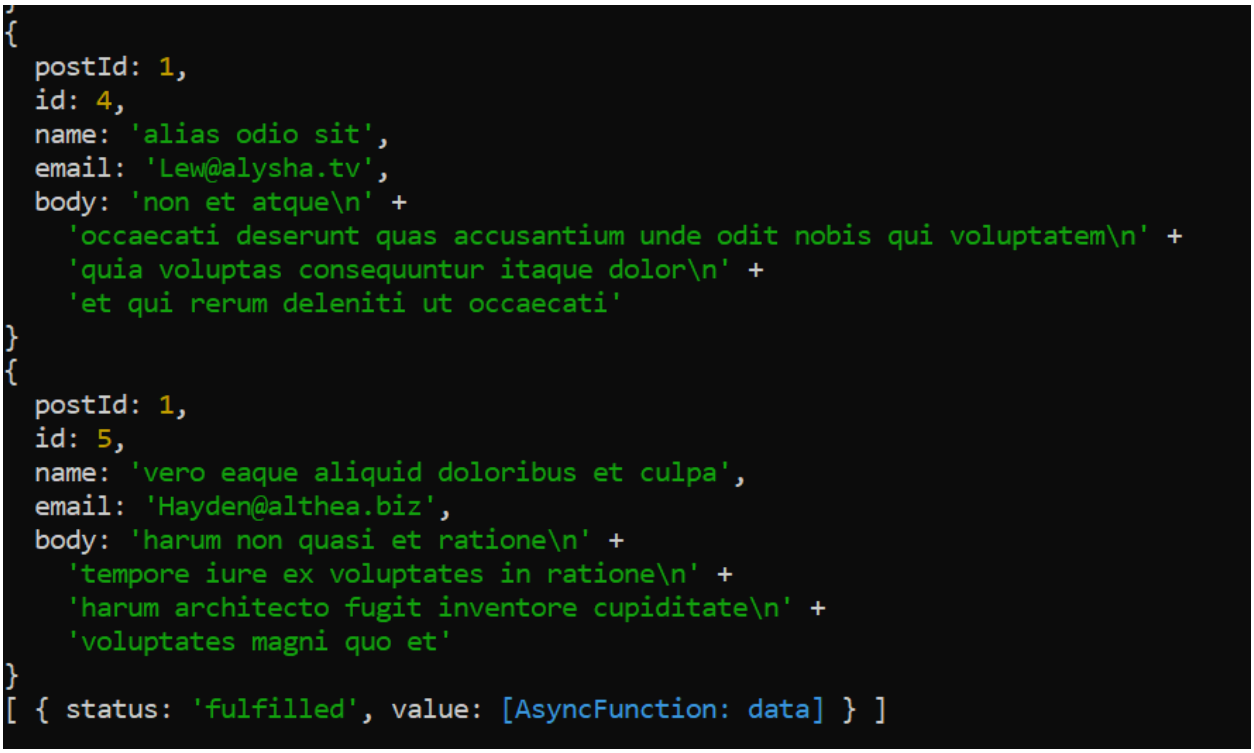
Asynchronous mean things can happen independently of the main program. JavaScript uses **async** and **await** keyword to perform asynchronous programming. In this way, program waits for the single instruction to get done before moving for the another instructions to execute.

for example -

```
import fetch from "node-fetch";

const url = `https://jsonplaceholder.typicode.com/comments`;

const data = async ()=>{
    let comments = await fetch(url);
    comments = await comments.json()
    // console.log(comments.allSettled())
    let top10 = 1;
    comments.forEach(val => {
        if(top10 <=5){
            // console.log(val.email +" "+val.id)
            console.log(val)
        }
        top10++;
    });
}

await data().catch(err => {
    console.log("got error");
    console.error(err)
});

Promise.allSettled([data]).then(mes => console.log(mes))
```

Output of the above function ➖

```
C:\Users\mindpath\Desktop\Intro JS\Code\Async and Await>node getPost.js
{
  postId: 1,
  id: 1,
  name: 'id labore ex et quam laborum',
  email: 'Eliseo@gardner.biz',
  body: 'laudantium enim quasi est quidem magnam voluptate ipsam eos\n' +
    'tempora quo necessitatibus\n' +
    'dolor quam autem quasi\n' +
    'reiciendis et nam sapiente accusantium'
}
{
  postId: 1,
  id: 2,
  name: 'quo vero reiciendis velit similique earum',
  email: 'Jayne_Kuhic@sydney.com',
  body: 'est natus enim nihil est dolore omnis voluptatem numquam\n' +
    'et omnis occaecati quod ullam at\n' +
    'voluptatem error expedita pariatur\n' +
    'nihil sint nostrum voluptatem reiciendis et'
}
{
  postId: 1,
  id: 3,
  name: 'odio adipisci rerum aut animi',
  email: 'Nikita@garfield.biz',
  body: 'quia molestiae reprehenderit quasi aspernatur\n' +
    'aut expedita occaecati aliquam eveniet laudantium\n' +
    'omnis quibusdam delectus saepe quia accusamus maiores nam est\n' +
    'cum et ducimus et vero voluptates excepturi deleniti ratione'
}
```



JavaScript is a single threaded programming language means it doesn't wait task to get completed before moving for another task. Say a function takes 3s to get executed but another instruction can be executed in 10ms, so what JS will do is it call the function and execute it and move to another instruction without even waiting for the result of that function

for example ➖

```
function fun(){
    setTimeout(() => {
        console.log("function");
    }, 3000);
}
fun()
console.log("Hello,")
```

output of the above function

```
C:\Users\mindpath\Desktop\Intro JS\Code\Synchronous>node syncExample.js
Hello,
function
```

Above example using async features

```
async function promise(){
    const a =  await new Promise((res,err)=>{
        setTimeout(() => {
            res('function');
        }, 3000);
    });
    return a;
}

console.log(await promise())
console.log("Hello");
```

result of this is

```
C:\Users\mindpath\Desktop\Intro JS\Code\Async and Await>node simpleEx.js
function
Hello
```