```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Async Await</title>
</head>

<body>

    <script>
        /*
,            ***** Async Await ****
,
,            async/await 는 두 Promise의 업데이트이다.
,            async/await를 만든 이유는 보기 좋은 코드를 위해서이다.
,            then 이나 catch 같은것들은 구식이다.
,            왜냐면 많은 사람들이 결국 많은 then을 사용해야 하기 때문.
,            then.then.then.then 이렇게 많은 function을 사용해야한다.
,
,            async/await는 기본적으로 Promise를 사용하는 코드를 더 좋게 보이게 하는 문법이다.
,            기본적으로 Promise, reject, resolve는 유효하다.
,            Promise.all, Promise.race도 .
,
,            async/await 는 Promise 밖에서 값을 가져올 수 있는 방법이다.
,            많은 then이나 catch를 사용하지 않고.
,
,            먼저 await는 혼자 사용할 수 없다.
,            await는 항상 async function안에서 사용할수있다.
,            async는 이렇게 만든다.
,
,            const getMovies = async() => {...}
,            arrow function을 사용하고 싶지 않다면
,            async function getMovies(){...}
,            기억해야할것은 async먼저 써줄것.
,
,            이제 해야할 것은 Promise를 끝내기 위해 수많은 then을 사용하는 대신에 await를 사용할것.
,            await로 쓰지만 a는 신경쓰지마라. wait라고 이해해도됨.
,            Promise가 끝나길 기다리는것
,
,            async에서는 then이나 다른것을 하는 대신에
,            Promise를 넣을 변수를 선언해주면 된다.
,
,            const getMoviesAsync = async() => {
,                const response = fetch("https://yts.mx/api/v2/list_movies.json")
,                console.log(response)
,            }
,
,            getMoviesAsync()
,            이런식으로 호출하면 pending중인 Promise라고 나오는데
,            fetch 앞에 await를 붙혀주면 호출이 제대로 된다.
,
,            await 는 Promise가 끝나길 기다린다.
,            그리고 response를 getMoviesPromise에서 했던것처럼 넣어준다.
,            기본적으로 resolve된 값을 넣어주는것.
,
,            * Promise, resolve, reject는 같다. 우린 여전히 fetch를 사용하고있고,
,            하지만 await는 뒤에서 해주고있고, response로 넣어준다.
,
```

```
,            그리고 await response.json()을 통해 JSON도 얻을 수 있다.
,
,            const getMoviesAsync = async() => {
,                const response = await
fetch("https://yts.mx/api/v2/list_movies.json")
,                const json = await response.json();
,                console.log(json)
,            }
,
,            기존과 비교해보면 코드가 많이 간결해진것을 볼 수 있다.
,
,            Promise를 설명한 이유는, 뒤에서 어떤 일이 일어나는지 알아야 하기 때문
,            다른것은 await가 더 쉽게 만들어준다는것이다. 왜냐면 await는 Promise가 끝나길 기다려주
는것이기 때문.
,            * 성공 * 을 기다려주는것이아님.
,            resolve나 reject는 상관없다 단지 끝나기를 기다려주는것.
,
,         */

        // Promise then catch
        const getMoviesPromise = () => {
            fetch("https://yts.mx/api/v2/list_movies.json")
            .then(response => {
                console.log(response);
                return response.json();
            })
            .then(json => console.log(json))
            .catch(err => console.log(`❌${err}`));

        }

        // async await
        const getMoviesAsync = async() => {
            const response = await fetch("https://yts.mx/api/v2/list_movies.json")
            const json = await response.json();
            console.log(json)
        }

        getMoviesAsync()
    </script>
</body>

</html>
```