```
<!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>RealWorldPromises</title>
</head>

<body>

    <script>
        /*
,            ***** Real world Promises ****
,
,            이제 나의 Promise를 사용해볼것이다.
,            먼저 Promise를 return하는 fetch이다.
,            fetch는 Promise를 return한다.
,
,            ** fetch가 하는일은 무언가를 가지고 오는 것이다.
,
,            fetch("https://google.com")
,                .then(response => console.log(response))
,                .catch(err => console.log(`❌${err}`));
,            구글은 허용하지 않기때문에 fetch할 수 없지만 Promise는 연습할 수 있다.
,
,            보다시피 에러가 catch됐다.
,            Access to fecth at ...  < fetch하지 못한것
,            동작하지는 않지만 이것이 Promise라는 것은 알 수 있다.
,
,            localhost를 fetch하면 어떻게 response를 가지고 올까.
,            response에 body, header, ok 등등 사이트가 동작을 하고있다.
,
,            여기 보다시피 body가 있는데 ReadableStream이라고 뜬다.
,            Stream은 기본적으로 byte다. 0과 1이다.
,
,            이경우에 우리는 이것을 가지고 와서 읽을것이다.
,            이제 response를 가지고와서 body를 JSON으로 바꾸어야한다.
,            .then(response => console.log(response).json()) 을 해보면
,
,            Promise가 reject되었다고 나올것이다
,            여기서 무엇을 했냐면 다른 Promise를 콘솔에 찍은것이다
,
,            왜냐면 response가 Promise를 리턴하는 함수를 가지고 있기 때문
,            일단 텍스트에 어떤 값을 넣어보자
,            왜냐면 이 텍스트는 JSON이 아니기 때문
,
,            .then(response => console.log(response.text());
,            를 해보면 function이 나온다. pending중인 Promise
,            이제 할것은 response를 텍스트로 바꾼것을 return하는것이다.
,
,            그러니까 API 요청을 하면 API는 나한테 response를 준다.
,            그러면 그 response를 텍스트로 변환하는것을 시도한다.
,            ** 그리고 그것은 나한테 Promise를 준다 이뜻은 then을 또 써야만 한다는것.
,
,            fetch("http://127.0.0.1:5500/08.promises/8-7_realWorldPromises.html")
,                .then(response => response.text())
,                .then(potato => console.log(potato))
,                .catch(err => console.log(`❌${err}`));
,            이렇게 해보면 document들이 불러와질것이다
```

```
,
,            Promise를 이용한것.
,            이 Promise를 다른 Promise를 리턴하고 다른 Promise의 결과인 response text를 가져왔다.
,            물론 response.json()도 사용이 가능하다. 만약 JSON을 얻을 수 있다면.
,
,            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}`));
,
,            response.json()을 했고 모든 영화들을 가지고 왔다.
,            response에서는 아무것도 가지고 오지않았지만
,            json을 사용해서 body를 가지고왔다.
,
,            https://yts.mx/api/v2/list_movies.json
,            이것으로 접속해보면 우리가 얻은것과 일치하는 json.data가 나오게 된다.
,
,            ** 보통 나만의 Promise를 만들지는 않는다.
,            Promise.all은 할 수 있지만 나만의 Promise를 만들지 않아도 된다.
,            여기서 Fetch를 쓴 것처럼 다른사람이 만든 Promise를 사용하면 된다.
,
,         */

        // fetch

        // fetch("https://google.com")
        /*
,         fetch("http://127.0.0.1:5500/08.promises/8-7_realWorldPromises.html")
,             .then(response => response.text())
,             .then(potato => console.log(potato))
,             .catch(err => console.log(`❌${err}`));
,         */

        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}`));



    </script>
</body>

</html>
```