```
<!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>Promise</title>
</head>

<body>

    <script>
        /*
,            ***** Promise ****
,
,            Promise는 무엇이고 어떻게 만드는것일까.
,            간단하다. 생성자처럼 new Promise()를 변수에 넣어주면된다.
,
,            ** Promise는 비동기 작업이 맞이할 미래의 완료 또는 실패와 그 결과 값을 나타낸다.
,            그래서 Promise를 만들때는 실행할 수 있는 function을 넣어야한다.
,
,            여기서 new Promise()는 forEach에서 처럼 executor이다.
,            이 Promise를 호출할 때 자바스크립트는 이 Promise를 실행할것이다.
,            내가 설정한 다른 function과 함께.
,
,            executor를 만들어보자.
,            const amISexy = new Promise((resolve, reject) => { 변수명은 바꿔도됨
,                이 안에서 내가 원하는것을 할 수 있다.
,                해야 할것은
,                * 이 Promise를 resolve하거나 reject하는것.
,            })
,
,            * resolve
,            resolve는 이것이 너의 값이다. 자바스크립트로 돌아가 이런뜻.
,
,            * reject
,            reject는 미안한데 여기 에러가 있어이다.
,
,            이전 파일에서 google을 fetch했는데 Promise가 reject됐다.
,            에러가 생긴것이여서 값이 없었다.
,
,            const amISexy = new Promise((resolve, reject) => {});
,            console.log(amISexy)
,            이런식으로 찍어보면 promise가 pending중이라고 나온다.
,            내가 amISexy를 다시 호출하면, 아직도 Promise가 pending중이라고 나온다. 끝나지않았다고.
,
,            자바스크립트가 이 파일의 실행을 끝내고 우리가 새로운 Promise를 시작했고 이 Promise가 끝나기를 기다리고있는것이다.
,            자바스크립트는 기다리고있는것 우리는 아무것도 가진것이 없다. 저 Promise가 끝나기까지 기다려야함.
,
,            * 어떻게 Promise를 끝낼 수 있을까.
,            resolve function을 실행하면 된다. 이것이 Promise를 resolve할 것이고 Promise를 끝낼것이다.
,            그렇게 되면 난 값을 가지게 되는것.
,
,            const amISexy = new Promise((myResolve, reject) => {
,                setTimeout(() => {
,                    myResolve("yes you are")
,                }, 3000)
```

```
,            });
,            setInterval(() => {
,                console.log(amISexy);
,            }, 1000)
,
,            Promise {<pending>}
,            Promise {<pending>}
,            Promise {<fulfilled>: 'yes you are'}
,
,            이런식으로 resolve 함수가 3초있다가 실행되게하고
,            1초간격으로 amISexy를 콘솔에 찍도록해보면
,            3초후에 resolve함수에 넣은것이 나온다.
,
,            * 우리는 Promise를 만들었다.
,            밖에선 Promise를 매초마다 콘솔에 찍을것이다.
,
,            * setTimeout과 setInterval 은 handler와 timeout과 인자를 가지고있기 때문에
,            사실 setInterval(console.log, 1000, amISexy) 이런식으로도 바로 할 수 있다.
,
,            계속 pending상태이다가 3초후에 Promise가 yse you are 메세지와함께 resolved되었다.
,
,            **** Prmoise의 핵심.
,
,            내가 아직 모르는 value와 함께 작업할 수 있게 해준다는것.
,            자바스크립트에서 이거 해줘 이건 바로 안될거야 이걸하고 끝나면 다시 알려줘.
,            그러니까 이게 끝나면 이걸 다시 나한테 돌려줘. 이런식이다.
,
,            우리는 어떻게 yes you are이 사용될지 Promise에서 yes you are을 어떻게 꺼낼지 모른
다.
,            Prmoise가 API를 호출하거나, 파일시스템에서 파일을 연다면? 유저가 설정을 연다면?
,            로딩이 다되면 그걸 다시 자바스크립트에게 돌려주어야한다.
,            Promise의 사용방법은 8-2.
,         */

        const amISexy = new Promise((resolve, reject) => {
            setTimeout(resolve, 3000, "yes you are");
        });
        setInterval(console.log, 1000, amISexy)

    </script>
</body>

</html>
```