```html
<!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</title>
</head>

<body>

    <script>
        /*
,            ***** async *****
,            async(비동기성)
,            promises를 시작하기 전에 자바스크립트의 비동기성과 동기성에 대해 생각해볼 필요가 있다.
,
,            우리는 보통 뭔가를 먼저 한다. 사러가고 어떻게 요리를 할지 정하고
,            이것들을 동시에 할 수 없다. 우리는 무언가를 먼저 해야한다.
,            두가지를 한번에 할 수는 없다.
,
,            멀티태스킹은 한가지 이상의 것을 한번에 동시에 생각하는것이 아니다.
,            단지 사이를 빠르게 스위칭 하는것이다.
,
,            컴퓨터는 가능하다. 동시에 두가지일을 할 수 있다.
,            promises를 이해하려면 이런것들을 알고 있어야한다.
,            왜냐면 자바스크립트는 동시에 많은 일을 할 수 있기 때문.
,
,            자바스크립트는 내 프로그램을 볼 수 있다.
,            한쪽에서 어떤 일을 하고 있어도 내 프로그램은 그대로 실행중일것이다.
,            예를들어, 자바스크립트 프로그램이있는데
,            API에서 데이터들을 가지고 온다. 기본적으로 가져오는 작업이 끝나면 계속될것이다.
,
,            ** 웹사이트에서 데이터를 어떻게 가져올까?
,            fetch를 이용한다.
,
,            ** fetch
,
,            const hello = fetch("http://google.com");
,            console.log("something");
,            console.log(hello);
,
,            something이 출력되고
,            promise가 오고
,            google.com을 fetch하지 못했다는 에러가 출력된다.
,
,            현재 난 google.com을 fetch하는 것을 먼저 했다.
,            이론적으로 저 에러는 something전에 나와야한다.
,            이것이 자바스크립트의 비동기성(async)이다.
,
,            * 자바스크립트는 프로그램의 실행을 멈추지않는다.
,            단지 fetch를 한다는것으로는.
,            자바스크립트는 google fetch를 실행하고 계속해서 something을 콘솔로 찍는다
,            그리고 google fetch가 끝나면 에러가 발생한다.
,            위에서부터 아래로 코드를 읽지만
,            fetch를 실행하자마자 한쪽에선 어떤 action을 시작하고,
,            그 다음 코드를 읽고 fetch에서 문제가 생기고 에러가 뜨는것
,
,            * 이것이 비동기적 프로그래밍이다.
,            순차적으로 처리되는것이 아니라 한꺼번에 실행된다.
```

```
,            이것이 promises에 대한 기초이다.
,
,
,         */

        const hello = fetch("http://google.com");

        console.log("something");
        console.log(hello);
    </script>
</body>

</html>
```