```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>finally</title>
</head>

<body>

    <script>
        /*
,            ***** finally ****
,
,            finally는
,
,            가끔 나는 무언가 하고싶을것이다.
,            Promise가 성공하거나 실패했을때.
,
,            예를들어 코멘트를 API를 통해 저장하고싶다.
,            그래서 유저가 save버튼을 눌렀을때 Spinner를 보여주고싶다.
,            만약 에러가 발생하면 Spinner를 멈추고 유저에게 에러를 해결하라고 보여주고싶고,
,            업로드에 성공하면 Spinner를 멈추고싶다.
,
,            이럴때 finally , finalizing 이라고 부르는것이다.
,
,            const p1 = new Promise((resolve, reject) => {
,                setTimeout(resolve, 1000, "First");
,            })
,                .then(value => console.log(value))
,                .finally(() => console.log("i'm done"))
,
,            출력값은
,            First
,            i'm done 이 나온다.
,            보다시피 이것은 then다음에 실행될것이다.
,            then에도 되고 catch에도 동작한다.
,
,            ** 정리
,            어쨌거나 내가 원하는 값을 얻을것이고 마지막 i'm done을 얻을것이다.
,            결과에 대해 신경쓰지 않아도 된다. finalize 하는데 성공하든지 실패하든지는 상관이 없다.
,
,            보통 finally를 API를 호출할 때에 쓴다.
,            로딩할 때, 하나를 얻고 두개를 얻고 세개를 얻고 마지막으로 데이터를 보여주거나
,            로딩을 멈추거나 뭔가를 하거나 할때.
,
,         */

        // resolve finally
        const p1 = new Promise((resolve, reject) => {
            setTimeout(resolve, 1000, "First");
        })
            .then(value => console.log(value))
            .finally(() => console.log("i'm done"))

                    // resolve finally
        const p2 = new Promise((resolve, reject) => {
            setTimeout(reject, 1000, "Second");
        })
```

```
62              .catch(err => console.log(`${err}❌`))
63              .finally(() => console.log("i'm done"))
64
65
66      </script>
67  </body>
68
69  </html>
```