JavaScript

JavaScript

- **JavaScript**

  JavaScript는 브라우저에서 실행되는 언어로써 인터프리터 방식의 객체지향 프로그래밍 언어이다.

- **ES6**

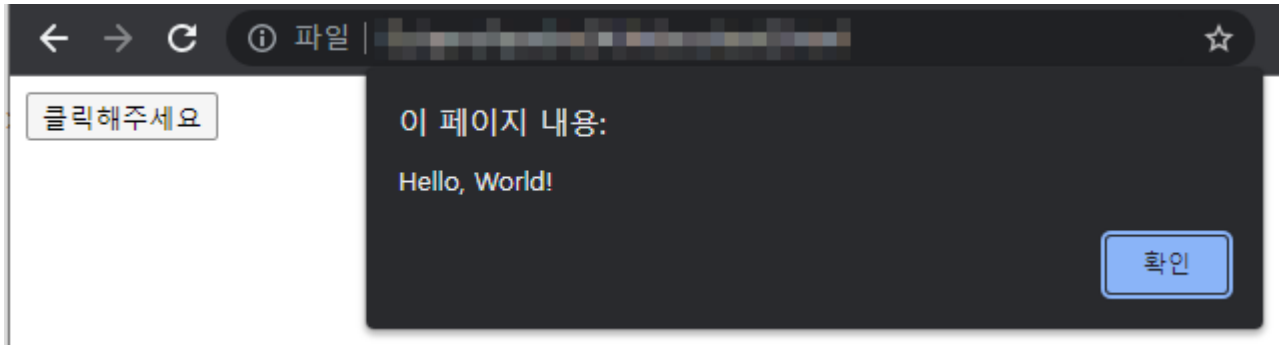
  ES란 JavaScript의 표준 규격을 의미 하는데, 이는 ECMA에서 정하게 된다.
  ES6은 가장 최근의 표준 규격으로써 JavaScript 개발 시 ES6 이상의 버전을 사용하는것이 권고됨

JavaScript 작성 방식-1

- **inline**

자바스크립트의 코드가 매우 짧거나 간단한 용도로 사용해야할 때 HTML 태그 안에 직접 코드를 작성하는 방식

```
<button type="button" onclick="alert('Hello, World!')">
    클릭해주세요
</button>
```



JavaScript 작성 방식-2

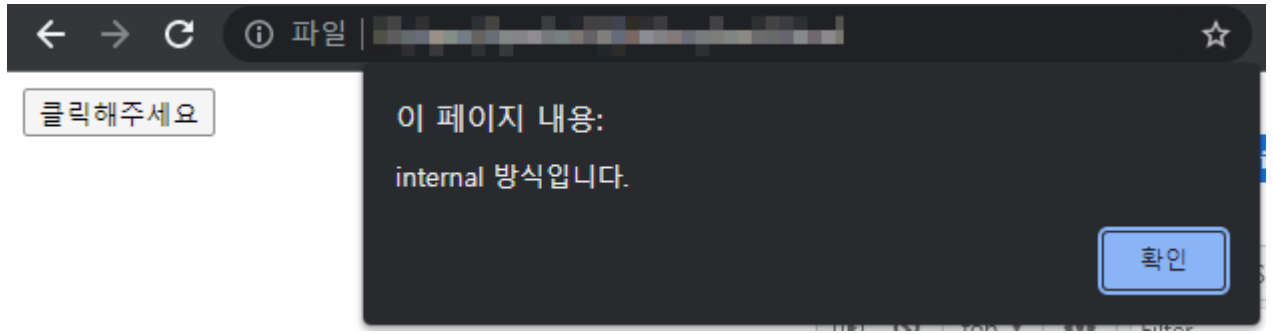
- **internal**

HTML 태그 내에 직접 작성하는 것이 아니라, <script> 코드 </script>를 별도로 작성하여 사용하는 방식.

자바스크립트 코드의 위치는 head, body 안에 작성 하거나 </html> 이후에 작성할 수 있음

```
<button id="btn" onclick="btnClick()">
    클릭해주세요
</button>

<script>
    function btnClick() {
        alert("internal 방식입니다.");
    }
</script>
```

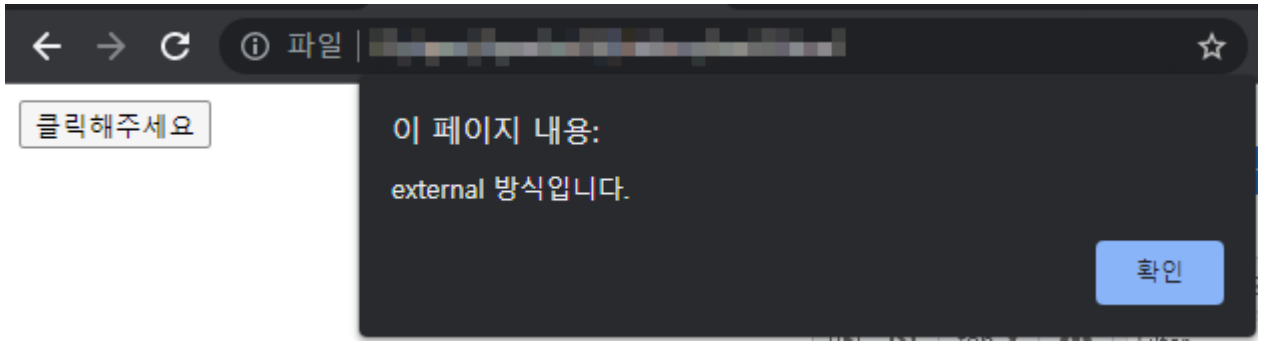


JavaScript 작성 방식-3

- **external**

별도의 js 파일을 작성하여 참조해서 사용하는 방법으로써, 유지보수 및 코드의 재사용성을 위해 가장 많이 사용되고 권장되는 방법
* js 파일을 불러올때는 head 안에 작성됨

```
<!DOCTYPE html>
<html>
<head>
    <script src="./test.js"></script>
    <title>JavaScript 변수 예제</title>
</head>
<body>
    <button id="btn" onclick="btnClick()">
        클릭해주세요
    </button>
</body>
</html>
```



웹에서의 통신 흐름1

- **서버들의 종류**

DNS: 도메인 주소의 IP를 사용자에게 알려주는 서버
WEB : 사용자로부터 HTTP 요청을 받아 정적인 컨텐츠(HTML, CSS, JavaScript...)를 응답해주는 서버
WAS : DB와 통신하여 동적인 컨텐츠를 처리하기 위한 서버
DB : 데이터를 관리하기 위한 서버

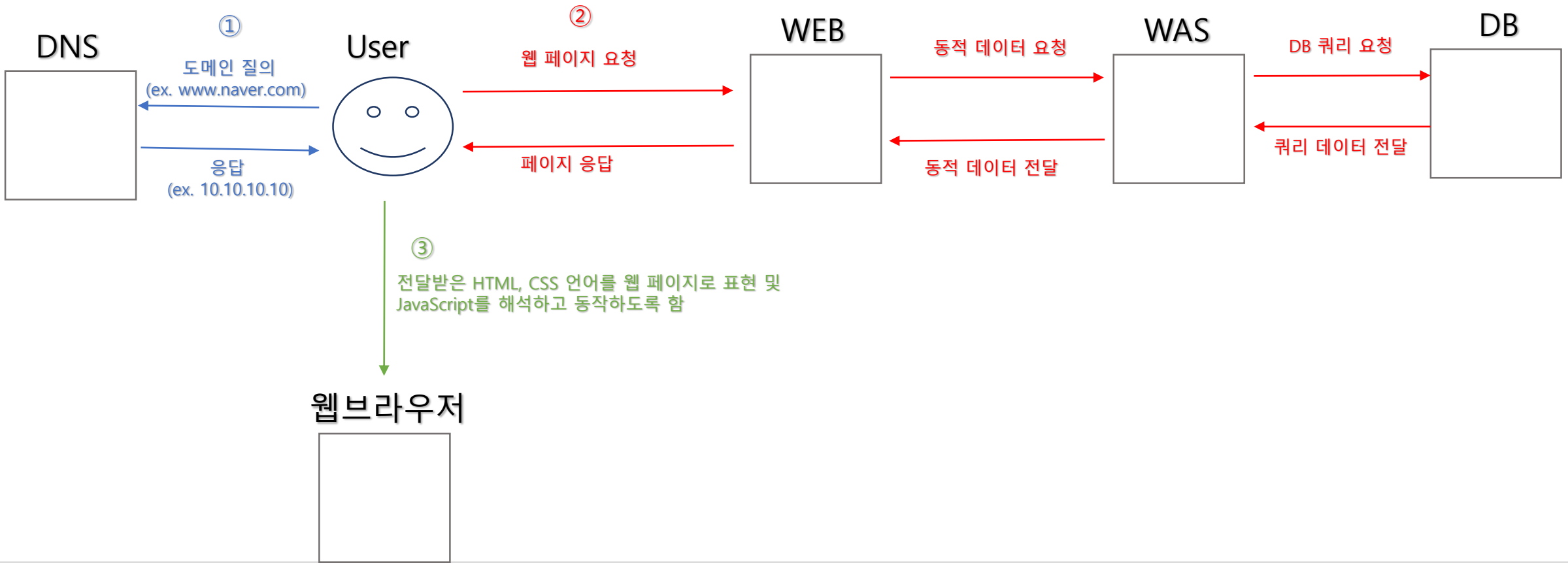
- **WEB vs WAS의 언어**

WEB : HTML, CSS, JavaScript, ...
WAS : JAVA, PHP, Python, ...

웹에서의 통신 흐름2

- **프론트엔드 vs 백 엔드**

일반적으로 프론트 엔드는 클라이언트 측(브라우저)에서 실행되는 언어를 뜻하고,
백 엔드는 서버 측에서 실행되는 언어를 뜻함

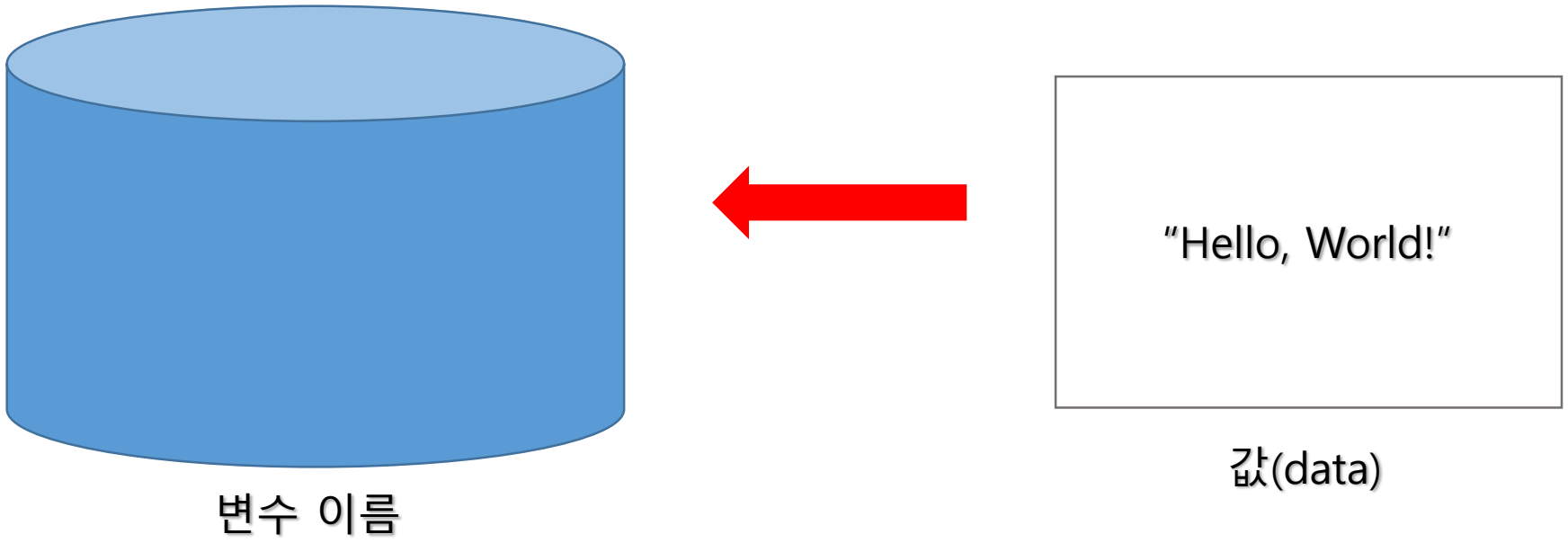


JavaScript 구문 : 변수

JavaScript 구문 : 변수

- **변수(Variable)**

어떠한 값(data)을 저장할 수 있는 저장 공간



JavaScript 구문 : 변수

- **스코프(scope)**

  변수에 접근할 수 있는 범위

- **전역 스코프(Global scope)**

  전역에 선언되어 있어 어디에서든지 변수에 접근할 수 있는 범위

- **지역 스코프(Local scope)**

  변수를 선언한 지역에서만 접근할 수 있는 범위이며 2가지로 나뉨
  1. 블록 스코프 : {}, if문, for문 등 블록 내부에서 선언된 변수의 범위
  2. 함수 스코프 : 함수 내부에서 선언된 변수의 범위

JavaScript 구문 : 변수의 종류1

- **var**

  ES6 이전에 사용하던 변수 선언 키워드로써, 함수 스코프를 가진 변수

- **let**

  ES6에서 도입된 변수 선언 키워드로써, 블록 스코프를 가진 변수

- **const**

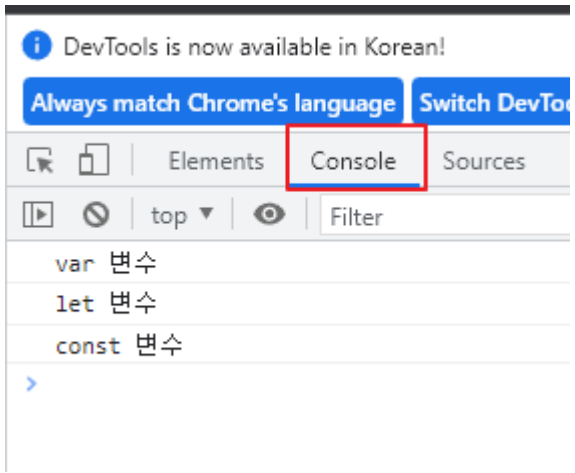
  ES6에서 도입된 변수 선언 키워드로써, 블록 스코프를 가진 변수(상수)

JavaScript 구문
변수 사용해보기 – 선언과 할당

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript 변수 예제</title>
</head>
<body>
    <script>
        var a = "var 변수";
        let b = "let 변수";
        const c = "const 변수";

        console.log(a);
        console.log(b);
        console.log(c);
    </script>
</body>
</html>
```

웹 브라우저 실행 → F12 → Console



JavaScript 구문
변수 사용해보기 – 중복 선언

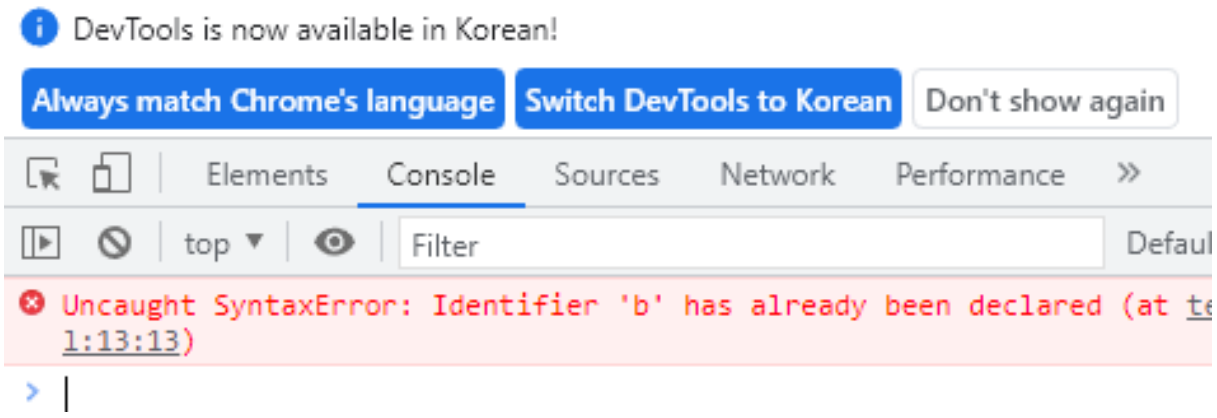
- 설명

1. var : 중복 선언 가능
2. let, const : 중복 선언 불가능

```
<script>
    var a = 1;
    let b = 2;
    const c = 3;

    var a = 4;
    let b = 5; // 불가
    const c = 6; // 불가
</script>
```

웹 브라우저 실행 → F12 → Console



JavaScript 구문
변수 사용해보기 – 외부 변수 참조

- 설명

1. var : 함수 스코프를 가지므로 전역변수인 x가 10으로 변경됨
2. let, const : 블록 스코프를 가지므로 지역변수로만 존재하며 값이 유지됨
   즉, 블록 스코프를 가진 변수는 {}의 외부에서 사용할 수 없음

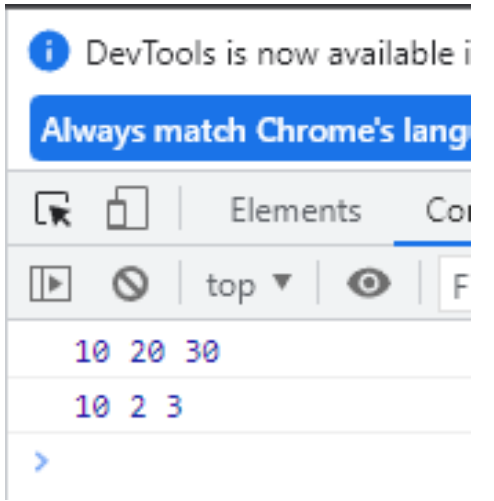
```
<script>
    function example() {
    var a = 1;
    let b = 2;
    const c = 3;

    if (true) {
        var a = 10;
        let b = 20;
        const c = 30;
        console.log(a, b, c); // 10 20 30
    }

    console.log(a, b, c); // 10 2 3
    }

    example();
</script>
```

웹 브라우저 실행 → F12 → Console



JavaScript 구문
변수 사용해보기 – 호이스팅

- 설명

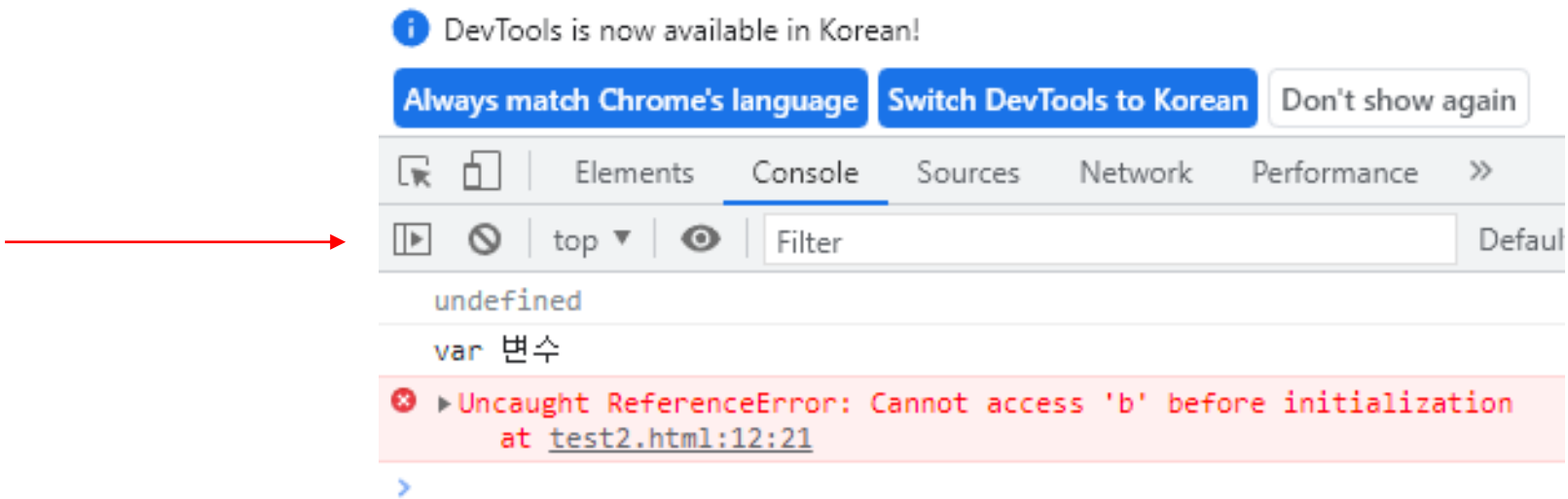
1. var : 호이스팅으로 인해 선언을 하지 않아도 undefined 반환
2. let, const : 호이스팅을 하지 않기 때문에 에러 발생

```
<script>
    console.log(a);
    var a = "var 변수";
    console.log(a);

    console.log(b);  // 에러 발생
    let b = "let 변수";
    console.log(b);

    console.log(c);
    const c = "const 변수";
    console.log(c);
</script>
```

웹 브라우저 실행 → F12 → Console



JavaScript 구문
변수 사용해보기 – const(상수)

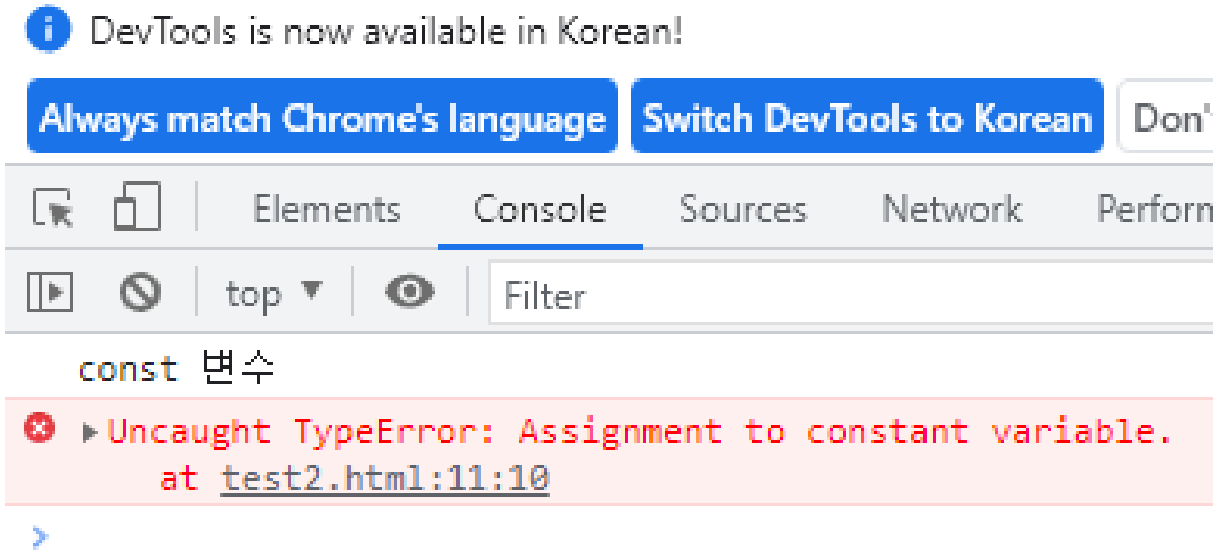
- 설명

const는 한번 값을 할당하면 이후 변경할 수 없기 때문에 에러 발생 (상수)

```
<script>
    const c = "const 변수";
    console.log(c);

    c = "값을 변경 해봅시다.";
    console.log(c);
</script>
```

웹 브라우저 실행 → F12 → Console



JavaScript 구문 : 변수 정리1

- **var의 특징**

1. 변수 중복 선언이 가능하여 개발 시 예기치 못한 값을 반환할 가능성이 있음
2. 함수 외부에서 선언한 변수는 모두 전역 변수로 처리됨
3. 변수를 선언하기 이전에 변수를 참조하면 undefined를 반환함
   → 호이스팅이 발생함

- **호이스팅(Hoisting)?**

코드가 실행되기 전 변수선언/함수선언이 최상단으로 끌어올라와져 별도의 선언을 하지 않았음에도 에러가 발생하지 않고 undefined를 반환하게 됨

* 함수 표현식은 호이스팅이 불가

JavaScript 구문 : 변수 정리2

- **let의 특징**

1. 변수 중복 선언이 불가능
2. 블록 스코프를 가지고 있기 때문에, 외부에서 변수를 참조할 수 없음
3. 변수를 선언하기 이전에 변수를 참조하면 에러를 반환함
   → 호이스팅 발생하지 않음

- **const의 특징**

1. 변수 중복 선언이 불가능
2. 블록 스코프를 가지고 있기 때문에, 외부에서 변수를 참조할 수 없음
3. 변수를 선언하기 이전에 변수를 참조하면 에러를 반환함
   → 호이스팅 발생하지 않음
4. 한번 값이 할당되면 변경 불가
   → 상수