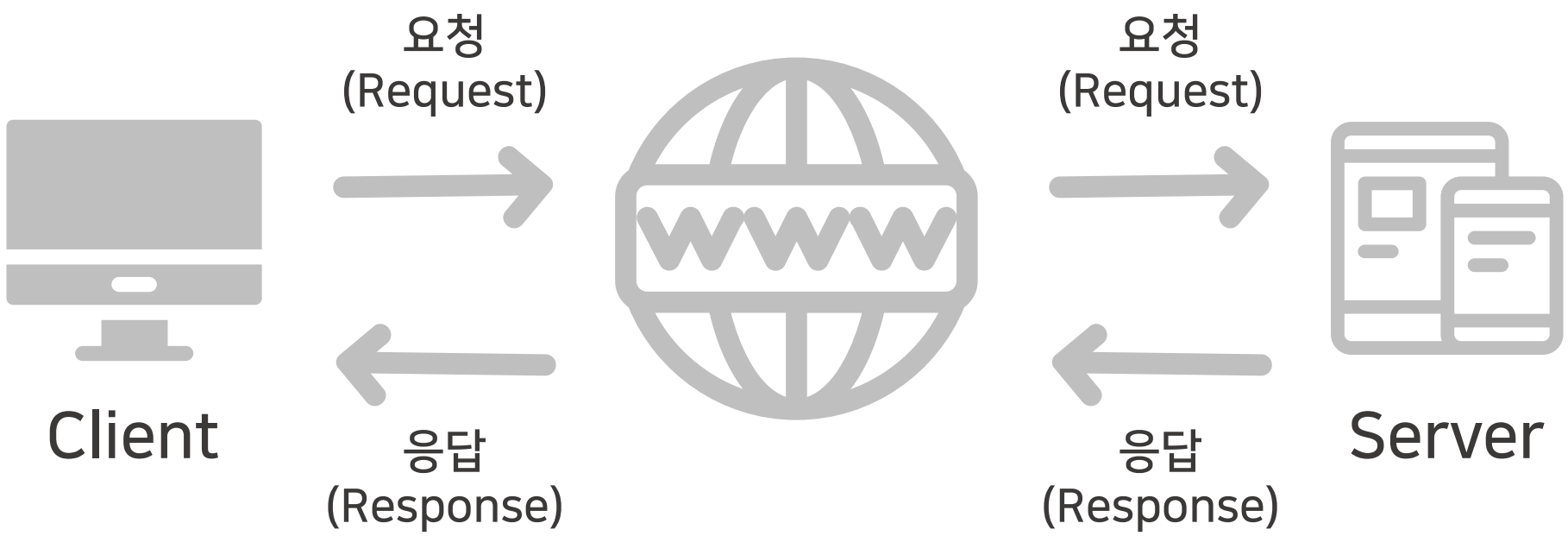
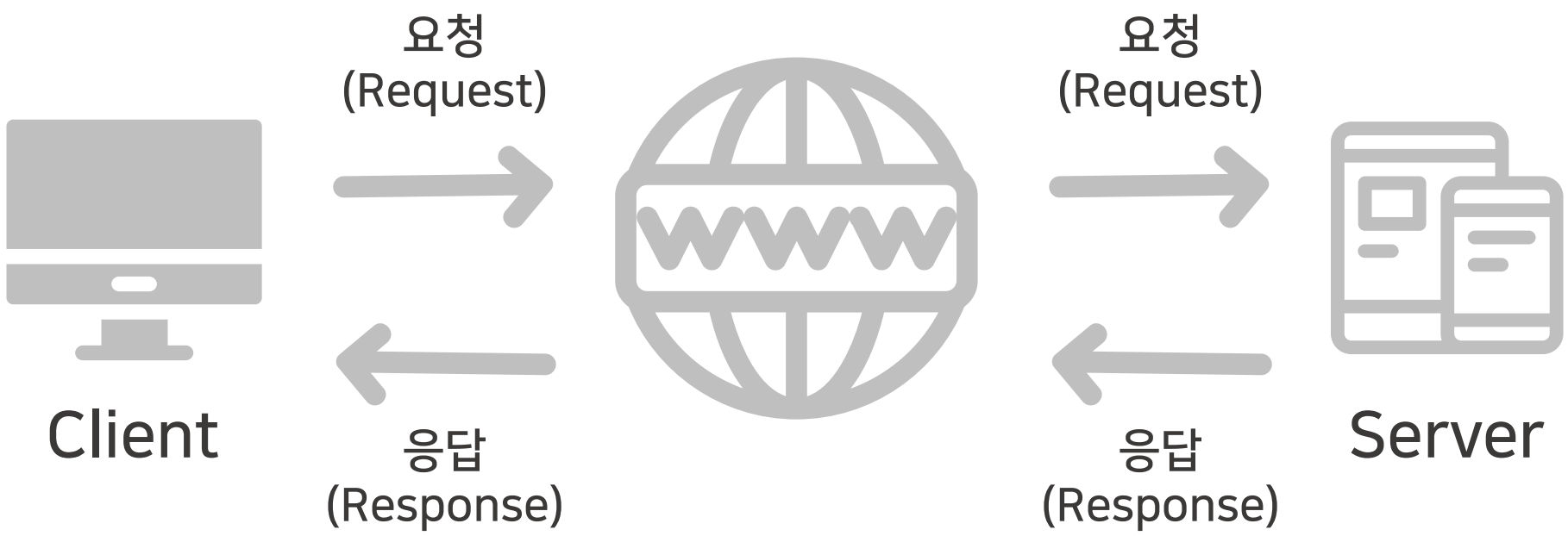
JS

1. JavaScript?

Web Browser 위에서 동작하는 언어

HTML CSS JS


요청
(Request)
요청
(Request)
Client
Server
응답
(Response)
응답
(Response)


요청
(Request)
요청
(Request)
Client
응답
(Response)
응답
(Response)
Server

node.js

Web Browser 위에서 동작하는 언어

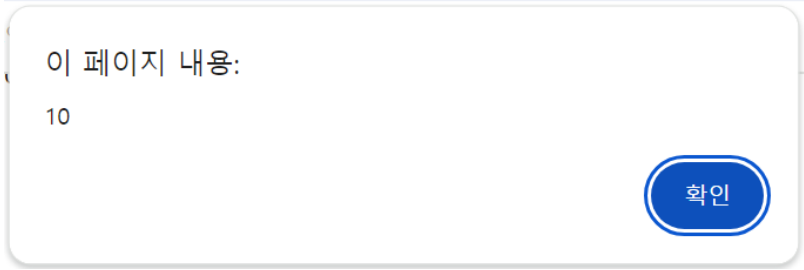
기초 함수

alert() 함수

- 알림창 표시
- alert() 함수의 괄호 안에 메시지를 입력하거나 변수를 넣어 사용

```
alert("안녕하세요")
```

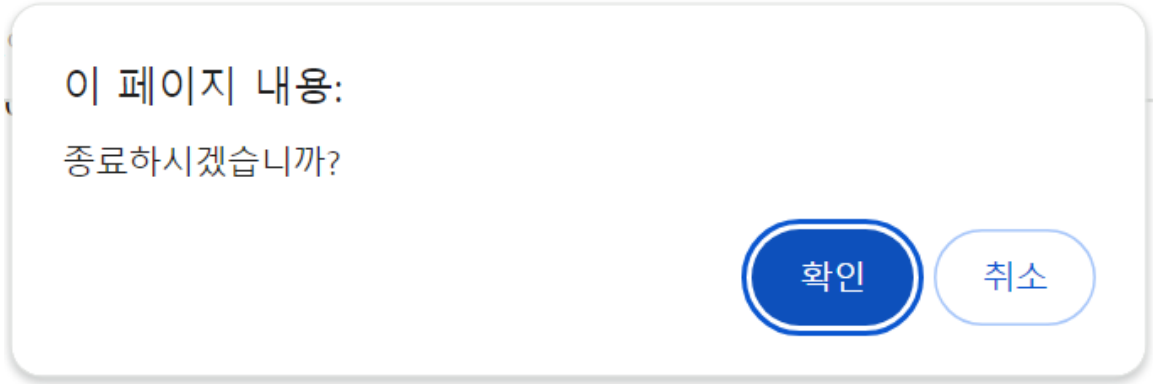

```
a = 10
alert(a)
```



confirm() 함수

- 확인창 표시
- [확인], [취소] 버튼이 있어서 사용자가 어떤 버튼을 클릭했는가에 따라 다르게 동작하도록 가능

```
confirm("종료하시겠습니까?")
```



confirm() 함수

확인 창에서 [확인] 버튼을 누르면

```
> confirm("종료하시겠습니까?")
<· true
>
```

결괏값 true

확인 창에서 [취소] 버튼을 누르면

```
> confirm("종료하시겠습니까?")
<· false
>
```

결괏값 false

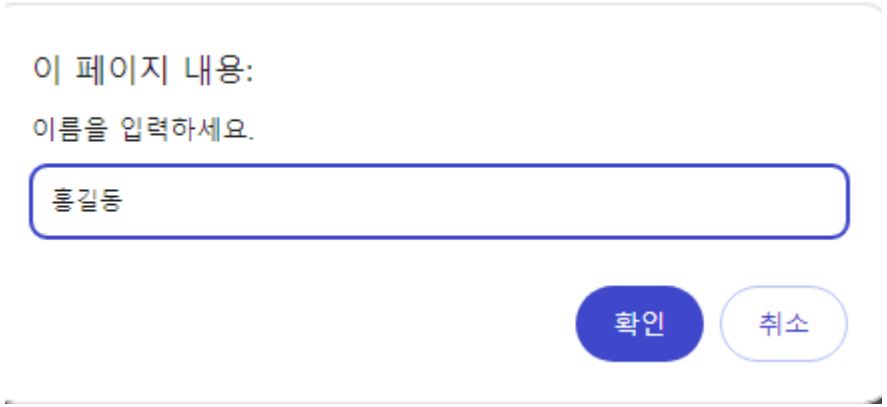
결괏값을 확인하면 사용자가 [확인]을 눌렀는지 [취소]를 눌렀는지 알 수 있음

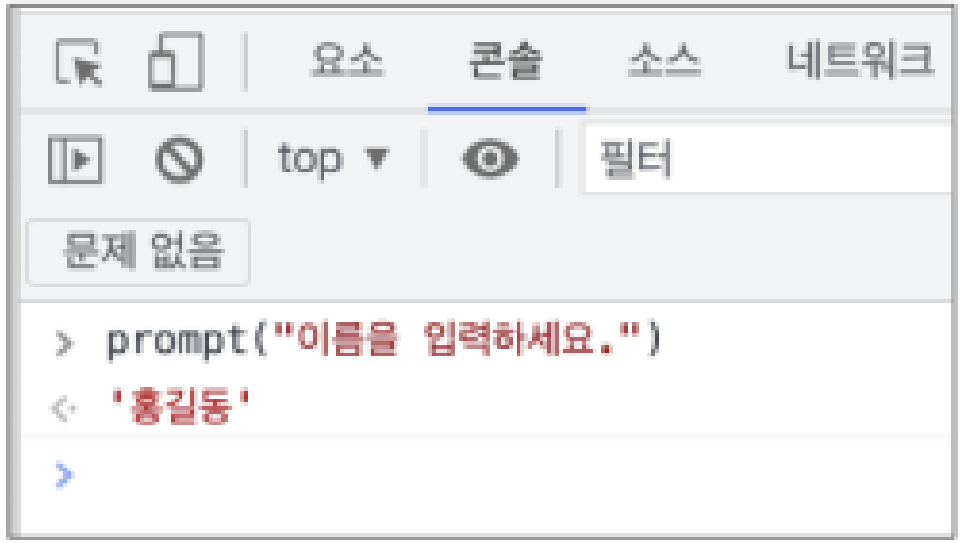
[확인]인지 [취소]인지에 따라 프로그램이 다르게 동작하도록 소스 작성할 수 있음

prompt() 함수

- 사용자가 간단한 값을 입력할 수 있는 창 표시
- 프로그램 실행에 필요한 값을 받을 때 주로 사용
- 기본값을 지정하지 않으면, 텍스트 필드가 빈 상태로 표시됨

```
prompt("이름을 입력하세요.")
```

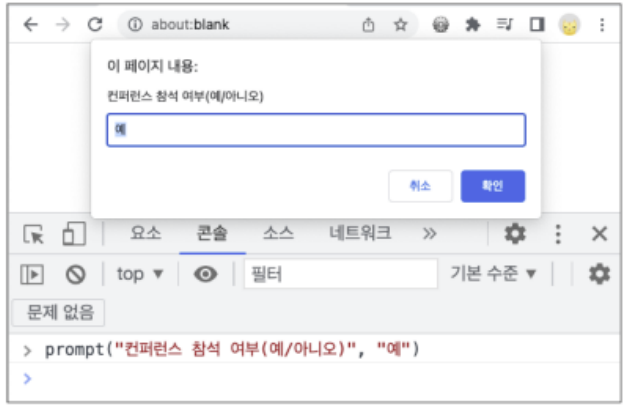




prompt() 함수

사용자가 많이 입력할 것 같은 값을 기본 값으로

```
prompt("컨퍼런스 참석 여부(예/아니오)", "예")
```

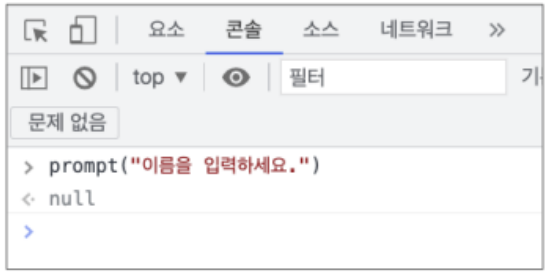


- 기본값을 사용한다면 [Enter]만 누르면 되기 때문에 편리함.
- 기본값을 지운 후 다른 내용을 입력해도 됨

```
prompt("이름을 입력하세요.")
```

프롬프트 창에서 입력하지 않고 [취소]를 누르면?

→ 결괏값 null



프로그램을 작성하면서 prompt() 함수를 사용할 때 사용자가 값을 입력했는지 확인하려면

prompt() 반환값이 null인지의 여부를 확인한다.

console.log() 함수

- 콘솔 창에 괄호 안의 내용을 표시함
- 자바스크립트 소스를 작성하면서 중간중간에 프로그램이 제대로 동작하는지 확인하는 용도로 자주 사용
- 콘솔 창에 결과를 표시하는 함수는 많지만 주로 console.log()를 많이 사용함
- 괄호 안에 텍스트나 변수를 사용할 수 있음

```
console.log(내용)
```

document.write() 함수

- 괄호 안의 내용을 웹 브라우저 화면에 표시함
- 실제 웹 브라우저 화면 에 내용을 표시할 때에는 DOM을 이용하지만, 아직 DOM을 공부하지 않았기 때문에 일단 document.write() 함수 사용.
- doument.write()문에서 연결 연산자(+)를 사용할 수도 있고, 템플릿 리터럴을 사용할 수도 있다.

```
document.write(내용)
```

변수
(Variable)

변수

값을 저장할 수 있는 메모리 공간에 붙은 이름
즉, 값을 담는 그릇(컨테이너)

- 통상적으로 변수는 값이 달라질 수 있는 데이터를 가리킨다.
- 그러나 프로그램에서 변수는 값이 바뀌지 않더라도 변수로 만들어서 사용한다. (상수)

- 프로그램 안에서 사용할 값이 메모리의 어느 위치에 있는지 신경쓰지 않고,
  변수 이름만을 기억하면 된다.
- 변수 이름을 쉽게 가져와서 그 안의 값을 사용할 수도 있고, 같은 위치에 바뀐 값을 저장할 수도 있다.

  따라서 변수 이름은 서로 다르게 만들어주어야 한다.

```
name = '손흥민';
birthYear = 1992;
```

```
name = '손흥민';
Name = '박지성';
NAME = ‘안정환';

birthYear = 1992;
```

```
name = '손흥민';
Name = '박지성';
NAME = '안정환';

birthYear = 1992;
```

변수명 정하는 규칙

- 숫자로 시작할 수 없다.
- 이름 안에 공백이 포함되어 있을 수 없다.
- 대소문자를 구분한다.
- 예약어는 사용할 수 없다.
- 무의미한 변수 이름은 피한다.

camelCase

합성어를 대문자로 구분
(각 단어의 첫글자를 대문자로 작성, 첫 글자 제외)

snake_case

합성어를 _로 구분

camelCase birthYear, myName, userInfoList

snake_case place_of_birth, my_age, address_code_list

예약어는 사용 불가

```
name = '손흥민';
birthYear = 1992;
class = 'Top of The World';
```

```
name = '손흥민';
birthYear = 1992;
class = 'Top of The World';
```

abstract	arguments	await	boolean
break	byte	case	catch
char	class	const	continue
debugger	default	delete	do
double	else	enum	eval
export	extends	false	final
finally	float	for	function
goto	if	implements	import
in	instanceof	int	interface
let	long	native	new
null	package	private	protected
public	return	short	static
super	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

alert()

console.log()

```
name = '손흥민';
birthYear = 1992;

alert(name);
alert(birthYear);
alert('축구선수');
```

```
name = '손흥민';
birthYear = 1992;

console.log(name);
console.log(birthYear);
console.log('축구선수');
console.log('축구선수:', name, '태어난 년도:', birthYear);
```

```
name = '손흥민';
birthYear = 1992;
```

```
name = '손흥민';
birthYear = 1992;
.
.
.
name = '방탄소년단';
console.log(name);
```

변수 선언 및 할당

- 변수 선언 : 키워드 let이나 const 다음에 변수 이름을 적어서 변수를 선언
(const는 상수를 위한 키워드로 프로그램 안에서 바뀌지 않는 값을 의미)

- 변수에 값 할당 : 변수 오른쪽에 = 기호를 붙이고, 오른쪽에 저장할 값이나 식을 작성

```
let name;
name = '손흥민';
```

- 변수 선언과 할당은 동시에 가능

```
let name = '손흥민';
```

```
let name = '손흥민';
birthYear = 1992;

.

.

.
let name = '방탄소년단';
console.log(name);
```

```
let name = '손흥민';
birthYear = 1992;

.

.

.
let name = '방탄소년단';
console.log(name);
```

```
let name = '손흥민';
birthYear = 1992;

.

.

.
name = '방탄소년단';
console.log(name);
```

최초로 선언되는 변수에 let을 붙이는 습관을 들인다면,
이미 선언된 변수가 변경되는 것을 방지할 수 있다.

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
```

변수명에 대한 규칙

1. 문자, 숫자, $, _ 만 사용 가능합니다.
2. 절대 첫 글자에는 숫자가 올 수 없습니다.
3. 절대 예약어는 사용할 수 없습니다.
4. 변수명은 읽기 쉽고 이해할 수 있게 선언하는 것이 좋습니다.

변수명으로 사용될 수 없는 것을 모두 고르세요.

A. `let 토트넘 = "England Football Club";`
B. `let name = '손흥민';`
C. `let 1th_position = 'Forward';`
D. `let number = 7;`
E. `let $ = '€ 80,000,000';`
F. `let _birth_Year = 1992;`
G. `let true = 'The best football player';`

변수명으로 사용될 수 없는 것을 모두 고르세요.

A. let 토트넘 = "England Football Club";
B. let name = '손흥민';
C. ~~let 1th_position = 'Forward';~~ // 숫자는 변수명 맨 앞에 올 수 없습니다.
D. let number = 7;
E. let $ = '€ 80,000,000';
F. let _birth_Year = 1992;
G. ~~let true = 'The best football player';~~ // true는 예약어입니다.

자료형

(Data Types)

자료형이란?

- 프로그램에서 처리하는 자료의 형태
- 자바스크립트에서는 자료형을 기본형(원시형, primitive type)과 객체형(object)으로 구분

typeof() 함수

- 괄호 안에 값이나 변수를 넣으면, 어떤 자료형인지 알려주는 함수

```
typeof("안녕하세요")

let data = 5
typeof(data)
```

숫자 자료형(number)

- 자바스크립트에서는 정수와 실수를 함께 묶어서 숫자형이라고 한다.
  (최근 숫자형의 한계를 넘는 큰 정수를 다루기 위해 Bigint라는 자료형이 추가되었다.)

- 숫자라고 하더라도 따옴표로 묶이면 숫자가 아닌 문자열로 인식한다.

```
typeof(10)      // 'number'
typeof("10")    // 'string'
typeof(3.145)   // 'number'
```

```
const BIRTH_YEAR = 1992;
const PI = 3.14;
let age = 30;
let avgScore = 95.2;
```

```
let add = 2+7;
let subtract = 2-7;
let multiply = 2*7;
let square = 2**7;
let divide = 7/2;
let remainder = 7%2;
```

Infinity? NaN?

```
let unlimitedNumber = 5/0;

console.log(unlimitedNumber);
```

```
let unlimit = Infinity;

console.log(unlimit+3);
console.log(unlimit-3);
console.log(3-unlimit);
console.log(unlimit*3);
console.log(unlimit**3);
console.log(3**unlimit);
console.log(unlimit/3);
console.log(unlimit%3);
console.log(3/unlimit);
console.log(3%unlimit);
```

```
let unlimit = Infinity;


console.log(3-unlimit);



console.log(unlimit%3);
console.log(3/unlimit);
console.log(3%unlimit);
```

```
let unlimit = Infinity;


console.log(3-unlimit);          // -infinity



console.log(unlimit%3);          // NaN
console.log(3/unlimit);          // 0
console.log(3%unlimit);          // 3
```

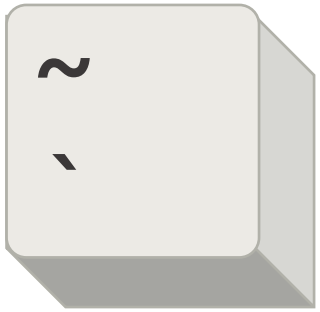
```
console.log('two'/2);
```

```
console.log('two'/2);      // NaN
```

문자 자료형(string)

- 작은 따옴표나 큰 따옴표 또는 백틱으로 묶은 문자열 데이터


ESC
F1
~
`
!
1
@
2
#
3
Tab
Q
W
Caps Lock
A
S


~
`

프로그래밍 입문자를 위한 '자바스크립트'

message = '프로그래밍 입문자를 위한 '자바스크립트'';

```
message = '프로그래밍 입문자를 위한 '자바스크립트'';
```

```
message = "He's the Best!";
console.log(message);
```

```
message = 'He\'s the Best!';
console.log(message);
```


Backspace
Ins
15ms
}
]
|
\ ₩
Del
Enter


|
\ ₩

```
let name = '손흥민';
let team = "토트넘";
let position = `윙백`;
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
let message = `제가 응원하는 선수는 ${name}입니다.`;

console.log(message);
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
let message = `제가 응원하는 선수는 ${2024-BIRTH_YEAR}살의 ${name}입니다.`;

console.log(message);
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
let message = '제가 응원하는 선수는‘ + name + '입니다.';

console.log(message);
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
let message = '제가 응원하는 선수는 ' + 2024-BIRTH_YEAR + '살의 ‘ + name + '입니다.‘;

console.log(message);
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;
let message = '제가 응원하는 선수는 ' + 2024 - BIRTH_YEAR + '살의 ' + name + '입니다.';

console.log(message);
```

불린 자료형(boolean)

- 참이나 거짓을 표현하기 위한 논리형 데이터 유형

```
let isEasy = true;
let isHard = false;
```

```
let name = '손흥민';
const BIRTH_YEAR = 1992;

console.log(name == '손흥민');
console.log(BIRTH_YEAR < 1990);
```

undefined

- 변수를 선언하기만 하고
  값을 할당할지 않을 때 변수의 초기값.
- undefined는 값이면서 동시에 자료형

```
let userName
userName          // undefined
```

null

- 유효하지 않은 값
- null 역시 값이면서 동시에 자료형

```
let age = null
```

undefined과 null

```
let first, second;
second = null;
console.log(first)
console.log(second)
```

undefined	null
선언만 하고 할당하지 않음	null 값을 할당함
주로 사용자의 실수에 의해 발생	주로 사용자가 의도적으로 null 을 할당

```
console.log(notDefined);
```

```
let noAssign;

console.log(noAssign);
```

```
let nullValue = null;

console.log(nullValue);
```

```
let message = '손흥민의 등 번호는 ';
let backNumber = 7;

console.log(message + backNumber);
```

```
let message = '손흥민의 등 번호는 ';
let backNumber = 7;

let newMessage = message + backNumber
console.log(typeof(newMessage));
```

```
let textTen = '10';
let five = 5;
console.log(textTen - five);
console.log(textTen * five);

let textFive = '5';
console.log(textTen - textFive);
console.log(textTen * textFive);
```

```
let textTen = '10';
let five = 5;
console.log(textTen + five);

let textFive = '5';
console.log(textTen + textFive);
```

배열

- 하나의 변수에 여러 값을 할당할 수 있는 형태
- 대괄호([ ])로 묶고, 그 안에 값을 나열함. 각 값은 쉼표(,)로 구분
- 대괄호 안에 아무 값도 없으면 '빈 배열'이라고 하고, 이것 역시 배열

```
배열명 = [값1, 값2, ...]
```

```
emptyArr = []                          // 빈 배열
colors = ["red", "blue", "green"]  // 문자열 배열
arr = [10, "banana", true]           // 여러 자료형으로 구성된 배열
```

배열과 인덱스

```
season = ["봄", "여름", "가을", "겨울"]
```

```
> season = ["봄", "여름", "가을", "겨울"]
▼(4) ['봄', '여름', '가을', '겨울'] i
    0: "봄"
    1: "여름"
    2: "가을"
    3: "겨울"
    length: 4
  ▶ [[Prototype]]: Array(0)
>
```

인덱스 → 0, 1, 2, 3

length: 4 ← 배열의 크기

- 인덱스 : 배열에 있는 여러 값을 저장하는 방 번호
- 인덱스는 0부터 시작!

- 두번째 값을 알고 싶다면 season[1]
- 배열에 있는 요소의 갯수를 알고 싶다면 season.length
- 배열의 마지막 값을 알고 싶다면 season[season.legth - 1]

심볼

- ES6에 새롭게 추가된 원시 유형의 자료형
- 심볼의 가장 큰 특징은 유일성을 보장한다는 것
- 심볼은 객체 프로퍼티의 키로 사용할 수 있다

(예) 자바스크립트 프로그램에서 오픈 소스를 가져와 사용하거나
다른 팀원이 만든 객체들을 함께 사용할 경우 객체의 키 이름이 중복될 수도 있다.
➔ 키 이름을 심볼로 지정하면 서로 충돌이 발생하지 않는다.

심볼

- 심볼을 만들 때는 Symbol() 함수 사용
- 심볼은 한 번 만들면 변경할 수도 없고, 같은 값을 가진 심볼을 만들 수도 없다.

```
Symbol()
```

```
let var1 = Symbol()
let var2 = Symbor()
```

var1과 var2는 똑같아 보이지만, 심볼은 유일한 값이기 때문에 두 변수는 같지 않다!

```
var1 === var2        // false
```

- 심볼을 키로 사용할 때에는 [키]처럼 대괄호로 묶어서 표현
- 키에 접근할 때도 마침표가 아닌 대괄호 사용

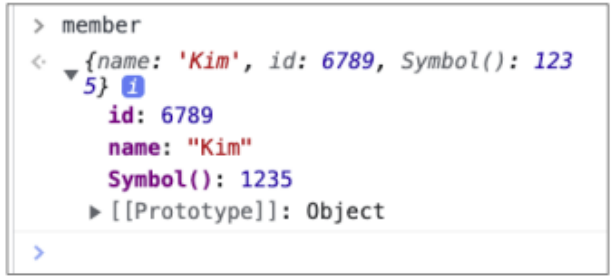
예) member 객체를 만들면서 id 키를 고유하게 만들기

```
let id = Symbol()
const member = {
  name : "Kim",
  [id] : 12345
}
```

```
member          // {name: "Kim", Symbol(): 12345}
member[id]      // 12345
```

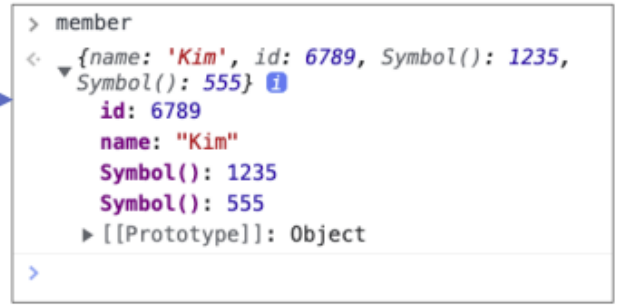
다시 id 키를 지정하면?

```
member.id = 6789
```



다시 심볼형 id 키를 지정하면?

```
id = Symbol();
member[id] = 555;
```



형 변환

(Data Type Conversion)

prompt()를 이용해,
세 과목 [국어(kor), 영어(eng), 수학(math)]의 점수를 입력받고,

console.log()를 이용해,
세 과목 점수의 평균(avg)를 콘솔 창에 출력해보세요.

```
let kor = prompt('국어 점수를 입력하세요');
let eng = prompt('영어 점수를 입력하세요');
let math = prompt('수학 점수를 입력하세요');

let avg = (kor + eng + math) / 3;
console.log(avg);
```

```
let kor = prompt('국어 점수를 입력하세요');
let eng = prompt('영어 점수를 입력하세요');
let math = prompt('수학 점수를 입력하세요');

console.log(
    typeof(kor),
    typeof(eng),
    typeof(math)
);
```

자바스크립트의 형 변환

- 자바스크립트는 다른 언어와 다르게 프로그램 실행 중에 자료형이 변환되는 언어
- 자동으로 형이 변환될 때에도 있다 → 이런 상황을 미리 알아 두지 않으면 오류를 발생시키기도 하고, 처음에 예상했던 것과 다른 결과가 나올 수도 있습니다.

C 언어나 자바 등 일반 프로그래밍 언어

- 변수를 선언할 때 변수의 자료형을 결정
- 자료형에 맞는 값만 변수에 저장 가능
- 자료형으로 인한 프로그램의 오류 방지 가능

```
int num = 20          // 정수형 변수 num
char *name = "John"   // 문자형 변수 name
```

자바스크립트

- 변수를 선언할 때 자료형 지정하지 않음
- 변수에 값을 저장할 때 자료형 결정
- 편리하긴 하지만 변수를 일관성 있게 유지하기 힘들다

```
num = 20        // 숫자형
num = "John"    // 문자열
```

자동 형 변환 (Promotion)

- 연산을 통해 자동으로 자료형이 변환되는 것
- 숫자와 문자열을 이용한 사칙연산에 주의!

강제 형 변환 (Casting)

- 직접 자료형을 변환시키는 것

- Number(), parseInt(), parseFloat()
- String(), toString()
- Boolean()

```
let promotion = "708090“ / 3;

console.log(typeof(promotion));
```

강제 형 변환 = 명시적 형변환 (Casting)

Number()

String()

Boolean()

```
let kor = Number(prompt('국어 점수를 입력하세요'));
let eng = Number(prompt('영어 점수를 입력하세요'));
let math = Number(prompt('수학 점수를 입력하세요'));

let avg = (kor + eng + math) / 3;
console.log(avg);
```

```
console.log(
    Number('7'),
    Number('Two'),
    Number(true),
    Number(false),
    Number(null),
    Number(undefined)
);
```

```
console.log(
    String(3),
    String(true),
    String(false),
    String(null),
    String(undefined)
);
```

```
console.log(
    Boolean(2),
    Boolean('0'),
    Boolean(' '),
    Boolean('true'),
    Boolean('false'),

    Boolean(0),
    Boolean(''),
    Boolean(null),
    Boolean(NaN),
    Boolean(undefined)
);
```

```
console.log(

    Boolean('0'),
    Boolean(' '),



    Boolean(0),
    Boolean('')



);
```