KB금융그룹 | 국민의 평생 금융파트너

기본 문법 학습 이후
js vanilla가 웹브라우저에 어떻게 작동하는지

2025년 상반기 K-디지털 트레이닝

자바스크립트 기본 문법

[KB] IT's Your Life

KB 국민은행

js는 태생적으로 웹브라우저에서 돌아가게끔 설계됨.
웹브라우저가 js 입장에서는 pc/OS.
이후로 웹브뿐만 아니라 다른 플랫폼(OS, 앱 등)에서도 사용 가능해짐

1 자바스크립트 기본 용어와 출력 방법

HTMLPage.html

```
<!DOCTYPE html>
<html>
    <head>
    <title>JavaScript Basic</title>
    <script>

    </script>
</head>
<body>

</body>
</html>
```

혹은 <script src="path">

script요소는 헤드태그에 맨 아래에

자바스크립트 기본 용어

표현식과 문장

- 표현식: 값을 만들어 내는 간단한 코드.
- 문장: 프로그래밍 언어에 실행할 수 있는 코드의 최소 단위.
- 문장 마지막에 세미콜론(;) 또는 줄 바꿈을 넣어 종결을 나타냄.

```
273;
10 + 20 + 30 * 2;
var name = '윤' + '인' + '성';
alert('Hello JavaScript');
```

(a) 문장 예

```
273
10 + 20 + 30 * 2
'JavaScript'
```

(b) 표현식 예

자바스크립트 기본 용어

키워드

- 자바스크립트를 처음 만들 때 정해진 특별한 의미가 부여된 단어

break	else	instanceof	true	case	false
new	try	catch	finally	null	typeof
continue	for	return	var	default	function
switch	void	delete	if	this	while
debugger	import	export	extends	super	yield
do	in	throw	with	const	class

1 자바스크립트 기본 용어와 출력 방법

자바스크립트 기본 용어

식별자

- 자바스크립트에서 변수나 함수 등에 이름을 붙일 때 사용하는 단어

사용할 수 있는 예	사용할 수 없는 예
alpha	break
alpha10	273alpha
_alpha	has space
$alpha	
AlPha	
ALPHA	

표기법들
kebap : hello-www.
snake : _붙이는거
camel : 띄어쓰기대신단어첫글자대문자
pascal : 단어 첫글자 대문자

html에서 kebap, snake많이 씀

js, java에서 클래스명은 pascal, 함수/변수는 camel

```
i love you → iLoveYou
i am a boy → iAmABoy
create server → createServer
```

글자가 모두 붙어 있지만 대·소문자로 구분했으므로 쉽게 끊어서 읽을 수 있습니다.

1 자바스크립트 기본 용어와 출력 방법

자바스크립트 기본 용어

식별자 종류

구분	단독으로 사용	다른 식별자와 함께 사용
식별자 뒤에 괄호 없음	변수	속성
식별자 뒤에 괄호 있음	함수	메서드

```
alert('Hello World')          → 함수
Array.length                  → 속성
input                         → 변수
prompt('Message', 'Defstr')   → 함수
Math.PI                       → 속성
Math.abs(-273)                → 메서드
```

1 자바스크립트 기본 용어와 출력 방법

자바스크립트 기본 용어

주석

방법	형태
❶ 한 행 주석 처리	// 주석문
❷ 여러 행 주석 처리	/* 주석문 주석문 */

```
<script>
	// 주석은 코드 실행에 영향을 주지 않습니다.
	/*
	alert('Hello JavaScript .. !');
	alert('Hello JavaScript .. !');
	alert('Hello JavaScript .. !');
	*/
</script>
```

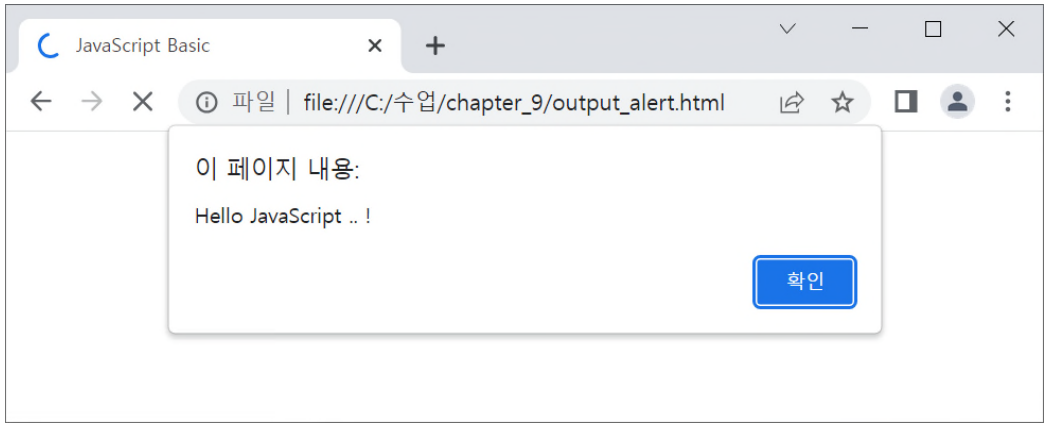
1 자바스크립트 기본 용어와 출력 방법

자바스크립트 출력

- 웹 브라우저에 경고 창 띄우기.

```
alert("메시지")
```

alert함수



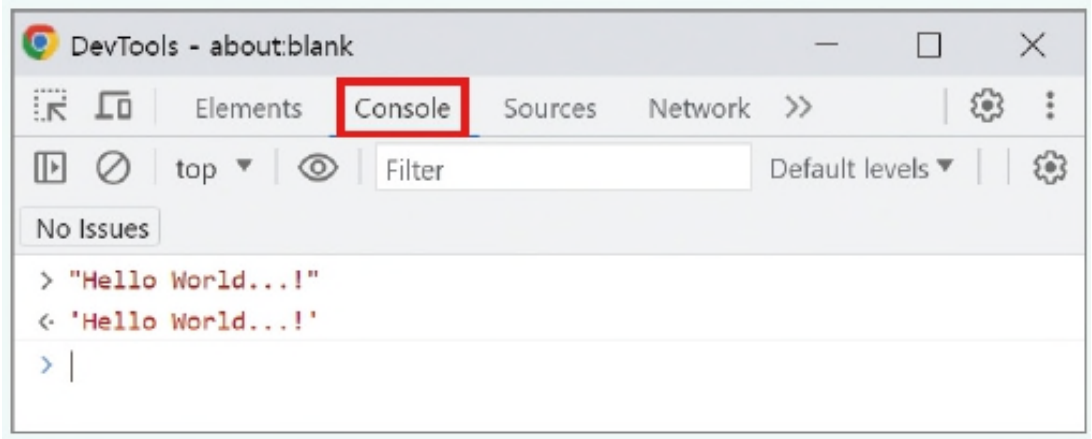
output_alert.html

```
<!DOCTYPE html>
<html>
<head>
	<title>JavaScript Basic</title>
	<script>
		alert('Hello JavaScript .. !');
	</script>
</head>
<body>

</body>
</html>
```

자바스크립트 출력

- 개발자 도구의 console에 출력하기

```
console.log("Hello World...!")
```



디버깅 때
브라우저에서의 콘솔창을
많이 이용한다

자료형

숫자

- 정수, 실수

정수 실수가 나뉘어져 있지 않음.
실수로 모든 것을 표현.
소숫점 안 붙이면 정수지 그게

사칙 연산자

연산자	설명	예
+	덧셈	> 52 + 273 325 > 52.273 + 103.57 155.843
-	뺄셈	> 52 - 273 -221 > 52.273 - 103.57 -51.29699999999999
*	곱셈	> 52 * 273 14196 > 52.273 * 103.57 5413.91461

부동소수점을 계산할 때
약간의 오차 발생.

연산자	설명	예
/	나눗셈	> 52 / 273 0.19047619047619047 > 52.273 / 103.57 0.504711789128126 > 1 / 0 Infinity

숫자를 0으로 나누었을 때,
무한을 나타내는 값이 됨.

반올림 ㄱㄱ

자료형

숫자

- 나머지 연산자

연산자	설명	예
%	나머지	> 10 % 5 0 > 7 % 3 1

숫자에 돈 처럼 000,000,00 ...를 찍고 싶으면
tolocalstring 변환함수 사용`

자료형

문자열

- 문자 집합

스크립트 언어에서는
단어, 문자열
모두 문자열로 표현함

방법	예
작은따옴표 사용	> 'Hello JavaScript .. !' "Hello JavaScript .. !" > '"문자열"입니다.' ""문자열"입니다."
큰따옴표 사용	> "Hello JavaScript .. !" "Hello JavaScript .. !" > "'문자열'입니다." "'문자열'입니다."

작은 따옴표로 통일해서 사용하자

사전에 prettier 설정을 통해서
문자열은 작은 따옴표로 설정했었음

자료형

문자열

- 이스케이프 문자

이스케이프 문자	설명	예
\t	수평 탭	> '한빛\t아카데미' "한빛 아카데미"
\n	행 바꿈	> '한빛\n아카데미' "한빛 아카데미"
\\	역 슬래시	> '\\\\' \\
\'	작은따옴표	> '\'\'\'' "'''"
\"	큰따옴표	> "\"\"\"" """"

템플릿 리터럴
백틱을 활용한다

`${변수명}`

근데 문자열+변수+문자열 ... 이 더 편하다

자료형

문자열

- 문자열 연결 연산자

연산자	설명	예
+	문자열 연결	> '가나다' + '라마' + '바사아' + '자차카타' + '파하' "가나다라마바사아자차카타파하"

+연산에서
다른 타입과
문자열이 피연산자라면

결과는 문자열로 변환되어 합쳐짐

'10'+5 => '105'

+아닌 연산자에서
다른 타입과 문자열이 피연산자면

오류가 나는게 아니라

문자열이 다른 타입에 맞게 형변화되고
연산을 실시한다.
보통 문자열이 숫자로 형변환된다.

'10'-5 => 5

자료형

불(Boolean)

- true/false 값 중 하나를 가짐
- 비교 연산자

연산자	설명	예
>=	좌변이 우변보다 크거나 같음	> 10 >= 20 false > '가방' >= '하마' false
<=	우변이 좌변보다 크거나 같음	> 10 <= 20 true
>	좌변이 우변보다 큼	> 10 > 20 false
<	우변이 좌변보다 큼	> 10 < 20 true
==	좌변과 우변이 같음	> 10 == 20 false
!=	좌변과 우변이 다름	> 10 != 20 true

형변환 후 연산하기에

'10' == 10
이
10 == 10 로 변환되어
true로 나온다.

=== 연산자는
타입부터 비교하고 그다음에 ==연산자처럼
비교를 진행한다.
'10 === 10 의 결과는
타입부터 다르기에 false

자료형

불(Boolean)

- 논리 연산자

연산자	설명	예
!	논리 부정(참이면 거짓, 거짓이면 참)	> !true false > !(10 == 10) false
&&	논리곱(둘 다 참이어야 참)	> true && true true > true && false false > false && true false > false && false false > (10 < 20) && (20 < 30) true

둘 다 참일 때만 참.

20이 10과 30 사이에 있다는 의미

자료형

불(Boolean)

- 논리 연산자

연산자	설명	예
||	논리합(둘 중 하나만 참이어도 참)	> true || true true > true || false true > false || true true > false || false false

둘 중 하나만 참이어도 참.

변수 선언은
옛날에는 ㅍvar했지만 문제가 많아서 권장X

let을 써라

변수

let말고 다른 변수 키워드로는 const가 있다

변수: 값을 저장할 때 사용하는 식별자

- 변수 선언
- 변수에 값을 할당

```
① > let pi;                    // 변수 선언
  undefined
② > pi = 3.14159265;           // 값 할당
  undefined
```

- 변수 초기화

```
> let pi = 3.14159265;
undefined
```

이처럼 하나의 표현식을 한 번에 대하는 시스템을
repl
read evaluate print loop라고 한다.

표현식을 읽고 평가하고 출력하고 반복한다.

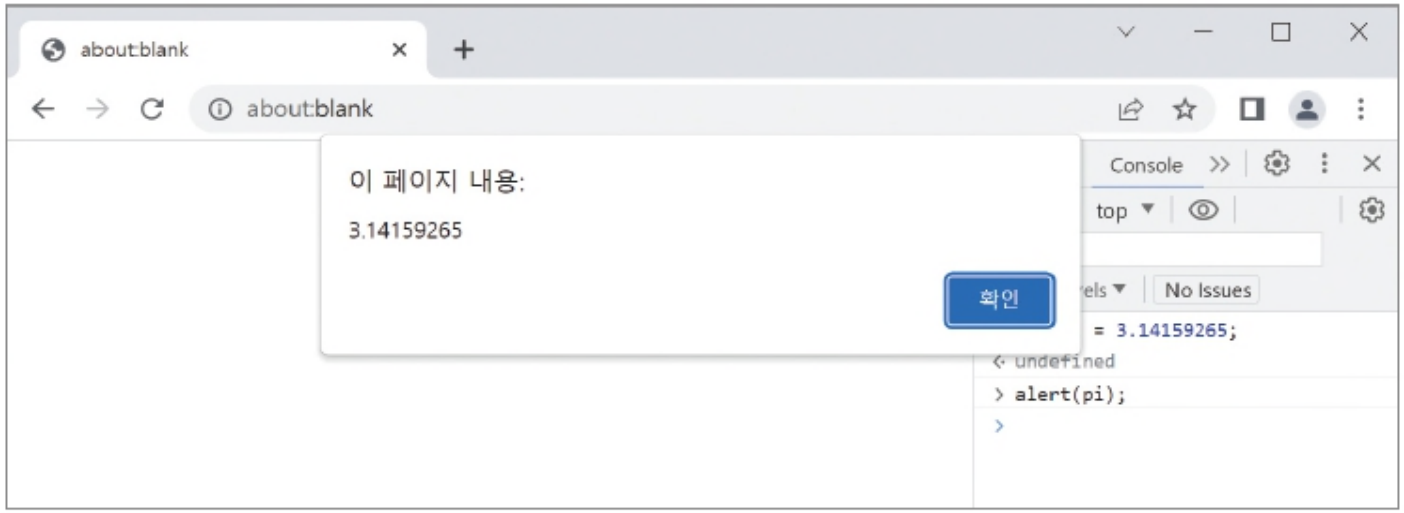
pi만 쳐도 console.log함수를 사용하지 않아도
해당 문장 ,표현식 자체에 대한 repl을 진행하기에
그 결과가 출력창에 출력함

js에는 undefined라는 특수한 값이 있다.
메모리 공간은 잡혔는데, 공간에 값이 배정되지 않았을 때. null아니냐?

변수

변수에 저장된 값 출력

```
> let pi = 3.14159265;
undefined
> alert(pi);
undefined
```



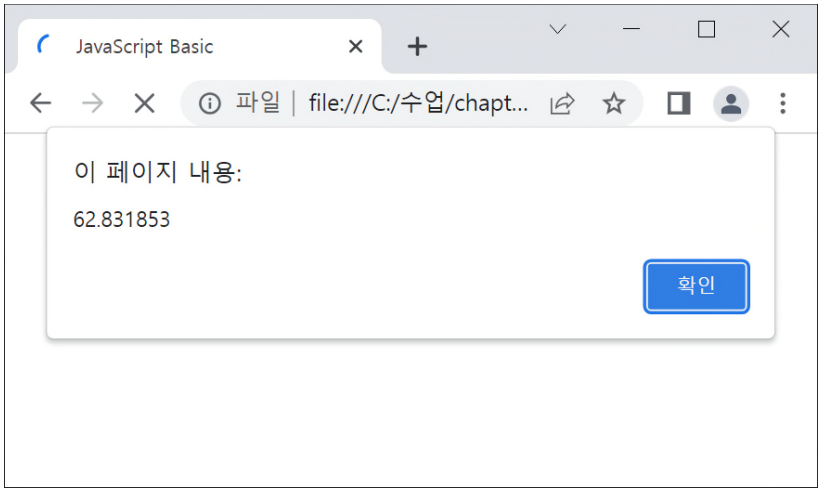
var는 선언된 위치 전에서도 사용ㄱㄴ
오히려 안좋아

let 키워드 특.

선언된 위치에서부터 사용할 수 .
있다. 오히려 좋아
재선언시 오류.
오히려 좋아

variable_use.html 변수

```
<script>
    // 변수를 선언 및 초기화합니다.
    let radius = 10;
    let pi = 3.14159265;

    // 출력합니다.
    alert(2 * radius * pi);
</script>
```



변수

- const 키워드
- let을 붙인 식별자는 변수, const를 붙인 식별자는 상수.

const는 상수 표현

```
let a = 10
const b = 10
```

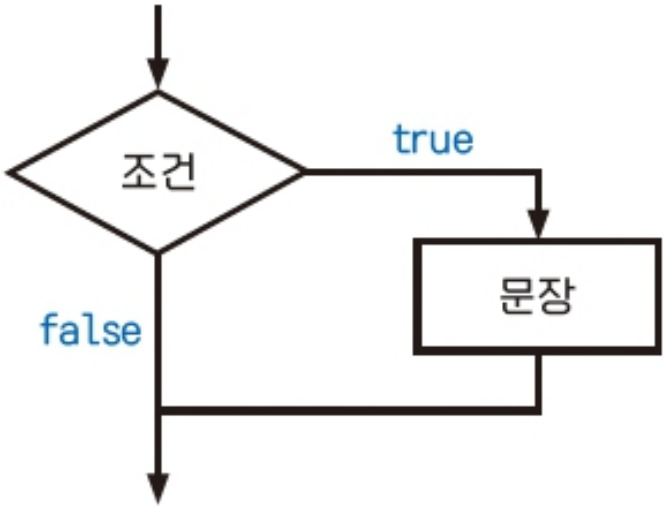
상수변수는 대문자로만+snake로 표현하는게 관용.

```
<script>
	// radius와 pi가 이후로 바뀌지 않으므로
	// let을 const로 변경합니다.
	const radius = 10;
	const pi = 3.14159265;

	alert(2 * radius * pi)
</script>
```

조건문

if 조건문

```
if (조건) {
    문장
}
```



condition_basic.html if 문으로 참과 거짓 판별

```
<script>
    // 조건문
    if (273 < 52) {
        // 표현식 "273 < 52"가 참일 때 실행합니다.
        alert("273 < 52 => true");
    }
    // 프로그램 종료
    alert("프로그램 종료");
</script>
```

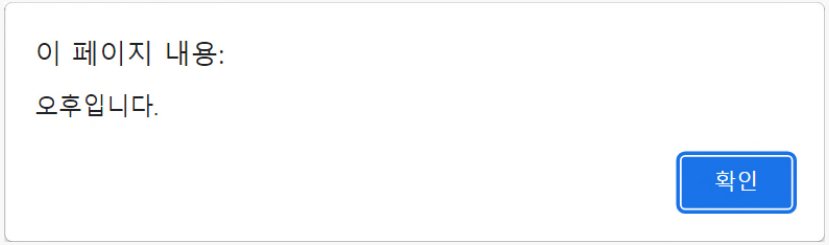
condition_getDate.html 현재 시간 구하기

```
<script>
    // Date 객체를 선언합니다.
    let date = new Date();          객체 생성 초기화

    // 요소를 추출합니다.
    let year = date.getFullYear();
    let month = date.getMonth() + 1;
    let day = date.getDay();
    let hours = date.getHours();
    let minutes = date.getMinutes();
    let seconds = date.getSeconds();
</script>
```

getMonth()는 0부터 시작.
우리에게 익숙한 월 표현으로
변경하기 위해 1씩 더함.

condition_date.html if 조건문으로 오전 오후 판별

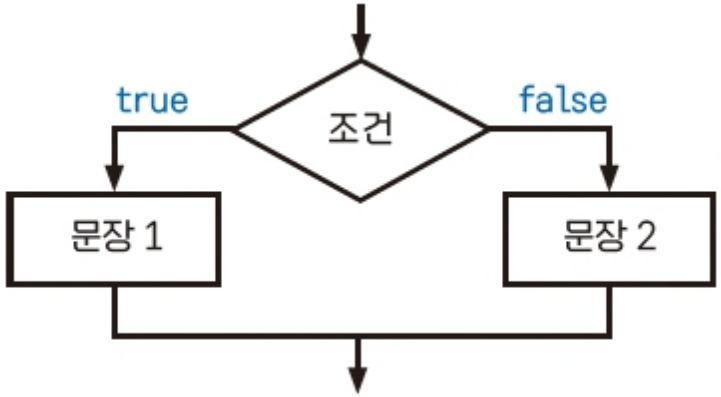
```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let hours = date.getHours();

    // 조건문
    if (hours < 12) {
        // 표현식 "hours < 12"가 참일 때 실행합니다.
        alert('오전입니다.');
    }

    if (12 <= hours) {
        // 표현식 "12 <= hours"가 참일 때 실행합니다.
        alert('오후입니다.');
    }
</script>
```



조건문

if else 조건문

```
if (조건) {
    문장 1
} else {
    문장 2
}
```



condition_else.html　　if else 조건문으로 오전과 오후 판별

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let hours = date.getHours();

    // 조건문
    if (hours < 12) {
        // 표현식 "hours < 12"가 참일 때 실행합니다.
        alert('오전입니다.');
    } else {
        // 표현식 "12 <= hours"가 참일 때 실행합니다.
        alert('오후입니다.');
    }
</script>
```

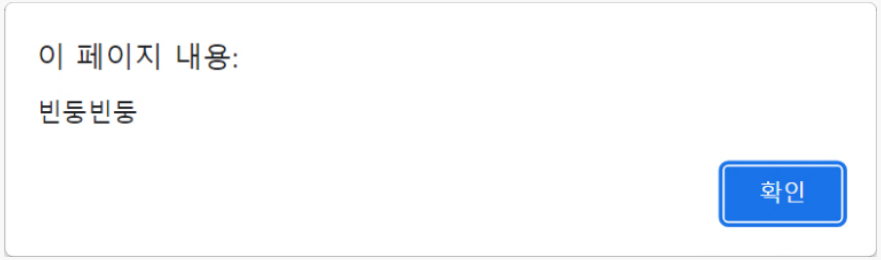
조건문

중첩 조건문과 if else if 조건문

```
if (조건) {
    if (조건) {
        문장
    } else {
        문장
    }
} else {
    if (조건) {
        문장
    } else {
        문장
    }
}
```

condition_duplication.html 중첩 조건문으로 하루 일정 표현

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let hours = date.getHours();

    // 조건문
    if (hours < 5) {
        alert('잠을 자렴....');
    } else {
        if (hours < 7) {
            alert('준비');
        } else {
            if (hours < 9) {
                alert('출근');
            } else {
                if (hours < 12) {
                    alert('빈둥빈둥');
                } else {
                    if (hours < 14) {
                        alert('식사');
                    } else {
                        // 여러 가지 업무를 수행
                    }
                }
            }
        }
    }
</script>
```



조건문

- if else if 조건문: 중복되지 않는 조건 세 가지 이상을 구분할 때 사용

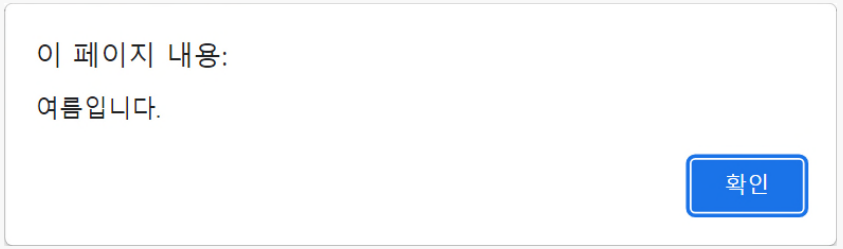
```
if (조건) {
    문장
} else if (조건) {
    문장
} else if (조건) {
    문장
} else {
    문장
}
```

condition_ifelseif.html　　if else if 조건문으로 하루 일정 표현

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let hours = date.getHours();

    // 조건문
    if (hours < 5) {
        alert('잠을 자렴....');
    }  else if (hours < 7) {
        alert('준비');
    } else if (hours < 9) {
        alert('출근');
    } else if (hours < 12) {
        alert('빈둥빈둥');
    } else if (hours < 14) {
        alert('식사');
    } else {
        // 여러 가지 업무를 수행합니다.
    }
</script>
```

note_logic.html 논리 연산자 사용

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let month = date.getMonth() + 1;

    // 조건문
    if (3 <= month && month <= 5) {
        alert('봄입니다.');
    } else if (6 <= month && month <= 8) {
        alert('여름입니다.');
    } else if (9 <= month && month <= 11) {
        alert('가을입니다.');
    } else {
        alert('겨울입니다.');
    }
</script>
```



반복문

```
<script>
	alert('출력');
	alert('출력');
	alert('출력');
	alert('출력');
	alert('출력');
</script>
```

```
<script>
	for (let i = 0; i < 1000; i++) {
		alert('출력');
	}
</script>
```

반복문

배열

- 변수 여러 개를 한꺼번에 다룰 수 있는 자료형.

배열 선언시에도
변수, 객체 똑같이.

```
// 변수를 선언합니다.
let array = [273, 32, 103, 57, 52];
```

js에서는 배열과 객체가 주

js에서는 배열의 크기를 동적으로 조절할 수 있다.
딱히 list같은 자료구조가 필요 없다!

근본 없다

array_basic.html

```
<script>
    // 변수를 선언합니다.
    let array = [273, '문자열', true, function () { }, {}, [32, 103]];
    alert(array);
</script>
```

함수도 품을 수 있음 ㅋㅋ

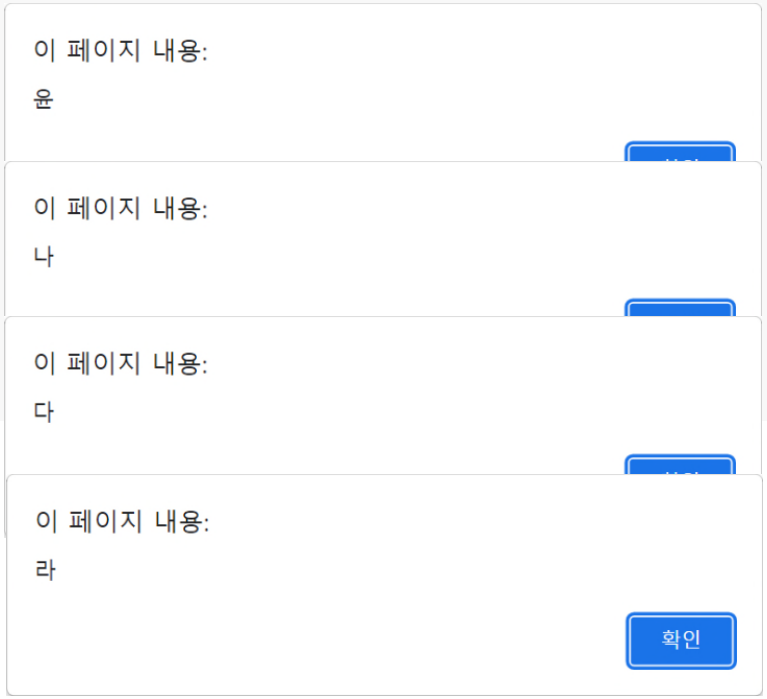
다양한 자료형을 품을 수 있다.

배열안에 배열도 ㅋㅋ

객체 리터럴 표현이다.

array_index.html 배열 생성과 배열 요소 접근

```
<script>
    // 변수를 선언합니다.
    let array = ['가', '나', '다', '라'];

    // 배열 요소를 변경합니다.
    array[0] = '윤';

    // 요소를 출력합니다.
    alert(array[0]);
    alert(array[1]);
    alert(array[2]);
    alert(array[3]);
</script>
```



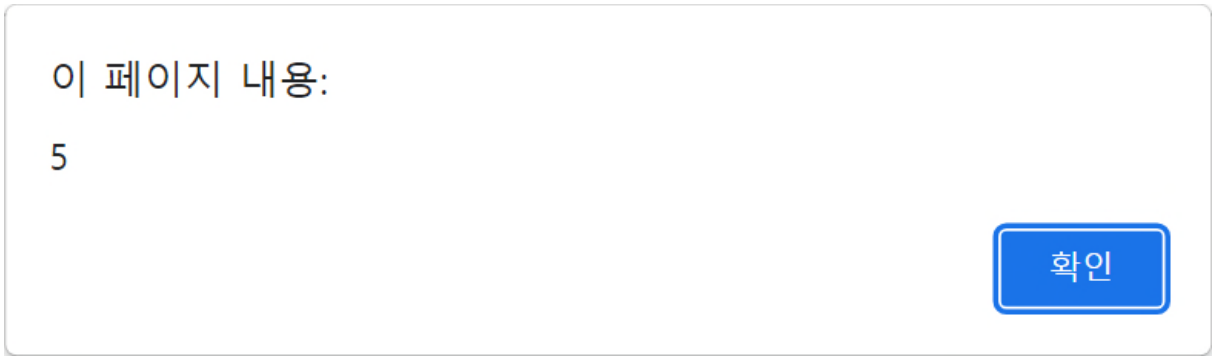
배열엔 속성 말고도
기본적으로 제공하는 메소드가 있다.

note_arrayLength.html 배열 요소의 개수

```
<script>
    // 변수를 선언합니다.
    let array = [10, 20, 30, 40, 50];

    // 출력합니다.
    alert(array.length);
</script>
```

lenght속성. => 배열의 길이.

다른 언어에서는 read-only속성이지만.
js에서는 역으로 해당 경로를 통해 length 속성 값을 대입 연산으로 바꿀 수 있다.
가변적이다. 사이즈를 바로 변경시킬 수 있다!

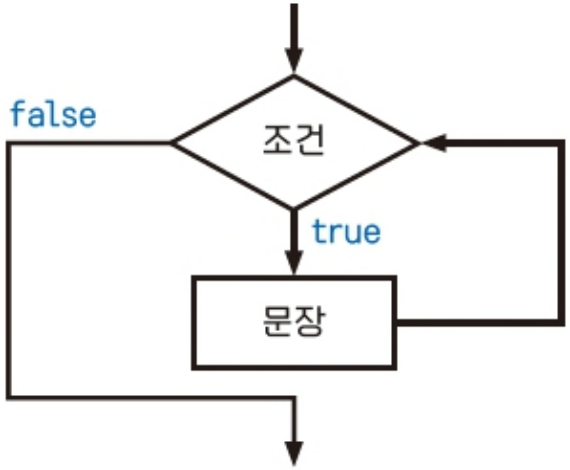


반복문

while 반복문

- 불 표현식이 참이면 중괄호 안 문장을 계속 실행.

```
while (조건) {
    문장
}
```

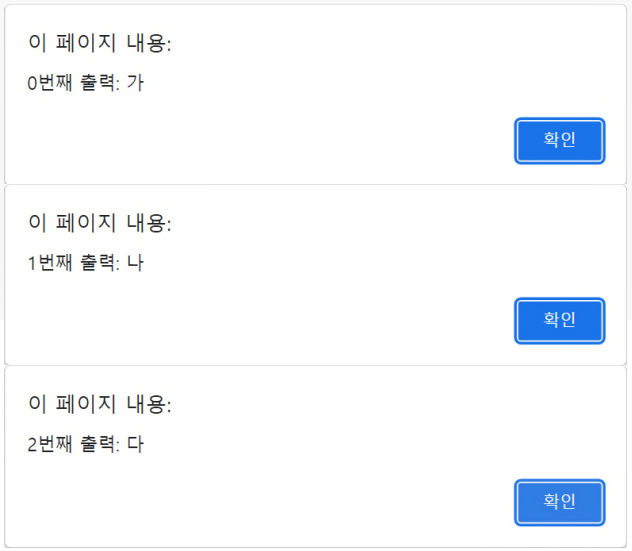


- 조건을 거짓으로 만드는 문장이 없으면 무한 반복.

```
<script>
    // 반복을 수행합니다.
    while (true) {
        alert('무한 반복');
    }
</script>
```

loop_while.html while 반복문

```
<script>
    // 변수를 선언합니다.
    let i = 0;
    let array = ['가', '나', '다'];

    // 반복을 수행합니다. i가 배열 원소 개수인 3보다 작을 때 반복합니다.
    while (i < array.length) {
        // 출력합니다.
        alert(i + '번째 출력: ' + array[i]);

        // 탈출하려고 변수를 더합니다.
        i++;
    }
</script>
```

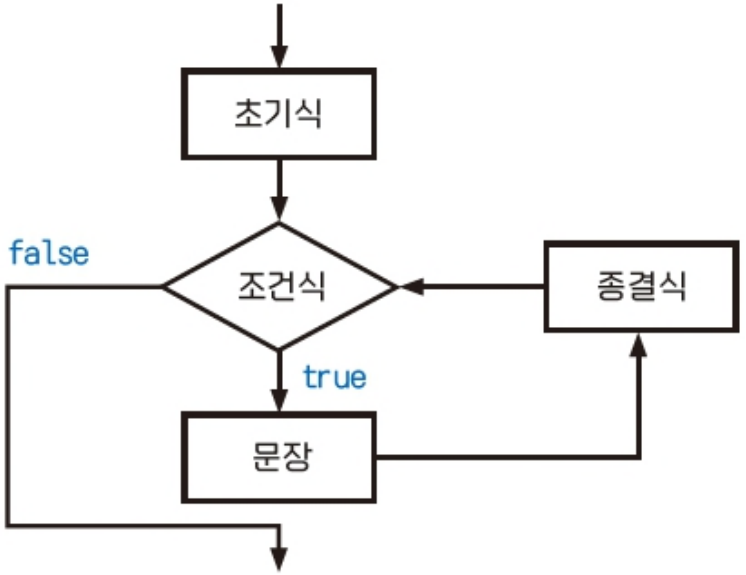


반복문

for 반복문

- 원하는 횟수만큼 반복하고 싶을 때 사용

```
for (초기식; 조건식; 종결식) {
    문장
}
```



반복문

for 반복문

- for 반복문 단계

① 초기식을 비교합니다.
② 조건식을 비교합니다. 조건이 거짓이면 반복문을 종료합니다.
③ 문장을 실행합니다.
④ 종결식을 실행합니다.
⑤ 앞의 단계로 이동합니다.

```
for (let i = 0; i < 반복 횟수; i++) {

}
```

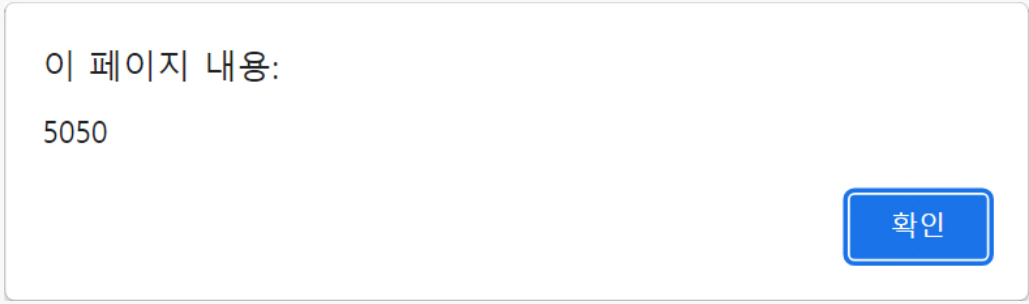
loop_for.html for 반복문

```
<script>
    // 변수를 선언합니다.
    let array = ['가', '나', '다'];

    // 반복을 수행합니다.
    for (let i = 0; i < 3; i++) {
        // 출력합니다.
        alert(i + '번째 출력: ' + array[i]);
    }
</script>
```

console.log() 시
콤마로 값들을 나열하면 띄어쓰기하며 나열한다

loop_forSum.html　　for 반복문을 사용한 0부터 100까지 합 계산

```
<script>
    // 변수를 선언합니다.
    let output = 0;

    // 반복을 수행합니다.
    for (let i = 0; i <= 100; i++) {
        output += i;
    }

    // 출력합니다.
    alert(output);
</script>
```



물론 js에는 do while문도 있따

for (in) 과 for(of) 도 있따.
순회용 반복문

for(let i in arr) {...} – 키에 대한 순회
i에는 값에 접근할 수 있는 키가 주어진다.
arr에서는 그 키가 인덱스죠?
주의할 것은 undfined값을 가지는 요소가 있다면
해당 요소에 대한 키는 건너뛴다!

for (let i of arr) {...} – 값에 대한 순회
i에는 값이 들어간다.
undefined 안따지고 다 대입됨.

배열 말고 객체도 가능하다.
객체는 필드 명과 필드 값을 순회한다.
참고로 js에서 객체는 키와 값 쌍들로 구성된다.
단 객체를 대상으로 of 연산시 iterable하지 않는 대상이라고 오류가 난다.
in연산시에는 되는데... in 연산시에는 똑같이 키들을 순회한다

함수는 재사용 가능한 코드 블록이다

KB국민은행

선언과 호출, 실행 우선순위

변하지 않는 부분 일반화된 코드
변하는 부분 변수로

선언과 호출

- 함수 생성 방법

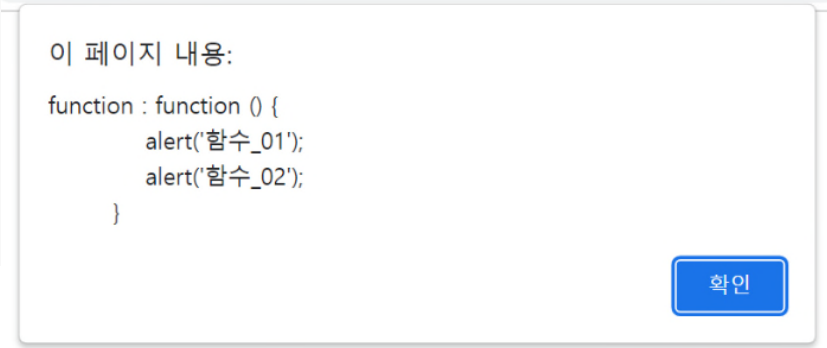
방법	표현
익명 함수	`function () {  }`

선언적 함수	`function 함수() {`
	
	`}`

변수에 할당해서 사용하거나, 일회용으로 사용ㅇ하거나

function_noname.html 익명 함수 선언하기

```
<script>
    // 함수를 선언합니다.
    let 함수 = function () {
        alert('함수_01');
        alert('함수_02');
    };

    // 출력합니다.
    alert(typeof (함수) + ' : ' + 함수);
</script>
```

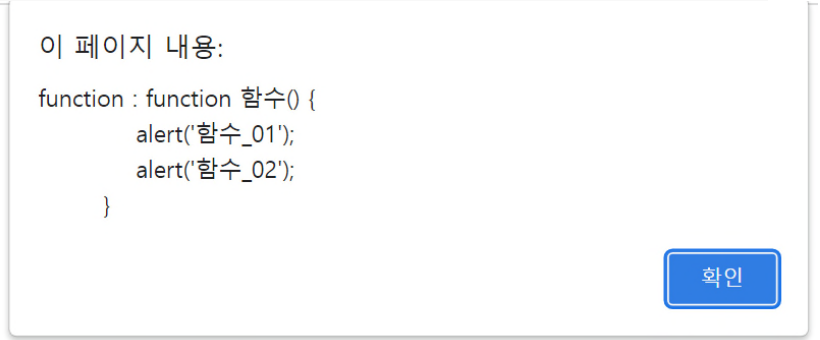


function_name.html 선언적 함수 선언하기

```
<script>
    // 함수를 선언합니다.
    function 함수() {
        alert('함수_01');
        alert('함수_02');
    };

    // 출력합니다.
    alert(typeof (함수) + ' : ' + 함수);
</script>
```

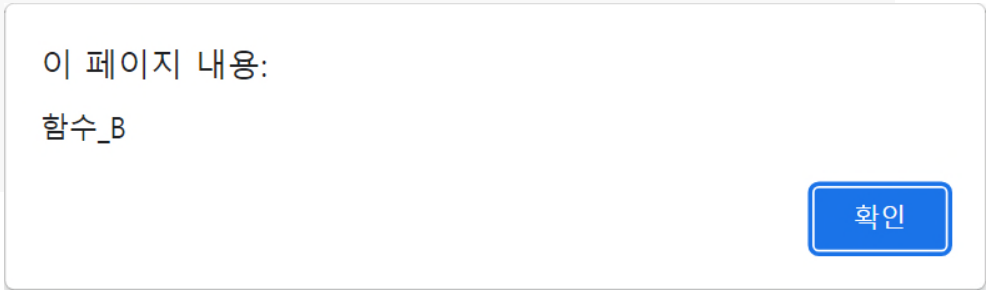
function 타입`` 함수 본문 출력

변수의 타입을 확인하는 typeof연산자
반환형은 문자열



function_priorityBetweenNoname.html 실행 우선 순위

```
<script>
    // 함수를 선언합니다.
    var 함수 = function () { alert('함수_A'); };
    var 함수 = function () { alert('함수_B'); };

    // 함수를 호출합니다.
    함수();
</script>
```



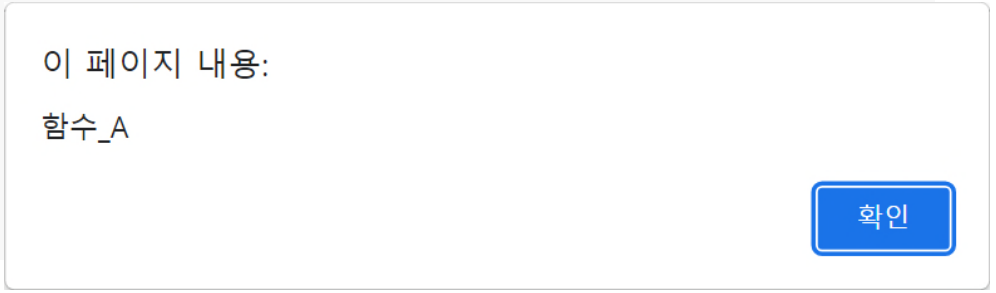
function_priority.html 실행 우선 순위

```
<script>
    // 함수를 선언합니다.
    함수 = function () {
        alert('함수_A');
    };

    function 함수() {
     alert('함수_B');
    }

    // 함수를 호출합니다.
    함수();
</script>
```

익명 함수 반수 식별자.

명명 함수 식별자가 같을 때는

변수가 더 높다



매개변수와 반환 값

- 매개변수: 함수의 괄호 안에 집어넣어 함수 쪽에 추가적인 정보를 전

```
// 함수를 호출합니다.
alert('매개변수');
```

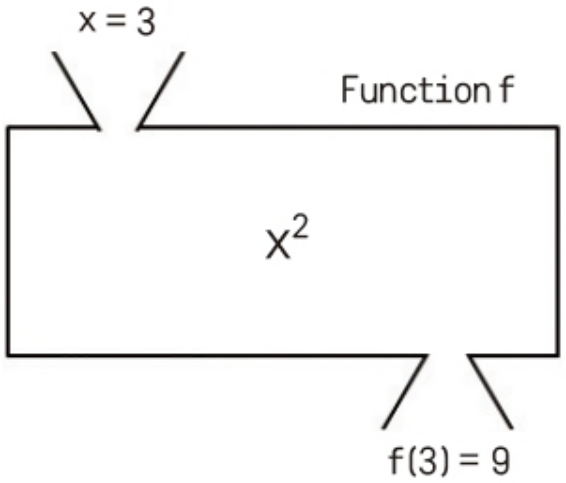
- 리턴 값: 함수를 실행한 결과

```
let minutes = date.getMinutes();
let seconds = date.getSeconds();
```

```
function 함수 이름(매개변수, 매개변수, 매개변수) {
    // 함수 코드
    // 함수 코드
    // 함수 코드
    return 반환 값;
}
```

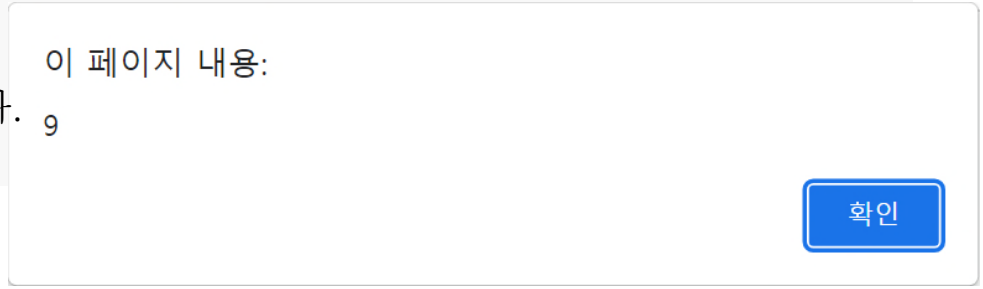
번외 2 날짜 사이에 일수를 구하는 함수

```
const getDateDiff = (d1, d2) => {
  const date1 = new Date(d1);
  const date2 = new Date(d2);

  const diffDate = date1.getTime() - date2.getTime();

  return Math.abs(diffDate / (1000 * 60 * 60 * 24));
}

getDateDiff("2021-09-01", "2021-10-01");
// 30
```



function_return.html매개변수와 반환 값이 있는 함수

```
<script>
    // 함수를 선언합니다.
    function f(x) {
        return x * x;
    }

    // 함수를 호출합니다.
    alert(f(3));
</script>
```



js는 매개변수의 갯수를 체크하지 않는다.

f(3,4,5 ...) 가능

넘으면 버린다.

모자르다면? undefined값 자동으로 대입됨

중요!

function_callback.html 콜백 함수

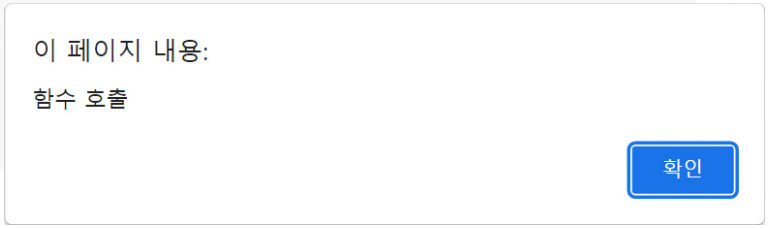
```
<script>
    // 함수를 선언합니다.
    function callTenTimes(callback) {
        // 10회 반복합니다.
        for (let i = 0; i < 10; i++) {
            callback(); // 매개변수로 전달된 함수를 호출합니다.
        }
    }

    // 변수를 선언합니다.
    let fun = function () {
        alert('함수 호출');
    };

    // 함수를 호출합니다.
    callTenTimes(fun);
</script>
```

함수포인터라고 생각하자.

주소를 넘겨 매개변수에. 참조형 매개변수가 되어버려

매개변수로 함수를 넘기는 것.

호출에 대한 주도가
다른 함수가 하는 것.



이벤트 핸들러가 이러한 형식

function_nonameCallback.html 콜백함수

```
<script>
    // 함수를 선언합니다.
    function callTenTimes(callback) {
        for (let i = 0; i < 10; i++) {
            callback();
        }
    }

    // 함수를 호출합니다.
    callTenTimes(function() {
        alert('함수 호출');
    });
</script>
```

함수를 받는 매개변수(콜백용)은
매개변수 나열시
항상 마지막에 하는 것이 관례

익명함수 활용.
매개변수로 넘겨주는 거 말고는 재사용할 필요 없으니

뒤에서 나오는 클로져 기법에 이런식으로 익명함수를 활용할 수 있다.

클로저에서 명명함수로는 활용X

객체 개요

○ 배열 선언과 요소 접근 예 – 인덱스 이용

```
<script>
    // 배열을 선언합니다.
    let array = ['사과', '바나나', '망고', '딸기'];
</script>
```

(a) 배열 선언

```
array[0] → '사과'
array[2] → '망고'
```

(b) 배열 요소 접근

배열은 위치정보가 지표지만

부가정보를 , 명치정보를 통해서 접근하고 싶다면 객체

인덱스	요소
0	사과
1	바나나
2	망고
3	딸기

배열은 인덱스를 통해서 값에 접근. 배열은 인덱스가 키, 위치정보가 키다.
객체는 키를 통해서 값에 접근. 객체는 필드명(속성명)이 키. 테이블(맵) 이라고 생각해

객체 개요

객체는 요소에 접근할 때 키(속성명)를 사용.

```
<script>
    // 객체를 선언합니다.
    let product = {
        제품명: '7D 건조 망고',
        유형: '당절임',
        성분: '망고, 설탕, 메타중아황산나트륨, 치자황색소',
        원산지: '필리핀'
    };
</script>
```

```
객체 = {
키: 값,
...
};
```

키는 문자열 or symbol타입 이어야함.
값은 숫자 문자열 함수 객체 배열 등 모든 타입이 될 수 있다

키	속성
제품명	7D 건조 망고
유형	당절임
성분	망고, 설탕, 메타중아황산나트륨, 치자황색소
원산지	필리핀

객체 개요

- 객체 뒤에 대괄호를 사용해 키를 입력하면 객체 속성에 접근.
- 객체 뒤에 .을 입력해 객체 속성에 접근.

대괄호 안에 문자열이 아니면
해당 키워드를 그냥 변수 인식해
변수 값을 통해서 속성에 접근할 것이다

```
product['제품명'] → '7D 건조 망고'
product['유형'] → '당절임'
product['성분'] → '망고, 설탕, 메타중아황산나트륨, 치자황색소'
product['원산지'] → '필리핀'

product.제품명 → '7D 건조 망고'
product.유형 → '당절임'
product.성분 → '망고, 설탕, 메타중아황산나트륨, 치자황색소'
product.원산지 → '필리핀'
```

js는 객체 리터럴를 지원하는 몇 안되는 언어.

다른 언어들은 new를 사용해야 하는데
js는 {}로
객체 인스턴스 바로 생성 ㄱㄴ

객체 개요

- 다음과 같은 형식으로 for in 반복문을 작성해 객체를 순환.

```
for (let 키 in 객체) {
    문장
}
```

object_withForIn.html 객체 속성 순회

```
<script>
    // 객체를 선언합니다.
    let product = {
        제품명: '7D 건조 망고',
        유형: '당절임',
        성분: '망고, 설탕, 메타중아황산나트륨, 치자황색소',
        원산지: '필리핀'
    };

    // 출력합니다.
    for (let i in product) {
        alert(i + ':' + product[i]);
    }
</script>
```

속성과 메서드

- 요소: 배열에 있는 값 하나하나.
- 속성: 객체에 있는 값 하나하나.

```
// 객체를 선언합니다.
let object = {
        number: 273,
        string: 'rintiantta',
        boolean: true,
        array: [52, 273, 103, 32],
        method: function () {
        }
};
```

속성과 메서드

- 메서드: 객체 속성 중 자료형이 함수인 속성

```
// 객체를 선언합니다.
let person = {
        name: '윤인성',
        eat: function (food) {
                alert(food + '을/를 먹습니다.');
        }
};

// 메서드를 호출합니다.
person.eat('밥');
```

그냥 일반 객체 메소드 처럼
명명함수를 객체 안에서 정의해도 된다.

속성과 메서드

- 객체에 있는 속성을 메서드에서 사용하고 싶을 때는 자신(this)이 가진 속성임을 분명하게 표시해야 함.

- this: 현재 인스턴스에 대한 참조

object_this.html

```
<script>
    // 객체를 선언합니다.
    let person = {
        name: '윤인성',
        eat: function (food) {
            alert(this.name + '이 ' + food + '을/를 먹습니다.');
        }
    };

    // 메서드를 호출합니다.
    person.eat('밥');
</script>
```

this는 정의될때가 아닌 호출될 때 정의된다.

반환 값이 없는 함수면
undefined 반환한다고 생각하면 된다.?

let result=person.eat('rice');

js에도 class키워드가 있다!!!