KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

자바스크립트 기본 문법

[KB] IT's Your Life

KB 국민은행

HTMLPage.html

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

    </script>
</head>
<body>

</body>
</html>
```

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

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	

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

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

자바스크립트 기본 용어

ㅇ 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
```

자바스크립트 기본 용어

○ 주석

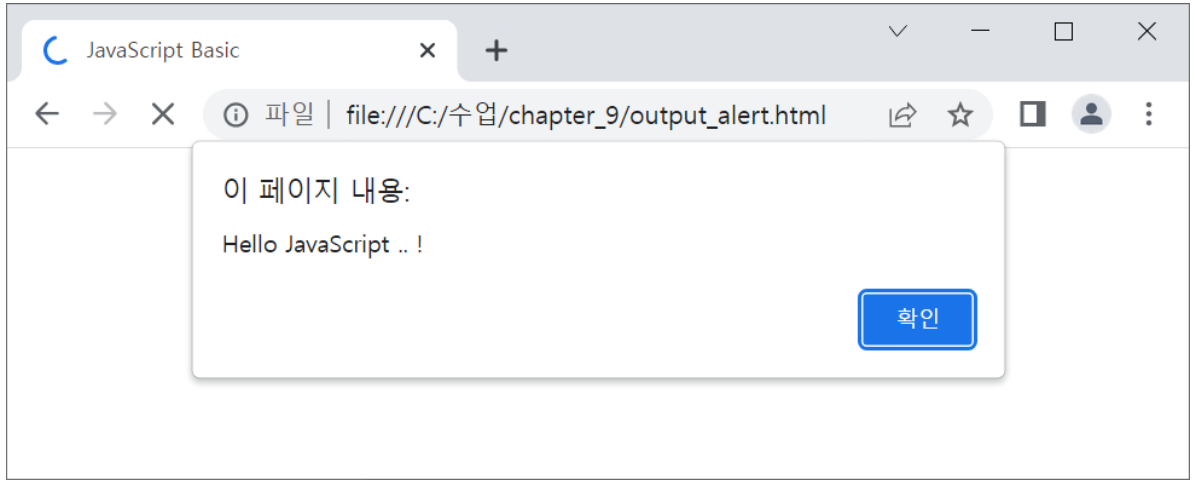
방법	형태
❶ 한 행 주석 처리	// 주석문
❷ 여러 행 주석 처리	/* 주석문 주석문 */

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

자바스크립트 출력

- 웹 브라우저에 경고 창 띄우기.

```
alert("메시지")
```



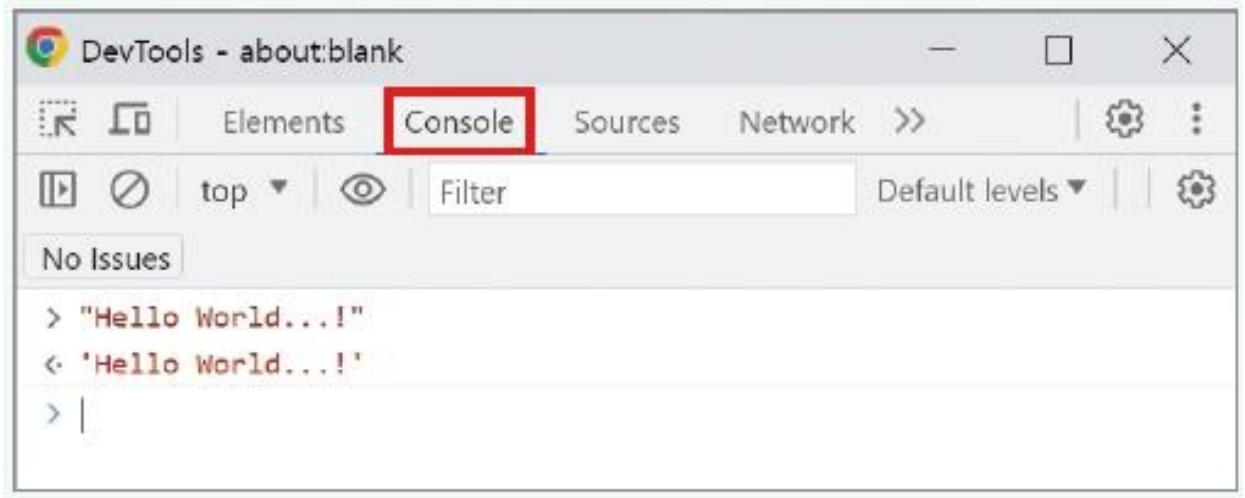
output_alert.html

```
<!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...!")
```



자료형

- 숫자
  - 정수, 실수

- 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.

자료형

- 숫자
  - 나머지 연산자

연산자	설명	예
%	나머지	> 10 % 5 0 > 7 % 3 1

자료형

○ 문자열

- 문자 집합

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

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	> "\"\"\"" """""

자료형

ㅇ 문자열

- 문자열 연결 연산자

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

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

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

- 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
```

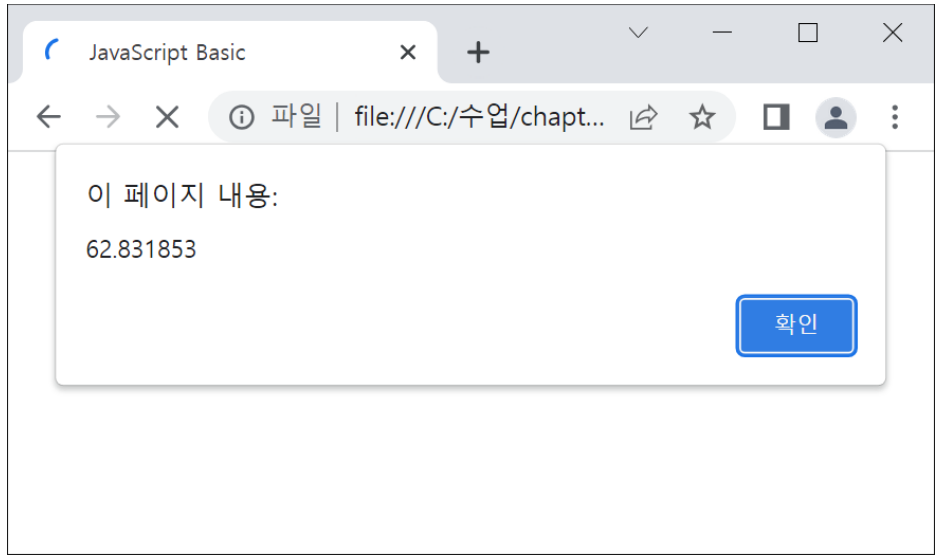
변수

- 변수에 저장된 값 출력

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

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.

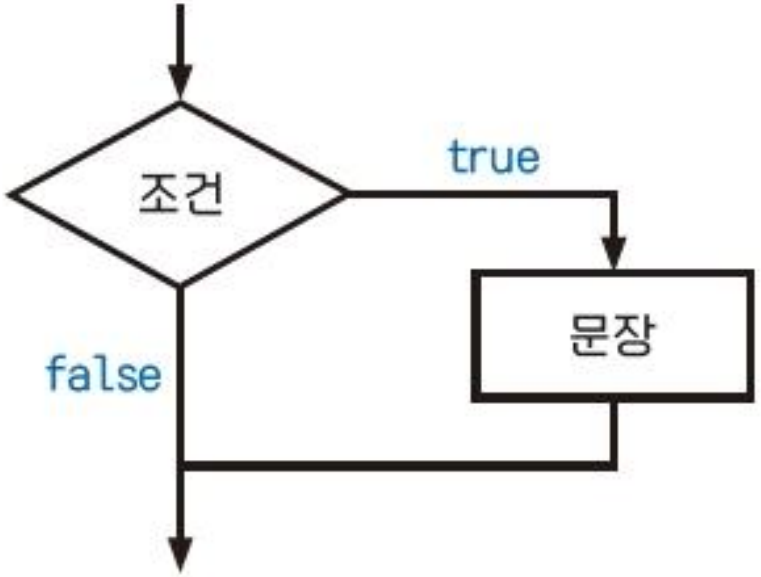
```
let a = 10
const b = 10
```

```
<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>
```

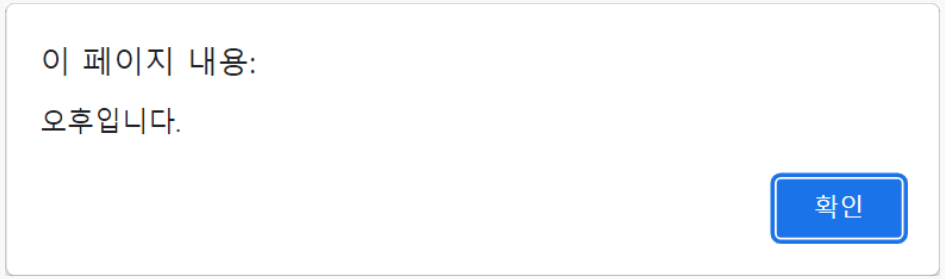
condition_getDate.html 현재 시간 구하기

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

    // 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

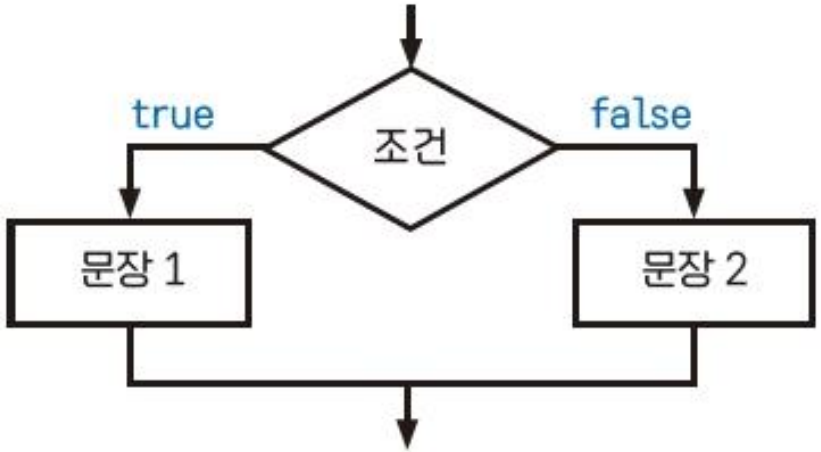
```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    let date = new Date();
    let hours = date.getHours();

    // 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>
```

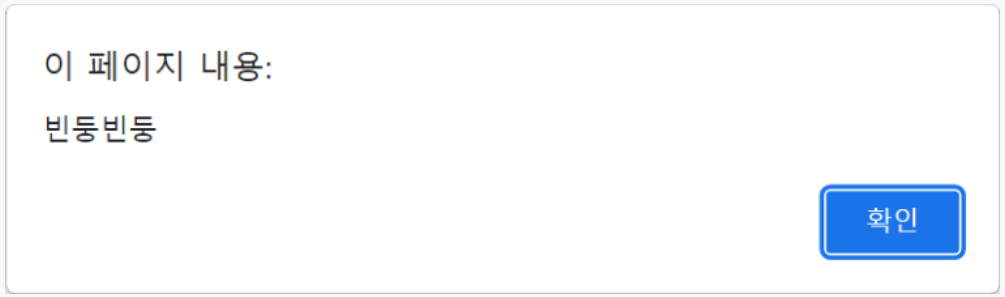
조건문

- 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

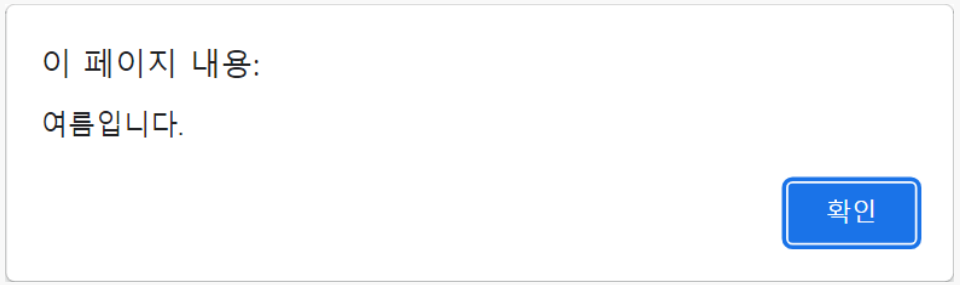
```
if (조건) {
    문장
} else if (조건) {
    문장
} else if (조건) {
    문장
} else {
    문장
}
```

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

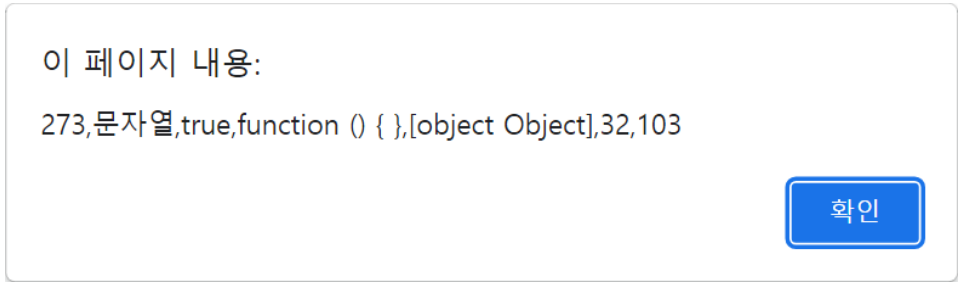
배열

- 변수 여러 개를 한꺼번에 다룰 수 있는 자료형.

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

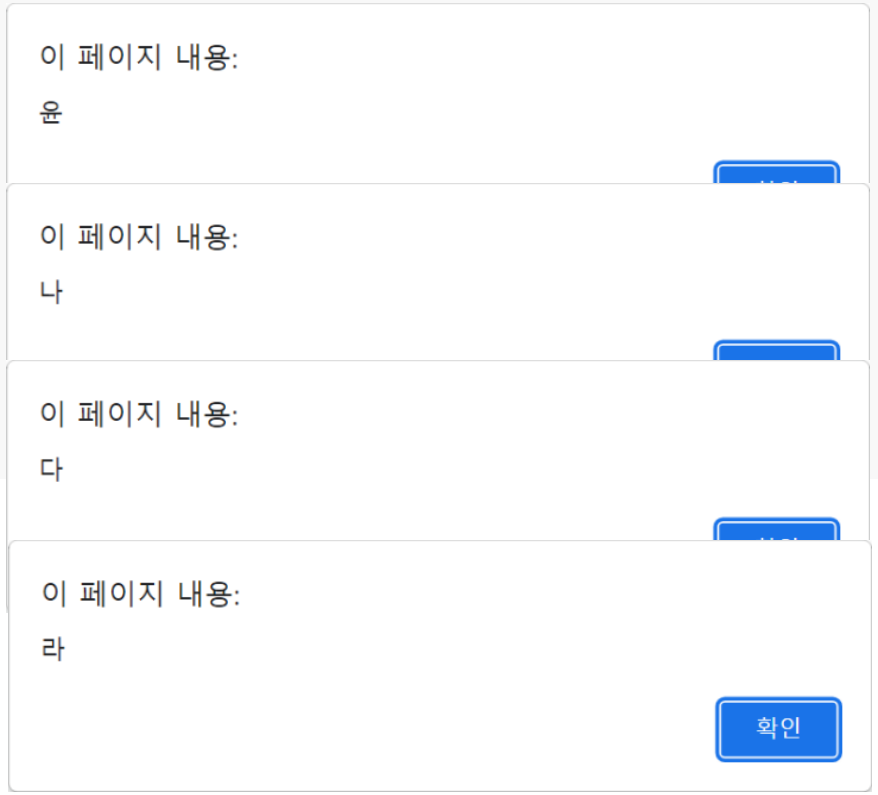
array_basic.html

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



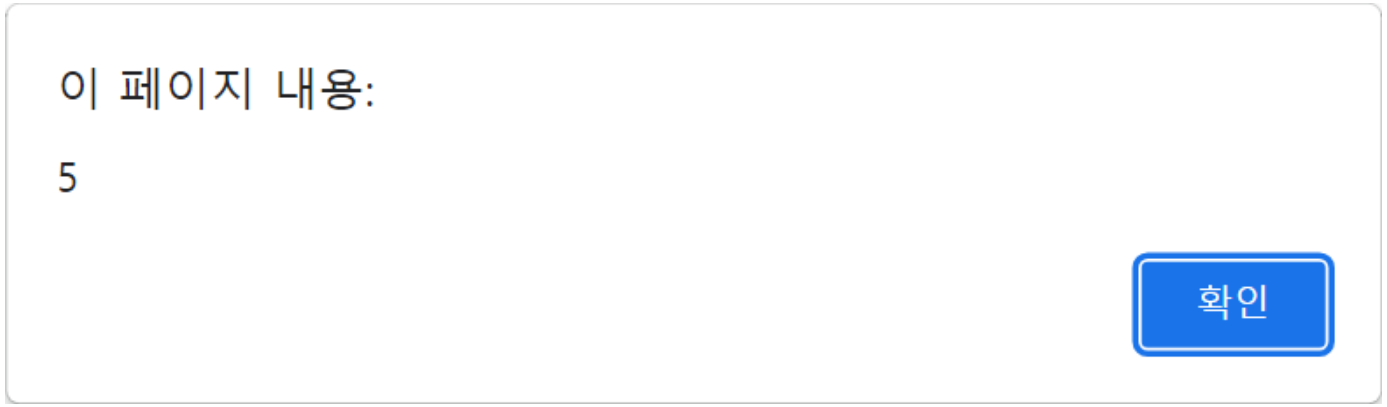
array_index.html 배열 생성과 배열 요소 접근

```
<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>
```



note_arrayLength.html 배열 요소의 개수

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

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

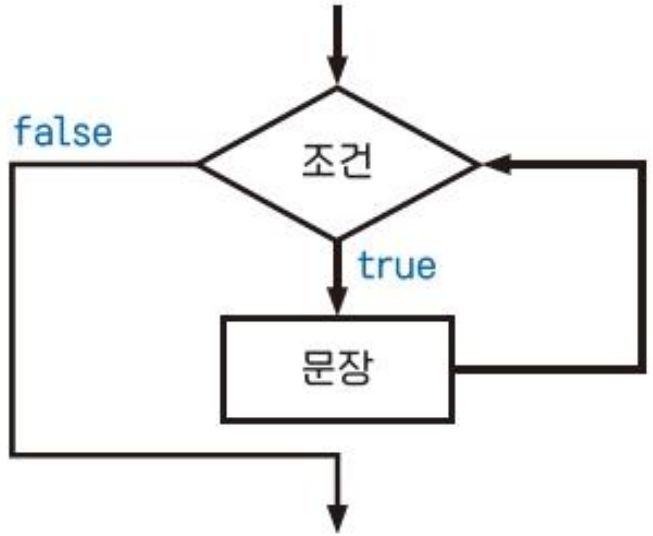


반복문

while 반복문

- 불 표현식이 참이면 중괄호 안 문장을 계속 실행.

```
while (조건) {
    문장
}
```

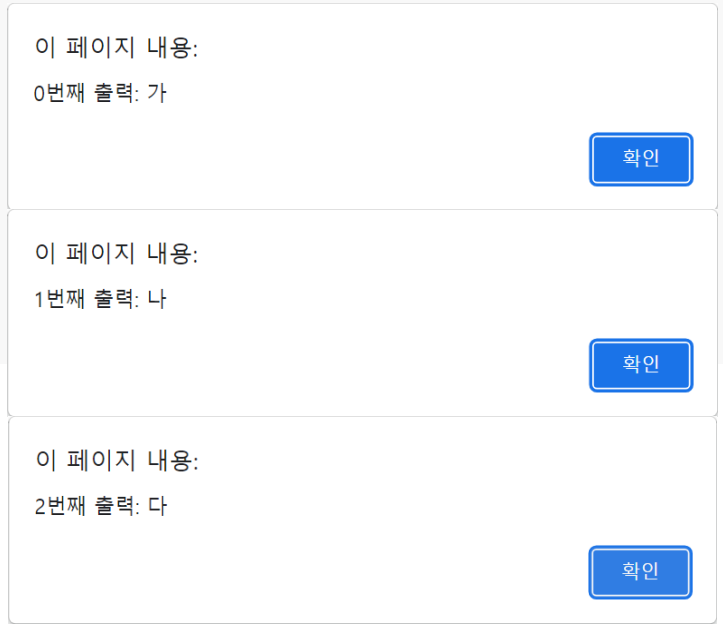


- 조건을 거짓으로 만드는 문장이 없으면 무한 반복.

```
<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>
```

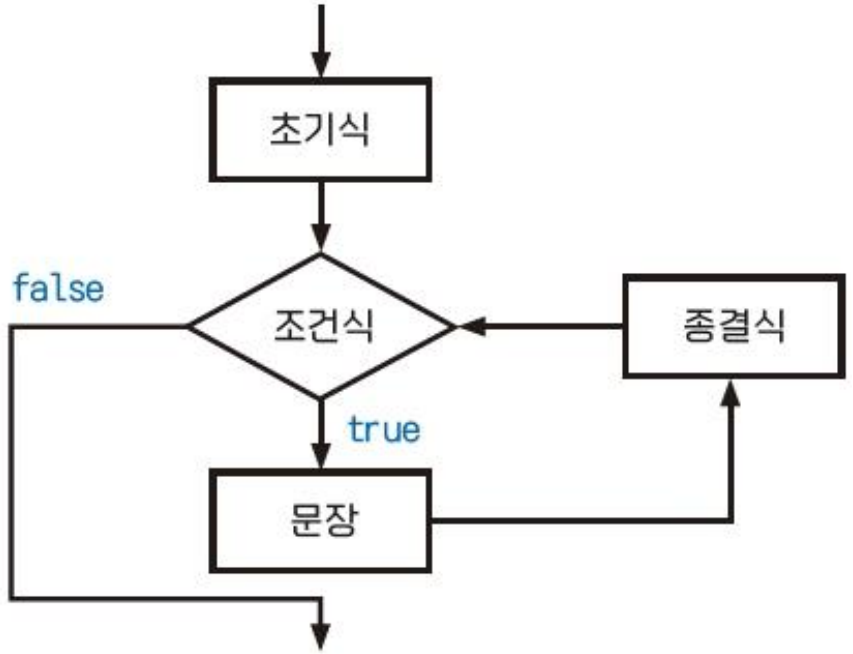


반복문

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 ❺ 앞의 ❷ 단계로 이동합니다.

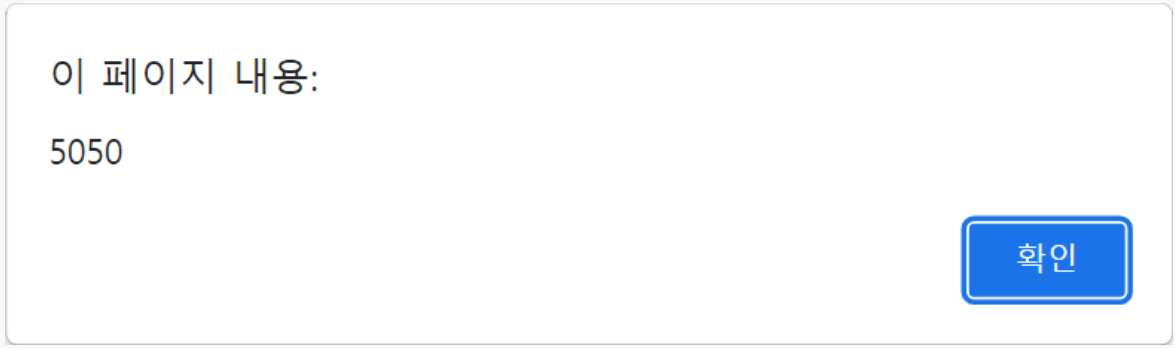
```
for (let i = 0; i < 반복 횟수; i++) {

}
```

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>
```

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>
```



선언과 호출, 실행 우선순위

선언과 호출

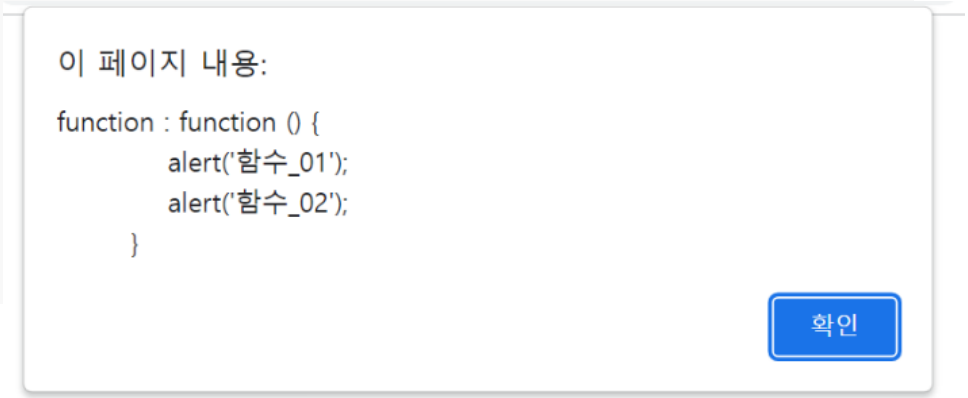
- 함수 생성 방법

방법	표현
익명 함수	`function () {  }`

선언적 함수	`function 함수() {`
	
	`}`

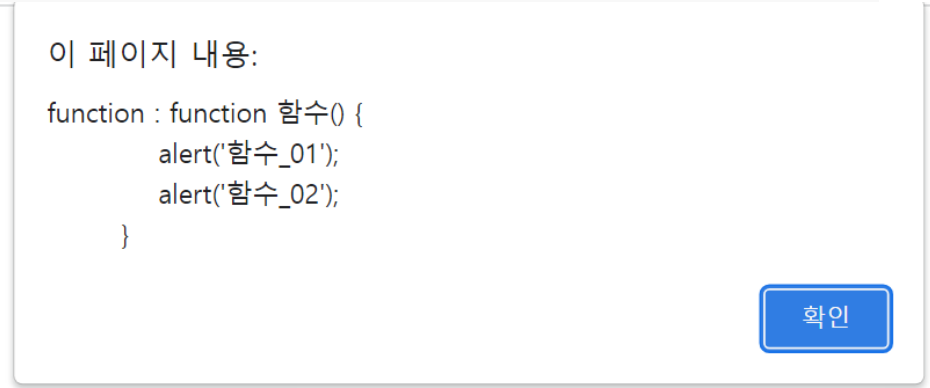
function_noname.html 익명 함수 선언하기

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

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



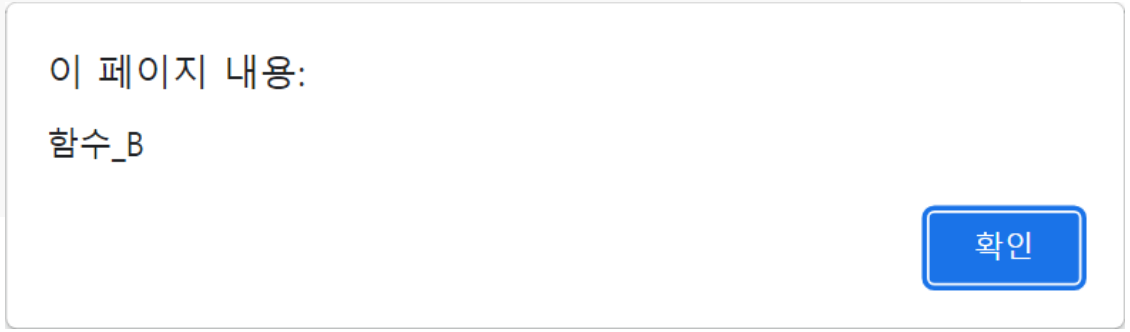
function_name.html 선언적 함수 선언하기

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

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



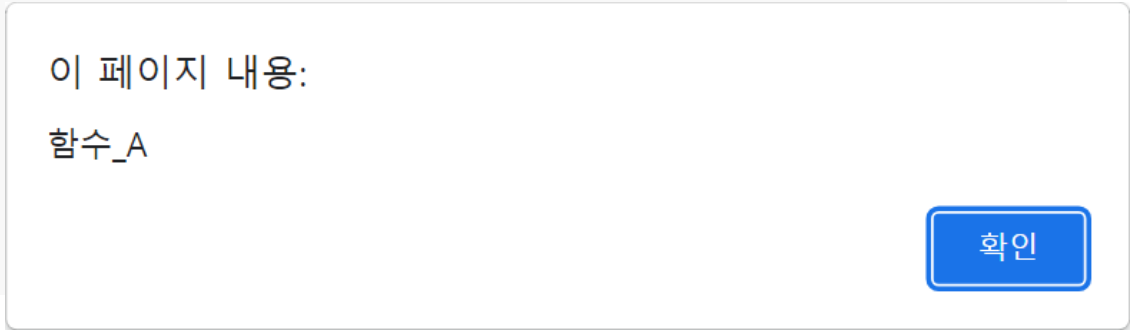
function_priorityBetweenNoname.html 실행 우선 순위

```
<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>
```



매개변수와 반환 값

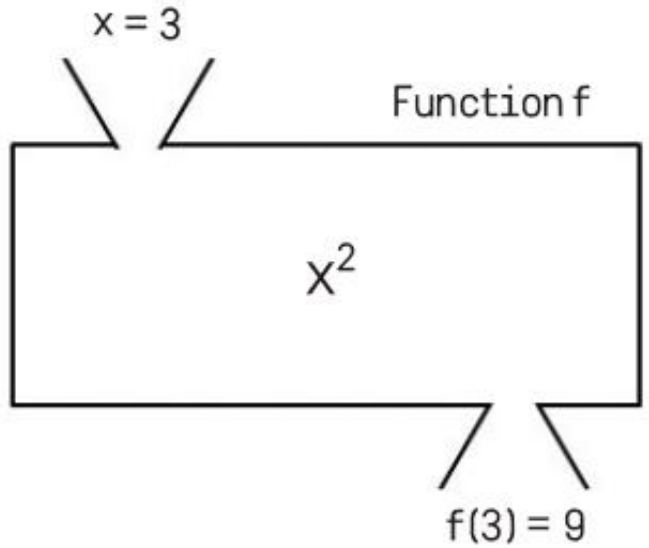
- 매개변수: 함수의 괄호 안에 집어넣어 함수 쪽에 추가적인 정보를 전달.

```
// 함수를 호출합니다.
alert('매개변수');
```

- 리턴 값: 함수를 실행한 결과

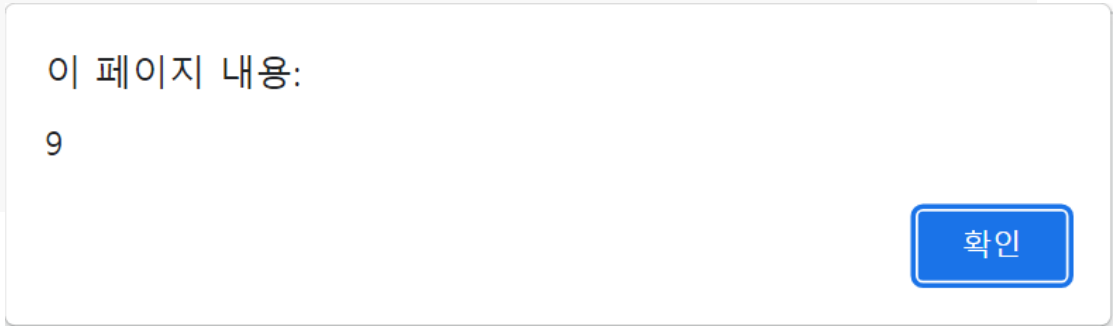
```
let minutes = date.getMinutes();
let seconds = date.getSeconds();
```

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



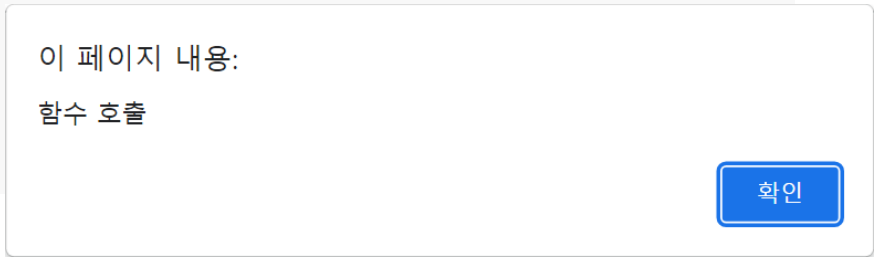
function_return.html 매개변수와 반환 값이 있는 함수

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

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



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>
```



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>
```

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

인덱스	요소
0	사과
1	바나나
2	망고
3	딸기

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>
```

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.

```
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'
```

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');
```

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>
```