Chapter 9 Basic JavaScript Grammar

c언어(컴퓨터가 돌림)와는 다르게 웹 브라우저가 프로그램을 돌림

- **JavaScript basic terminology**
  - Expression - Simple code to generate a value
  - Sentence - the smallest unit of code that can be executed in a programming language
  - Closing - A semicolon at the end of the sentence (;) or a line break

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

(a) 표현식 예

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

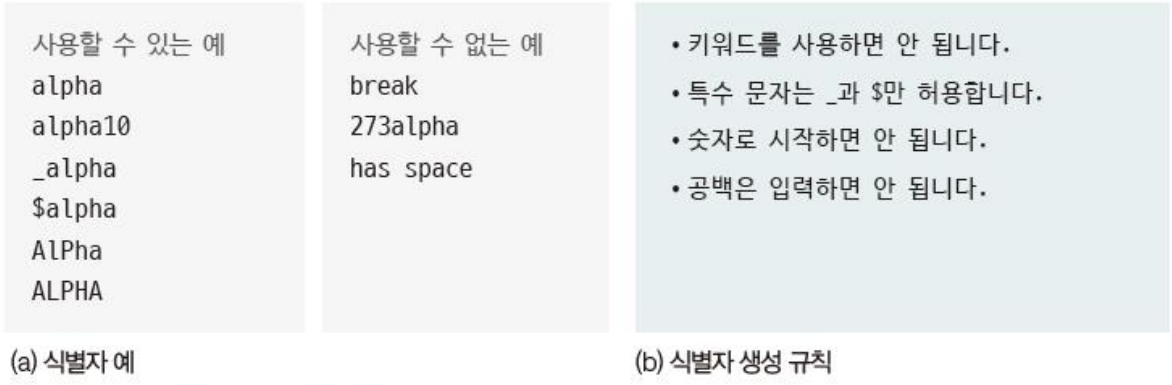
(b) 문장 예

그림 9-1 표현식과 문장 예

- **JavaScript basic terminology**
  - Keywords - words that have been given a special meaning that were set when JavaScript was first created
  - Identifier - A word used in JavaScript to name a variable, function, etc.

표 9-1 자바스크립트 키워드

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



(a) 식별자 예 (b) 식별자 생성 규칙

그림 9-2 식별자 예와 식별자 생성 규칙

- **JavaScript basic terminology**
  - Identifier generation rules

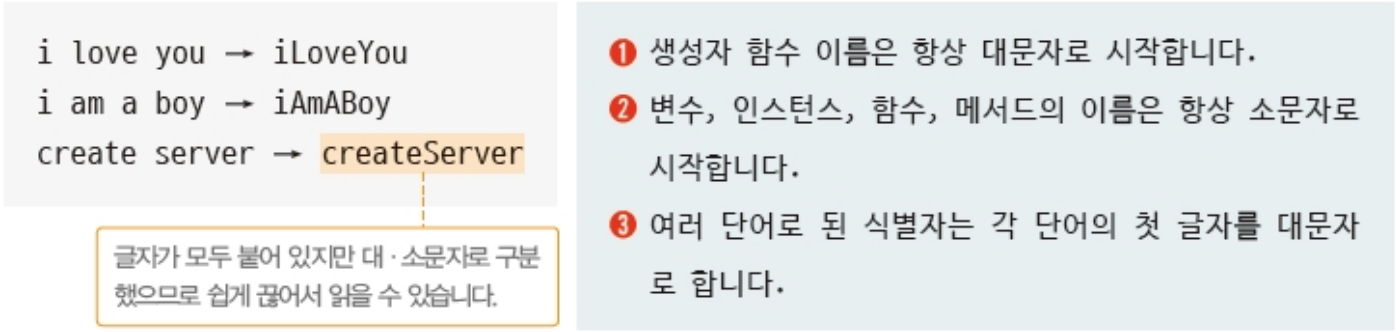


그림 9-3 식별자 생성 관례

표 9-2 자바스크립트의 식별자 종류

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

- **JavaScript basic terminology**
  - Comments—Code that has no effect on program progress, used to describe the program

표 9-3 주석 처리 방법

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

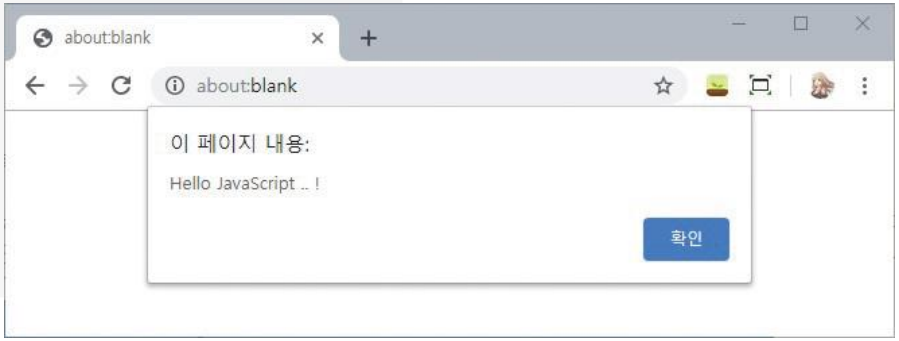
```
<script>
    // 주석은 코드 실행에 영향을 주지 않습니다.
    /*
    alert('Hello JavaScript .. !');
    alert('Hello JavaScript .. !');
    alert('Hello JavaScript .. !');
    */
</script>
```

- **JavaScript output**
  - The most basic output method - Use the alert( ) function to bring up a warning window in a web browser

그림 9-4 alert() 함수 형태

코드 9-1 output_alert.html

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

</body>
</html>
```



- **Data types**
  - number
  - The most basic data types
  - No distinction between integer and real number

실수 정수 구별이 없다

표 9-4 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

자바스크립트는 부동소수점(소수점을 포함한 숫자)을 계산할 때 약간의 오차를 발생시킵니다.

표 9-5 나머지 연산자

연산자	설명	예
%	나머지	> 10 % 5 0 > 7 % 3 1

연산자	설명	예
*	곱셈	> 52 * 273 14196 > 52.273 * 103.57 5413.91461
/	나눗셈	> 52 / 273 0.19047619047619047 > 52.273 / 103.57 0.504711789128126 > 1 / 0 Infinity

자바스크립트는 어떤 숫자를 0으로 나누었을 때, 무한을 나타내는 값Infinity이 됩니다.

- **Data types**
  - String
    - Character set
    - 'abcdefg', 'Hello World', 'Hello.'

표 9-6 문자열 생성

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

표 9-7 이스케이프 문자

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

표 9-8 문자열 연결 연산자

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

- **Data types**
  - Bool
    - Resources used to express true and false

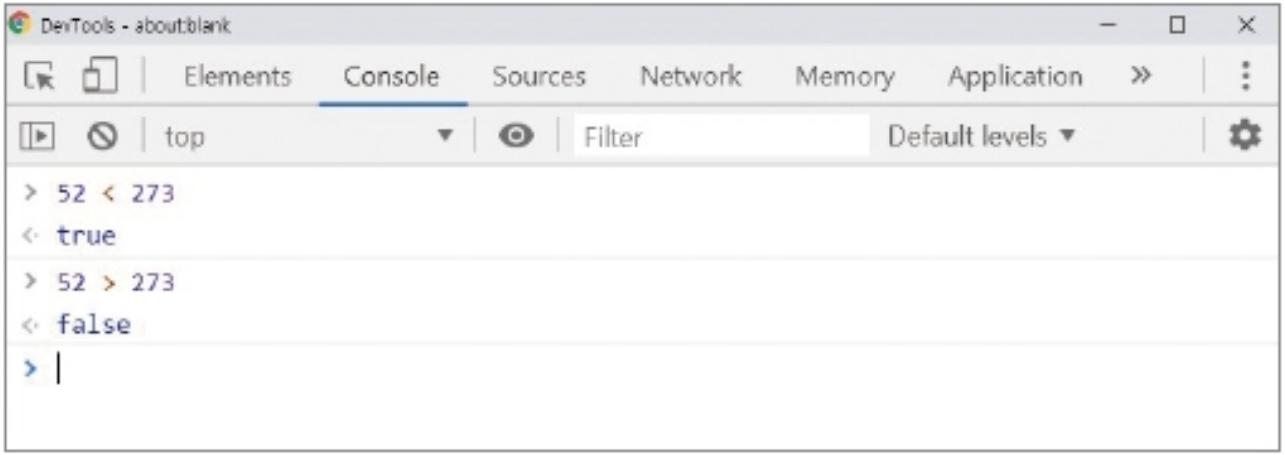


그림 9-5 불 사용 예

- **Data types**
  - Comparison Operators
    - Operators that can compare two targets

표 9-9 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

문자열 순서도 비교 가능합니다.

- **Data types**
  - Logical operators

표 9-10 논리 연산자

연산자	설명	예
!	논리 부정(참이면 거짓, 거짓이면 참)	> !true false > !(10 == 10) false
&&	논리곱(둘 다 참이어야 참)	> true && true true — 둘 다 참일 때만 참입니다. > true && false false > false && true false > false && false false > (10 < 20) && (20 < 30) true — 20이 10~30 사이에 있다는 것을 나타낼 때는 이러한 형태로 표현해야 합니다.
||	논리합(둘 중 하나만 참이어도 참)	> true || true true > true || false true > false || true true — 둘 중 하나만 참이어도 참입니다. > false || false false

- **Variable**
  - The identifier used to store the value.
  - ❶ Declare a variable.
  - ❷ Initialize the variable.

코드 9-3 변수 선언과 초기화

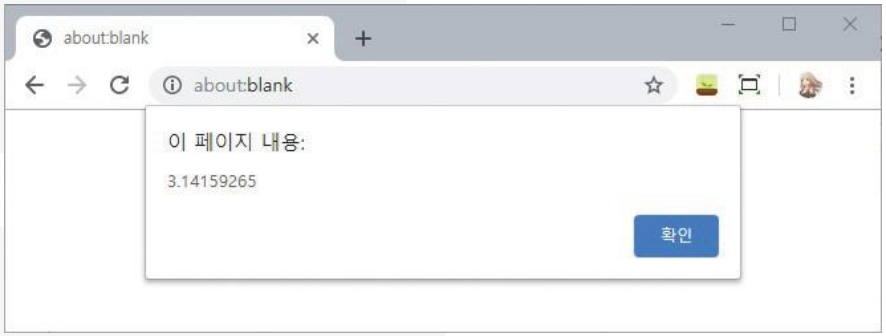
```
> var pi = 3.14159265;
undefined
```

코드 9-2 변수 선언과 값 할당

```
❶ > var pi;                 // 변수 선언
   undefined
❷ > pi = 3.14159265;        // 값 할당
   undefined
```

코드 9-4 변수에 저장된 값 출력

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



- **Message output using JavaScript (1)**
  - 1.HTML Create a page

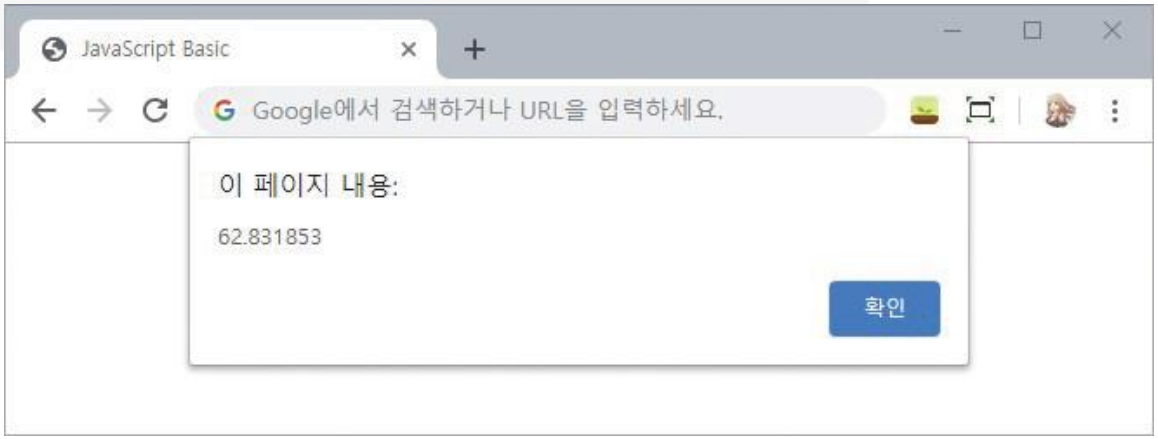
코드 9-5 variable_use.html

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

    </script>
</head>
<body>

</body>
</html>
```

- **Message output using JavaScript (2)**
  - 2. Using variables

코드 9-6 variable_use.html

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

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



- **conditional statement**
  - If conditional statement
    - Run the sentence if the condition is true, and ignore the sentence if it is false
    - If the execution sentence is one line, braces can be omitted
    - If the execution sentence is multiple lines, braces are required

```
if (조건) {
    문장
}
```

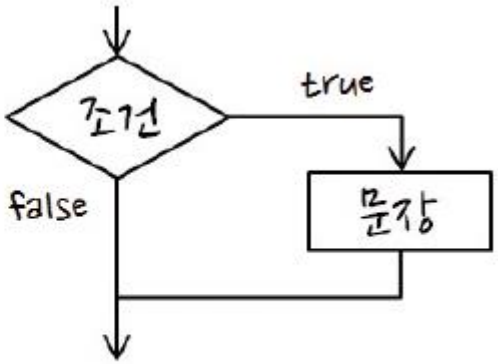
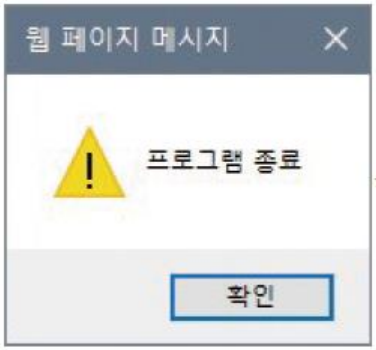


그림 9-6 if 조건문 기본 형태와 순서도

- **If conditional statements determine true and false**

코드 9-7 condition_basic.html

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



크롬은 경고 창이 너무 넓게 출력되므로 인터넷 익스플로러의 경고 창을 스크린샷으로 넣었습니다. 크롬에서 이러한 모양의 경고 창이 출력되지 않는다고 당황하지 마세요.

- **If conditional statements to determine morning and afternoon**
  - **1. Find the current time**
  -

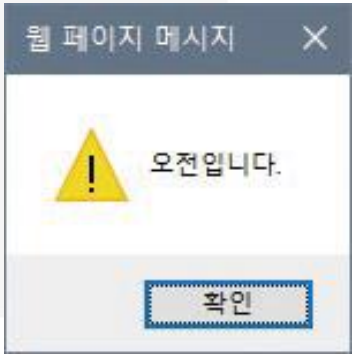
코드 9-8 condition_getDate.html

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    var date = new Date();

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

- **If conditional statements to determine morning and afternoon**
  - 2. Distinguish between morning and afternoon

코드 9-9 condition_date.html

```
<script>
    // 변수를 선언합니다.
    var date = new Date();
    var 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>
```



- **conditional statement**
  - If else conditional statement
    - Convenient to use in situations where there is a clear division of the two
    - If the execution sentence is one line, braces can be omitted
    - If the execution sentence is multiple lines, braces are required

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

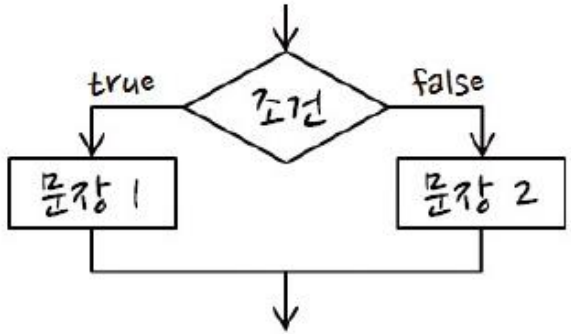


그림 9-7 if else 조건문 기본 형태와 순서도

- **If else conditional statement to determine morning and afternoon**

코드 9-10 condition_else.html

```
<script>
    // 변수를 선언합니다.
    var date = new Date();
    var hours = date.getHours();

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

- **conditional statement**
  - Nested conditional statements and if else if conditional statements

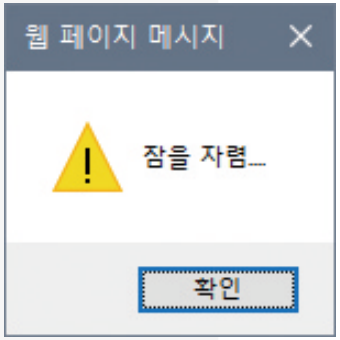
```
if (조건) {
    if (조건) {
        문장
    } else {
        문장
    }
} else {
    if (조건) {
        문장
    } else {
        문장
    }
}
```

그림 9-8 중첩 조건문 형태

- **Representation of the day schedule with nested conditional statements**

코드 9-12 condition_ifelseif.html

```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    var date = new Date();
    var 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>
```



- **conditional statement**
  - A conditional statement that you create when you omit the braces of nested conditionals
    - if else if conditional statement

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

그림 9-9 if else if 조건문 형태

- **Expressing the schedule of the day with an if else if conditional statement**

코드 9-11 condition_duplication.html

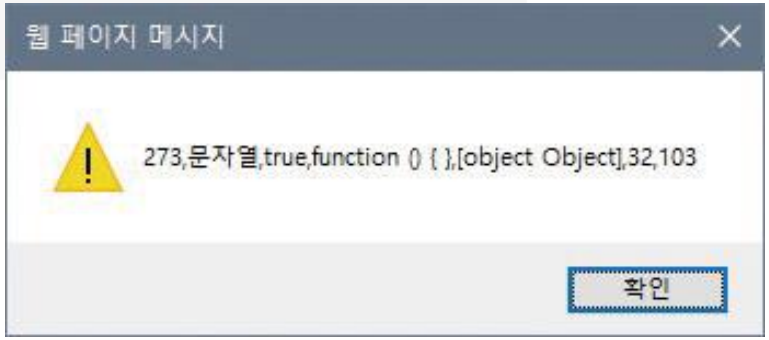
```
<script>
    // Date 객체를 선언합니다: 현재 시간 측정
    var date = new Date();
    var hours = date.getHours();

    // 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>
```

Iteration

- Array
  - A data type that can handle multiple variables at once
  - Elements - each piece of data entered inside an array
  - Various data types can be entered inside the array
  - If you output the entire array, the elements appear in order.

코드 9-14 array_basic.html

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

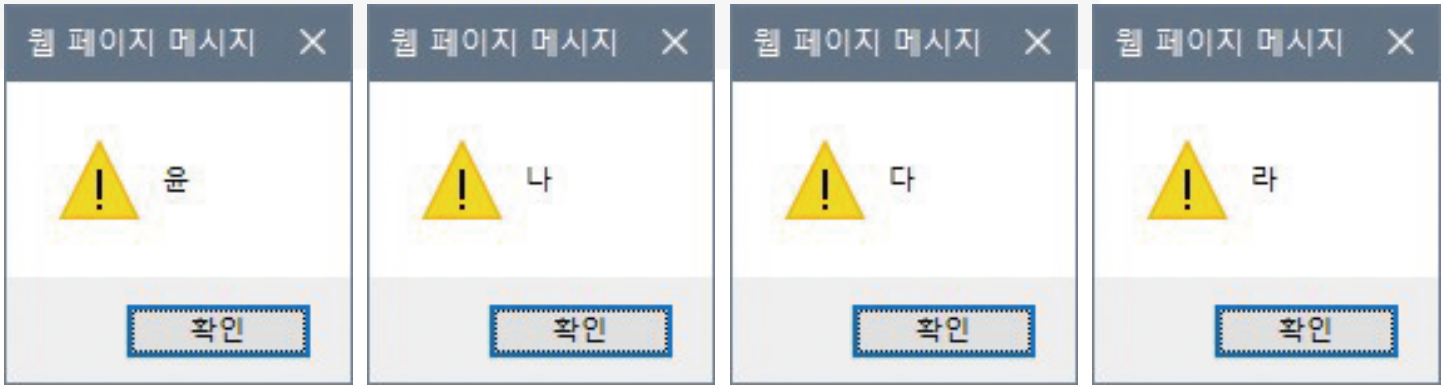


Array creation and array element access

코드 데모

코드 9-15 loop_array.html

```
<script>
    // 변수를 선언합니다.
    var 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>
```



Iteration

- while iteration
  - The most basic repetition
  - It is similar in format to an if conditional statement, but unlike an if conditional, if the bool expression is true, the non-brace statement continues to execute.

```
while (조건) {
    문장
}
```

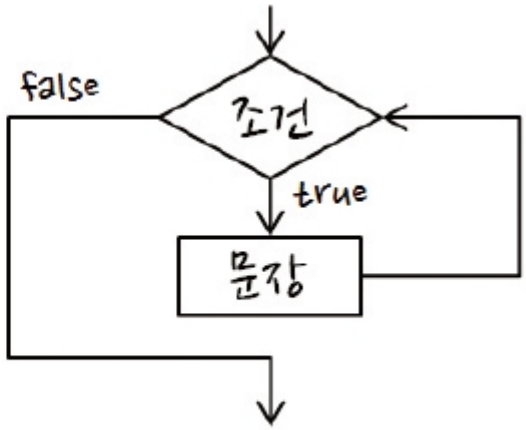
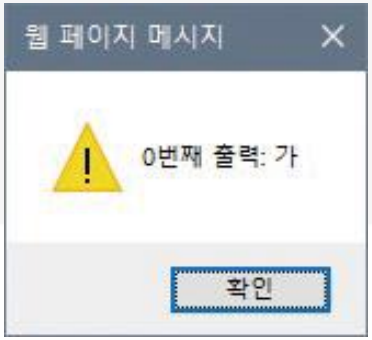


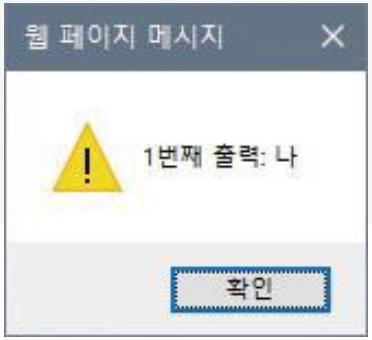
그림 9-10 while 반복문 기본 형태와 순서도

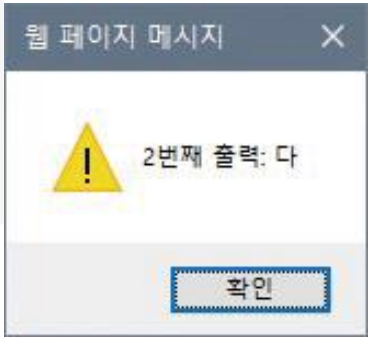
- **while iteration**

코드 9-17 loop_while.html

```
<script>
    // 변수를 선언합니다.
    var i = 0;
    var 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>
```







Iteration

for iteration

- Use the for iteration when you want to repeat it as many times as you want
- ❶ Compare the initial expressions.
- ❷ Compare conditional expressions. If the condition is false, the iteration terminates.
- ❸ Run sentences.
- ❹ Run a closing expression.
- ❺ Go to the previous ❷ step.

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

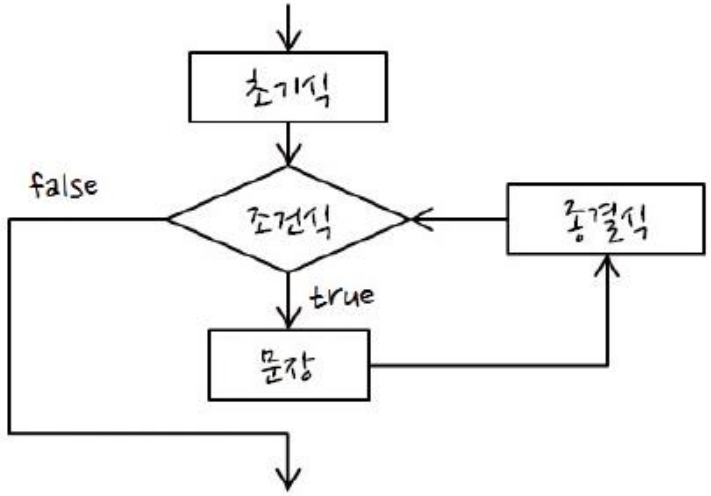
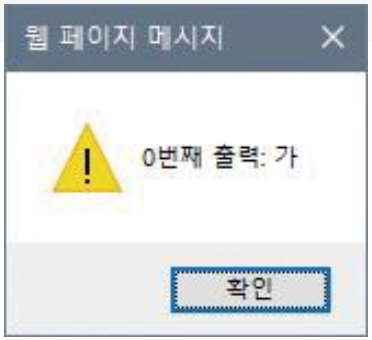


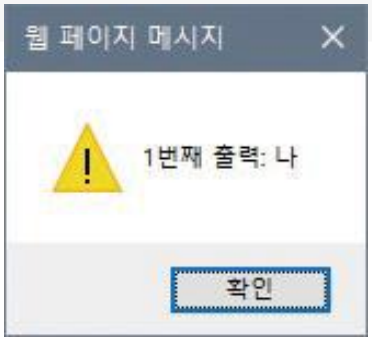
그림 9-11 for 반복문 기본 형태와 순서도

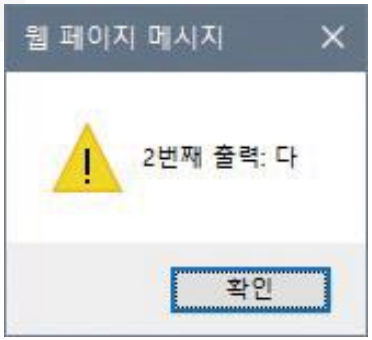
- **for iteration**

코드 9-18 loop_for.html

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

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







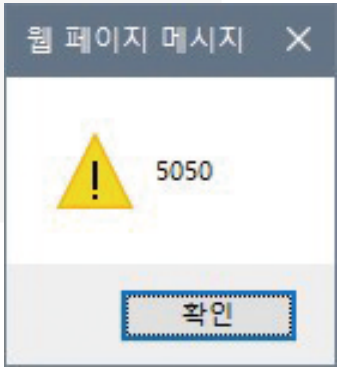
- **Calculate the sum from 0 to 100 using the for iteration**

코드 9-20 loop_forPlus.html

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

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

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

output += 3이랑
output = output +3과 같다

100까지 더해야 하므로 <= 연산자를 사용합니다.



- **Declaration, invocation, and execution priorities**
  - Declarations and Calls
    - Function—A data type that represents a set of codes.
    - Create an anonymous function - Create without entering a function name
    - Create a declarative function - Create by typing a function name

표 9-11 함수 생성 방법

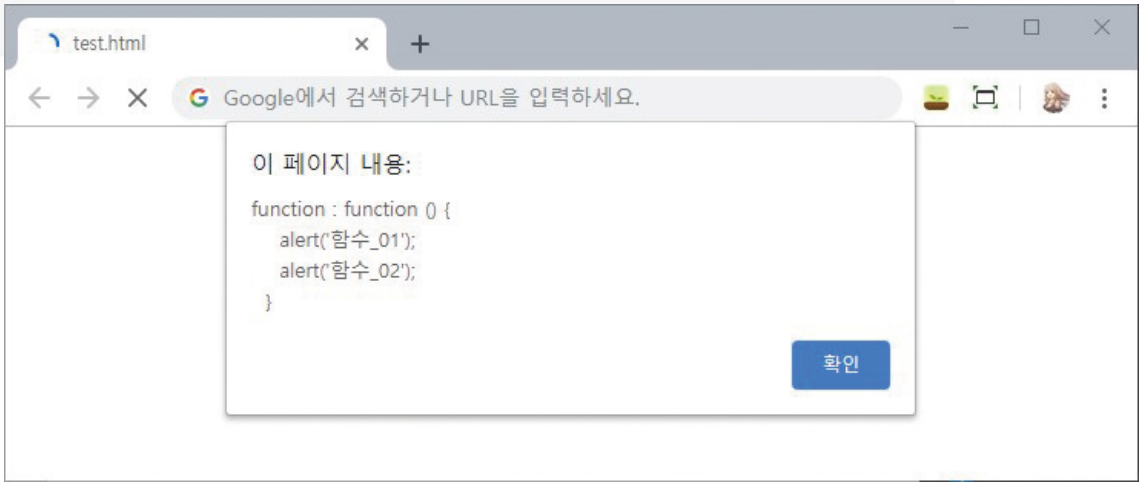
방법	표현
익명 함수	`function () {  }`

선언적 함수	`function 함수() {`
	
	`}`

- **Function declarations**
- **1. Declaring an anonymous function**

코드 9-21 function_noname.html

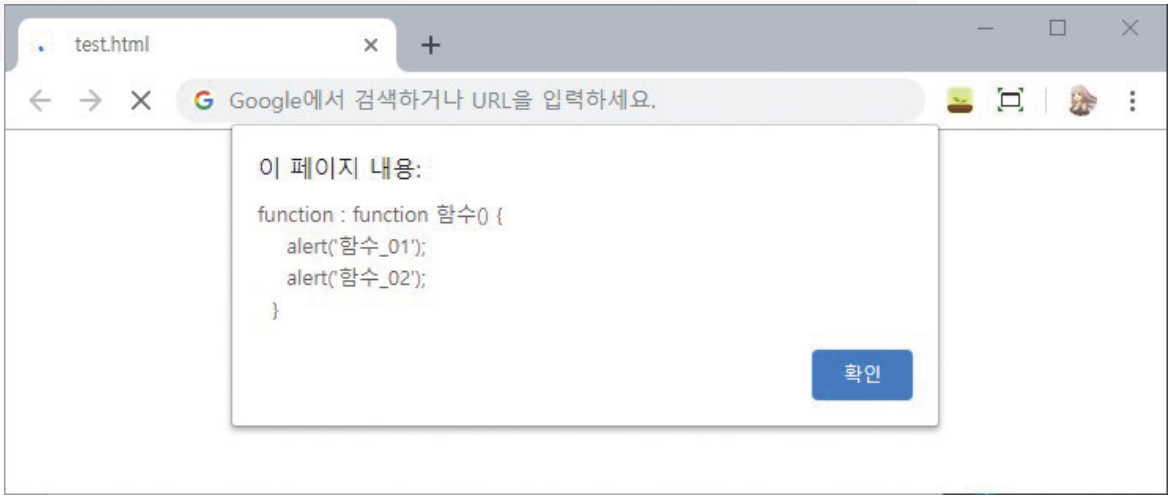
```
<script>
    // 함수를 선언합니다.
    var 함수 = function () {
        alert('함수_01');
        alert('함수_02');
    };
    // 출력합니다.
    alert(typeof (함수) + ' : ' + 함수);
</script>
```



- **Function declarations**
- **2. Declaring a Declarative Function**

코드 9-22 function_name.html

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

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



시험!

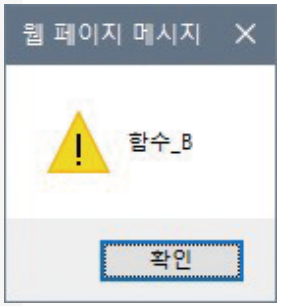
Declaration, invocation, and execution priorities

Execution Priority

- The last input value is stored

코드 9-23 function_priorityBetweenNoname.html

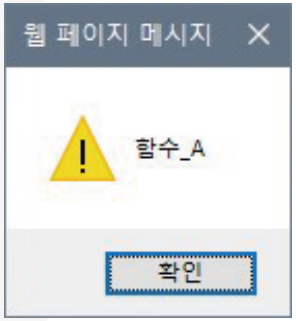
```
<script>
    // 함수를 선언합니다.
    var 함수 = function () { alert('함수_A'); };
    var 함수 = function () { alert('함수_B'); };

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



- ※Caution ※ When using declarative functions and anonymous functions together
  - Read the declarative function first before reading all the code

코드 9-24 function_priority.html

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

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



매개변수

- **Parameters and return values**
- **Parameter**
  - Putting it in parentheses of the function to pass additional information to the function side.
  - Return Value
  - The result of executing the function returns the value

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

그림 9-12 매개변수와 반환 값을 가지는 함수 형식

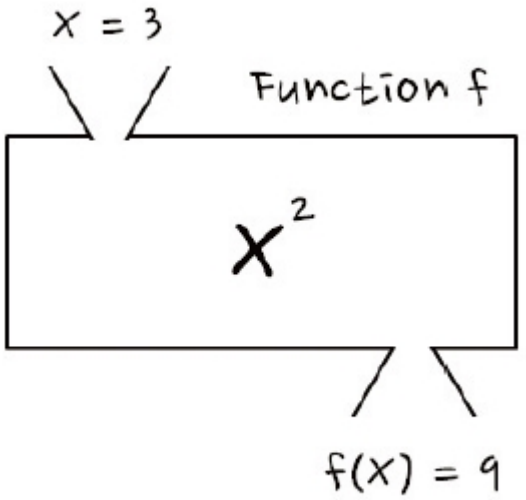
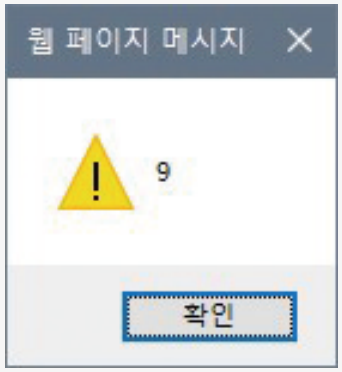


그림 9-13 함수

- **Functions with parameters and return values**
-

코드 9-25 function_return.html

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

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



- **Callback function**
  - Functions passed as parameters
  -

코드 9-26 function_callback.html

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

    // 변수를 선언합니다.
    var callback = function () {
        alert('함수 호출');
    };
    callTenTimes(callback); // 함수를 호출합니다.
</script>
```

열 번 출력됩니다.

- **Callback function**
  - Running as an anonymous function

코드 9-27 reference_nonameCallback.html

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

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

Object Overview

- An object stores multiple data types at once
- Arrays use indexes when accessing elements, but objects use keys

코드 9-28 object_create.html

```
<script>
  // 객체를 선언합니다.
  var 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

그림 9-15 객체 생성 예

- **Object Overview**
  - Using the square brackets after the object to access the object's properties by typing a key

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

그림 9-16 [코드 9-28]에서 생성한 객체 속성에 접근하는 예 1

  - The dot after the object (.) to access object properties

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

그림 9-17 [코드 9-28]에서 생성한 객체 속성에 접근하는 예 2

Object Overview

- for in iteration
  - You can look at object elements one by one

```
for (var 키 in 객체) {
    문장
}
```

그림 9-18 for in 반복문 형태

코드 9-30 object_withForIn.html

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

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

- **Properties and methods**
  - element
    - Each value in an array
  - Property
    - Each value in an object

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

그림 9-19 다양한 자료형의 객체 속성 예

- **Properties and methods**
- **this keyword**
  - When you want to use properties in an object in a method, you must clearly indicate that they are the properties that you have.

코드 9-31 object_this.html

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

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