

8장

자바스크립트 시작하기

8.1 자바스크립트 코드 작성 방법

8.2 프로그래밍 시작 전 알아 두기

8.1 자바스크립트 코드 작성 방법

1. HTML 파일과 자바스크립트 연결하기

- 내부 스크립트 방법(internal script) : HTML 문서 안에서 script 태그의 콘텐츠 영역에 자바스크립트 코드를 작성
- 외부 스크립트 방법(external script) : 별도의 js 확장자 파일을 만들어 자바스크립트 코드를 작성하고 이 파일을 HTML 문서에서 script 태그로 연결
- script 태그의 사용 위치 : script 태그는 웹 브라우저에 화면이 표시되는 것에 영향을 미치지 않도록 body 태그가 끝나기 전에 사용

실무에서는 유지보수의 용의성 때문에 외부스크립트 방법을 사용

8.1 자바스크립트 코드 작성 방법

1. HTML 파일과 자바스크립트 연결하기

- 내부 스크립트 예제

08/01/internal_script.html

```
<body>
  <script>
    document.write("내부 스크립트 방법");
  </script>
</body>
```

8.1 자바스크립트 코드 작성 방법

1. HTML 파일과 자바스크립트 연결하기

- 외부 스크립트 예제

```
document.write("외부 스크립트 방법");
```

08/01/script.js

```
<body>  
  <script src="script.js"></script>  
</body>
```

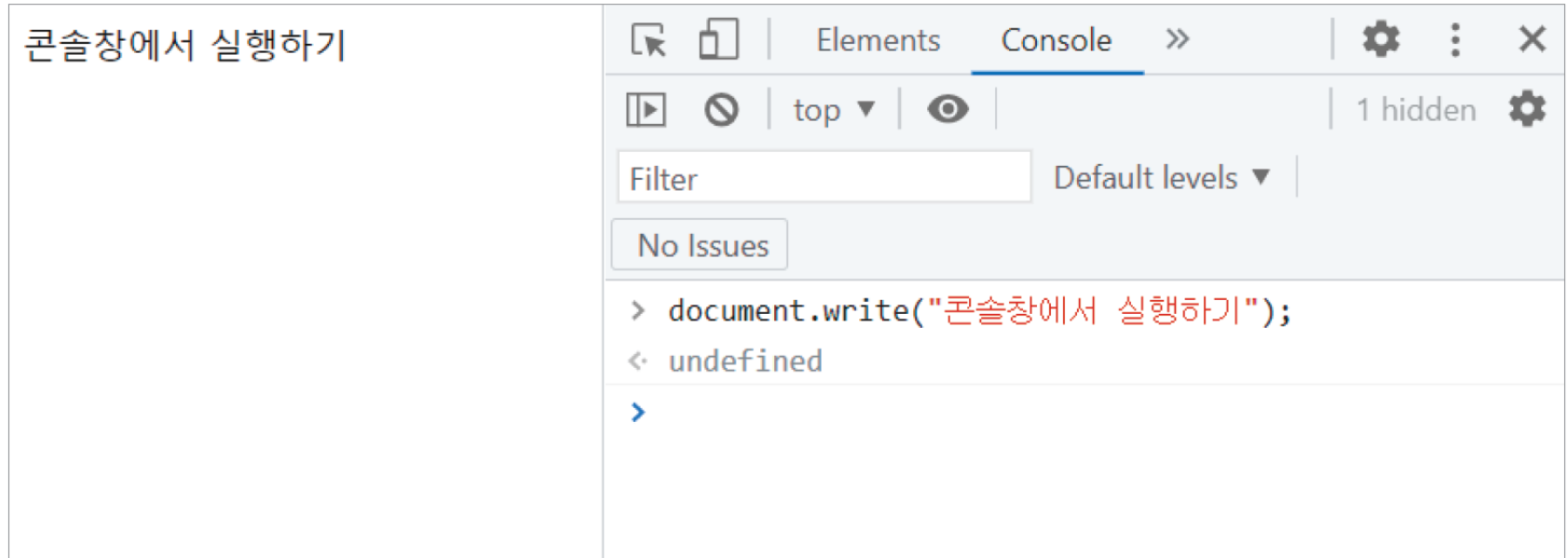
08/01/external_script.html

8.1 자바스크립트 코드 작성 방법

2. 자바스크립트 코드 실행하기

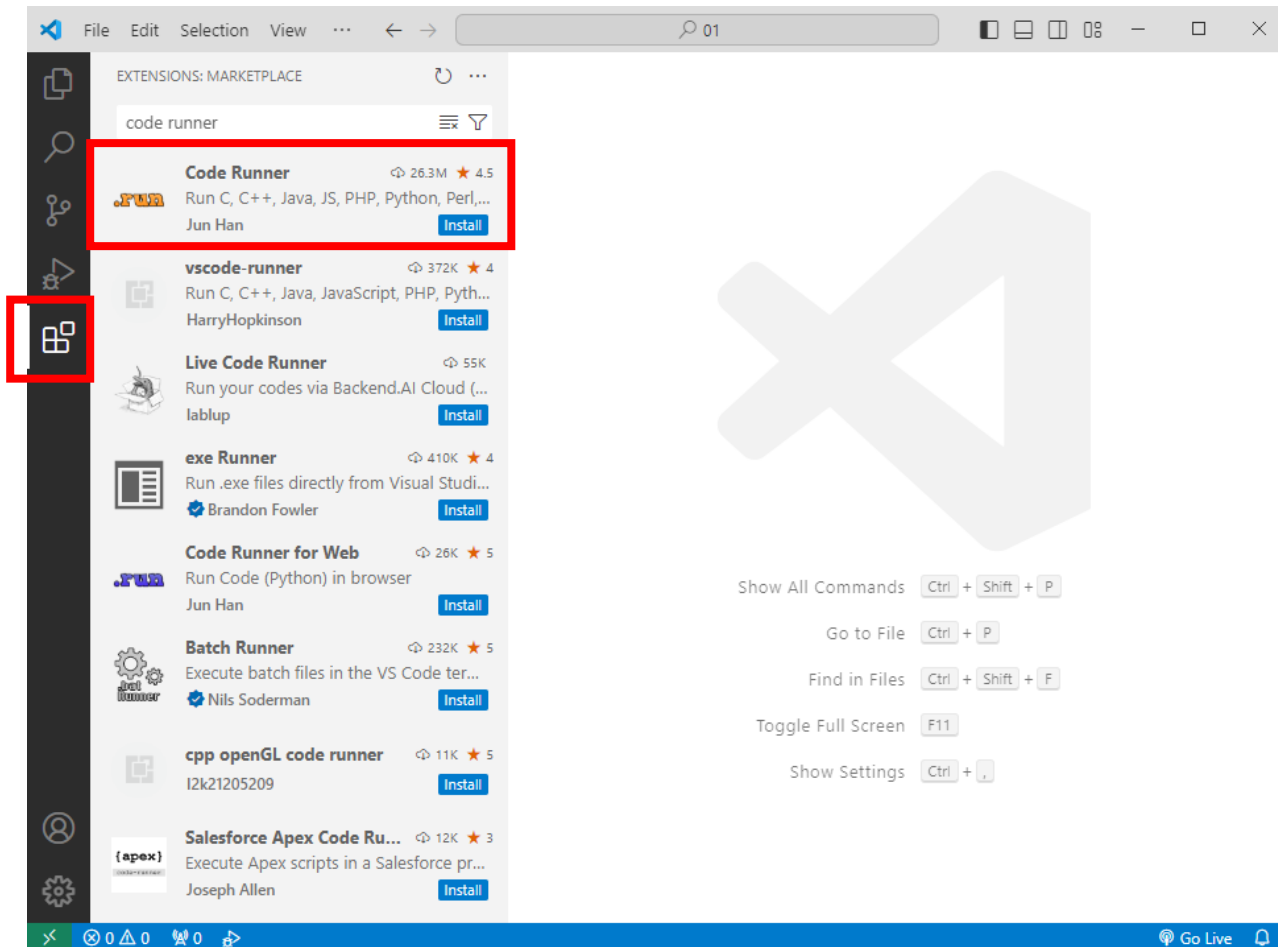
- 웹 브라우저의 개발자 도구(F12)에서 지원하는 콘솔창 활용하기

그림 8-4 콘솔창에서 자바스크립트 코드 실행하기



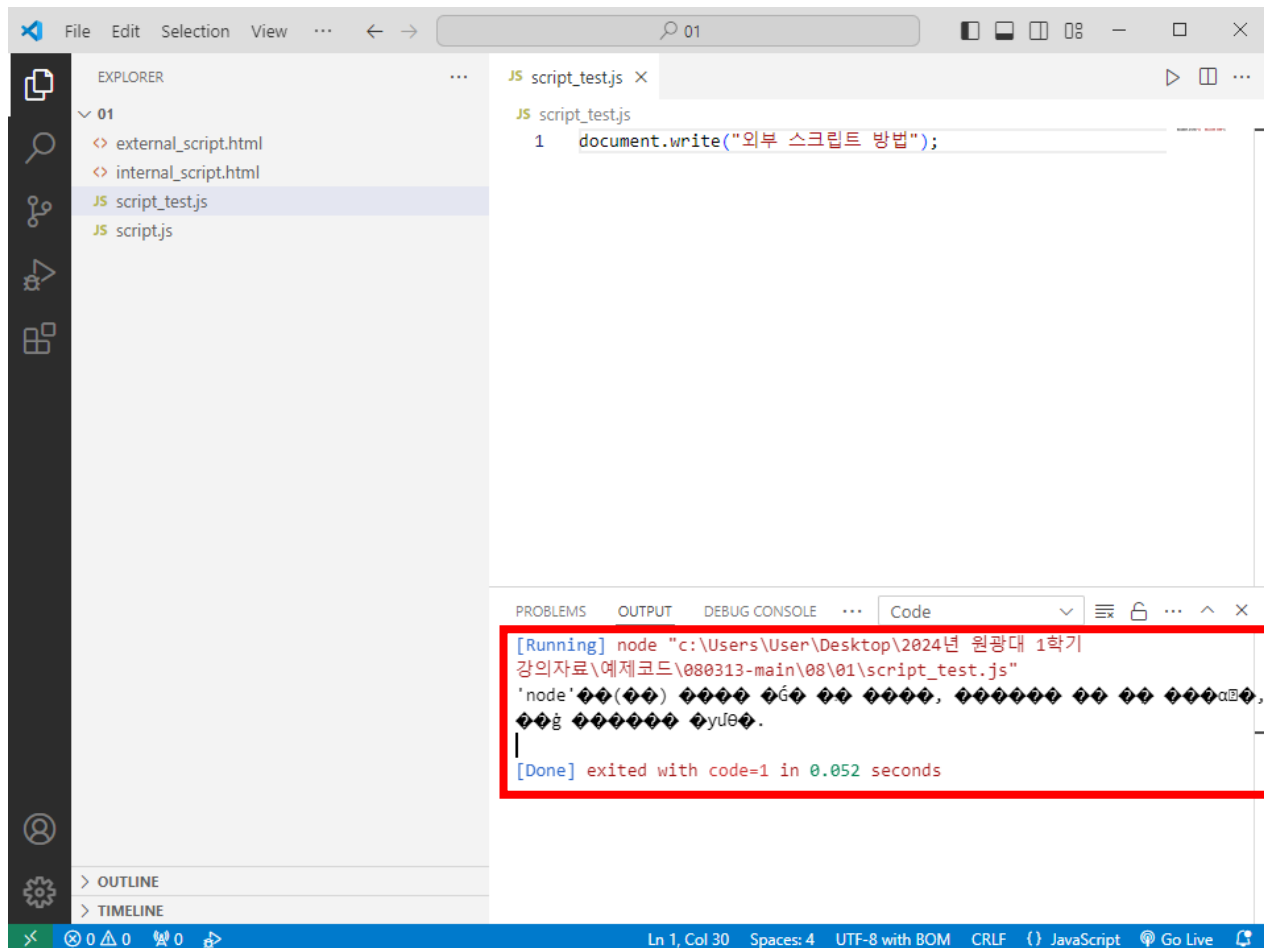
8.1 자바스크립트 코드 작성 방법

- VSCode의 Code Runner 확장 프로그램 설치하기



8.1 자바스크립트 코드 작성 방법

- VSCode의 Code Runner 확장 프로그램 활용하기



8.1 자바스크립트 코드 작성 방법


- 한글이 깨져서 출력되는 이유는 Node.js가 설치 되지 않아서 해당 현상이 나타남
- Node.js 사이트(<https://nodejs.org/en/download>)
- 버전 : v16.20.2

Download Node.js®

Download Node.js the way you want.

Prebuilt Installer Prebuilt Binaries Package Manager Source Code

I want the version of Node.js for running

 **Download Node.js v16.20.2**

Node.js includes npm (8.19.4) ↗.

Read the changelog for this version ↗

Read the blog post for this version ↗

Learn how to verify signed SHASUMS ↗

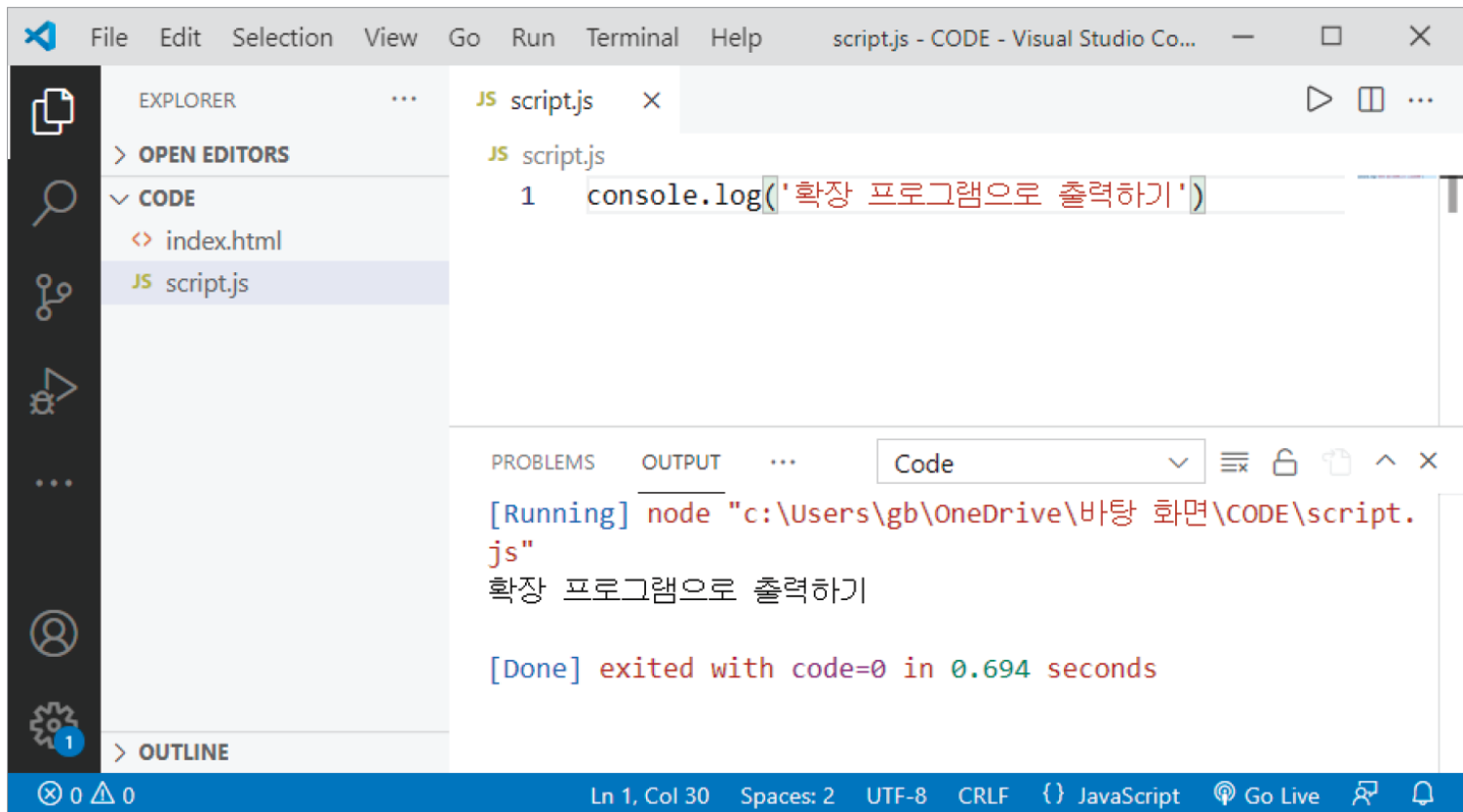
Check out all available Node.js download options ↗

Learn about Node.js Releases ↗

8.1 자바스크립트 코드 작성 방법

- VSCode의 Code Runner 확장 프로그램 활용하기

그림 8-7 Code Runner 실행결과



8.2 프로그래밍 시작 전 알아 두기

1. 주석

- 한 줄 주석 : // 기호(슬래시 2개)로 작성
- 여러 줄 주석 : /* 기호와 */ 기호 사이에 작성

2. 자바스크립트 오류 확인 방법

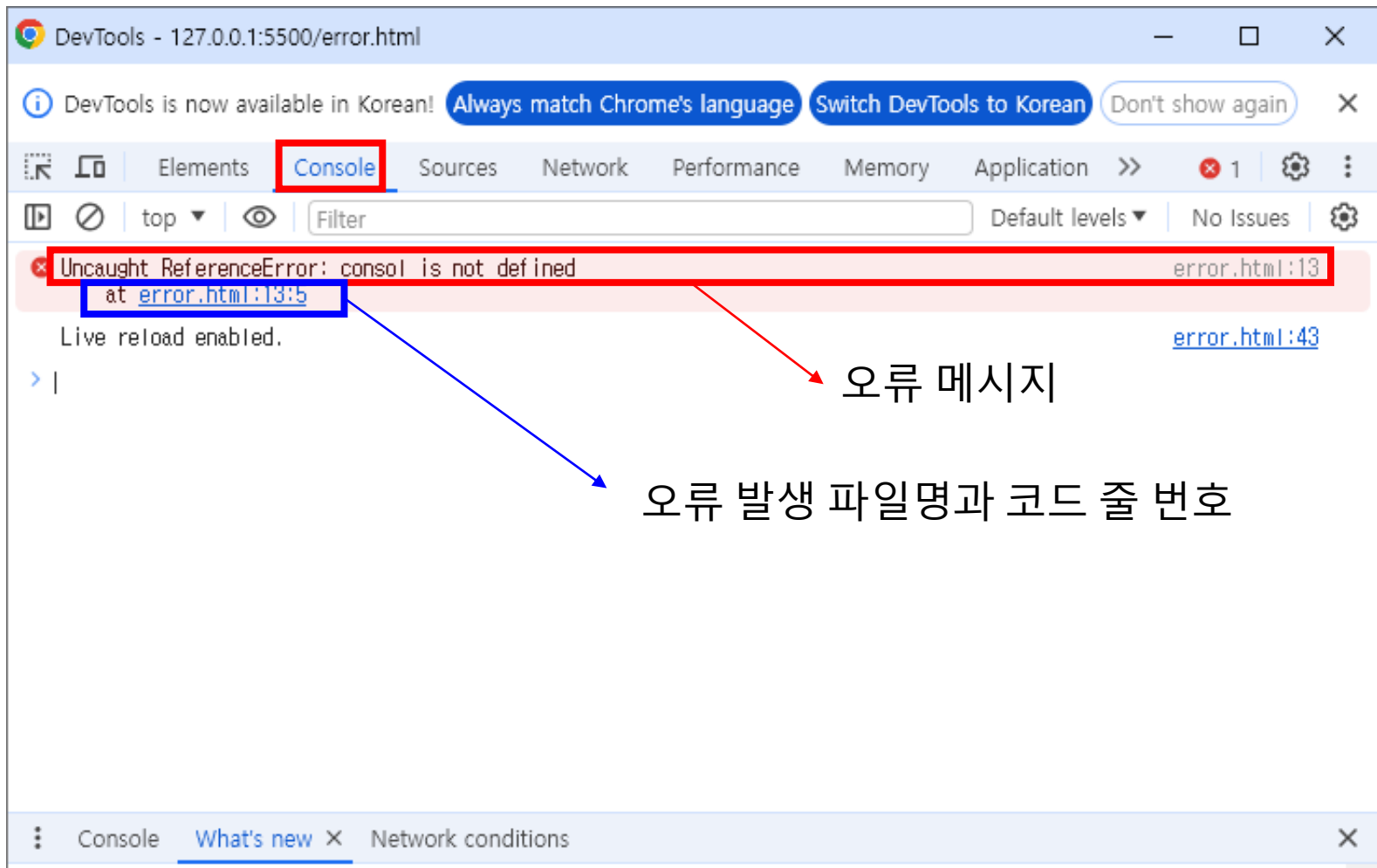
- 프로그래밍 언어의 실행 방법
 - 컴파일 방식 : 모든 코드를 기계어로 변환 후 실행
 - 인터프리터 방식 : 코드를 한 번에 한 줄씩 실행
- 오류가 발생하면 그 즉시 실행을 멈추고 오류 메시지와 오류가 발생한 줄 번호를 웹 브라우저의 콘솔창에 출력
- 모든 오류 관련 메시지는 웹 브라우저의 콘솔창에서 확인 가능

8.2 프로그래밍 시작 전 알아 두기

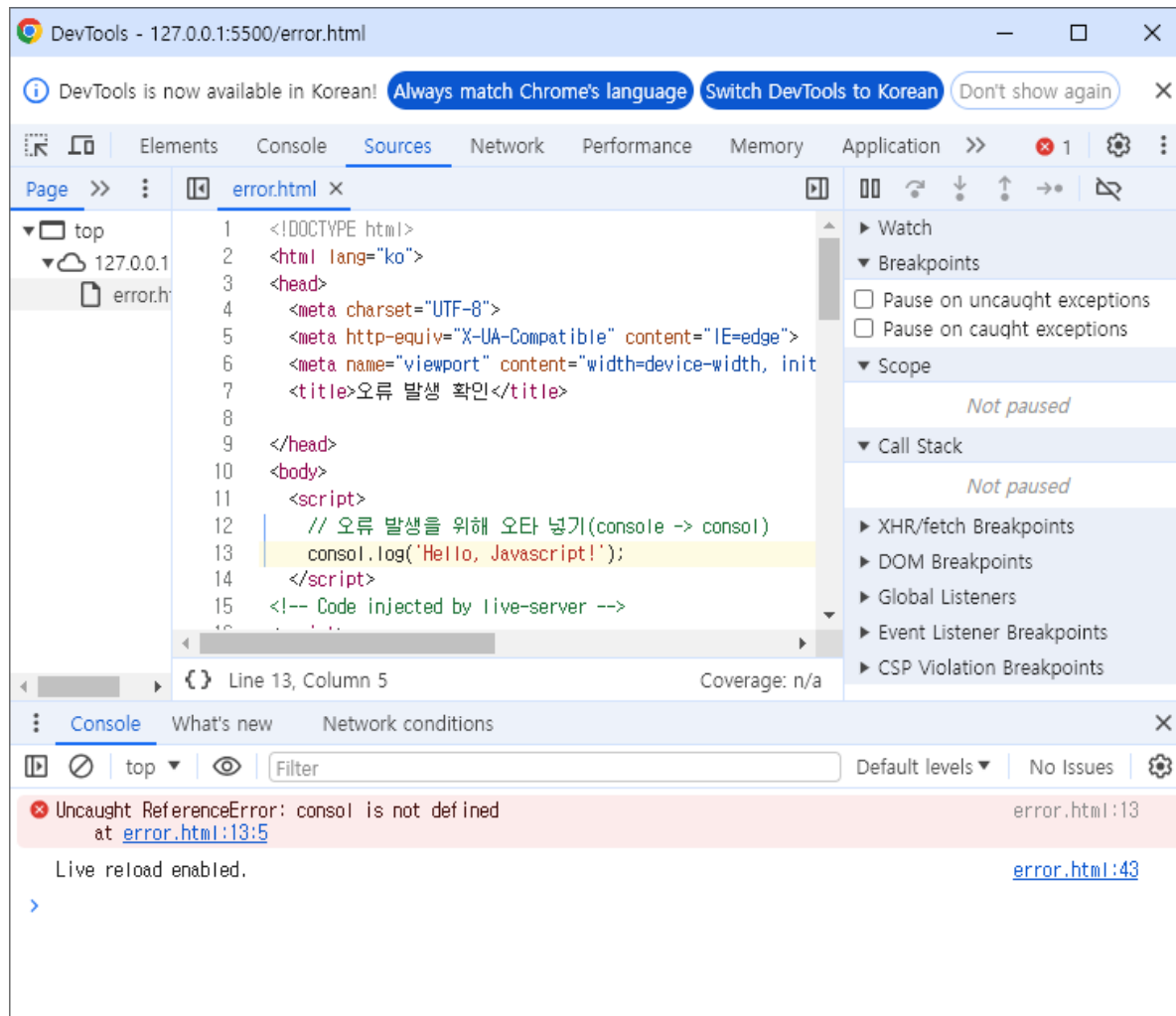
08/02/error.html

```
<body>
  <script>
    // 오류 발생을 위해 오타 넣기(console -> consol)
    consol.log('Hello, Javascript!');
  </script>
</body>
```

8.2 프로그래밍 시작 전 알아 두기



8.2 프로그래밍 시작 전 알아 두기



9장

자바스크립트 기초 문법 살펴보기

9.1 변수와 상수

9.2 자료형

9.3 연산자

9.4 조건문 다루기

9.5 반복문 다루기

9.1 변수와 상수

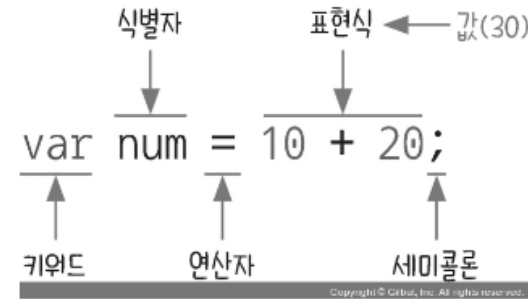


그림 9-1 변수 문법

1. 변수

- 변수(variant) : 변하는 수. 값이 변하는 데이터를 저장하고 관리하기 위한 공간
- 키워드(keyword) : 자바스크립트 프로그래밍 언어에서 어떤 역할이나 기능이 정해진 특별한 단어. 예약어(reserved word)라고도 함
- 식별자(identifier) : 자바스크립트 내부에서 변수, 함수 등에 부여되는 이름
- 연산자(operator) : 어떠한 연산 작업을 하는 데 사용하는 기호
- 표현식(expression) : 평가되어 하나의 값을 반환하는 식 또는 코드
- 값(value) : 더 이상 평가할 수 없는 데이터
- 세미콜론(semicolon) : 하나의 문(statement, 문법)이 끝났음을 의미

9.1 변수와 상수

표 9-1 자바스크립트의 키워드

async	await	break	case	catch
class	const	continue	debugger	default
delete	do	else	enum	export
extends	false	finally	for	function
if	implement	important	in	instanceof
interface	let	new	null	package
private	protected	public	return	static
super	switch	this	throw	try
ture	typeof	void	while	with
yield				

9.1 변수와 상수

- 선언, 할당, 초기화
 - 변수 선언 : 변수를 생성하고 값을 저장하는 문법에서 var, let, const 키워드를 사용해 변수의 식별자를 지정하는 것
 - 값 할당 : 할당 연산자인 = 기호로 우변에 있는 값을 변수 공간에 대입(저장)하는 것
 - 변수 초기화 : 선언과 할당을 같이(한 번에) 하는 것

```
var num;
```

```
var num = 10 + 20; // 변수 num을 초기화합니다.
```

9.1 변수와 상수

2. 새로운 변수 선언 키워드 let

- 변수명 중복이 불가능(var 키워드는 중복생성 가능)
- 호이스팅되지 않음
 - 호이스팅(hoisting) : var 키워드로 변수를 선언하고 할당했을 때, 변수 선언을 자바스크립트의 스코프(scope) 맨 위로 올려 실행하는 것
- 스코프의 범위가 다름

```
let num = 10 + 20;  
let num = 50;
```

실행결과

Uncaught SyntaxError: Identifier 'num' has already been declared

9.1 변수와 상수

3. 상수

- 상수(constant) : 변하지 않는 수
- const 키워드는 재할당이 안 되는 특징 때문에 상수 변수(constant variable)를 선언할 때 사용하는 키워드라고 하기도 함

4. 식별자 명명 규칙

- 강제적 식별자 명명 규칙
 - 식별자에 키워드 사용 불가 **예** var, let, const
 - 식별자에 공백 포함 불가 **예** my School, like food
 - 식별자의 첫 글자는 영문 소문자, _(언더스코어), \$ 기호만 사용
예 *name, #age, @email

9.1 변수와 상수

3. 상수 예제

09/01/let_variable.js

```
let num = 10;  
num = 30;  
console.log(num);
```

실행결과

30

09/01/constant_variable.js

```
const num = 10;  
num = 30;  
console.log(num);
```

실행결과

Uncaught TypeError: Assignment to constant variable.

9.1 변수와 상수

3. 상수 에러 예제

09/01/constant_variable.js

```
const num = 10;  
num = 30;  
console.log(num);
```

실행결과

Uncaught TypeError: Assignment to constant variable.

09/01/constant_variable_error.js

```
const num; // 선언을 먼저 하고  
num = 10; // 할당을 나중에 해도 오류가 납니다.
```

실행결과

Uncaught SyntaxError: Missing initializer in const declaration

9.1 변수와 상수

- 관용적 식별자 명명 규칙
 - 식별자는 영문으로만 작성 **예** name, age
 - 식별자는 의미 있는 단어로 작성 **예** name, age(이름과 나이 저장 시)
- 식별자 표기법
 - 카멜 표기법 : 변수명과 함수명 작성 시 사용
예 firstName, lastName
 - 언더스코어 표기법 : 상수명 작성 시 사용
예 FIRST_NAME, last_name
 - 파스칼 표기법 : 생성자 함수명 작성 시 사용
예 FirstName, LastName

9.2 자료형

1. 문자열

- 큰따옴표나 작은따옴표로 둘러싸인 값
- 문자열에 따옴표가 포함된 경우
 - 문자열에 포함되지 않은 따옴표로 감싸서 정의
 - 문자열 연결 연산자(+) 또는 이스케이프 문자 사용
- 문자열 연결 연산자(+)
- 이스케이프 문자(₩, 역슬래시)
- 템플릿 문자열(`, 백틱)

9.2 자료형

1. 문자열 예제

09/02/string_type.js

```
let string1 = "Hello, World!";  
let string2 = 'Hello, World!';  
console.log(string1);  
console.log(string2);
```

실행결과

Hello, World!

Hello, World!

9.2 자료형

2. 숫자형

- 정수, 실수를 포함한 모든 숫자

3. 논리형

- 논리 값(true, false)

4. undefined

- 변수에 아무런 값도 할당되지 않는 상태를 나타내는 값

9.2 자료형

2. 숫자형 예제

09/02/number_type.js

```
let num1 = 10;  
let num2 = 0.1;  
console.log(num1);  
console.log(num2);
```

실행결과

10

0.1

9.2 자료형

3. 논리형 예제

09/02/boolean_type.js

```
let boolean1 = true;
let boolean2 = false;
console.log(boolean1); // true
console.log(boolean2); // false
```

09/02/boolean_type_ex.js

```
let boolean1 = 10 < 20;
let boolean2 = 10 > 20;
console.log(boolean1); // true
console.log(boolean2); // false
```

9.2 자료형

4. undefined 예제

09/02/undefined_type.js

```
let empty;  
console.log(empty); // undefined
```

9.2 자료형

5. null

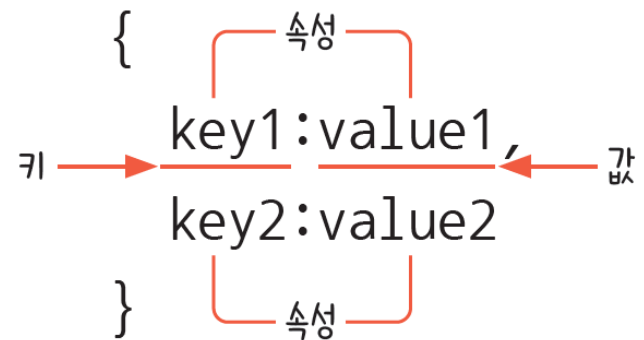
- 변수를 의도적으로 비워 두기 위해 사용하는 값

6. 객체

- 배열, 함수, 객체 리터럴 등으로 파생되는 상위 자료형

- 배열(array) : 복수의 데이터를 정의할 수 있는 자료형
- 객체 리터럴 : 중괄호({})를 사용하며, 키(key)와 값(value)의 한쌍으로 이루어짐

그림 9-3 객체의 구성 요소 명칭



9.2 자료형

5. null 예제

09/02/null_type.js

```
let empty = null;  
console.log(empty); // null
```

6. 배열 예제

09/02/array.js

```
let studentScore = [80, 70, 90, 60]; // 국어, 영어, 수학, 과학 점수  
console.log(studentScore[1]); // 70, 1번 인덱스의 데이터에 접근
```

9.2 자료형

6. 객체 예제

09/02/object_literal.js

```
let studentScore = {  
  koreanScore:80,  
  englishScore:70,  
  mathScore:90,  
  scienceScore:60  
};  
console.log(studentScore.koreanScore);    // 80  
console.log(studentScore['englishScore']); // 70
```

실행결과

80

70

9.3 연산자

1. 산술 연산자

- 이항 산술 연산자 : 연산을 수행하는 데 피연산자가 2개 필요한 연산자

- $x + y$ x 에 y 를 더함
- $x - y$ x 에 y 를 뺀
- $x * y$ x 에 y 를 곱함
- x / y x 를 y 로 나눔
- $x \% y$ x 를 y 로 나누어 나머지를 구함
- $x ** y$ x 의 y 거듭제곱

9.3 연산자

- 단항 산술 연산자 : 연산을 수행하는 데 피연산자가 1개만 필요한 연산자
 - 연산자를 앞에 사용하면 전치 연산, 뒤에 사용하면 후치 연산
 - $x++$ (후치 연산), $++x$ (전치 연산) : x 를 1 증가시킴
 - $x--$ (후치 연산), $--x$ (전치 연산) : x 를 1 감소시킴
- 단항 부정 연산자 : 항상 피연산자 앞에 위치하며 피연산자의 부호를 부정하는 연산자
 - $-x$: x 의 부호를 부정(음수 \rightarrow 양수, 양수 \rightarrow 음수)

9.3 연산자

2. 대입 연산자와 복합 대입 연산자

- 대입 연산자 : 데이터를 대입(할당)하는 연산을 수행하는 연산자
 - $x = y$: x 에 y 를 대입
- 복합 대입 연산자 : 산술 연산자와 대입 연산자를 함께 사용해 산술과 할당을 한 번에 수행하는 연산자
 - $x += y$: x 에 $x + y$ 를 대입
 - $x -= y$: x 에 $x - y$ 를 대입
 - $x *= y$: x 에 $x * y$ 를 대입
 - $x /= y$: x 에 x / y 를 대입
 - $x \% = y$: x 에 $x \% y$ 를 대입
 - $x ** = y$: x 에 $x ** y$ 를 대입

9.3 연산자

3. 비교 연산자

- 피연산자를 비교한 뒤, 논리형 값인 참(true), 거짓(false)을 반환
 - $x == y$: x 와 y 의 값이 같으면 true를 반환
 - $x === y$: x 와 y 의 값과 자료형이 같으면 true를 반환
 - $x != y$: x 와 y 의 값이 다르면 true를 반환
 - $x !== y$: x 와 y 의 값과 자료형이 다르면 true를 반환
 - $x < y$: x 가 y 보다 작으면 true를 반환
 - $x <= y$: x 가 y 보다 작거나 같으면 true를 반환
 - $x > y$: x 가 y 보다 크면 true를 반환
 - $x >= y$: x 가 y 보다 크거나 같으면 true를 반환

9.3 연산자

4. 논리 연산자

- 연산자를 논리적으로 평가한 뒤, 조건에 맞는 피연산자를 반환
 - $x \ \&\& \ y \rightarrow$ x 가 참이면 y 를 반환하고, 거짓이면 x 를 반환
 - $x \ || \ y \rightarrow$ x 가 참이면 x 를 반환하고, 거짓이면 y 를 반환
 - $!x \rightarrow$ x 가 참이면 `false`를 반환하고, 거짓이면 `true`를 반환

5. 삼항 연산자

- 세 항 중 가장 왼쪽에 있는 피연산자의 참, 거짓에 따라 나머지 두 항에 있는 피연산자를 선택적으로 반환
 - $x \ ? \ y \ : \ z \rightarrow$ x 가 참이면 y 를 반환하고, x 가 거짓이면 z 를 반환

9.3 연산자

6. 연산자 우선순위

- 연산자를 여러 개 사용했을 때 어떤 연산자를 먼저 연산할지를 결정하는 기준
- 가능한 한 우선순위가 가장 높은 그룹 연산자를 사용해 식의 우선순위를 단순하게 정리하는 것이 좋음

그림 9-5 연산자의 우선순위

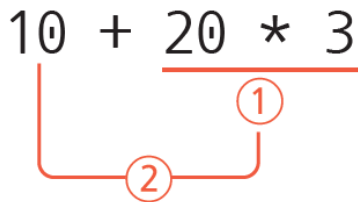


그림 9-6 우선순위 변경

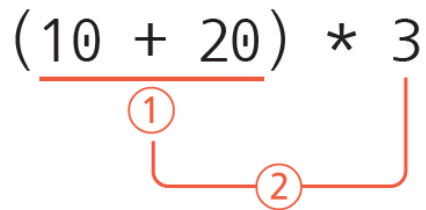
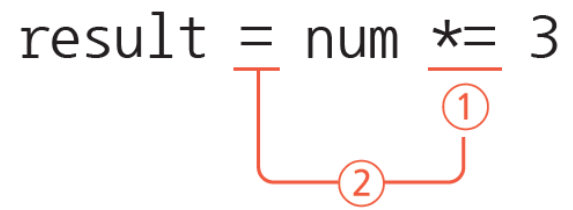


그림 9-7 결합 순서에 의한 연산 순서



9.3 연산자

```
const result = 10 + "10";  
console.log(result); // 1010
```

7. 형 변환

- 데이터의 자료형이 다른 자료형으로 바뀌는 것
- 암시적 형 변환
 - 사용자가 형 변환을 의도하지 않았지만, 자바스크립트에서 자체적으로 형 변환하는 것
 - 개발자가 놓친 부분이라는 의미이므로 암시적 형 변환이 발생하지 않도록 코드에 형 변환을 명확하게 표시하는 것이 좋음
- 명시적 형 변환
 - 드러나게 형 변환을 처리하는 것

```
let num = 10;  
let strNum = "10";  
if(String(num) == strNum){  
  console.log(`equals`);  
}
```

9.4 조건문 다루기

1. if, else, else if 문

- if 문 : if 뒤에 오는 소괄호() 안의 조건식이 참으로 평가되면 중괄호 안의 코드를 실행하는 조건문

형식 `if(조건식){`

`// 조건식이 참이면 실행`

`}`

- 블록문(block statement) : 한 개 이상의 자바스크립트 코드를 중괄호로 묶은 것. 블록 또는 코드 블록

9.4 조건문 다루기

- else 문 : if 문의 조건식이 거짓일 때 실행되는 블록문

형식 `if(조건식){`

`// 조건식이 참이면 블록문 실행`

`}else{`

`// 조건식이 거짓이면 블록문 실행`

`}`

9.4 조건문 다루기

09/04/if.js

```
let num = 10;
if(num % 2 === 0){
  console.log("변수 num에 할당된 숫자는 짝수입니다.");
}
```

실행결과

변수 num에 할당된 숫자는 짝수입니다.

09/04/else.js

```
let num = 5;
if(num % 2 === 0){
  console.log("변수 num에 할당된 숫자는 짝수입니다.");
}else{
  console.log("변수 num에 할당된 숫자는 홀수입니다.");
}
```

9.4 조건문 다루기

- else if 문 : if 문에 조건을 추가하고 싶을 때 사용하는 블록문

형식 `if(조건식1){`

`// 조건식1이 참이면 블록문 실행`

`}else if(조건식2){`

`// 조건식2가 참이면 블록문 실행`

`}else{`

`// 조건식이 모두 거짓이면 블록문 실행`

`}`

- 분기 처리 : 어떤 조건식을 만족할 때 어떤 블록문을 실행할지 결정하는 것. if 문은 작성하려는 코드의 분기 처리에 따라 중첩해서 사용할 수 있음

9.4 조건문 다루기

09/04/else_if.js

```
let num = 0;
if(num > 0){
  console.log("양수");
}else if(num < 0){
  console.log("음수");
}else{
  console.log("0");
}
```

실행결과

0

9.4 조건문 다루기

2. switch 문

- switch 뒤에 오는 소괄호 안의 값과 일치하는 case 문이 있을 때 해당 코드를 실행하는 조건문

형식 `switch(key){`
 `case value1:`
 `// key가 value1일 때 실행할 블록문`
 `break;`
 `case value2:`
 `// key가 value2일 때 실행할 블록문`
 `break;`
 `default:`
 `// 아무것도 일치하지 않을 때 실행할 블록문`
 `break;`
}

9.4 조건문 다루기

09/04/switch.js

```
let food = "melon";
switch(food){
  case "melon":
    console.log("fruit");
    break;
  case "apple":
    console.log("fruit");
    break;
  case "banana":
    console.log("fruit");
    break;
  case "carrot":
    console.log("vegetable");
    break;
  default:
    console.log("It's not fruits and vegetables.");
    break;
}
```

실행결과

fruit

9.4 조건문 다루기

3. if 문과 조건식

- if 문은 조건에 식을 사용
- 논리 연산자나 비교 연산자를 식에 이용할 수 있음

4. if 문 vs switch 문

- if 문 : 조건에 식(statement)을 사용, 범위를 이용한 조건을 작성할 때
- switch 문 : 조건에 값(value)을 사용, 값이 하나일 때

```
let score = 90;
if(score >= 90 && score <= 100){
  console.log("A++ 학점");
}
```

9.5 반복문 다루기

- 반복문(loop) : 지정한 조건이 참(true)으로 평가되는 동안 지정한 블록문을 반복해서 실행

1. while 문

- 특정 조건을 만족하는 동안 블록문을 실행

형식 `while(조건식){`

`// 조건식이 참이면 실행`

`}`

```
let num = 1;
while(num <= 9999){
  console.log(num);
  num++;
}
```

09/05/while.js

9.5 반복문 다루기

2. 무한 반복문

- 반복문의 조건이 계속 참으로 평가되어 반복문이 끝나지 않고 무한히 실행되는 것

3. do...while 문

- 특정 조건이 참으로 평가되는 동안 do 다음에 오는 블록문을 반복 실행

```
형식 do{  
    // 블록문  
}while(조건식);
```


9.5 반복문 다루기

```
let num = 1;
while(num <= 9999){
  console.log(num);
  num++; // 코드가 한 번 반복될 때마다 num 변수를 1씩 증가시킵니다.
}
```

09/05/do_while.js

```
do{
  console.log("무조건");
  console.log("한 번은 실행");
}while(false);
```

실행결과

무조건

한 번은 실행

9.5 반복문 다루기

4. for 문

- 지정한 횟수가 끝날 때까지 블록문을 반복 실행하는 반복문

형식 `for`(초깃값; 조건식; 증감식){

 // 블록문

}

- 초깃값 → 조건식 → 블록문(조건식이 참일 경우) → 증감식 → 조건식 순서로 실행
- 중첩해서 사용할 수 있음

9.5 반복문 다루기

5. for 문과 배열

- 배열과 같은 자료형을 반복 횟수 용도로 사용할 수 있음

6. for...in

```
형식 for(가변수 in 배열/객체 리터럴){  
    // 블록문  
}
```

- 객체 리터럴을 반복할 경우 : 탐색 결과로 가변수에 객체 리터럴의 키가 할당되어 객체 리터럴의 키와 값을 출력할 수 있음
- 배열을 반복할 경우 : 배열의 순서대로 접근하는 것을 보장하지 않으므로 코드를 작성할 때 주의할 것!

9.5 반복문 다루기

09/05/for_in_obj.js

```
let obj = {name:"철수", age:"20"};
for(let key in obj){
  console.log(key + ": " + obj[key]);
}
```

실행결과

name: 철수

age: 20

09/05/for_in_arr.js

```
let arr = ["orange", "banana", "apple"];
for(let index in arr){
  console.log(index + ": " + arr[index]);
}
```

실행결과

0: orange

1: banana

2: apple

9.5 반복문 다루기

7. break 문

- 종료 조건을 만족하지 않아도 인위적으로 반복문을 종료하게 할 때

8. continue 문

- 반복문을 건너뛰게 할 때

9.5 반복문 다루기

09/05/for_break.js

```
for(let i = 0; i < 10; i++){  
  console.log(i);  
  if(i === 5) break;  
}
```

실행결과

0
1
2
3
4
5

9.5 반복문 다루기

09/05/for_break_ex.js

```
let obj = {name:"철수", age:20};  
for(let key in obj){  
  if(key === "age") break;  
  console.log(obj[key]);  
}
```

실행결과

철수

09/05/for_continue.js

```
for(let i = 1; i <= 10; i++){  
  if(i % 2 === 1) continue;  
  console.log(i);  
}
```

실행결과

2

4

6

8

10

9.5 반복문 과제

```
2단
2X1=2
2X2=4
2X3=6
2X4=8
2X5=10
2X6=12
2X7=14
2X8=16
2X9=18
```

.

.

```
9단
9X1=9
9X2=18
9X3=27
9X4=36
9X5=45
9X6=54
9X7=63
9X8=72
9X9=81
```

자바스크립트로 for문을 이용한 2~9단까지 출력하는 구구단 코드를 작성하고 코드만 캡처해서 과제로 제출

구구단 출력 방법

n단

$n \times 1 = n$

$n \times 2 = 2n$

.

.

$n \times 9 = 9n$

다음단이 나올때 한줄만 띄운다.

n+1단

$(n+1) \times 1 = n+1$

[Done] exited with code=0 in 0.154 seconds