

WEEK 2 JAVASCRIPT STUDY (1)

최강 프론트.





JAVASCRIPT STUDY WEEK1 세션 내용

01. 자바스크립트란? 탄생 배경을 알아보자.
02. 변수 (변수 / 식별자 / 호이스팅 / 값 할당 / 네이밍 규칙까지)
03. 표현식 & 문 & 데이터 타입
04. 연산자 (산술 / 할당 / 비교 / 삼항 / 논리 연산자 / ...)
05. 제어문 (블록문 / 조건문 / 반복문 / ...)
06. 타입 변환 살펴보기
07. Q&A + 과제 공지



초기

HTML & CSS 만으로 정적인 웹사이트 제작

후기

동적인 웹사이트 제작을 위해 Mocha 탄생 → Live script → Java script

Microsoft : JScript vs Netscape : Java script

자바스크립트의 표준화 : ECMA Script 1 (1997) ~

ES6 (2015) : default parameter, class, arrow function, let, const ...



Babel : Javascript transcompiler

- ECMA 최신 버전을 사용자에게 배포할 때 ES5 or ES6로 문법 변환 (2020: ES11)
- 항상 최신 문법의 자바스크립트로 코딩할 수 있도록 도와주는 트랜스파일러

SPA : Single Page Application

- 하나의 페이지 내에서 데이터를 받아와 필요한 부분만을 부분적으로 업데이트



V8 자바스크립트 엔진

- 모든 브라우저 : 자바스크립트의 해석 & 실행이 가능하도록 자바스크립트 엔진 내장
- 브라우저에서 동작하는 코드 → Node.js 환경에서도 동일하게 동작

브라우저 vs Node.js

- 브라우저 : HTML, CSS, JS를 실행해 웹페이지를 브라우저 화면에 렌더링하는 것
- Node.js : 브라우저 외부에서 JS 실행 환경 제공

Java vs Java Script

Java

- 객체지향 프로그래밍 언어 (객체 사이 관계 기반)
- JVM 상에서 실행됨 (JRE + JDK)
- 다양한 OS에서 실행될 수 있는 독립적인 언어
- 블록 기반, 변수 : 블록 밖에서 사용 X

Java Script

- 객체지향 스크립트 언어
- 웹 브라우저 위에서 실행, 추가적 환경 설정 필요 X
- HTML, CSS에 의존해 실행됨
- 함수 기반, 변수 : 함수 안에서만 사용 O



VS

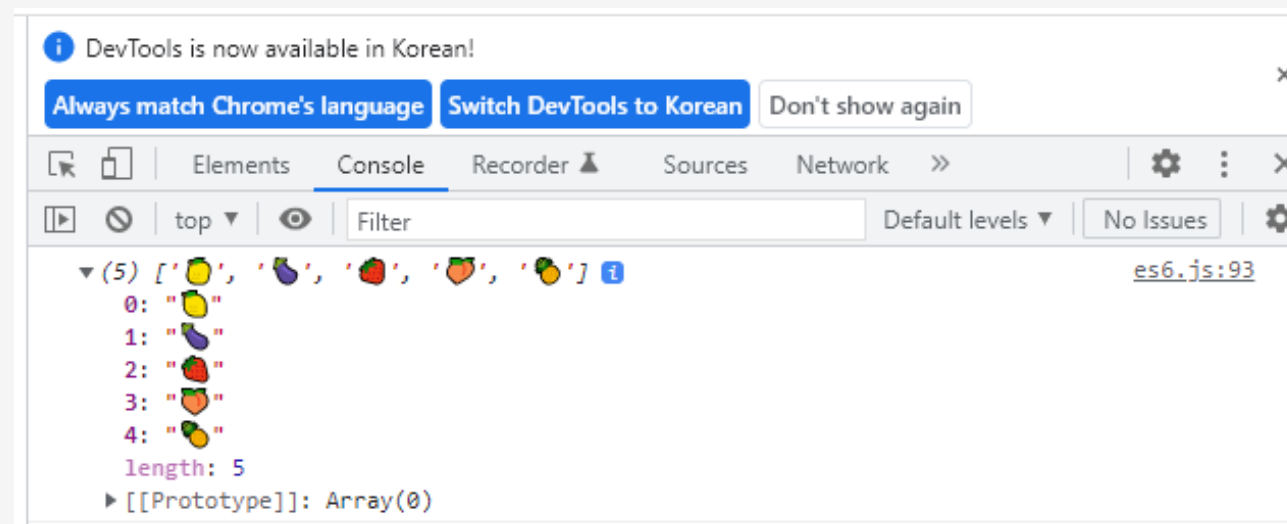




Javascript 실행 환경 셋팅

- node.js 설치 : <https://nodejs.org/ko/download/>
- node.js 버전 확인 : `node -v`
- VS Code 상 js 실행 : `node [JS 파일명]`
- open live server 상 js 실행 : F12 **or** `ctrl + shift + I` (개발자 도구)

```
UserK@Win1010 MINGW64 ~/Documents/GitHub/likelion_10th_session/FrontTeam-Javascript-study (main)
$ node javascript_week1.js
undefined
2022
undefined
1234
```





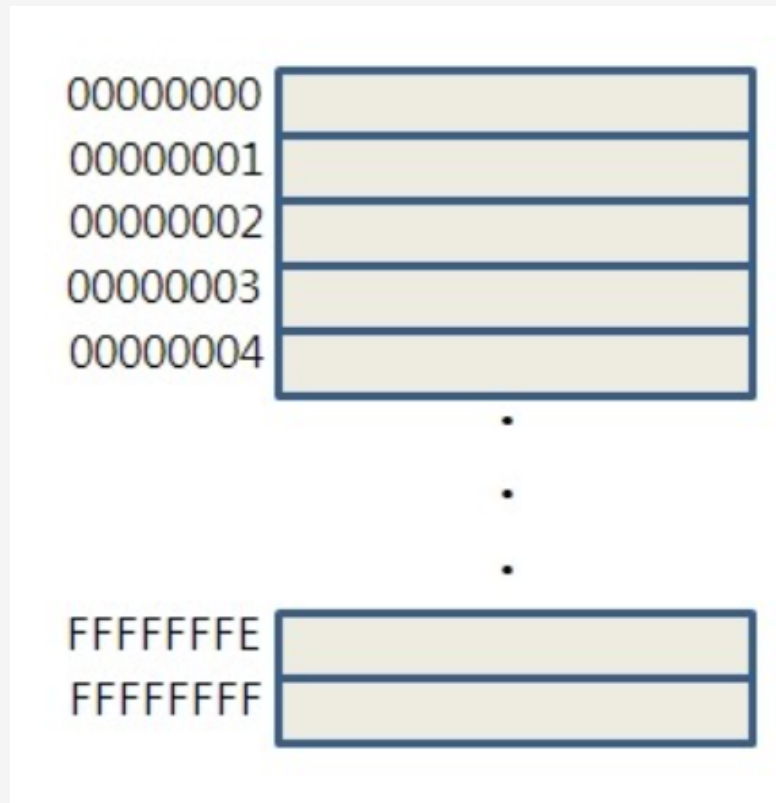
Javascript 공식 문서

<https://www.ecma-international.org/>

<https://developer.mozilla.org/ko/docs/Web/JavaScript>

변수

- 프로그래밍 언어에서 데이터를 관리하기 위한 핵심 개념
- 하나의 값을 저장하기 위해 확보한 메모리 공간 자체
또는 그 메모리 공간을 식별하기 위해 붙인 이름



- 프로그래밍 언어에서 값을 저장 & 참조하는 메커니즘
값의 위치를 가리키는 상징적인 이름

식별자

- 변수 이름 == 식별자
- 어떤 값을 구별하여 식별할 수 있는 고유한 이름

```
// 변수에 값을 저장하는 방법
var userId = 1;
var userName = "yoo";

// 객체 or 배열을 사용한 변수의 저장
var user = { id : 1, name : 'yoo'};

var users = [
  { id : 1, name : 'yoo'},
  { id : 1, name : 'kim'},
]
```

변수 선언

- 변수를 생성하는 것
- var, let, const 키워드 사용 (let, const : ES6 ~)
- var와 let의 차이 : 변수 선언 키워드, 예제 코드를 통해 같이 살펴봐요!
- const : 상수 변수 선언 키워드, 불변의 값 생성
- 변수 선언 : 런타임(소스코드가 한줄씩 순차적으로 실행되는 시점)이 아닌, 그 이전 단계에서 먼저 실행
 - 선언 단계 : 변수 이름을 등록해 JS 엔진에 변수의 존재를 알림
 - 초기화 단계 : 값을 저장하기 위한 메모리 공간 확보, 암묵적으로 undefined 할당하여 초기화 2단계에 걸쳐 이루어짐
- 자바스크립트 엔진 : 실행 전 선언해둔 변수를 확인해 메모리에 기억해둠

호이스팅

- 변수 선언문이 코드의 선두로 끌어 올려진 것처럼 동작하는 자바스크립트 고유의 특징
- var 변수 : 선언 단계 & 초기화 단계가 동시에 진행됨



var 쓰지마세요 ...

그래서 왜! var를 쓰면 안되나요?



var로 선언된 변수

1. 전역변수와 지역변수에 대한 개념이 확실하지 않음
 - 함수만을 지역변수로 호이스팅
 - 나머지의 것들은 다 전역변수로 올림 ex) for, if ...
2. 동일한 식별자를 가진 변수의 재선언이 가능함
 - ex) 본인과 같은 주민등록번호를 가진 사람이 또 생겨난다면?

식별자 네이밍 규칙

- 특수문자를 제외한 문자, 숫자, 언더스코어(_), 달러기호(\$) 포함 가능
- 식별자 네이밍의 시작 : 특수문자를 제외한 문자, 언더스코어(_), 달러기호(\$)로 시작함, 숫자로 시작하는 것 허용 X
- 예약어 : 식별자로 사용 X

프로그래밍에서 언어로 사용되고 있거나 사용될 예정인 언어

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



값

- 식이 평가되어 생성된 결과

```
10 + 20;
```

리터럴

- 사람이 이해할 수 있는 문자 또는 약속된 기호를 사용해 값을 생성하는 표기법
ex) 정수, 2/8/16진수, 문자열, 불리언, null, undefined, 객체, 배열, 함수, 정규표현식 리터럴 ...

```
// 숫자 리터럴 37  
37
```

표현식

- 값으로 평가될 수 있는 문
- 표현식이 평가되면 새로운 값의 생성 및 기존 값의 참조가 이루어짐

```
var score = 50 + 40;  
score;
```

문

- 프로그램을 구성하는 기본 단위 / 최소 실행 단위
- 여러 토큰으로 구성됨

자바스크립트 엔진

- 세미콜론 자동 삽입 기능(ASI)의 암묵적 수행



데이터 타입

- 값은 메모리에 저장 & 참조할 수 있어야 함
- 메모리로의 값 저장을 위해 확보해야 할 메모리 공간의 크기를 결정해야 함

구분	데이터 타입	설명
원시타입 (primitive type)	숫자(number)	숫자, 정수와 실수 구분없이 하나의 타입만 존재
	문자열(string)	문자열
	불리언(boolean)	논리적 참(true) 과 거짓(false)
	underfined	var 키워드로 선언된 변수에 암묵적으로 할당되는 값
	null	값이 없다는 것을 의도적으로 명시 할때 사용하는 값
	심벌(symbol)	es6에서 추가되는 7번째 타입
객체 타입(object/refernece type)		객체, 함수 배열 등



데이터 타입

- 배열 : 같은 자료구조를 담는 배열

```
// 배열 생성 (빈 배열)  
let arr = [];  
  
arr[0] = 'zero';  
arr[1] = 'one';  
arr[2] = 'two';  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}
```

```
// 배열 생성 (초기 값 할당)  
let arr = ['zero', 'one', 'two'];  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}
```

```
// 배열 생성 (배열 크기 지정)  
// 쉼표 개수만큼 크기가 지정됨  
let arr = [,,,];  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}  
  
// 값이 할당되지 않아서 undefined 3번 출력
```



데이터 타입

- 배열 : 같은 자료구조를 담는 배열

```
// 배열 생성 (빈 배열)  
let arr = new Array();  
  
arr[0] = 'zero';  
arr[1] = 'one';  
arr[2] = 'two';  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}
```

```
// 배열 생성 (초기 값 할당)  
let arr = new Array('zero', 'one', 'two');  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}
```

```
// 배열 생성 (배열 크기 지정)  
// 원소가 1개이고 숫자인 경우 배열 크기로 사용됨  
let arr = new Array(3);  
  
for (let i = 0; i < arr.length; i++) {  
  console.log(arr[i]);  
}  
  
// 값이 할당되지 않아서 undefined 3번 출력
```



데이터 타입

- 배열명.length : 배열 요소의 개수 (인덱스 개수)

```
const fruits = ['apple', 'banana', 'kiwi', 'melon'];

console.log(fruits.length);

// 배열의 원소 개수만큼 반복 (배열의 모든 인덱스에 방문)
for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

데이터 타입

- 값은 메모리에 저장 & 참조할 수 있어야 함
- 메모리로의 값 저장을 위해 확보해야 할 메모리 공간의 크기를 결정해야 함

```
3 // 문자열 : 작은따옴표, 큰따옴표, 백틱 표기 모두 가능
4 let string;
5 let name = '지민';
6 string = '작은 따옴표의 문자열';
7 string = "큰 따옴표의 문자열";
8 string = `백틱 표기의 문자열`;
9 let newString = `이름은 ${name} 이다.`;
10 console.log(newString);
11
```

문제 출력 디버그 콘솔 터미널

```
UserK@Win1010 MINGW64 ~/Documents/GitHub/likelion_10th_session/FrontTeam-Javascript-study (main)
$ node javascript_basic_practice.js
백틱 표기의 문자열
이름은 지민 이다.
```

```
// 심벌 타입 : 변경 불가능한 원시 타입의 값
// 이름이 충돌할 위험이 없는 객체의 유일한 프로퍼티 키를 만들기 위해 사용
let key = Symbol('key');
console.log(typeof key); // symbol

let obj = {}; // 객체 생성
obj[key] = 'value';
console.log(obj[key]); // value
```

연산자

- 산술 연산자 : 이항 / 단항 / 문자열 연결 연산자
- 할당 연산자
- 비교 연산자 : 동등/일치 비교 연산자, 대소 관계 비교 연산자
- 삼항 조건 연산자
- 논리 연산자
- typeof 연산자
- instanceof 연산자



연산자

- 산술 연산자 : 이항 / 단항 / 문자열 연결 연산자

```
// 이항 산술 연산자
5 + 2; // 7 : 덧셈
5 - 2; // 3 : 뺄셈
5 * 2; // 10 : 곱셈
5 / 2; // 2.5 : 나눗셈
5 % 2; // 1 : 나머지
```

```
// 단항 산술 연산자
let x = 1;
x++; // x = x + 1 : 증가
x--; // x = x - 1 : 감소
+x; // 아무런 효과 x, 음수 -> 양수의 반전 x
-x; // 양수 -> 음수, 음수 -> 양수
```

```
// 문자열 연결 연산자
'1' + 2; // 12
1 + '2'; // 12

1 + 2; // 3

1 + true; // 2 : true - 1
1 + false; // 1 : false - 0
1 + null; // 1 : null - 0
```

연산자

- 할당 연산자
- 비교 연산자 : 동등/일치 비교 연산자, 대소 관계 비교 연산자

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

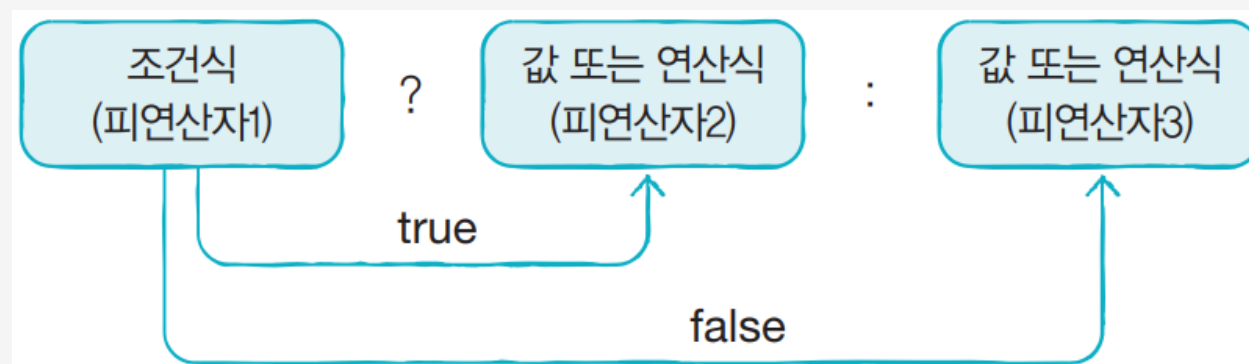
연산자	설명
==	equal to 같은지
===	equal value and equal type 값과 타입이 모두 같은지
!=	not equal 같지 않은
!==	not equal value or not equal type 값과 타입이 모두 같지 않은
>	greater than 보다 큰
<	less than 보다 작은
>=	greater than or equal to 크거나 같은
<=	less than or equal to 작거나 같은
?	ternary operator 삼항 연산자



연산자

- 삼항 조건 연산자
- 논리 연산자

- typeof 연산자
- instanceof 연산자



연산자	설명
&&	logical and
	logical or
!	logical not

연산자	설명
typeof	변수의 타입을 반환한다
instanceof	object가 해당 object 타입이면 true를 반환한다

제어문

- 블록문
- 조건문
- switch문
- 반복문 (for문, while문)
- break문
- continue문

제어문

- 블록문 : 0개 이상의 문을 중괄호로 묶은 것 (코드 블록 / 블록)
 - Javascript : 블록문을 하나의 실행 단위로 취급
 - 제어문 / 함수를 정의할 때 사용하는 것이 일반적

```
{  
  var foo = 10;  
}
```


제어문

- 조건문 : 주어진 조건식의 평가 결과에 따라 코드 블록의 실행을 결정하는 문
 - 불리언 값으로 평가될 수 있는 표현식

```
let num = 2;
let kind;

if (num > 0) {
  kind = "양수";
} else if (num < 0) {
  kind = "음수";
} else {
  kind = "0";
}
```

제어문

- switch문 : 주어진 표현식을 평가해

그 값과 일치하는 표현식을 갖는 case문으로 실행 흐름을 옮김

```
const Test = 80;
switch(Test) {
  case 100 : // case : 조건
    console.log('100점 통과');
    break; // 멈추기
  case 90 :
    console.log('90점 통과');
    break;
  case 80 :
    console.log('80점 통과'); // 조건값과 동일한 case, 이 구문을 실행한다.
    break;
  default : // 조건 외의 값 else와 기능이 동일
    console.log('80점 미만 탈락입니다.');
```

```
    break;
}
```



제어문

- 반복문 (for, while) : 조건식의 평가 결과가 참인 경우 코드 블록을 실행

```
for (var i = 0; i < 2; i++) {  
    console.log(i);  
}
```

① i=0 ② true ③ i=0
 ⑤ true ④ i=1
 ⑧ false ⑦ i=2
 ⑥ i=1


```
for(초기화 조건식 증감문) {  
    //코드를 작성하는 곳  
}
```

```
초기화  
while(조건식) {  
    증감문  
    //코드를 작성하는 곳  
}
```

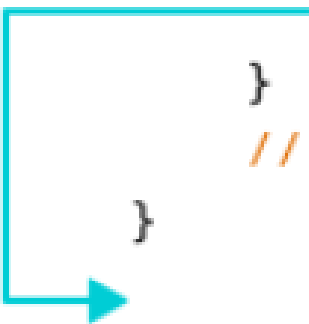
제어문

- break문

```
for (init; condition; update) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```

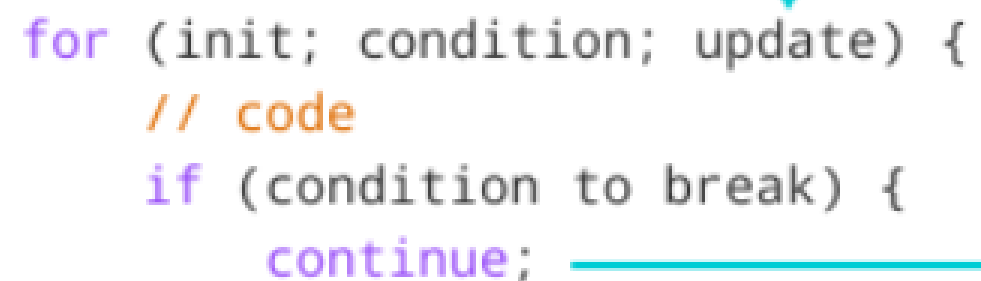


```
while (condition) {  
    // code  
    if (condition to break) {  
        break;  
    }  
    // code  
}
```

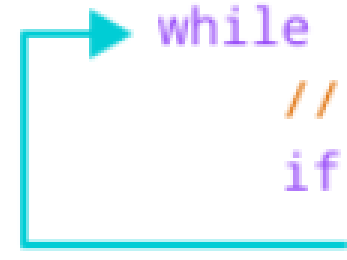


- continue문

```
for (init; condition; update) {  
    // code  
    if (condition to break) {  
        continue;  
    }  
    // code  
}
```



```
while (condition) {  
    // code  
    if (condition to break) {  
        continue;  
    }  
    // code  
}
```



타입 변환

- 암묵적 타입 변환

```
number + number // number
number + string // string
string + string // string
string + boolean // string
number + boolean // number
50 + 50; //100
100 + "점"; //"100점"
"100" + "점"; //"100점"
"10" + false; //"10"
99 + true; //100
```

```
//다른 연산자(-,*,/,%)
string * number // number
string * string // number
number * number // number
string * boolean //number
number * boolean //number
"2" * false; //0
2 * true; //2
```

타입 변환

- 명시적 타입 변환 : 표준 빌트인 생성자 함수(String, Number, Boolean)

```
const falsy1 = null;  
Number(falsy1); // 0;  
  
const falsy2 = '';  
Number(falsy2); // 0;  
  
const falsy3 = false;  
Number(falsy3); // 0;  
  
const truthy1 = [];  
Number(truthy1); // 0;  
  
const truthy2 = true;  
Number(truthy2); // 1;  
  
const truthy3 = {};  
Number({}); // NaN;
```

Number() : 정수형과 실수형의 숫자로 변환

parseInt() : 정수형의 숫자로 변환

parseFloat() : 부동 소수점의 숫자로 변환



타입 변환

- 명시적 타입 변환 : 표준 빌트인 생성자 함수(String, Number, Boolean)

```
String(123); // "123"  
String(123.456); // "123.456"
```

```
var trans = 123.456789;  
var roundOff = 99.987654;  
trans.toFixed(); // "123"  
trans.toFixed(0); // "123"  
trans.toFixed(2); // "123.46"  
trans.toFixed(8); // "123.45678900"  
roundOff.toFixed(2); // "99.99"  
roundOff.toFixed(0); // "100"
```

String() : 다른 자료형을 문자 타입으로 변환

toString() : 주어진 값을 문자열로 변환 수 반환
(인자로 기수 선택 가능)

toFixed() : 인자값만큼 반올림하여 소수점 표현

타입 변환

- 명시적 타입 변환 : 표준 빌트인 생성자 함수(String, Number, Boolean)

```
Boolean(100); //true
Boolean("1"); //true
Boolean(true); //true
Boolean(Object); //true
Boolean([]); //true
Boolean(0); //false
Boolean(NaN); //false
Boolean(null); //false
Boolean(undefined); //false
Boolean( ); //false

const numb1 = 0;
Boolean(numb1); // false
!!numb1; // false
!numb1; // true
```

Boolean() : Boolean 타입으로의 변경 (! 사용)



node.js로 BOJ 풀기 (입출력 연산)

```
// 1. 하나의 값을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim();

// 2. 공백으로 구분된 한 줄의 값들을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim().split(' ');

// 3. 여러 줄의 값들을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim().split('\n');

// 4. 첫 번째 줄에 자연수 n을 입력받고, 그 다음줄에 공백으로 구분된 n개의 값들을 입력받을 때
const [n, ...arr] = require('fs').readFileSync('/dev/stdin').toString().trim().split(/\s/);

// 5. 첫 번째 줄에 자연수 n을 입력받고, 그 다음줄부터 n개의 줄에 걸쳐 한 줄에 하나의 값을 입력받을 때
const [n, ...arr] = require('fs').readFileSync('/dev/stdin').toString().trim().split('\n');
```

node.js로 BOJ 풀기 (입출력 연산)

```
// 1. 하나의 값을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim();

// 2. 공백으로 구분된 한 줄의 값들을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim().split(' ');

// 3. 여러 줄의 값들을 입력받을 때
const input = require('fs').readFileSync('/dev/stdin').toString().trim().split('\n');

// 4. 첫 번째 줄에 자연수 n을 입력받고, 그 다음줄에 공백으로 구분된 n개의 값들을 입력받을 때
const [n, ...arr] = require('fs').readFileSync('/dev/stdin').toString().trim().split(/\s/);

// 5. 첫 번째 줄에 자연수 n을 입력받고, 그 다음줄부터 n개의 줄에 걸쳐 한 줄에 하나의 값을 입력받을 때
const [n, ...arr] = require('fs').readFileSync('/dev/stdin').toString().trim().split('\n');
```

node.js로 BOJ 풀기 (입출력 연산)

입출력

Hello World : <https://www.acmicpc.net/problem/2557>

사칙연산 : <https://www.acmicpc.net/problem/10869>

조건문

시험 성적 : <https://www.acmicpc.net/problem/9498>

윤년 : <https://www.acmicpc.net/problem/2753>

반복문

N 출력 <https://www.acmicpc.net/problem/2741>

별찍기 -1 <https://www.acmicpc.net/problem/2438>

배열

최소, 최대 <https://www.acmicpc.net/problem/10818>

