

23-11-02

- 첫 언어 배움

<언어에 대한 이야기>

- 언어 : 컴퓨터에서 사용자가 원하는 결과를 만들어내기 위해  
기계적, 프로그램적 작동 명령의 집합된 CODE

저급 -> 기계어 (0,1) : 컴퓨터만 사용한다, 우리는 사용할 일이 없다.

고급 -> 자연어 : 인간이 사용하는 언어 (영어)

- 기호등을 사용 (코볼\_사무, 파스칼\_과학, 포트란\_수학처리,통계)
- 언어만 사용 (C\_win운영체제,유닉스운영체제)

**User ---coding-> 컴파일러 ---compile-> 기계어**

Coding : 자연어로 이루어져 있다.

컴파일러 : 자연어를 기계어로 바꾸어준다. (한번에 번역)

- 단점 : 크기가 크면 시간이 오래 걸림, 오류가 나면 오류수정에 시간이 많이 걸린다.
- 장점 : 미리 규격화된 것을 사용해서 오류를 빨리 찾을 수 있다.

기계어 : 0,1로 이루어진 기계어로 번역한다.

**basic ---coding-> 인터프리터 사용 ---compile-> 기계어**

인터프리터 : 줄단위번역, 실시간 (ex\_F12 개발자 도구에서 콘솔에 직접 입력하는 경우)

- 단점 : 프로그램 커지면 그때마다 번역 -> 번역 실행 속도 느림, 문장 전체 오류 찾기 힘들다

<언어적구조>

1. 구조적언어 : C, 포트란, 코볼, (파스칼)

-문법이 매우 어렵다, 시스템 프로그램 개발에 사용

-c언어는 잘 쓰지 않는다, 보안할 때 필요하다.

## 2. 객체 지향 : C++, JAVA

-상속의 개념을 갖는다. -> 부모(PK)와 자식(FK)의 관계(ERD에서 했었다)

부모 지정해주면 자식은 부모에서 가지고 올거야 하면 다 가지고 올 수 있음

-Html태그 -> css에서 받아옴 -> 하나씩 쓰지 않아도 됨

## 3. 스크립트 언어 : C#, python, **Javascript**, \_\_.JS(로 되는프로그램)

-컴파일러 없다(컴파일 하지 않는다)

-인터프리터 : 구동하는 화면에서 바로바로 처리한다.

-(스크립트 언어들이 많아지는 이유 : 웹에서 빨리 처리하는 경우가 생겨서)

- 우리는 Javascript 언어 사용할 것이다 -> JAVA에 자료형이 없이 간략화한 언어

<자바스크립트 node 설치>

VS 코드에 터미널 부분

Md javascript //javascript의 디렉토리를 만든다

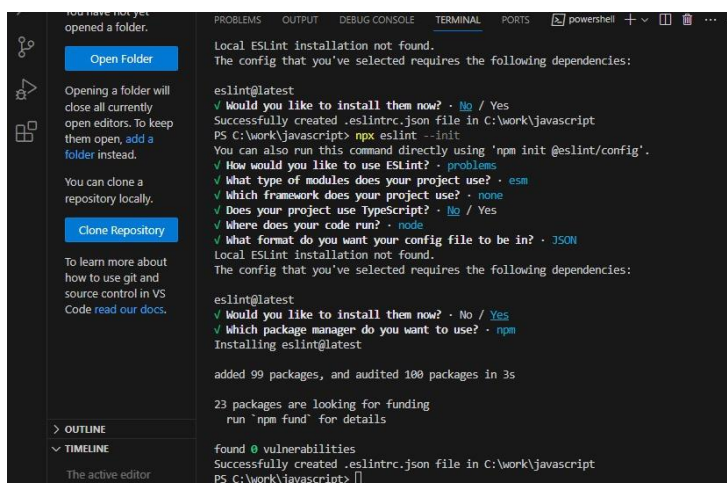
Nvm -version //nvm 버전 몇인지 알려줌

Node -v // 노드의 버전

Nvm init -yes //git의 init 처럼 nvm 실행

nvm install -g eslint //eslint 설치, g를 붙인 것은 하위 포함해서 사용하려면 붙여야 한다.

g를 붙이지 않으면 내 것만 사용하는 것!

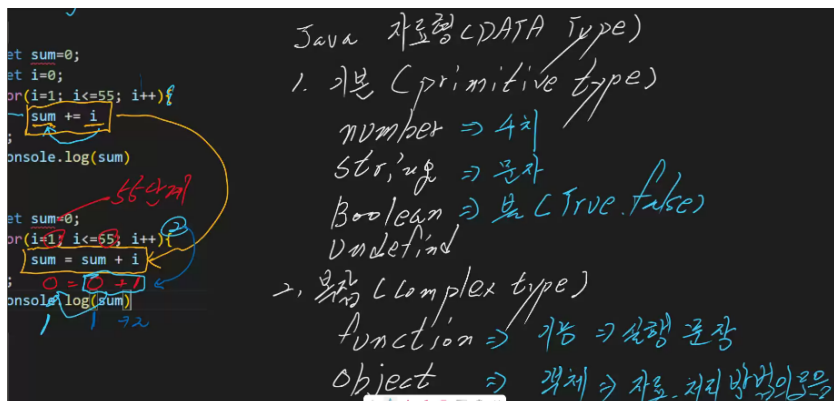


Node 부분 할 때 ctrl+enter 누르면 사용하는 것 변경할 수 있다 -> 노드만 활성화 해주기



node eslint 설치 이후 자바스크립트 안에 node 파일이 생겼다 그 중에 package.json 파일설명!

- Debug -> 오류 났을 때 어떻게 할 것인지
- "test" : "echo -> echo"
- "keywords" : [] -> 태그, 해쉬태그와 같은 것. 프로그램 안내 태그를 붙여주는 자리가 키워드다
- "author" : "" -> 작성자
- "license" : "ISC" -> 라이선스
- "devDependencies" : -> 라이브러리 목록이다 작업 도와주는 틀의 집합체
- "eslint" -> 지금은 eslint 하나만 있다, 나중에 점점 사용하는게 많아지면 늘어날 것



Java 자료형(DATA Type)

1. 기본(primitive type)
  - number => 수치
  - string => 문자
  - boolean => 불대수(true,false)
  - undefined
2. 기본(primitive type) :

컴퓨터 프로그래밍은 조건문에서 반복문 만드는 과정이다

```

a=1;
let b=3;
a = "a";
b = "b";
console.log("a=" +a);
console.log("b=" +b);
console.log(a+b); //

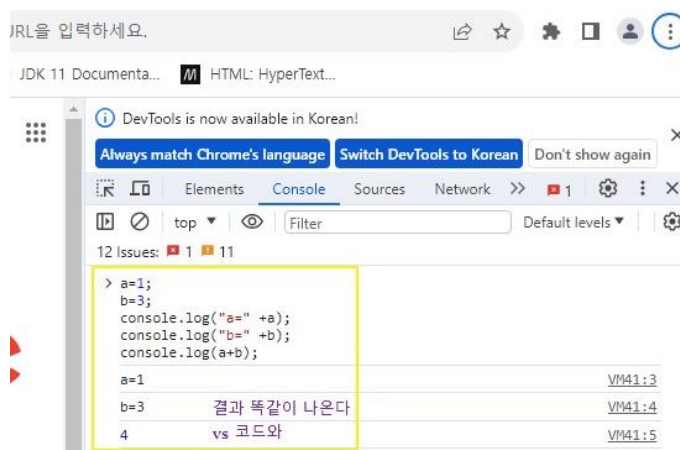
```

a : 변수명, 1 : 변수값

let을 써도 아래가 작동된다. Let을 써주는 것은 고정해주겠다는 것이다 자바스크립트에서 문자는 " 큰따옴표로 묶어준다

console.log(a+b);는 a+b의 값을 터미널의 값에 보여주라는 것

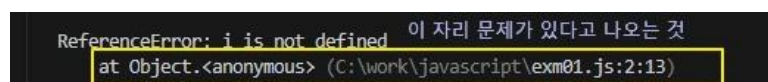
=> 터미널에서 해당 폴더에 node 만든파일명.js 입력해주면 터미널 창에서 값을 확인할 수 있다.



F12 개발자모드에서도 vs코드와 결과 똑같이 나오는 것을 확인할 수 있다.

구글과 크롬에는 자바스크립트가 구현가능하게 작동되어 있다. 프로그램 설치하지 않아도 구글에서 작동되기 때문에 개발자들이 좋아하는 프로그램이다.

(JDK 자바 툴 / JVM 자바 가상머신 / JVR 자바와 자바스크립트 돌아가는 플러그인)



오류가 발생하면 어느 자리에 문제가 발생했는지 알 수 있다 -> eslint로 확인가능한 것.

## <반복문>

For, while, do~while

- ➔ 모든 언어에 다 나옴
- ➔ 컴퓨터 프로그래밍은 반복하는 것을 컴퓨터가 잘하기 때문에 반복하는 것을 시키기 위해서 반복문은 필수이다.
- ➔ 이 반복문을 실행시키기 위해서는 조건문이 필요한데, 그래서 조건문도 필수이다.

```
for (let i = 0; i<10; i++)  
console.log(i)
```

for 이 반복 ( 시작값, 종료값, 증가값 ) / i++는 배후법으로 1씩 더하라는 것(++는 전위법)  
0~9까지 사용하라는 것

java int i=0; 으로 써야 하는데 //int는 자료형 정수 (나중에 자바에서 int, double 이런거 써줄거다)

let i=0; 으로 썼다 //자바스크립트에는 자료형이 존재하지 않는다

일반적으로 i는 줄의 개념으로 쓴다

상수 => 변하지 않는 값

변수 => 변하는 값

```
1 for (let i = 0; i<10; i++)  
2 console.log(i);
```

```
for(let a=0;a<10;++a)  
console.log(a);
```

0	VM160:2
1	VM160:2
2	VM160:2
3	VM160:2
4	VM160:2
5	VM160:2
6	VM160:2
7	VM160:2
8	VM160:2
9	VM160:2

F12개발자모드 콘솔창에서 작업

```
let s="hello papawon";
```

```
for (let i=0;i<s.length; ++i){  
  let ch = s[i];  
  console.log(ch);}
```

```
PS C:\work\javascript> node ex01.js  
h  
e  
l  
l  
o  
  
p  
a  
p  
a  
w  
o  
n
```

### 자바스크립트의 for 문

s는 변수, s.length는 s의 전체문장길이 개수(빈칸포함해서13개), ++i가 하나씩 증가하라고 하는 것  
ch도 변수, s[i] i의 의미 : i의 개수중 s의 몇번째인지 기억해서 출력

s[0]이면 s의 "hello papawon"안에서 0번째인 h를 말한다. (컴퓨터는 0부터 시작)

{ } : 괄호 안에는 실행 내용 작성

자바스크립트 구조가 이렇게 생겼구나 알면 됨

```
public class for{  
  public static void main(String args){  
    string s="hello papawon";  
    for(int i=0; i<s.length();++i){  
      char ch = s.charAt(i);  
      System.out.println(ch);  
    }  
  }  
}
```

### 자바의 for문

예시로 보여준 것. (뒤에가서 할것)

for라고 하는 작동문 main 문자열로 가지고 오겠다는 것, int는 자료형이다

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

**For문** 아래 **while**과 비교해보자!

(아래 while 같이 쓸 때 let을 준 거를 아래도 주면 오류가 나옴)

결과값 : 0~9 까지 하나씩 출력됨

```
let i=0;
while(i<10){
  console.log(i);
  ++i;
}
```

**While문**

결과값 : 0~9 까지 하나씩 출력됨

```
let k=0;
do{
  console.log(k);
  ++k;
} while(k<10);
```

**do~while문**

```
for(i=0;i<100;++i){
  if(i%2==1) continue;
  console.log(i)
  if(i>=20) break;
}
```

let으로 변수지정 안해놔어도 프로그램은 돌아간다.

for -> i는 0부터 100미만까지 i를 한칸씩 증가하면서 진행해라 (% 나누기해서 나머지 값)

if -> i를 2로 나눠서 나머지가 1이면 continue해라, 나머지가 1이아니면 console.log(i)를 실행해서 i값을 출력해라, i가 20보다 크면 break해라.

결과값 : 0,2,4,6,8,10,12,14,16,18,20 짝수의 값이 20까지 출력된다

```
let s1="papawon"; //let을 사용하면 지역변수이다 let을 사용한 공간에서만 작동
const s2="papawon"; //문자열 상에서 바꿀 수 없다.
```

```
console.log(s1);
console.log(s2);
```

**let을 사용하면 지역변수**

## <비교연산자>

> : 크다 초과 / >= : 크거나 같다, 이상 / < : 작다, 미만 / <= : 작거나 같다, 이하  
/ == : 같다 / != : 다르다

a > b : a가 b보다 크다 //기준은 항상 왼쪽이 기준이다.

```
let a=3, b="3"
console.log(a);
console.log(b);
console.log(a==b);
console.log(a===b);
console.log(a!=b);
console.log(a!==b);
```

PS C:\work\javascript\11-02> node exm03.js

```
3
3
true
false
false
true
```

‘악타입언어’이다. 결과 값이 반대로 나옴

==와 ===차이 앞에는 엄격하게 비교하지 않지만 뒤에는 엄격하게 비교한다

!= !== 일반적으로는 앞에것 사용하지만 확실하게 구분해야 할 때는 뒤에것 3개인것 사용한다

```
console.log(typeof a);
console.log(typeof b);
console.log(typeof c);
```

number  
string  
undefined

자료형 찾을 때 console.log(typeof 변수명); 이렇게 쓴다 그러면 결과 값으로 number인지 string인지 나온다, undefined는 값이 지정하지 않았다는 것 (화살표 그 변수 갖다 댈 때 나오기도 하지만 이렇게 찾을 수 있는 방법도 있다.

```
let a=100;
console.log(a);
let a=3.141592;
console.log(a);
```

위의 거는 오류가 난다 이유->let이 두개라서 let을 아래 하나 지워주니까 정상적으로 작동한다.  
Let 두개 쓰니까 무엇을 지정해준건지 혼란이 온다.

```
let a=100;
console.log(a);
a=3.141592;
console.log(a);
```

정수와 실수를 구분하지 않는다

```
let a=3.141592;
console.log(a);
console.log(typeof a);
```

3.141592  
number

a의 타입은 수치이다



```
let s=a.toString
console.log(s);
console.log(typeof s);
```

3.141592  
string

**s=a.toString** : a를 문자열로 바꾸라는 것이다 수치로 나오면 안되는 것들을 이렇게 문자열로 바꿔 준다 (ex\_상품코드)

```
let n = Number(s); //문자열을 수치로 바꿔주는 것이다
console.log(n);
console.log(typeof n);
```

**number(s)**: 문자열이 되었던 n을 다시 수치형으로 바꿔준 것 결과 값은 수치로 노란색으로 표시됨

```
for(let i=0; i<5; ++i ){
    let a = Math.floor(Math.random()*3+1);
    console.log(a);
}
```

랜덤값이 나온다, 난수로 임의의 수 발생시킨 것.

랜덤 함수는 많이 나온다 (ex\_가위바위보)

```
for(let i=0; i<1; ++i ){
    //let a = Math.floor(Math.random()*100 //임의의 수 0~99 까지
    let a = Math.floor(Math.random()*3+1 //임의의 수 0~2 까지 +1 이니까 => 1~3 까지
    console.log(a);
    let b=2;
    console.log(a===b);
```

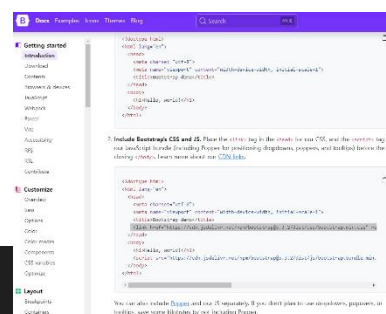
임의의 수가 나온 것을 b=2와 같으면 출력하라고 하는 것

<그 외 정리>

- 조건문 : if와 같이 조건에 따라 다른 값이 주어질 때 사용한다.
- Length의 의미는 길이로, 그 값의 길이를 말해준다.

<부트스트랩>

부트스트랩(bootstrap)



```
<!doctype html>
<html lang="en" data-bs-theme="auto">
  <head><script src="/docs/5.3/assets/js/color-modes.js"></script>

  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Buttons · Bootstrap v5.3</title>
  <!-- 이것이 html할 때 만들 반응형 구조이다. 앞으로 이렇게 만들 것 -->
```

html에서 만들 반응형 구조를 부트스트랩에서 복사해 왔다. 앞으로 이렇게 만들것

<문제>

1.

1 이상 55 이하의 모든 정수를 더해서 화면에 출력하는 코드를 반복문으로 구현하시오.

이상 55 이하의 모든 정수를 더해서 화면에 출력하는 코드 반복문으로 구현

```
let i=0;
let a=0;
for(i=0;i<=55;++i){
    a=i+a; }
console.log(a);
```

```
let b=0;
let k=0;
while(k<=55){
    b=k+b;
    ++k; }
console.log(b);
```

```
h=0; c=0;
do{
    c=c+h;
    ++h;
} while(h<=55);
console.log(c);
```

```
let i=0;
a=0
for(i=0;i<=55;++i){
    a+=i; }
console.log(a);
```

반복문 만들기 여러가지 방법이 있다

정답 :

```
let sum=0;
let i=0;
for(i=1;i<=55;i++){
    sum += i; //i 가 변경될 때마다 sum 에 더한다
}
console.log(sum);
```

```
PS C:\work\javascript> node exm02.js
1540
```

+=이라는 것은 뒤에 i값을 sum에 계속 넣어주겠다는 것

sum+=i 는 sum=sum+1과 똑같다.(문장의 구조가 똑같다는 것, 내가 쓰고 싶은 것으로 쓰기)

sum=0;으로 변수값을 지정해주지 않으면 sum의 값이 나오지 않는다.

sum은 i의 값이 변경될 때마다 sum가 변한다.

2.

1 이상 100 이하의 모든 3의 배수를 더해서 화면에 출력하는 코드를 반복문으로 구현하시오.

If 사용하기!

```
let sum=0;
let i=0;
for(i=1;i<=100;i++){
    if(i%3==0)
        sum += i;
}
console.log(sum);

let sum=0;
let i=0;
for(i=1;i<=100;i++){
    if(i%3==1) continue;
    if(i%3==2) continue;
    sum += i;
}
console.log(sum);
```

sum에 0을 주고, if로 3으로 나뉘서 나머지가 0이면 sum=sum+i를 진행시켜주라는 것  
continue를 해주는 것은 그 값은 그냥 넘어가겠다는 뜻으로 이해했다.

교과서적인 정답 :

```
let sum=0;
let i=0;
for(i=1;i<=100;i++){
    if(i%3==0){
        sum += i;}
}
console.log(sum);
```

if 뒤에 괄호 써주기

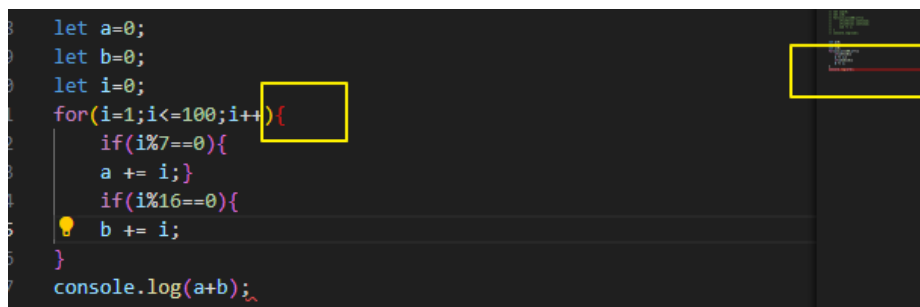
3. 1~100사이의 수중에서 16배수와 7의 배수의 합

```
let a=0;
let b=0;
let i=0;
for(i=1;i<=100;i++){
  if(i%7==0){
    a += i;}
  if(i%16==0){
    b += i;}
}
console.log(a+b);
```

7로 나눠서 나머지가 0인 것을 a에 값을 계속 더해주고

16으로 나눠서 나머지가 0인 것을 b에 값을 계속 더해준다.

그 두개의 값을 더해서(a+b) 나온 것이 7과 16배수의 합이다.



변수명에 빨간줄은 괜찮지만 { 괄호 같은 것 빨간 줄 나오면 오류가 있는 것 오른쪽에도 빨간색으로 뜬다, 괄호 닫기 안해줬다

4.

10 이상 20 이하의 정수를 임의로 30개 출력하는 코드를 구현하시오.

```
for(i=0;i<30;i++){
  let a = Math.floor(Math.random()*11)+10);
  console.log(a);
}
```

\*11+10의 의미 10~20까지인 것

\*10은 0부터 9까지의 랜덤 숫자인데, \*11하면 0~10까지의 숫자가 나온다

\*11에서 각 나오는 숫자에 +10을 하겠다는 것이다. => 1~10까지 나오니까 +10 하면 10~20.

<궁금한 것>

- 전위법, 후위법 ++의 위치

: 하나씩 증가할 때는 상관 없다. 나중에 수가 많이 증가할 때는 위치 상관 있다