node.js 프로젝트 생성

학습목표

javascript 프로젝트 생성

eslint 설정

javascript 반복문

지역변수 선언

목차

[1. 관련 개념 2](#_Toc128947306)

[1) 컴파일러(compiler), 인터프리터(interpreter) 2](#_Toc128947307)

[2) node.js 2](#_Toc128947308)

[3) 터미널(terminal) 3](#_Toc128947309)

[4) git bash 쉘 4](#_Toc128947310)

[5) bash 명령 5](#_Toc128947311)

[6) 파일 경로명 (file path) 6](#_Toc128947312)

[2. hello world 7](#_Toc128947313)

[1) 소스코드 폴더 생성 7](#_Toc128947314)

[2) 소스코드 파일 생성 8](#_Toc128947315)

[3) 터미널 열기 9](#_Toc128947316)

[4) javascript 파일 실행 12](#_Toc128947317)

[5) let 키워드로 변수 선언 13](#_Toc128947318)

[6) var 키워드로 변수 선언 14](#_Toc128947319)

[7) 세미콜론 생략 가능 14](#_Toc128947320)

[3. eslint 15](#_Toc128947321)

[1) eslint란? 15](#_Toc128947322)

[2) eslint 설치 15](#_Toc128947323)

[3) visual studio code 확장 설치 16](#_Toc128947324)

[4. node.js 프로젝트 설정 파일 17](#_Toc128947325)

[1) package.json 파일 생성 17](#_Toc128947326)

[2) eslint 설정 파일 생성 18](#_Toc128947327)

[3) 프로젝트 폴더 20](#_Toc128947328)

[5. 반복문 22](#_Toc128947329)

[1) chap02 폴더 생성 22](#_Toc128947330)

[2) chap02 폴더 아래 파일 생성 22](#_Toc128947331)

[3) for 문 23](#_Toc128947332)

[4) while 루프 25](#_Toc128947333)

[5) do while 루프 25](#_Toc128947334)

[6) break, continue 26](#_Toc128947335)

[6. 변수 선언 27](#_Toc128947336)

[1) 변수 선언 생략 27](#_Toc128947337)

[2) "use strict"; 28](#_Toc128947338)

[3) 상수 선언 const 29](#_Toc128947339)

[7. 내용 정리 30](#_Toc128947340)

[8. 과제 32](#_Toc128947341)

[1) 합계1 (sum1.js) 32](#_Toc128947342)

[2) 합계2 (sum2.js) 32](#_Toc128947343)

# 관련 개념

## 컴파일러(compiler), 인터프리터(interpreter)

C는 컴파일 방식의 프로그래밍 언어이다.

컴파일러는 소스 코드를 실행 파일로 변환한다.

실행 파일만 있으면 실행이 되기 때문에, 소스 코드를 배포할 필요 없다.

javascript 언어는 인터프리터 방식의 프로그래밍 언어이다.

인터프리터는 소스 코드를 실행 파일로 변환하지 않는다.

인터프리터는 소스 코드를 한 줄 씩 읽으며 실행하기 때문에,

실행할 때 마다 소스 코드 파일이 필요하다.

## node.js

javascript 언어는 인터프리터(interpreter) 방식으로 실행된다.

javascript 인터프리터가 javascript 소스코드를 읽어서 실행한다.

javascript 인터프리터는 웹브라우저에 내장되어 있다.

그렇기 때문에 웹브라우저에서 javascript 소스 코드가 실행될 수 있다.

javascript 소스 코드를 서버에서 실행하기 위한 엔진이 node.js 이다.

당연히 node.js 에도 javascript 인터프리터가 내장되어 있다.

가장 유명한 javascript 인터프리터는

구글이 만든 chrome v8 javascript 인터프리터이다.

이 javascript 인터프리터가 chrome 웹브라우저에 내장되어 있고,

node.js 에도 내장되어 있다.

node.js는, javascript로 구현된 코드를 서버에서 실행하기 위한 플랫폼이다.

플랫폼 = 실행 엔진(인터프리터) + 라이브러리 + 개발도구

node.js 플랫폼에서 실행되는 웹 서버, REST API 서버, 채팅 서버를

javascript 언어로 어렵지 않게 구현할 수 있다.

## 터미널(terminal)

Windows에서 명령 프롬프트(command prompt) 창에 해당하는 것을

Linux 에서는 터미널(terminal) 이라고 부른다.

visual studio code 에서도 이 창을 터미널이라고 부른다.

### Windows의 명령 프롬프트 창

Text

Description automatically generated

### Linux의 터미널 창

A screenshot of a computer

Description automatically generated with medium confidence

### 쉘(shell)

**명령 프롬프트 창**에서 입력된 명령을 실행하는 프로그램을 쉘(shell)이라고 한다.

Windows의 기본 쉘은 cmd.exe 이다.

Windows의 고급 쉘은 powershell.exe 이다.

Linux의 기본 쉘은 bash 이다.

Mac의 기본 쉘은 bash 이었는데, 최근에 zsh로 바뀌었다. (M1 맥북 개발도구 설치.docx 강의노트 참고)

Windows 에서도 bash 쉘을 설치하여 사용할 수 있다.

Git 클라이언트를 다운로드하여 설치하면,

Windows 용 bash 쉘이 설치된다.

Windows, Linux, Mac에서 bash 쉘을 사용하는 것을 권장한다.

## git bash 쉘

Wingows git 클라이언트를 설치하면 git bash 쉘도 함께 설치된다.

Graphical user interface, application

Description automatically generated

Shape, rectangle

Description automatically generated

그런데 git bash 쉘이 실행되고 있는 이 터미널 창에 약간에 문제가 있다.

가끔 화살표 키가 잘 작동하지 않는다.

그래서 가급적 Visual Studio Code 터미널 창에서 git bash 쉘을 실행하는 것을 권장한다.

## bash 명령

디렉토리는 폴더와 같은 개념이다.

### 디렉토리 생성

|  |
| --- |
| **mkdir 디렉토리명** |

### 하위 디렉토리로 이동하기

|  |
| --- |
| **cd 디렉토리명** |

### 현재 디렉토리의 파일 목록 출력

|  |
| --- |
| **ls -al** |

### 상위 디렉토리로 이동하기

|  |
| --- |
| **cd ..** |

### 디렉토리 삭제하기

|  |
| --- |
| **rm -rf 디렉토리명** |

디렉토리와 그 아래 파일들이 모두 삭제된다.

### 파일 삭제하기

|  |
| --- |
| **rm 파일명** |

### 파일 복사하기 #1

|  |
| --- |
| **cp 소스\_파일 새\_파일** |

### 파일 복사하기 #2

|  |
| --- |
| **cp 소스\_파일 대상\_폴더** |

## 파일 경로명 (file path)

### Windows

Windows 운영체제의 파일시스템에는 디스크 드라이브 개념이 있다. (예: C: D: E:)

Linux 에는 디스크 드라이브 개념이 없다.

Windows 에 설치한 Git bash 쉘에서, Windows 파일시스템의 디스크 드라이브를 다음과 같이 다룬다.

예를 들어 Windows 에서 파일 경로명이 다음과 같을 때

|  |
| --- |
| **C:**/PJ/node1/chap01/hello1.js |

위 경로명에 해당하는 **Git bash 경로명**은 다음과 같다.

|  |
| --- |
| **/c**/PJ/node1/chap01/hello1.js |

**Git bash**에서 C:/PJ/node1/ 디렉토리로 이동하는 명령은 다음과 같다.

|  |
| --- |
| cd /c/PJ/node1/ |

### Mac

맥에서 사용자 홈 디렉토리 위치는

/Users/계정명

이 홈 디렉토리 아래에 적당한 폴더를 만들고 그 아래 프로젝트 폴더를 만들자.

# hello world

## 소스코드 폴더 생성

소스코드들을 저장할 적당한 폴더를 만들자

**예: c:/PJ/frontend/node1**

visual studio code에서 소스코드 폴더를 열자

### 소스코드 폴더 열기 방법 #1

메뉴: File - Open Folder

### 소스코드 폴더 열기 방법 #2

A picture containing graphical user interface

Description automatically generated

## 소스코드 파일 생성

### 소스코드 파일 생성

Text

Description automatically generated

소스코드를 저장할 폴더(예: node1)을 선택하고,

파일 생성(New File) 버튼을 클릭하자.

### 폴더 생성

위 화면에서 폴더 생성 버튼을 클릭하여, node1 폴더 아래에 chap01 폴더를 만들자.

### 파일 생성

Graphical user interface, text, application

Description automatically generated

chap01 폴더를 선택하고, 파일 생성 버튼을 클릭하여, chap01 폴더 아래에 hello1.js 파일을 만들자.

Graphical user interface, text, application

Description automatically generated

### hello1.js

|  |
| --- |
| console.log("hello world"); |

console 객체의 log 메소드는, 화면(console)에 값을 출력한다.

Java의 System.out.println 메소드와 기능이 같다.

javascript 소스 코드에는 main 함수가 필요없다.

javascript 소스코드 파일을 실행하면, 파일의 소스 코드들이 한 줄씩 순서대로 실행된다.

### 파일 저장

단축키 Ctrl+S

## 터미널 열기

node.js는 기본적으로 명령 프롬프트 창의 쉘(shell)에서 실행된다.

visual studio code 에서 명령 프롬프트 창을 열자. (터미널을 열자)

### 터미널 열기

메뉴: Terminal - New Terminal

Graphical user interface, application

Description automatically generated

위 화면의 (2)를 클릭하여, 터미널에서 사용할 쉘(shell)을 선택하자.

### 디폴트 쉘 선택

터미널에서 사용할 쉘을 선택하자.

A screenshot of a computer

Description automatically generated

(1) Select Default Profile 클릭

(2) 선택 가능한 쉘 목록

Command Prompt : Windows의 기본 쉘

PowerShell : Windows의 고급 쉘

Git Bash : Git 클라이언트에 포함된 bash 쉘

오픈 소스 개발자들이 사용하는 운영체제는 리눅스이다.

실력있는 개발자라면 리눅스의 bash 쉘에 익숙해져야 한다.

Git Bash 항목을 선택하자.

### 터미널 다시 열기

Graphical user interface, text, application, email

Description automatically generated

터미널 창 위의 휴지통 버튼을 클릭하여 터미널 창을 닫자.

그리고 다시 터시널 창을 새로 열자 (단축키: Ctrl+`)

새로 열린 터미널 창에서 bash 쉘이 실행될 것이다.

현재 디렉토리의 파일 목록을 출력하는 리눅스 명령은 다음과 같다. (디렉토리 == 폴더)

|  |
| --- |
| ls -al |

|  |
| --- |
| Text  Description automatically generated |

현재 디렉토리 아래의 chap01 디렉토리로 들어가는 명령은 다음과 같다.

|  |
| --- |
| cd chap01 |

|  |
| --- |
| Text  Description automatically generated |

## javascript 파일 실행

### 쉘에서 실행

자바스크립트 파일을 쉘에서 실행하기 위한 명령은 다음과 같다.

|  |
| --- |
| node 파일명 |

hello.js 파일이 있는 chap01 디렉토리로 이동하자

|  |
| --- |
| cd chap01 |

hello.js 파일 실행

|  |
| --- |
| node hello.js |

|  |
| --- |
| Text  Description automatically generated with low confidence |

화면에 hello world 문자열이 출력되었다.

### 터미널 창 감추기

Ctrl+` 단축키를 계속해서 누르면, 터미널 창이 보였다 감춰졌다 한다.

쉘이 종료되는 것은 아니다.

터미널 창 위의 휴지통 버튼을 클릭하면, 터미널 창이 닫히고 쉘이 종료된다.

## let 키워드로 변수 선언

### 파일 생성

Graphical user interface, text, application

Description automatically generated

chap01 폴더를 선택하고, 파일 생성 버튼을 클릭하여, chap01 폴더 아래에 hello2.js 파일을 만들자.

### hello2.js

|  |
| --- |
| let message = "hello world";  console.log(message); |

let 키워드를 사용하여, 지역 변수를 선언한다.

javascript 언어에서는 변수를 선언할 때 변수의 타입을 지정하지 않는다.

let message = "hello world";

messsage 지역 변수에 "hello world" 문자열이 대입된다.

### hello2.js 실행

|  |
| --- |
| A picture containing logo  Description automatically generated |

## var 키워드로 변수 선언

### hello3.js

|  |
| --- |
| var message = "hello world";  console.log(message); |

출력

|  |
| --- |
| hello world |

var 키워드를 사용하여 변수를 선언할 수도 있다.

var 키워드로 선언한 변수는, let 키워드로 선언한 변수와 미묘한 차이점이 있다.

이 차이를 이해하는 것은 너무 고급 문법이라서 여기서 설명하지 않겠다.

기억해야할 요점은,

지역 변수를 선언할 때 var 이 아니고 let 을 사용해야 한다는 것.

## 세미콜론 생략 가능

### hello4.js

|  |
| --- |
| let message = "hello world"  console.log(message) |

출력

|  |
| --- |
| hello world |

javascript 문법에서는, 줄 끝의 세미콜론(;) 문자를 생략할 수 있다.

줄 중간에 있는 세미콜론은 생략할 수 없다.

# eslint

## eslint란?

오타 같은 실수에서 컴파일 에러가 발생해야 버그 찾기가 수월하다.

그런데 javascript는 인터프리터 방식의 언어라서 컴파일을 하지 않는다.

컴파일을 하지 않기 때문에, 컴파일 에러가 발생하지도 않는다.

소스코드에 오타가 있어도, 컴파일 에러가 발생하지 않고,

실행할 때 런타임 에러(runtime error)가 발생한다.

에러를 찾기 위해, 모든 기능을 다 실행해 보는 것은 힘들기 때문에,

컴파일 방식 언어보다 인터프리터 방식 언어는 버그 찾기가 힘들다.

그래서 인터프리터 언어는(스크립트 언어는) 큰 프로젝트에 적당하지 않고,

간단한 기능을 빠르게 개발하기에 적당하다.

이 불편함을 보완하는 도구가 eslint 이다.

eslint는 컴파일러처럼 소스코드를 분석해서 오타와 같은 문법 오류를 찾아준다.

바람직하지 않은 코딩 스타일도 지적해 준다.

## eslint 설치

### eslint 설치 명령

|  |
| --- |
| npm install -g eslint |

Windows 명령 프롬프트

Text

Description automatically generated

Mac 터미널

Text

Description automatically generated

## visual studio code 확장 설치

### visual studio code에 eslint 확장 설치

Graphical user interface

Description automatically generated

### Prettier - Code formatter 확장 설치

Graphical user interface, application

Description automatically generated

# node.js 프로젝트 설정 파일

## package.json 파일 생성

node.js 프로젝트의 설정 파일은 package.json 파일이다.

이 파일이 프로젝트 폴더에 있어야 한다.

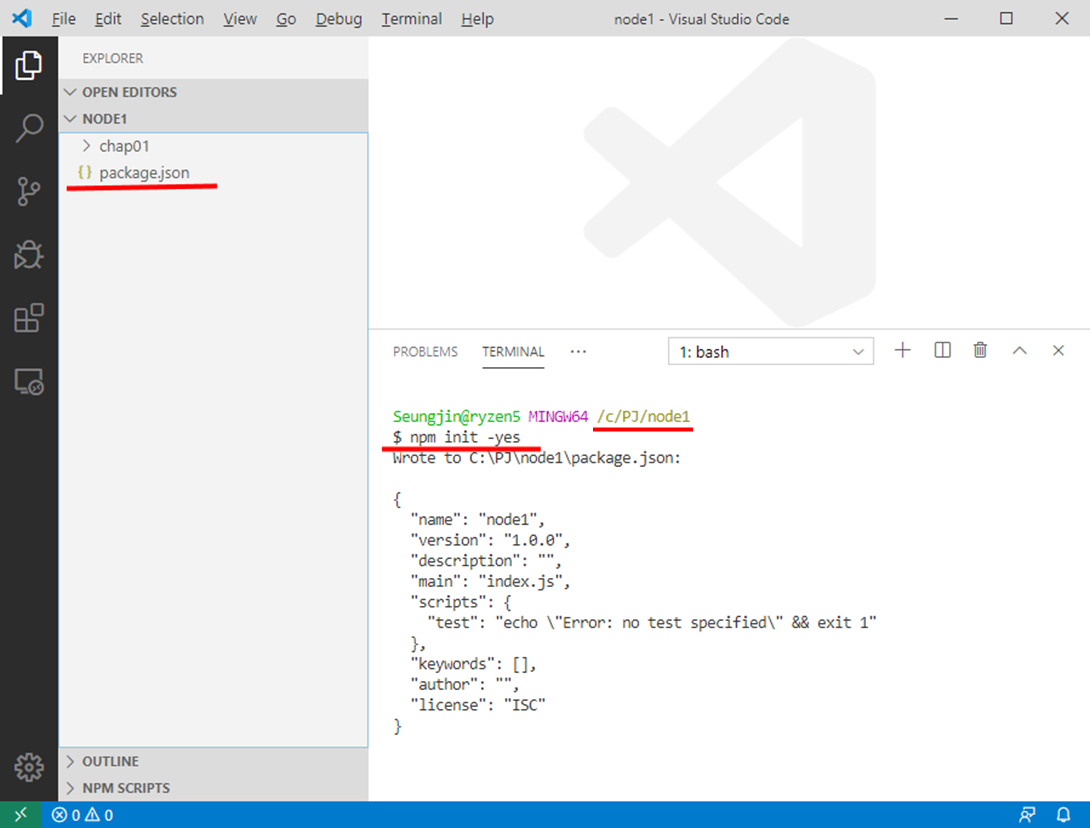
프로젝트 디렉토리에 package.json 파일을 생성하자.

### package.json 생성

프로젝트 디렉토리에서 아래 명령을 실행해야 한다.

|  |
| --- |
| npm init -yes |

node1 프로젝트 폴더에 package.json 파일이 생성된다.



## eslint 설정 파일 생성

프로젝트 폴더 에 eslint 설정 파일을 추가하자.

eslint 설정 파일은 .eslintrc 이다.

### 프로젝트에 eslint 설정 파일(.eslintrc) 추가 명령

주의: node1 프로젝트 폴더에서 아래 명령을 실행해야 한다.

|  |
| --- |
| npm init @eslint/config |

|  |
| --- |
|  |

y 엔터

|  |
| --- |
|  |

위 아래 화살표 키를 눌러서 'To check syntax and find problems' 항목을 선택하고

엔터키 입력

check syntax : 문법 오류를 검사한다

find problems : 흔한 실수를 검사한다

enforce code style: 코드 스타일을 검사한다

|  |
| --- |
|  |

Javascript modules (import/export) 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

None of these 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

No 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

Node 항목을 선택하고 엔터키 입력

Browser : 이 프로젝트의 코드가 웹브라우저에서 실행될 것이라면 선택

Node : 이 프로젝트의 코드가 node.js에서 실행될 것이라면 선택

|  |
| --- |
|  |

JSON 항목을 선택하고 엔터

(생성될 eslint 설정 파일의 포맷을 지정한다)

|  |
| --- |
|  |

Yes 항목을 선택하고 엔터키 입력

|  |
| --- |
|  |

npm 항목을 선택하고 엔터키 입력

위 명령에 의해서, eslint의 설정 파일인 .eslintrc.json 파일 생성된다.

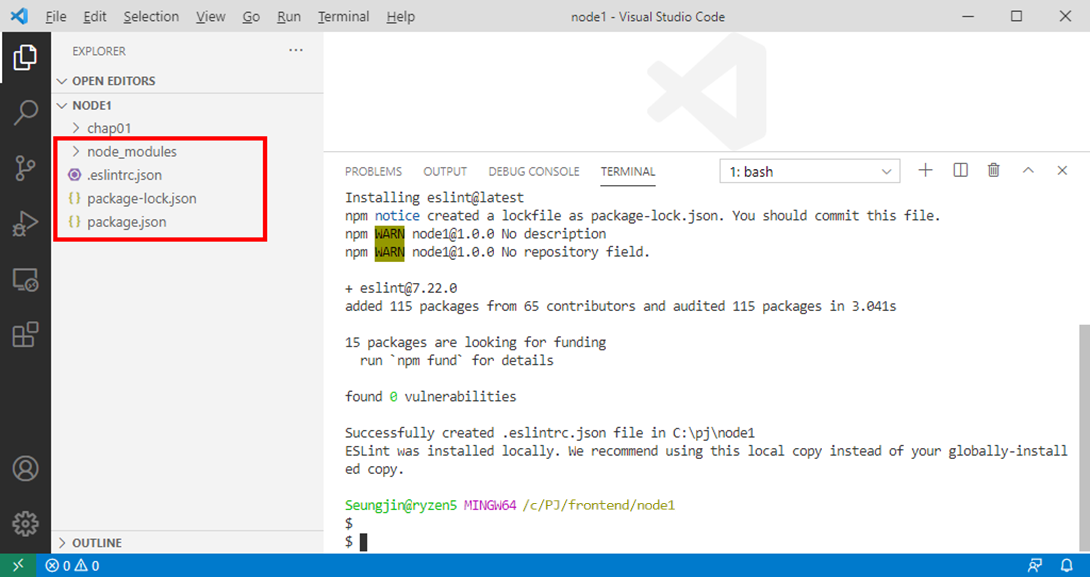
## 프로젝트 폴더

node1 프로젝트 폴더의 내용

- node\_modules 폴더

- package.json 파일

- .eslintrc.json 파일



### package.json 파일의 내용

node.js 프로젝트 설정 파일

|  |
| --- |
| {    "name": "node1",    "version": "1.0.0",    "description": "",    "main": "index.js",    "scripts": {      "test": "echo \"Error: no test specified\" && exit 1"    },    "keywords": [],    "author": "",    "license": "ISC",    "devDependencies": {      "eslint": "^8.35.0"    }  } |

|  |  |
| --- | --- |
| 항목 | 설명 |
| name | 프로젝트 이름 |
| version | 프로젝트 버전 |
| description | 프로젝트 설명 |
| main | 프로젝트의 실행이 시작될 첫 파일 |
| scripts | 프로젝트를 빌드, 테스트, 실행하기 위한 명령 |
| keywords | 프로젝트의 검색 키워드 |
| author | 작성자 |
| license | 라이센스 정보 |
| devDependencies | 프로젝트에 필요한 라이브러리들 목록 |

### node\_modules 폴더

프로젝트에 필요한 라이브러리들이 자동으로 저장되는 폴더

### .eslintrc.json 파일

eslint 설정 파일

### package-lock.json

프로젝트에 필요한 라이브러리들의 정확한 버전이 자동으로 등록되는 파일.

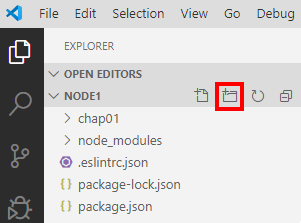
이 파일은 자동으로 생성된다.

지금은 이 파일을 무시하자.

# 반복문

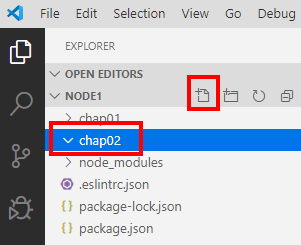
## chap02 폴더 생성

node1 프로젝트 폴더 아래에 chap02 폴더를 만들자.



node1 폴더를 클릭하고, 폴더 생성 버튼을 클릭하자.

## chap02 폴더 아래 파일 생성



chap02 폴더를 클릭하고, 파일 생성 버튼을 클릭하자.

## for 문

### for1.js

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

지역 변수 선언 키워드가 let 임에 주의하자.

javascript 언어는 변수를 선언할 때, 변수의 타입을 지정하지 않는다.

**int i = 0;** 문법 오류

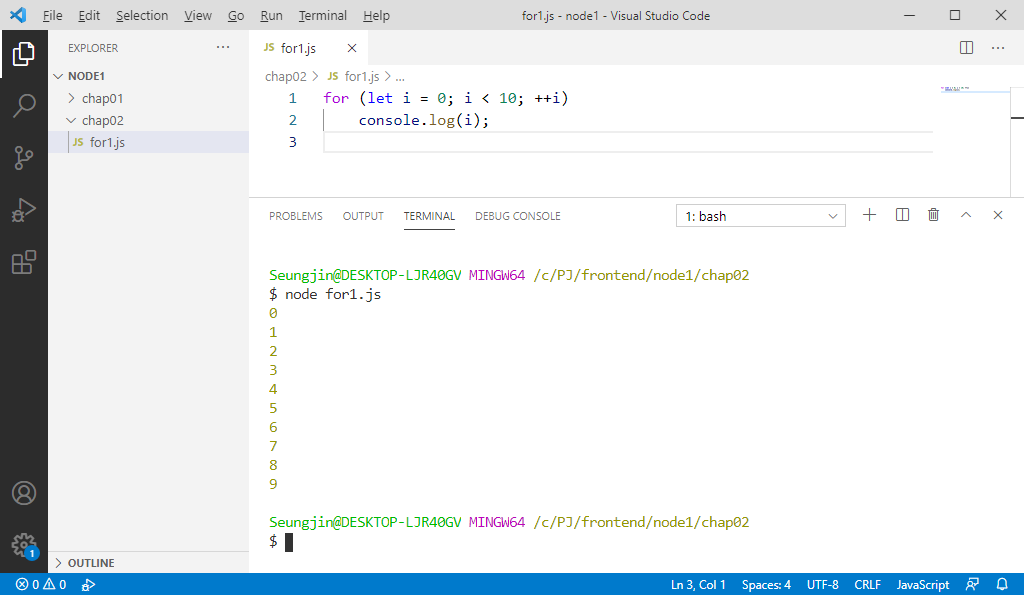
**let i = 0;** Ok.

**String s = "hello";** 문법 오류

**let s = "hello";** Ok

let을 제외하면 for 문의 문법은 java와 동일하다.

### 실행



### for2.js

|  |
| --- |
| let s = "hello world";  for (let i = 0; i < s.length; ++i) {      let ch = s[i];      console.log(ch);  } |

let s = "hello world";

지역변수 s에 "hello world" 문자열이 대입된다.

문자열은 객체이다.

s.length

문자열 객체의 length 속성의 값은 문자열의 길이이다.

let ch = s[i];

s 문자열의 인덱스 i 위치 문자를, 지역변수 ch 에 대입한다.

실행

|  |
| --- |
|  |

### 참고 : For2.java

이것은 java 코드이다. (java 코드 실행은 이클립스에서...)

|  |
| --- |
| public class For2 {  public static void main(String[] args) {  String s = "hello world";  for (int i = 0; i < s.length(); ++i) {  char ch = s.charAt(i);  System.out.println(ch);  }  }  } |

for2.js 코드와 For2.java 코드를 비교하자.

Java 는 main 메소드에서 실행이 시작된다.

Java String 클래스의 length() 메소드의 리턴값은, 문자열의 길이이다.

Java String 클래스의 charAt(int index) 메소드의 리턴값은, 인덱스 i 위치 문자이다.

## while 루프

### while1.js

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

while 문의 문법은 java와 동일하다.

실행

|  |
| --- |
|  |

## do while 루프

### do1.js

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

do while 문의 문법은 java와 동일하다.

실행

|  |
| --- |
|  |

## break, continue

반복문에서 break, continue 는 java 문법과 동일하다.

### loop1.js

|  |
| --- |
| let i = 0;  for (;;) {      ++i;      if (i % 2 == 1) continue;      console.log(i);      if (i >= 20) break;  } |

실행

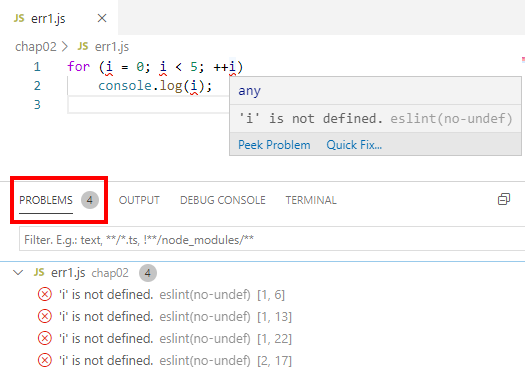
|  |
| --- |
|  |

# 변수 선언

## 변수 선언 생략

### err1.js

|  |
| --- |
| for (i = 0; i < 5; ++i)      console.log(i); |



위의 visual studio code 화면에서 변수 i 에 빨간색 밑줄이 표시되었다.

빨간색 밑줄 위에 마우스 포인터를 위치하고 기다리면,

위 화면과 같은 에러 메시지가 표시된다.

'i' is not defined.

변수 i 가 정의되지 않았다는 에러 메시지이다.

그런데 실행해 보면, 에러 없이 실행이 잘 된다.

|  |
| --- |
|  |

### 변수 선언을 생략해도 된다.

위 err1.js 에서 변수 i 를 선언하지 않고, 그냥 i = 0; 를 실행했다.

이것은 문법 오류가 아니다.

javascript 언어 문법에서는, 변수 선언을 생략해도 된다. 필수가 아니다.

i = 0; 문장을 처음 실행할 때, 변수 i 가 자동으로 생성되기 때문이다.

즉 javascript 문법에서는, 변수를 미리 선언하지 않아도, 어떤 변수에 값이 처음 대입될 때,

그 변수가 자동으로 생성된다.

### eslint 경고 메시지

위 화면에서, 변수 i 가 정의되지 않았다는 에러 메시지는,

javascript 문법 오류가 아니고, eslint 도구의 경고 메시지이다.

코딩 스타일이 바람직하지 않다는 경고 메시지이다.

변수를 미리 선언하지 않고 사용하는 것은 바람직하지 않기 때문이다.

## "use strict";

javascript 소스 코드 파일 선두에 **"use strict";** 선언이 있으면,

이 소스 파일에서는 변수를 미리 선언하지 않고 사용할 경우에 문법 오류가 발생한다.

### err2.js

|  |
| --- |
| "use strict";  for (i = 0; i < 5; ++i)      console.log(i); |

|  |
| --- |
|  |

err2.js를 실행하면, 위 화면에서처럼 문법 오류 메시지가 출력되고, 실행되지 않는다.

## 상수 선언 const

### const1.js

|  |
| --- |
| let s1 = "hello world";  const s2 = "hello world";  console.log(s1);  console.log(s2); |

const 키워드를 사용하여 상수를 선언한다.

s1 = "안녕하세요";

지역변수 s1의 값은 변경할 수 있다.

s2 = "안녕하세요";

const 변수의 값은 변경할 수 없으므로, 에러가 발생한다.

# 내용 정리

### 디렉토리 생성

mkdir 디렉토리명

### 하위 디렉토리로 이동하기

cd 디렉토리명

### 현재 디렉토리의 파일 목록 출력

ls -al

### 상위 디렉토리로 이동하기

cd ..

### 디렉토리 삭제하기

rm -rf 디렉토리명

### 파일 삭제하기

rm 파일명

### 파일 복사하기 #1

cp 소스\_파일 새\_파일

### 파일 복사하기 #2

cp 소스\_파일 대상\_폴더

### Git bash에서 D:/pj 디렉토리로 이동하는 명령

cd /d/pj

### javascript 프로젝트 설정 파일

package.json

### javascript 프로젝트 설정 파일 생성 명령

npm init

### eslint

eslint는 컴파일러처럼 소스코드를 분석해서 오타와 같은 문법 오류를 찾아준다.

바람직하지 않는 코딩 스타일도 지적해 준다.

### eslint 설정 파일

.eslintrc.json

### eslint 설정 파일 생성 명령

eslint --init

### node\_modules 폴더

프로젝트에 필요한 라이브러리들이 자동으로 저장되는 폴더

### 변수 선언

javascript 문법에서는, 변수를 미리 선언하지 않고 사용해도 된다.

### "use strict";

javascript 소스 코드 파일 선두에 **"use strict";** 선언이 있으면, 변수 선언이 필수이다.

즉 변수를 미리 선언하고 사용해야 한다.

### 문자열.length

문자열의 길이

### 문자열[i]

문자열에서 인덱스 i 위치 문자

### const

const 키워드를 사용하여 상수를 선언한다.

# 과제

## 합계1 (sum1.js)

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

## 합계2 (sum2.js)

1 이상 55 이하의 모든 3의 배수를 더해서 화면에 출력하는 코드를 반복문으로 구현하시오.