

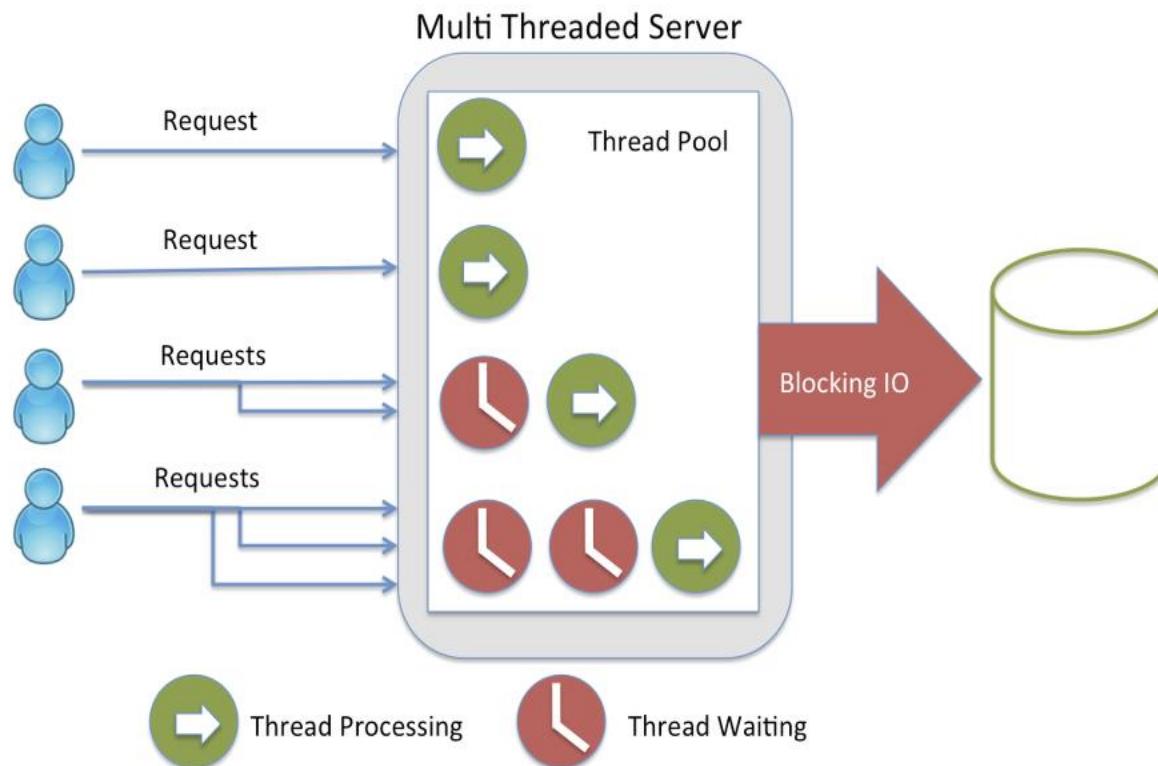
Node js

1. Node.js란

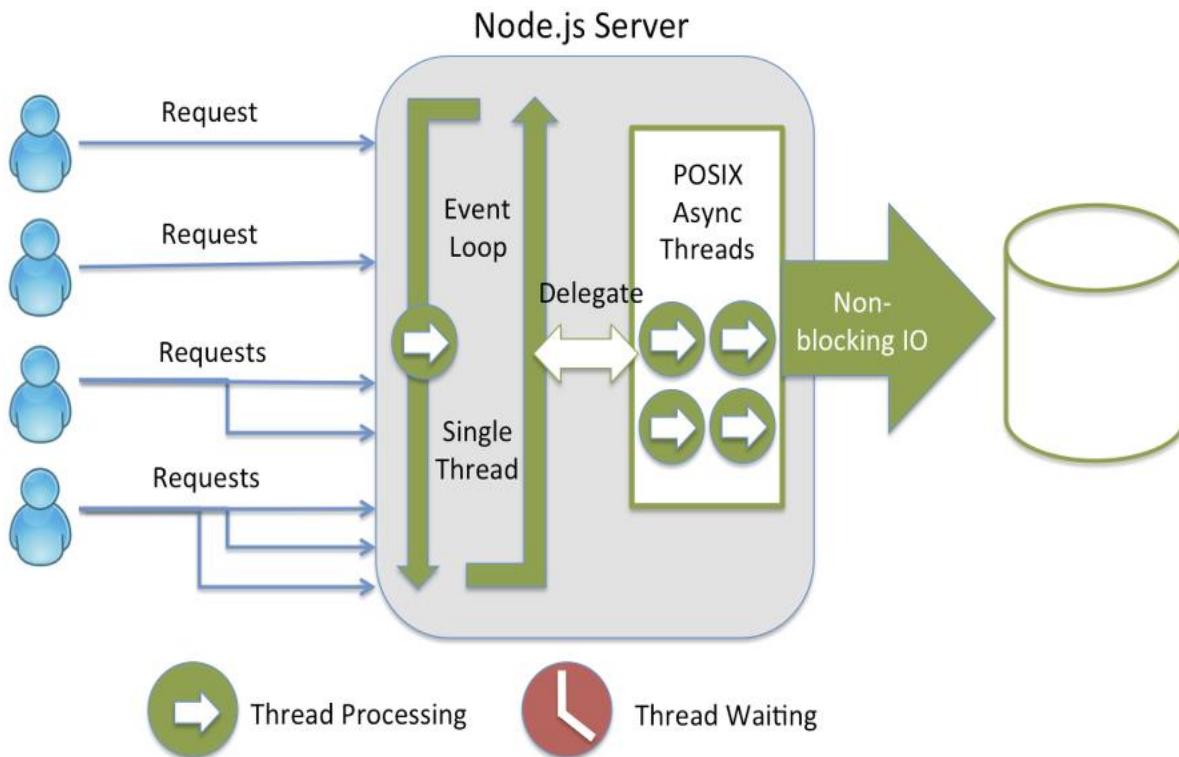
Node.js 소개

- Node.js의 js는 Javascript이다 !
- Node.js는 서버사이드 개발에 사용되는 소프트웨어 플랫폼이다.
- 작성 언어는 Javascript이며, 많은 module, library들이 제공된다
- 전 세계 곳곳에서 현재도 개발중이기 때문에 정보를 얻기가 굉장히 쉽다
- Node.js 의 구조적인 장점은, 비동기식 단일 쓰레드 서버라는 것이다.

동기식 다중 쓰레드 서버



비동기식 단일 쓰레드 서버

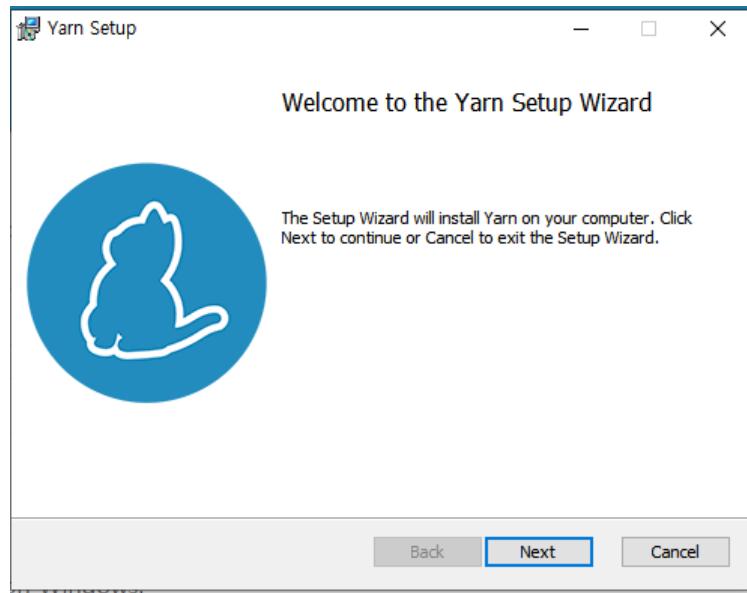


설치해야 할 것들

- Node.js: Webpack 과 Babel 같은 도구들이 자바스크립트 런타임인 Node.js 를 기반으로 만들어져있습니다. 그렇기에 해당 도구들을 사용하기 위해서 Node.js 를 설치합니다.
- Yarn: Yarn 은 조금 개선된 버전의 npm 이라고 생각하시면 됩니다. npm 은 Node.js 를 설치하게 될 때 같이 딸려오는 패키지 매니저 도구입니다. 프로젝트에서 사용되는 라이브러리를 설치하고 해당 라이브러리들의 버전 관리를 하게 될 때 사용하죠. 우리가 Yarn 을 사용하는 이유는, [더 나은 속도, 더 나은 캐싱 시스템](#)을 사용하기 위함입니다.(맥사용자)
- 코드 에디터: 그리고, 코드 에디터를 준비하세요. 여러분이 좋아하는 에디터가 있다면, 따로 새로 설치하지 않고 기존에 사용하시던걸 사용하셔도 됩니다. 저는 주로 VSCode 를 사용합니다. 이 외에도, Atom, WebStorm, Sublime 같은 훌륭한 선택지가 있습니다.
- 윈도우의 경우, [Git for Windows](#) 를 설치해서 앞으로 터미널에 무엇을 입력하라는 내용이 있으면 함께 설치되는 Git Bash 를 사용하세요.

yarn 설치

- <https://yarnpkg.com/lang/en/docs/install/#windows-stable>

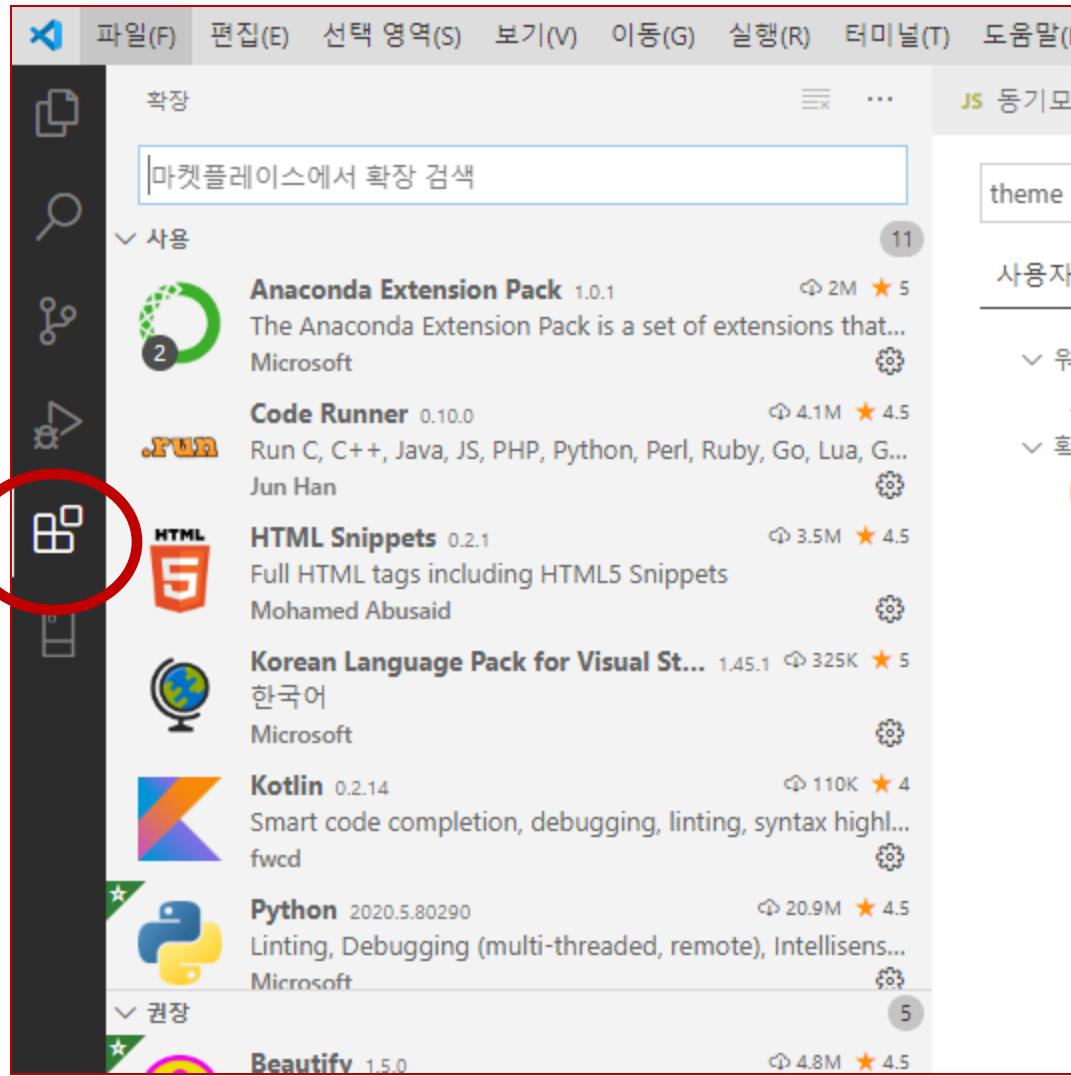


Visual studio code 설치

- 파일 – 기본설정 - 설정 – zoom 화면 크기 설정하기
- 파일 – 기본설정 - 설정 - theme
- 자동설정 체크
-

확장팩 설치하기

여기로 누르세요

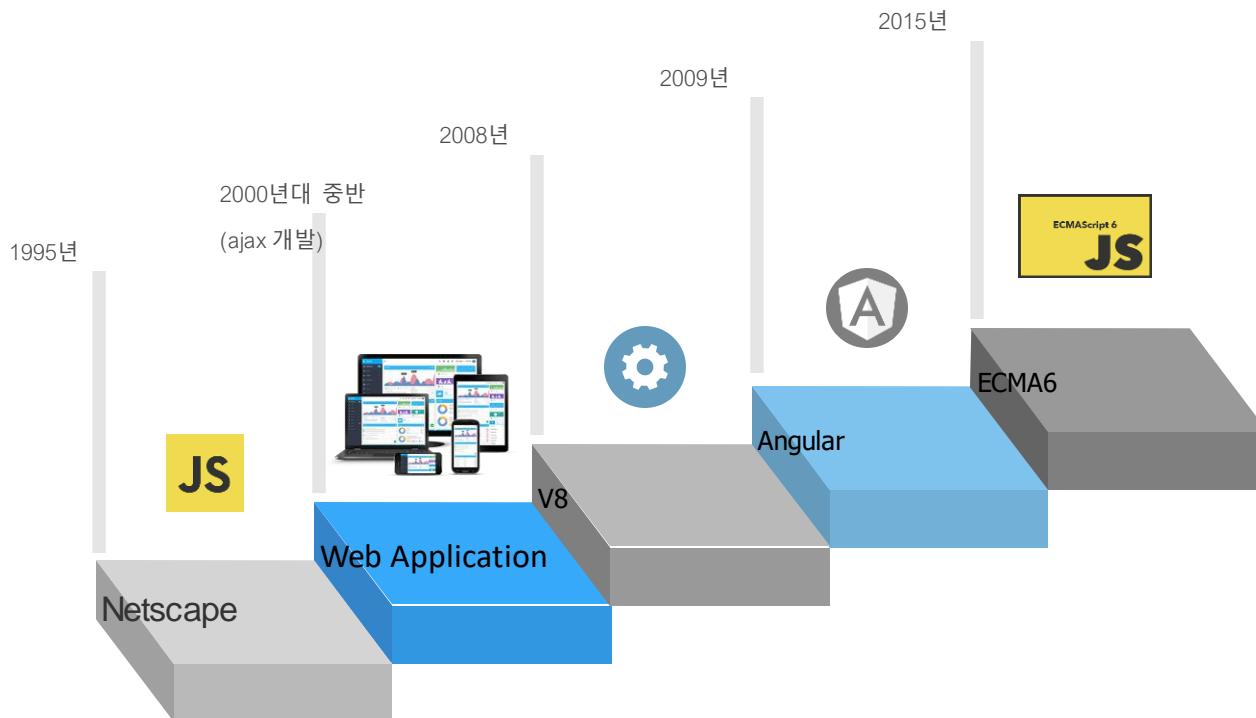


확장팩 설치하기

- **HTML Snippets**
- **TypeScript JavaScript and TypeScript Nightly**
-

typescript

웹의 발전



ECMA 스크립트에 대하여

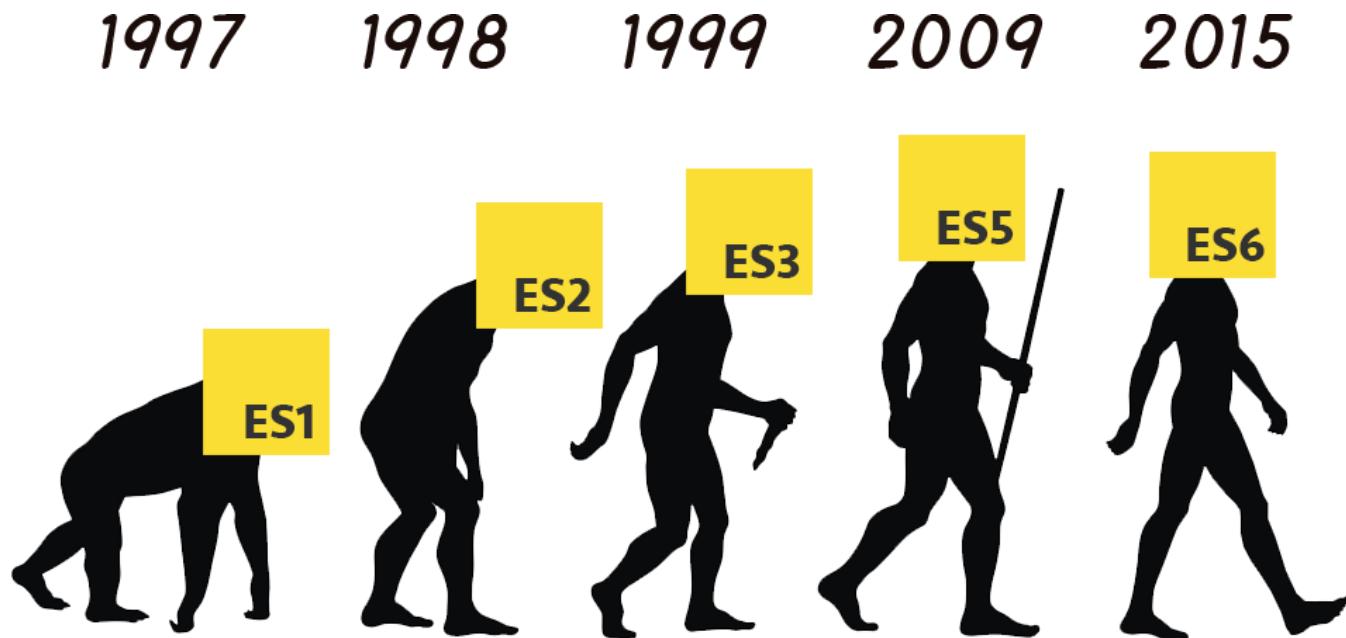
ECMA 스크립트는 ECMA International의 표준

최초 ECMA 스크립트는 브라우저 언어인 Javascript와 Jscript간 차이를 줄이기 위한 공통 스펙 제안으로 출발 하였다(1997, ECMA-262)

ECMA International은 전세계적인 표준기관으로 유럽 컴퓨터 제조협회로부터 기원하였다
ECMA(European Computer Manufacturers Association)

C#, JSON, Dart을 포함한 많은 언어표준을 관리함

ECMA Script 의 진화



ECMAScript 2015, 2016 소개

- Modules - 언어레벨에서 지원
- Classes - 프로토타입에서 클래스로
- Arrow Functions (=>) - 람다식 문법 지원
- Let and Const - 새로운 변수 선언 키워드
- Default, Rest, Spread - 함수 파라미터 세 가지 새로운 기능

www.typescriptlang.org



TypeScript의 역사

2012년 10월 첫 타입스크립트 버전 0.8 발표

2013년 6월 18일 타입스크립트 버전 0.9 발표

2014년 2월 25일 Visual Studio 2013 빌트인
지원

2014년 4월 2일 타입스크립트 1.0 발표

2014년 7월 타입스크립트 컴파일러 발표,
Github 이전

2016년 5월 타입스크립트 2.0 발표

2017년 타입스크립트 2.3

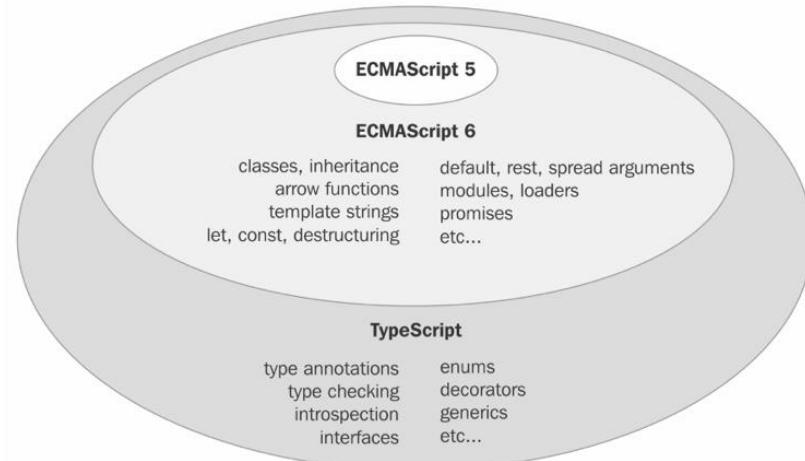
TypeScript 소개

TypeScript는 마이크로소프트(Anders Hejlsberg)에서 개발한 자바스크립트의 확장된 언어이며, ES2015의 기능에 추가적으로 엄격한 타입체크와 객체지향적 기능 등을 추가한 자바스크립트의 Superset이다.

Angular 2 소스는 TypeScript로 개발되었다.

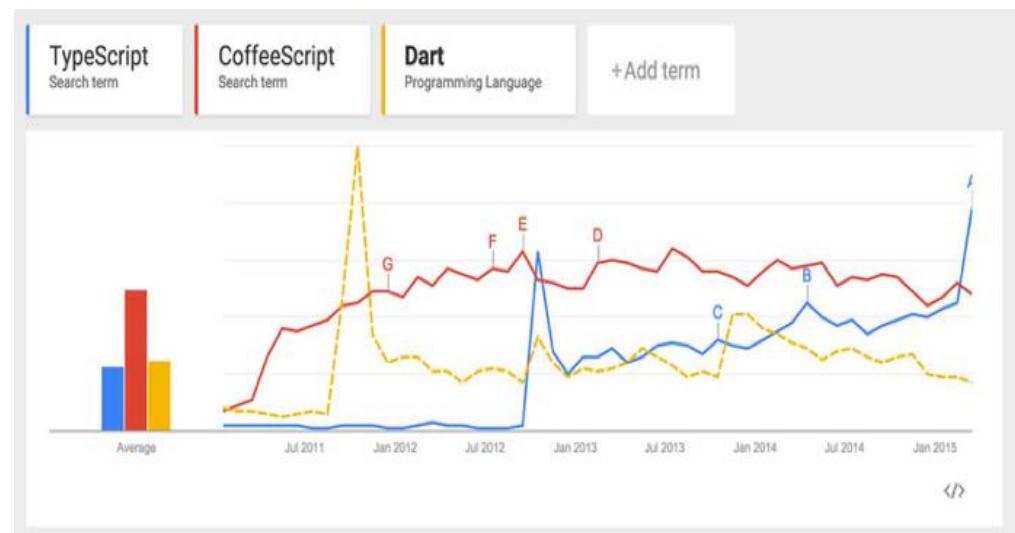


<http://www.typescriptlang.org>



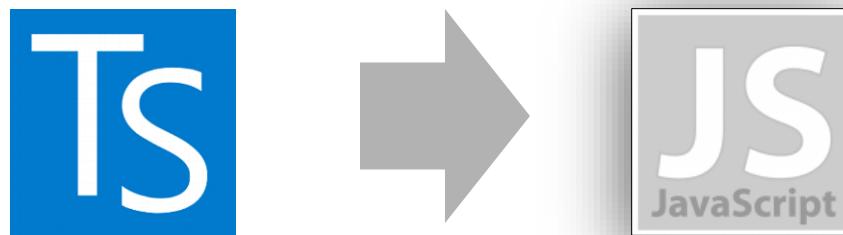
TypeScript 개요

- 자바스크립트 → ES2015(ES6) → ES2016(ES7) → TypeScript(?)
- 타입스크립트는 자바스크립트의 미래다?
- 자바스크립트(**ES6, ES7**) 기능에 타입스크립트의 필요한 기능을 추가하면 된다
- 새로운 언어가 아니다
- 클래스 관련기능
- 정적 타이핑(**static typing**)



트랜스파일러 : tsc

TSC는 타입스크립트를 자바스크립트로 변환(transpiling)해주는 도구이다.

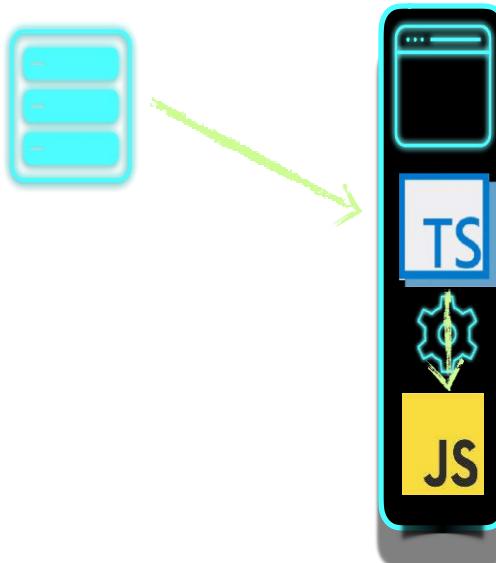


트랜스파일링

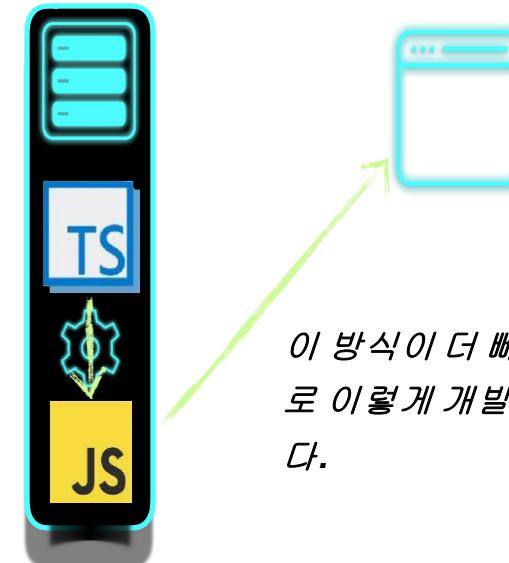
트랜스파일 위치

브라우저는 타입스크립트를 해석할 수 없으며, 자바스크립트로 변환하여 브라우저에서 처리되어야 한다. 다음 두 가지 방식이 사용되고 있다.

브라우저에서 자바스크립트로 변환



자바스크립트로 변환 후 브라우저로 로딩



이 방식이 더 빠르며, 주로 이렇게 개발하게 된다.

TypeScript : Playground

<http://www.typescript.org/play/index.html>

The screenshot shows the TypeScript Playground interface in a web browser. The top navigation bar includes tabs for 'TypeScript' (which is selected), 'Documentation', 'Samples', 'Download', 'Connect', and 'Playground'. A 'Fork me on GitHub' button is visible on the right. Below the navigation, a message promotes 'TypeScript 2.2 is now available. Download our latest version today!'. The main area features two code editors. The left editor has a dropdown menu labeled 'Select...' and contains the following TypeScript code:

```
1 let myAdd2 = function(x: number, y: number)
2
3 let myAdd3: (baseValue:number, increment:number) =>
4   function(x: number, y: number): number
5
6 let myAdd: (x: number, y: number) => number
7   function(x: number, y: number): number
8 console.log(myAdd(10,20));
9
```

The right editor has a 'Run' button and a 'JavaScript' tab, and contains the following JavaScript code:

```
1 var myAdd2 = function (x, y) { return x + y }
2 var myAdd3 = function (x, y) { return x + y }
3 var myAdd = function (x, y) { return x + y }
4 console.log(myAdd(10, 20));
5
```

TypeScript : Download

Get TypeScript

Node.js

The command-line TypeScript compiler can be installed as a Node.js package.

INSTALL

```
npm install -g typescript
```

COMPILE

```
tsc helloworld.ts
```

Visual Studio



Visual Studio 2017



Visual Studio 2015



Visual Studio Code

And More...



Sublime Text



Emacs



Atom



WebStorm



Eclipse



Vim

TypeScript 주요 문법

● Type

```
let firstName: string = "John";
let lastName = 'Smith';
let height: number = 6;
let isDone: boolean = false;
```

```
var numbers:number[] = [1, 2, 3];
var names:Array<string> = ['Alice', 'Helen', 'Claire'];
```

● Interface

```
interface IVehicle {
  wheels: number;
  engine: string;
  drive();
}
```

● Decorator

```
function Decorator(target: any) {
}

@Decorator
class MyClass {

}
```

```
function DecoratorWithArgs(options: Object) {
  return (target: Object) => {
    ...
  }
}

@DecoratorWithArgs({ type: 'SomeType' })
class MyClass {

}
```

TypeScript의 장점

- 정적 타입 지원
 - TypeScript는 정적 타입을 지원하므로 컴파일 단계에서 오류를 포착할 수 있는 장점이 있다. 명시적인 정적 타입 지정은 개발자의 의도를 명확하게 코드로 기술할 수 있다. 이는 코드의 가독성을 높이고 예측할 수 있게 하며 디버깅을 쉽게 한다.
- 도구의 지원
 - TypeScript를 사용하는 이유는 여러가지 있지만 가장 큰 장점은 IDE(통합개발환경)를 포함한 다양한 도구의 지원을 받을 수 있다는 것이다. IDE와 같은 도구에 타입 정보를 제공함으로써 높은 수준의 인텔리센스(IntelliSense), 코드 어시스트, 타입 체크, 리팩토링 등을 지원받을 수 있으며 이러한 도구의 지원은 대규모 프로젝트를 위한 필수 요소이기도 하다.

TypeScript의 장점

- **강력한 객체지향 프로그래밍 지원**
 - 인터페이스, 제네릭 등과 같은 강력한 객체지향 프로그래밍 지원은 크고 복잡한 프로젝트의 코드 기반을 쉽게 구성할 수 있도록 도우며, Java, C# 등의 클래스 기반 객체지향 언어에 익숙한 개발자가 자바스크립트 프로젝트를 수행하는 데 진입 장벽을 낮추는 효과도 있다.
- **ES6 / ES Next 지원**
 - 브라우저만 있으면 컴파일러 등의 개발환경 구축없이 바로 사용할 수 있는 ES5 와 비교할 때, 개발환경 구축 관점에서 다소 복잡해진 측면이 있지만 현재 ES6를 완전히 지원하지 않고 있는 브라우저를 고려하여 Babel 등의 트랜스파일러를 사용해야 하는 현 상황에서 TypeScript 개발환경 구축에 드는 수고는 그다지 아깝지 않을 것이다. 또한, TypeScript는 아직 ECMAScript 표준에 포함되지는 않았지만 표준화가 유력한 스펙을 선제적으로 도입하므로 새로운 스펙의 유용한 기능을 안전하게 도입하기에 유리하다.

개발환경 구축과 트랜스파일링

개발환경 구축

- TypeScript 파일(.ts)은 브라우저에서 동작하지 않으므로 TypeScript 컴파일러를 이용해 자바스크립트 파일로 변환해야 한다. 이를 컴파일 또는 트랜스파일링이라 한다.
- nodejs 설치
- <https://nodejs.org/ko/download/>

TypeScript 컴파일러 설치 및 사용법

- Node.js를 설치하면 [npm](#)도 같이 설치된다. 다음과 같이 터미널(윈도우의 경우 커맨드창)에서 npm을 사용하여 TypeScript를 전역에 설치한다.
- 설치하기
- `npm install -g typescript`
- 버전확인
- `tsc -v`

컴파일(트랜스파일링)

- 컴파일은 일반적으로 소스 코드를 바이트 코드로 변환하는 작업을 의미 한다. TypeScript 컴파일러는 TypeScript 파일을 자바스크립트 파일로 변환하므로 컴파일보다는 트랜스파일링(Transpiling)이 보다 적절한 표현이다.
- 원소스 : person.ts
- 트랜스파일링 : tsc person
- 실행: node person

개발도구

- atom
- subprime text
- edit plus
- node ++
- eclipse
- visual studio code 등 에디터이기만 하면된다. 여기서는 visual studio code를 사용한다 ms사가 내놓은 무료 툴이다

• <https://code.visualstudio.com/>

The screenshot shows the official website for Visual Studio Code at <https://code.visualstudio.com/>. The top navigation bar includes links for Docs, Updates, Blog, API, Extensions, and FAQ, along with a search bar and a green 'Download' button. A message at the top indicates 'Version 1.34 is now available! Read about the new features and fixes from April.' Below this, the main content area features a large heading 'Code editing. Redefined.' with the subtext 'Free. Built on open source. Runs everywhere.' A prominent download section for Windows, macOS, and Linux is shown, with options for 'Stable' or 'Insiders' builds. To the right, a code editor window displays a TypeScript file with syntax highlighting and code completion. Below the download section, there are four featured icons: 'IntelliSense', 'Debugging', 'Built-in Git', and 'Extensions'. At the bottom, a social media feed shows three user reviews:

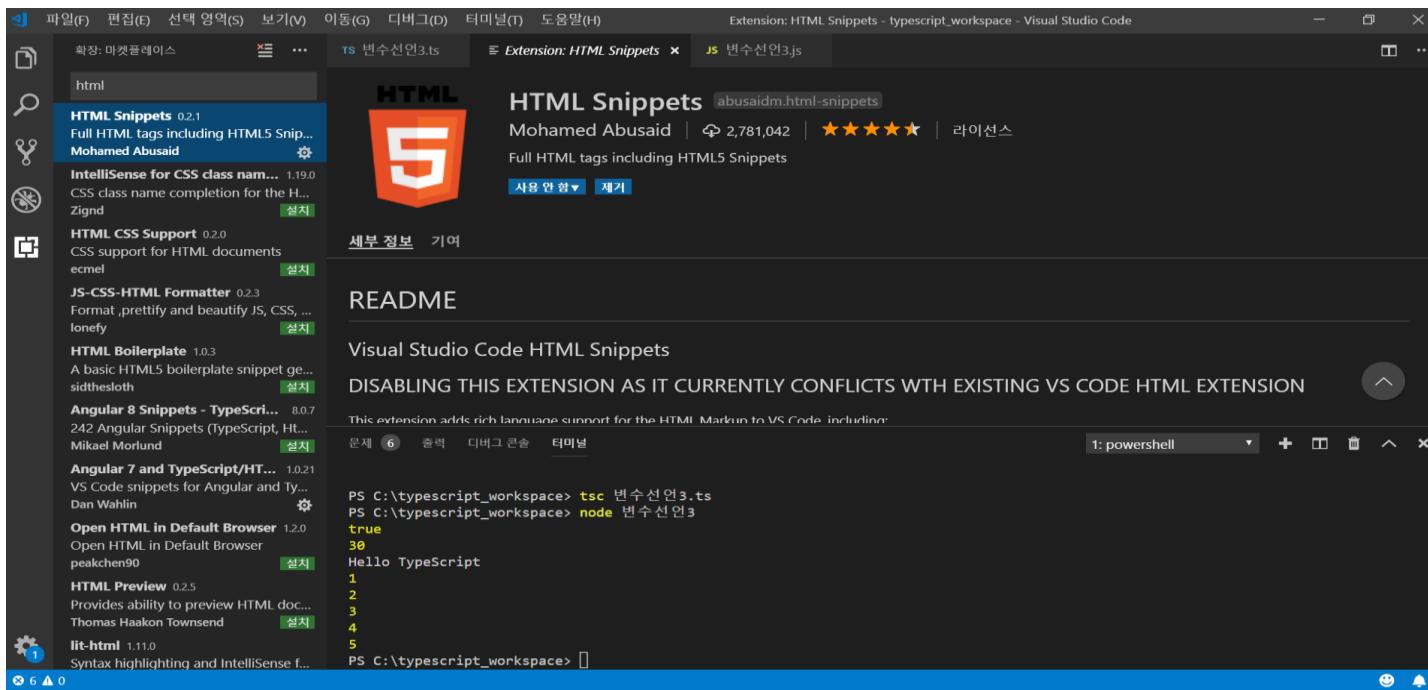
- Stefan Cosma @stefanbc** Pushed my first commit using @code in under 15s.
- Pavithra Kodmad @PKodmad** VS Code is my most used and favorite editor. I love being able to customize the editor - changing the layout, the icons, fonts and color scheme is so
- Lukas @McKbrother** What's going on with @code? The latest update is awesome! Think it will now become my primary tool for webdevelopment instead of WebStorm :)

visual studio code

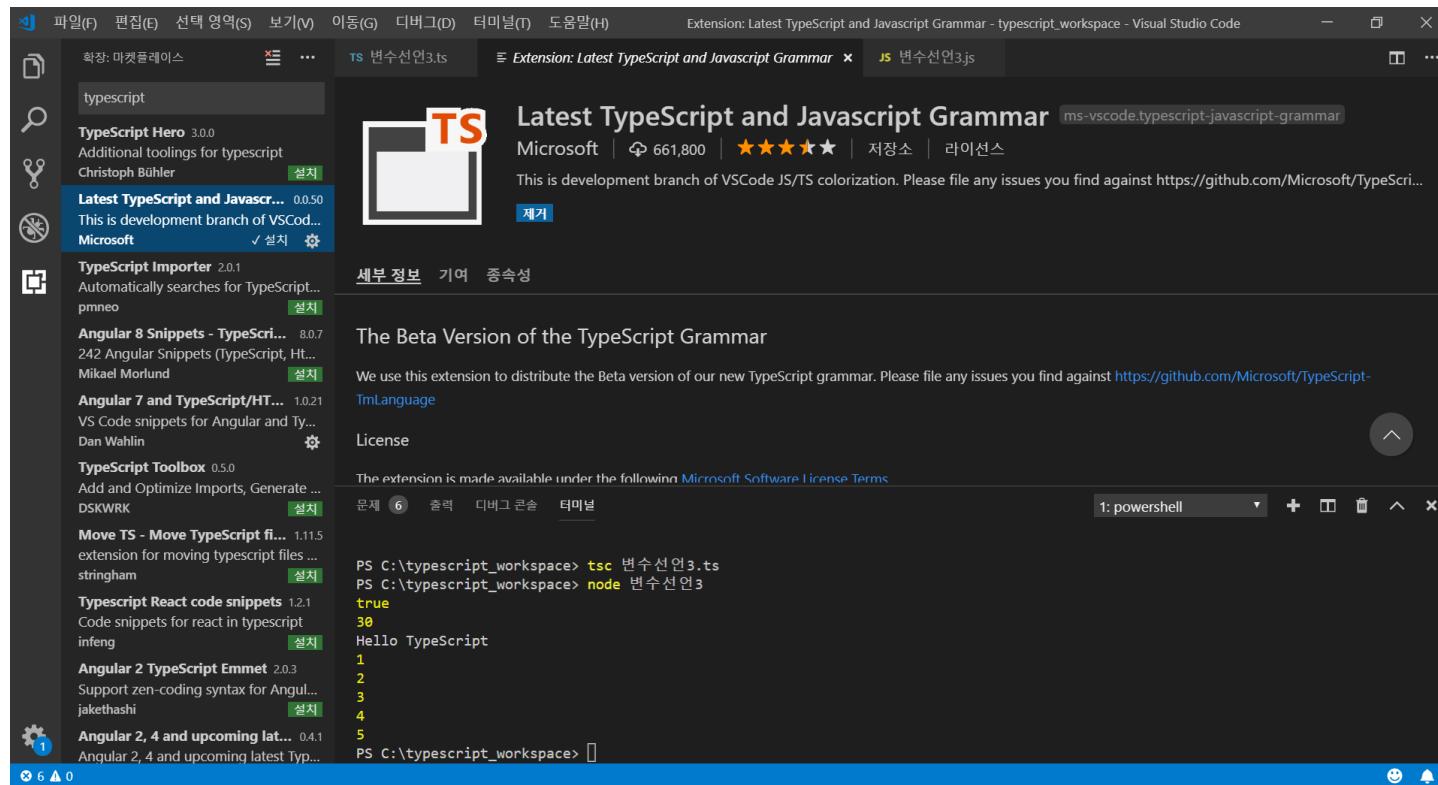
- 비주얼스튜디오 코드에서의 typescript 사용하기
- <https://code.visualstudio.com/docs/languages/typescript>

HTML 확장

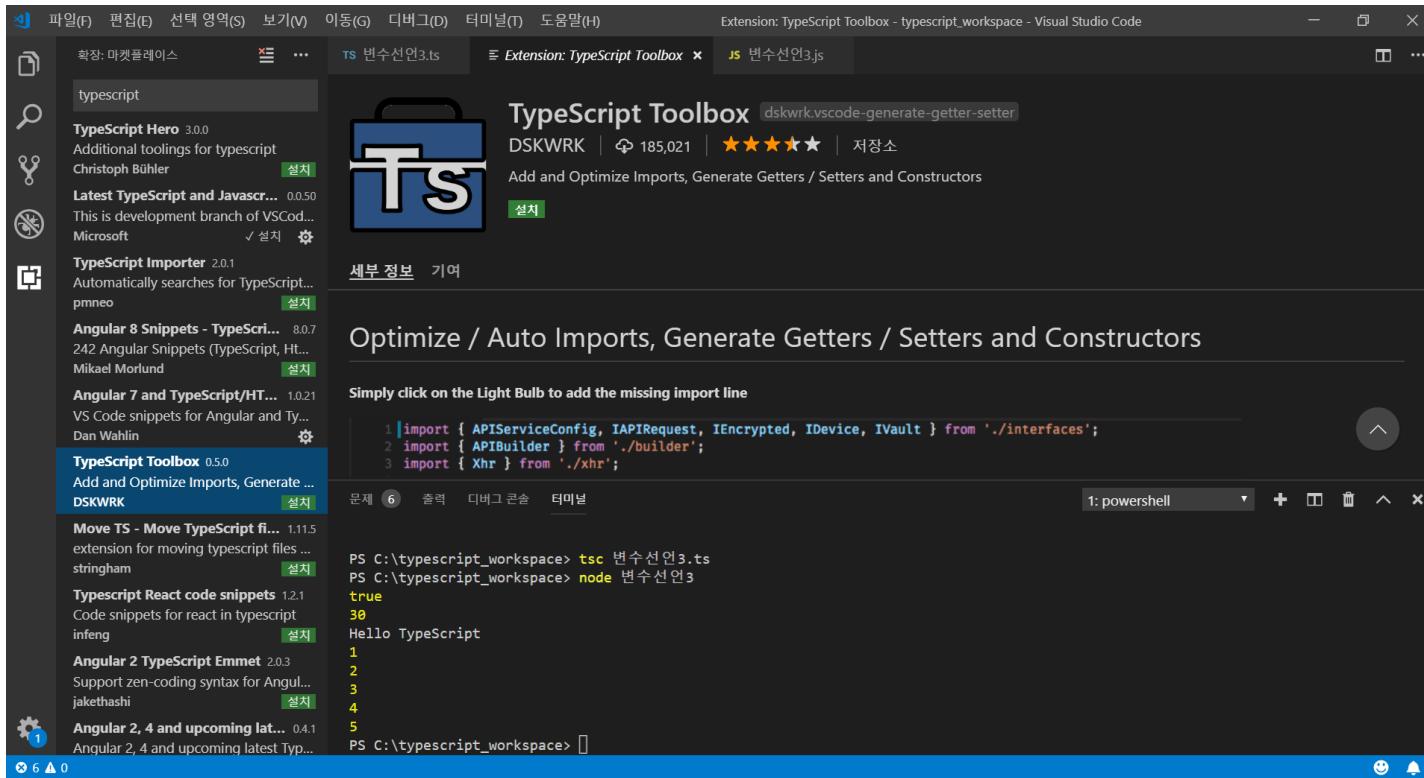
ctrl-shift-m 을 눌러서 확장 플러그인을 설치해보자. 좌변에서 html을 검색 후 첫번째 항목을 설치하자



typescript 확장기능 설치



typescript 확장기능 설치(toolbox)



typescript 구성요소

Strongly
Typed

Classes

Interfaces

Generics

Modules

Type
Definitions

Compiles to
JavaScript

EcmaScript 6
Features

변수선언과 기본타입

var로 변수 선언시

- 호이스팅 발생
- 블록레벨 스코프가 지원되지 않는다.
 - 호이스트란 변수의 정의가 그 범위에 따라 선언과 할당으로 분리되는 것을 의미한다.
 - 즉, 변수가 함수 내에서 정의되었을 경우, 선언이 함수의 최상위로,
 - 함수 바깥에서 정의되었을 경우, 전역 컨텍스트의 최상위로 변경이 된다.

```
a = 10;

console.log("a = ", a);
var a; //위의 값 10을 기억하고 있음
a = a + 1;
console.log("a = ", a);

//if 블럭내에서 선언된 변수가 별도로 취급되지 않는다.
happy가 출력된다.
var myName="sad";
if(true){
  var myName="happy";
}

console.log(myName);
```

var 과 let

- 자바스크립트에서는 var로 변수를 선언하였다
- 변수선언시, 변수의 타입도 지정하지 않았다
- 자바스크립트는 타입안정성을 포기하고 속도를 택했고(동적타이핑), 타입스크립트는 컴파일시간에 타입검사를 수행한다. 그러나 타입스크립트는 컴파일 단계까지만 사용되고 실행은 자바스크립트로 이루어진다
- var로 변수 선언시의 문제점은 var은 호이스팅이 일어나고 블록레벨스코프도 지원되지 않는다

변수선언(**var**)

```
var a; //기본적으로 any로 인식한다  
a = 10;  
console.log(a);  
//경고는 나오지만 작동은 한다  
a = "test";  
console.log(a);  
//타입체크를 정확하게 하지 않아서, 번역시에 문제가 있음을 알 수 없다.
```

호이스팅

- 변수 선언이 변수 사용보다 뒤에 있더라도 동작한다

```
helloMessage = '안녕하세요';
console.log(helloMessage);
var helloMessage;
console.log(helloMessage);
```

//동일하게 결과가 안녕하세요 라고 나온다

블록레벨스코프

```
//함수레벨 스코프는 적용됨  
helloMessage = "반갑습니다."  
function myfunc(){  
    var helloMessage="안녕히 가세요";  
    console.log(helloMessage);  
}  
  
myfunc();  
console.log(helloMessage);
```

결과

안녕히가세요
반갑습니다

```
//블록레벨스코프-안지켜짐  
helloMessage = "반갑습니다."  
if( true)  
{  
    var helloMessage="안녕히 가세요";  
    console.log(helloMessage);  
}  
  
console.log(helloMessage);
```

결과

안녕히가세요
안녕히가세요

let 선언자 특징

- 같은 블록 내에서 같은 이름의 변수를 선언할 수 없다
- 변수를 초기화하기 전에는 변수에 접근할 수 없게 해서 호이스팅을 방지한다
- 선언할 변수에 블록 레벨 스코프를 적용 한다

변수선언 (let, const 추가)

//엄격한 타입체크를 한다

```
let b1:number;
```

```
b1=10;
```

```
console.log(b1);
```

```
b1="test";      //error 발생
```

```
console.log(b1);
```

```
tsc 변수선언2.ts
```

```
PS C:\typescript_workspace> tsc 변수선언2.ts
변수선언2.ts:6:1 - error TS2322: Type '"test"' is not assignable to type 'number'.
6 b1="test";
  ~~
Found 1 error.
```

변수선언 : hello.ts

```
let a:number;  
let b:number;  
let c:number;  
  
a = 10;  
b = 20;  
c = a+b;
```

```
console.log("a = ", a);  
console.log("b = ", b);  
console.log("c = ", c);
```

```
tsc hello  
node hello
```



filename : hello.js

```
var a;  
var b;  
var c;  
a = 10;  
b = 20;  
c = a + b;  
console.log("a = ", a);  
console.log("b = ", b);  
console.log("c = ", c);
```

상수(const)

```
const birthMonth = 9;
// const는 블록 스코프가 적용됨
if (true) {
  const birthMonth = 12; //상수 선언
}
console.log(birthMonth);
```

```
const profile = {
  name: "happy",
  month: birthMonth,
};
```

// profile 변수에 대한 재할당은 불가능 함

```
const profile = "happy";
```

// 속성에 대한 할당은 가능함
profile.name = "happy2";
profile.month--;

```
console.log(profile);
```

```
상수.ts:17:7 - error TS2451: Cannot redeclare block-scoped variable 'profile'.
17 const profile = "happy";
      ~~~~~~
Found 2 errors.
```

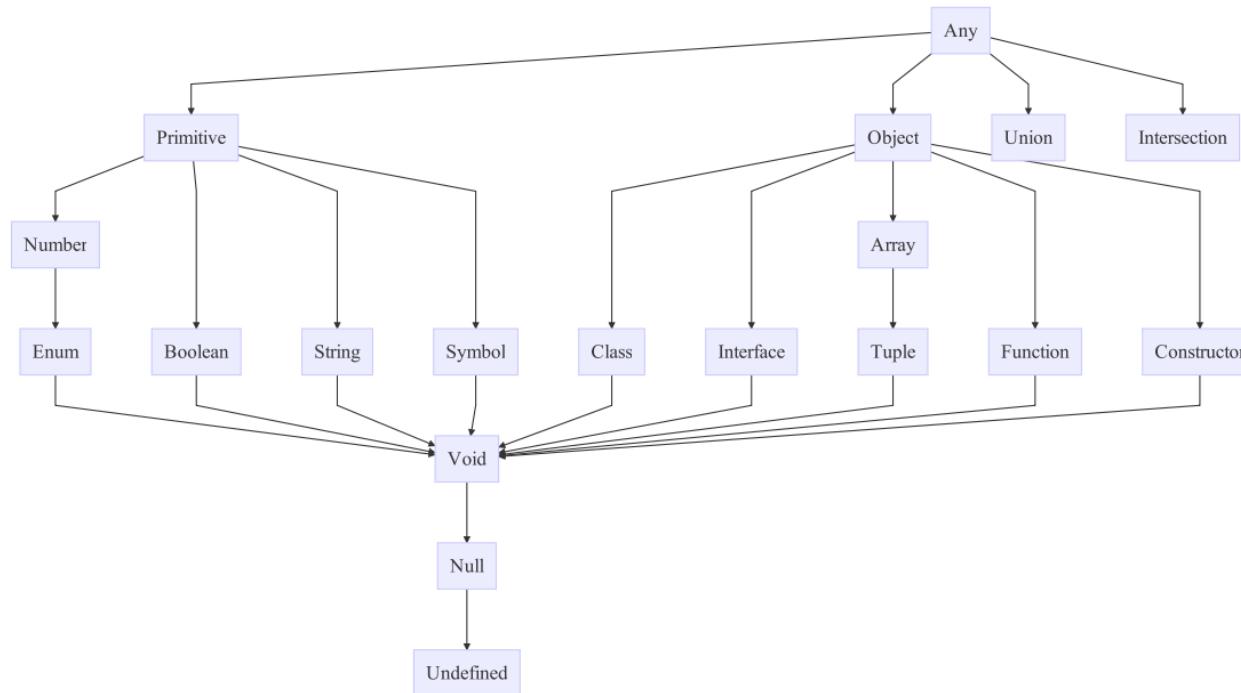
점진적 타입 검사

- 정적 타입 검사 – 컴파일 시간에 타입검사를 한다. c++, java등
- 동적 타입 검사 – 실행시간에 타입검사를 한다. 예)자바스크립트
- 점진적 타입 검사 – 컴파일 시간에 타입검사를 하면서, 필요에 따라 타입 선언의 생략을 허용한다. 타입선언을 생략할 경우 암시적 형변환이 일어난다 예)typescript, python

```
function add(a, b) {  
    return a+b;  
}  
  
console.log( add(3,4) );
```

함수의 매개변수 **a**에는 타입검사를 하지 않았지만 **any**타입으로 인식된다.
any – 타입스크립트의 모든 타입의 최상위 타입

typescript 타입 계층도



typescript가 제공하는 타입들

- 기본타입 : number, string, Boolean, symbol, enum, 문자열리터럴
- 객체타입 : Array, Tuple, function, 생성자, class, interface
- 기타타입 : union, intersection, 특수타입

변수에 타입 지정

- 형식
- var <변수식별자>:<타입>=<값>;

```
var.isTrue : boolean = true;  
var.width : number = 10;  
var.country : string ="korea";
```

typescript가 제공하는 기본 데이터 타입

- Boolean 형 , true/false값을 가진다

```
let isDone: boolean = false;
```

- number 형 – typescript의 숫자는 floating 형입니다

```
let decimal: number = 6;
let hex: number = 0xf00d;
let binary: number = 0b1010;
let octal: number = 0o744;
console.log(decimal);
console.log(hex);
console.log(binary);
console.log(octal);
```

typescript 의 타입들

```
let a:boolean;  
a=true;  
console.log(a);  
let b:number;  
b=30;  
console.log(b);  
let c:string;  
c="Hello TypeScript";  
console.log(c);  
  
//number 배열  
let d:number[]=[1,2,3,4,5];  
let i:number;  
  
for(i=0; i<d.length; i++)  
{  
    console.log(d[i]);  
}
```

typescript가 제공하는 기본 데이터 타입

- **String 형**

```
let color: string = "blue";
color = 'red';
```

- 여러 줄에 걸쳐 있고 표현식이 포함될 수 있는 템플릿 문자열을 사용할 수도 있습니다. 이 문자열은 backtick / backquote (`) 문자로 둘러싸여 있으며 포함된 표현식은 형태 \${ expr }입니다.

```
let fullName: string = `Bob Bobbington`;
let age: number = 37;
let sentence: string = `Hello, my name is ${ fullName }.
```

```
I'll be ${ age + 1 } years old next month.`;
```

type연습.ts

```
let fullName: string = `Bob Bobbington`; // ` 어포스트로피 아님, backquote임
let age: number = 37;
let sentence: string = `Hello, my name is ${ fullName }.

I'll be ${ age + 1 } years old next month.`;

console.log(fullName);
console.log(age);
console.log(sentence);
```

```
PS C:\typescript_workspace> tsc type연습.ts
PS C:\typescript_workspace> node type연습
Bob Bobbington
37
Hello, my name is Bob Bobbington.

I'll be 38 years old next month.
```

typescript가 제공하는 기본 데이터 타입

- 배열

```
let list: number[] = [1, 2, 3];
```

```
let list: Array<number> = [1, 2, 3];
```

typescript가 제공하는 기본 데이터 타입

- **tuple** : 서로 다른 데이터를 배열로 표현할 수 있다

```
// Declare a tuple type
let x: [string, number];

// Initialize it
x = ["hello", 10]; // OK //

Initialize it incorrectly
x = [10, "hello"]; // Error
```

typescript가 제공하는 기본 데이터 타입

- 열거형 , 추가됨

```
enum Color {Red, Green, Blue}  
let c: Color = Color.Green;
```

```
enum Color {Red = 1, Green, Blue}  
let c: Color  
Color.Green;
```

```
enum Color {Red = 1, Green = 2, Blue = 4}  
let c: Color = Color.Green;
```

typescript가 제공하는 기본 데이터 타입

- **Any형** – 외부(타사)의 라이브러리를 가져올때 타입을 알 수 없는 경우
동적 타이핑이 필요할경우 사용합니다

```
let notSure: any = 4;  
notSure = "maybe a string instead";  
notSure = false; // okay, definitely a boolean
```

typescript가 제공하는 기본 데이터 타입

- void 형 – 아무 값도 할당되지 않았음을 의미한다, 함수에 사용해서 함수가 반환값이 없음을 명문화 할때 사용한다

```
function warnUser(): void {  
    console.log("This is my warning message");  
    //함수에서 어떤 값도 반환하지 않겠다는 의미임  
    return "1"; //에러발생  
}
```

typescript가 제공하는 기본 데이터 타입

- never : 어떤 다른 타입도 Never에 사용 불가능하며 오로지 exception과 같이 throw되는 함수에 주로 사용한다. never에는 undefined, null, any 조차 할당이 불가능하다.

```
// Function returning never must have unreachable end point
function error(message: string): never {
    throw new Error(message);
}

// Inferred return type is never
function fail() {
    return error("Something failed");
}

// Function returning never must have unreachable end point
function infiniteLoop(): never {
    while (true) {
    }
}
```

typescript가 제공하는 기본 데이터 타입

- **Object** : non-primitive 타입, 즉 **number**, **string**, **boolean**, **symbol**, **null** 또는 **undefined**가 아닌 타입을 나타내는 타입입니다.
- **null**, **undefined** – 값이 할당되지 않았음을 의미한다

제어문

if문

```
let text: string = "";
let statusActive: number = 0;
let isEnabled: boolean = true;
```

// 첫번째 if문

```
if (statusActive || text) {
    console.log("1");
}
```

// 두번째 if문

```
if (isEnabled && 2 > 1) {
    console.log("2");
}
```

switch문

```
let command: number = 0

switch (command) {
    case 0:
        // 명령문
        break;

    case 1:
        // 명령문
        break;
}
```

```
let yourname:string;
let score:number;
let grade:String;

yourname="홍길동";
score = 91;

switch( parseInt(String(score/10)))
{
    case 10:
    case 9: grade = 'A'; break;
    case 8: grade = 'B'; break;
    case 7: grade = 'C'; break;
    case 6: grade = 'D'; break;
    default: grade = 'F'; break;
}
console.log(` ${yourname} 님의 성적은 ${grade}
입니다 `)
```

for문

```
let fruits:string[]=["오렌지", "사과", "배", "귤", "수박", "키위"];
```

```
let i:number=0;  
for(i=0; i<fruits.length; i++)  
    console.log(fruits[i]);
```

```
for (let index in fruits) {           // for ~ in 인덱스를 가져온다  
    console.log(index, fruits[index]);  
}
```

```
for (let item of fruits) {           //for ~ of 요소값을 가져온다  
    console.log(item);  
}
```

for문

```
let colors = { "red": "빨간색", "blue": "파란색", "green": "초록색" };
console.log(colors['red'])
console.log(colors['green'])
console.log(colors['blue'])

for (let key in colors) {
    console.log(key, colors[key]);
}
```

디스트럭쳐링(destructuring)

- typescript 는 디스트럭쳐링을 제공한다.
- 디스트럭쳐링은 객체의 구조를 제거하는 것을 말한다
- typescript는 객체의 구조를 분해 후 할당이나 확장과 같은 연산을 허용한다

```
let {id, country} = {"id": "1", "country": "한국"};  
  
console.log(id);  
console.log(country);
```

디스트럭쳐링(destructuring)

```
let {friendname, phone, email="test@hanmail.net"}  
    ={friendname:"홍길동", phone:"010-0000-0000"};  
console.log(friendname);  
console.log(phone);  
console.log(email);
```

디스트럭쳐링(destructuring)

```
// 새로운 이름 할당
let {friendname:newname, phone:newphone,
email:newemail="test@hanmail.net"}
    ={friendname:"홍길동", phone:"010-0000-0000"};
console.log(newname);
console.log(newphone);
console.log(newemail);
```

디스트럭쳐링 매개변수 선언

```
function printProfile(obj){  
    var name = "";  
    var nationality = "";  
  
    name = (obj.name == undefined)? "anonymous":obj.name;  
    nationality = (obj.nationality == undefined)? "?":obj.nationality;  
  
    console.log("이름 : ", name)  
    console.log("국적 : ", nationality);  
}  
  
printProfile({'name':'홍길동'});  
printProfile({'nationality':'한국'});  
printProfile({'name':'Tom', 'nationality':'America'});
```

디스트럭쳐링 매개변수 선언

```
function printProfile2({name, nationality='none'}={name:'anonymous'}){  
    console.log("이름 : ", name)  
    console.log("국적 : ", nationality);  
}  
  
printProfile2();  
printProfile2({'name':'홍길동'});  
printProfile2({'name':'Tom', 'nationallity':'America'});
```

객체 디스트럭쳐링시 **type** 키워드 활용

```
type C={a:string, b?:number};
```

```
function fruit({a, b}:C):void{
```

```
    console.log(a, b);
```

```
}
```

```
fruit( {a:"사과", b:10});
```

```
fruit( {a:"바나나"});
```

b는 ?(선택연산자) 를 사용하여 선택적 입력이 가능합니다

배열 디스트럭쳐링

```
let fruits1:string[]=["사과", "바나나", "수박", "감", "딸기", "포도"]
let f1:string = fruits1[0];
let f2:string = fruits1[1];
let [ff1,ff2,ff3]=fruits1;
console.log(ff1);
console.log(ff2);
console.log(ff3);
```

배열 디스트럭쳐링

```
let [,,ff4,ff5]=fruits1;  
console.log(ff4);  
console.log(ff5);  
[ff4, ff5] = [ff5, ff4];  
console.log(ff4);  
console.log(ff5);  
let [color1, color2="black"]=["blue"];  
console.log(color1);  
console.log(color2);
```

배열요소를 함수의 디스트럭처링 매개변수로 전달

```
function f1([first, second]:[number, string])  
{  
    console.log(first);  
    console.log(second);  
}  
  
f1([100, 'hello']);
```

전개연산자

- 타입스크립트는 E6의 전개연산자를 지원합니다 .
- 전개연산자는 ... 로 나타낸다
- 나머지 매개변수를 선언할때
- 배열요소를 확장할때
- 객체요소를 확장할때

```
let [first, ...second]=[1,2,3,4,5]
console.log(first)
console.log(second)
```

```
1
[2,3,4,5]
```

전개연산자

```
//배열 합치기  
let arr:number[] =[1,2,3];  
let arr2:number[]=[...arr, 4,5];  
  
console.log(arr)  
console.log(arr2)  
  
//배열 디스트럭처링  
let [firstitem, ...rest]:[number, number, number]=[1,2,3];  
console.log(firstitem);  
console.log(rest);  
console.log(rest[0]);
```

함수

타입스크립트 함수

- 타입스크립트는 함수의 매개변수나 반환타입을 추가해 타입안정성을 강화하였음

```
function max( x: number, y:number ) : number {  
    if (x > y)  
        return x;  
    else  
        return y;  
}
```

기본값을 갖는 매개변수

- 함수의 매개변수에 기본값을 부여하면 함수 호출시 매개변수 값을 생략 할 수 있다 오버로딩과 같은 효과, 기본값을 줄 경우 기본값을 부여한 변수들을 좌측에 배치해야 한다

```
function add(x:number=10, y:number=20, z:number=30):number{  
    return x+y+z;  
}  
  
console.log(add());  
  
console.log(add(1));  
  
console.log(add(1,2));  
  
console.log(add(1,2,3));
```

전개변수를 사용

```
function myfunc(a:number, ...rest):void{
    let i:number;
    console.log(a);
    for(i=0; i<rest.length; i++)
        console.log(rest[i]);
}
myfunc(1);
myfunc(1,2);
myfunc(1,2,3,4,5,6,7,8,9,10);
```

선택매개변수

```
function myfunc2(name:String, phone:String, email?:string){  
    if( email==undefined)  
        email = "test@hanmail.net";  
    console.log(name);  
    console.log(phone);  
    console.log(email);  
}  
myfunc2("홍길동", "010-0000-0000");  
myfunc2("임꺽정", "010-1111-1111", "lim@hanmail.net");
```

선택매개변수는 호출 시 값을 생략할 수 있다

선택매개변수는 ? 를 쓴다 이 변수에는 초기값을 할당할 수 없다.

예)email?string="test@hanmail.net" 에러 발생

위의 함수 예처럼 if 문을 써서 처리해야 한다

익명함수

- 함수의 이름이 없는 함수

//익명함수

```
let myfunc3 = x =>{ return x*x; };
```

```
let myfunc4 = (x,y) =>{ return x+y; }
```

```
console.log( myfunc3(10) );
```

```
console.log( myfunc4(10, 20) );
```

콜백함수

```
//콜백함수
function hello(message:string, callback){
    console.log(message+ " 자바스크립트 입니다");
    callback();
}
var callback1 = function():void{
    console.log("callback1 함수가 호출되었습니다
.");
}
var callback2 = function():void{
    console.log("callback2 함수가 호출되었습니다
.");
}
```

```
//함수호출
hello("안녕하세요", callback1);
hello("반갑습니다", callback2);
```

클래스와 인터페이스

TypeScript : 클래스

```
class car {  
    numTier: number;  
    carName: string;  
  
    constructor(  
        carName: string, numTier:  
number){  
        this.carName = carName;  
        this.numTier = numTier;  
    }  
    getNumTier(){  
        return this.numTier;  
    }  
    getCarName(){  
        return this.carName;  
    }  
}
```

```
let myCar = new car("해피카",4);  
console.log(myCar.getCarName()  
+"의 타이어 개수는  
"+myCar.getNumTier()+"개 입니  
다.");
```

[결과]

해피카의 타이어 개수는 4개입니다.

클래스 : person.ts

```
class Person{
    name : string;
    age : number;
    phone: string;
    email: string;
    constructor(name:string,
    age:number, phone:string,
    email:string){
        this.name = name;
        this.age = age;
        this.phone = phone;
        this.email = email;
    }
}
```

```
display(){
    console.log("이름 : ", this.name);
    console.log("나이 : ", this.age);
    console.log("전화 : ", this.phone);
    console.log("이메일 : ", this.email);
}
let p1 = new Person("강감찬", 23, "010-0000-0000", "kang@hanmail.net");
p1.name="홍길동";
p1.display();
```

변환 및 실행

```
tsc person
node person
```

접근제한자

- typescript에서도 접근제한자가 있으나, 특별히 지정안하면 public 이다
- public : 누구자 접근 가능
- protected : 상속관계에서는 public, 외부에서는 접근 불가
- private : 외부에서 접근 불가

TypeScript : 상속

- extends 키워드를 이용하여 부모 클래스를 상속

```
class HappyCar {
    numTier: number;
    carName: string;
    speed: number;

    constructor(carName: string,
    numTier: number){
        this.carName = carName;
        this.numTier = numTier;
    }

    setSpeed(speed:number){
        this.speed=speed;
    }

    getSpeed(){
        return this.speed;
    }
}
```

```
class bus extends HappyCar {
    constructor(carName: string,
    numTier: number) {
        super(carName,numTier);
    }
    setSpeed(speed = 0) {
        super.setSpeed(speed);
    }
}

class truck extends HappyCar {
    constructor(carName: string,
    numTier: number) {
        super(carName,numTier);
    }
    setSpeed(speed = 0) {
        super.setSpeed(speed);
    }
}
```

TypeScript : 상속

```
let myBus = new bus("myBus",6);
let myTruck: HappyCar = new truck("myTruck",10);

myBus.setSpeed(100);
myTruck.setSpeed(120);
console.log("현재 버스속도 : "+myBus.getSpeed());
console.log("현재 트럭속도 : "+myTruck.getSpeed());
```

[결과]

현재 버스속도 : 100

현재 트럭속도 : 120

인터페이스

```
interface MyInterface {  
    id: string;  
    title:string;  
    contents:string;  
}
```

//interface 타입을 함수의 매개변수로 전달한다

```
function printLabel(paramObj:  
MyInterface) {  
    console.log(paramObj.id);  
    console.log(paramObj.title);  
    console.log(paramObj.contents);  
}
```

```
let myObj = {size: 10, id:"1",  
title: "제목1",  
contents:"내용"};  
printLabel(myObj);
```

TypeScript : 인터페이스

- Interface에 선언된 변수나 메서드에 대한 사용을 강제함

```
interface AddressInterface {
    addressBookName:string;
    setBookName(addressBookName:
string);
    getBookName();
}

class AddressBook implements
AddressInterface {

    addressBookName:string;
    setBookName(addressBookName:
string) {
        this.addressBookName =
addressBookName;
    }
    getBookName(){
        return this.addressBookName;
    }
    constructor() { }
}
```

```
let myAddressBook = new
AddressBook();
myAddressBook.setBookName("나의
주소록");
console.log(myAddressBook.getBoo
kName());
```

[결과]
나의 주소록

TypeScript : module

Validation.ts

```
export interface StringValidator {  
    isAcceptable(s: string): boolean;  
}
```

ZipCodeValidator.ts

```
import { StringValidator } from "./Validation";  
  
export const numberRegexp = /^[0-9]+$/;  
export class ZipCodeValidator implements StringValidator  
{  
    isAcceptable(s: string) {  
        return s.length === 5 && numberRegexp.test(s);  
    }  
}
```

Test.ts

```
import { ZipCodeValidator } from "./ZipCodeValidator";  
let myValidator = new ZipCodeValidator();
```

네임스페이스

```
namespace Aname{
    export let a:number=10;
    export function myfunc():void{
        console.log("Aname space");
    }
    export class MyClass{
        name : String;
        phone : String;
        constructor(name:string="", phone:string ""){
            this.name = name;
            this.phone = phone;
        }
        display():void{
            console.log(this.name, this.phone);
        }
    }
}
```

네임스페이스

```
console.log(Aname.a);
Aname.myfunc();
let obj1:Aname.MyClass = new Aname.MyClass("홍길동", "010-0000-0000");
obj1.display();
```

map함수

- map함수는 배열의 요소 모두에 특정 함수를 적용하고자 할 때 사용합니다.

```
let a1:number[]=[1,2,3,4,5,6,7,8,9,10];
let b1:number[];
function getDouble(x:number):number
{
    return x*2;
}

let i:number;
b1=[];
for(i=0; i<a1.length; i++)
{
    b1.push( getDouble(a1[i]));
}
console.log(b1);
```



```
b1 = a1.map( (x)=> x*2 );
console.log(b1);
```

filter 함수

- filter함수는 배열의 요소 중 특정 조건을 만족하는 데이터를 추출하기 위해 사용합니다

```

let a1:number[]=[1,2,3,4,5,6,7,8,9,10];
let b1:number[];
function isGreater(x:number):boolean
{
    return x>5;
}

let i:number;
b1=[];
for(i=0; i<a1.length; i++)
{
    if( isGreater(a1[i]))
        b1.push( a1[i]);
}
console.log(b1);

```



```

b2 = a1.filter( (x)=>x>5 );
console.log(b2);

```

map과 filter이용

```
let a1:number[]=[1,2,3,4,5,6,7,8,9,10];
let b1:number[];
let b2:number[];
let b3:number[];
b1 = a1.map( (x)=> x*2 );
console.log(b1);
b2 = a1.filter( (x)=>x>5 );
console.log(b2);
b3 = a1.filter( (x)=>x>5 ).map((x)=>x+10);
console.log(b3);
```

Node.js 설치

- 다운로드 : <https://nodejs.org/ko/download/>
- 윈도우의 경우 msi 파일을 선택해 다운받는다



다운로드

최신 버전: v4.5.0 (includes npm 2.15.9)

플랫폼에 맞게 미리 빌드된 Node.js 인스톨러나 소스코드를 다운받아서 바로 개발을 시작하세요.

LTS 대다수 사용자에게 추천	Windows Installer node-v4.5.0-x86.msi	Macintosh Installer node-v4.5.0.pkg	Source Code node-v4.5.0.tar.gz
최신 버전 최신 기능	Windows Installer	Macintosh Installer	Source Code

Windows Installer (.msi)

Windows Binary (.exe)

Mac OS X Installer (.pkg)

Mac OS X Binaries (.tar.gz)

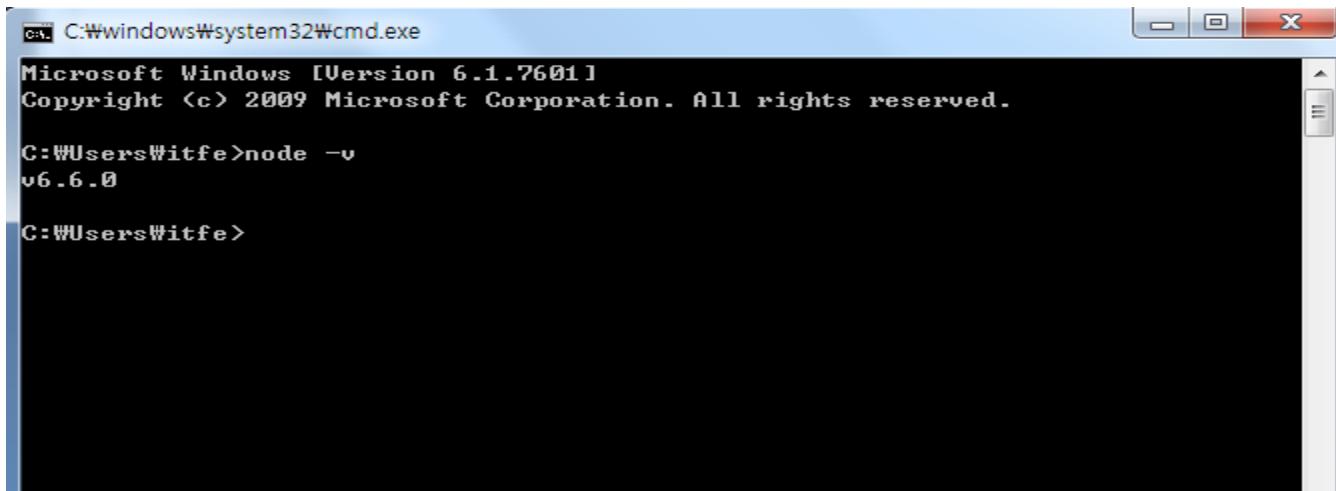
32-bit	64-bit
32-bit	64-bit
	64-bit
	64-bit

Node.js 의 개발 툴

- EditPlus 나 UltraEdit 등 일반 에디터도 사용 가능하다
- Eclipse
 - 조금 무겁고 속도도 느리다.
- Visual Studio
 - 마이크로소프트에서 제공하는 비쥬얼 스튜디오에 JS 플러그인을 설치하여 사용한다
- WebStorm
 - 상용 프로그램이다. 연단위로 정액을 내고 사용할 수 있다.
- Atom
 - Git에서 개발한 에디터 프로그램이다.
 - 간단하게 설치가 가능하고 대부분의 언어를 사용 가능하다
 - <https://atom.io/>

Node js 설치 확인하기

- cmd 창에서 node -v 를 쳐본다



```
cmd C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Witfe>node -v
v6.6.0

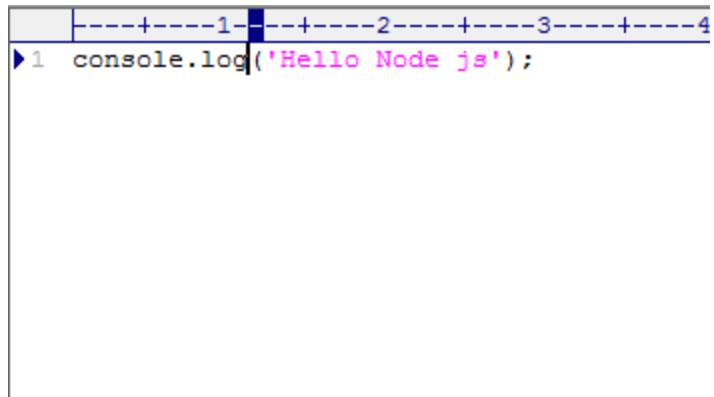
C:\Users\Witfe>
```

반응이 없을경우에

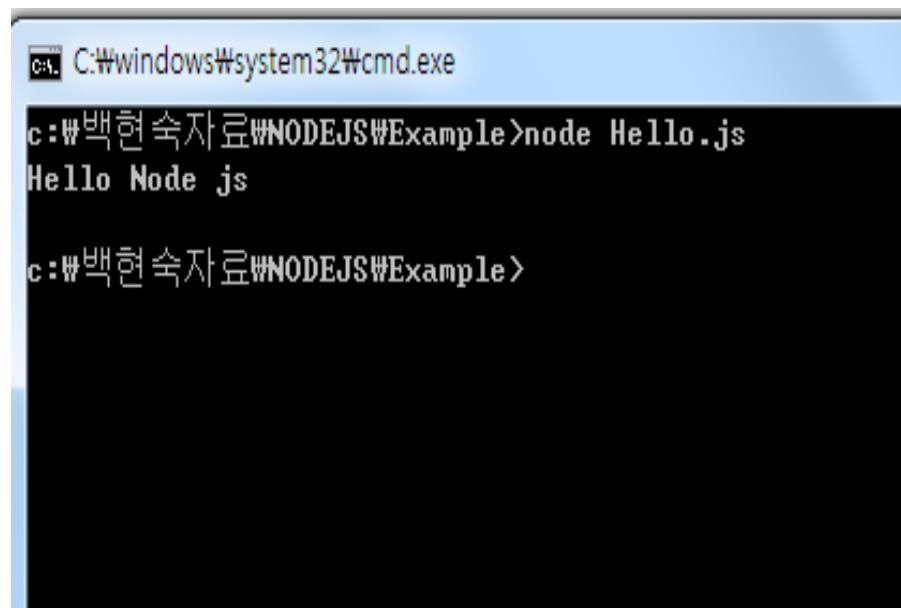
내컴퓨터 속성 - 고급 - 환경 변수 설정 - path에 node.js 설치 폴더 추가
path 마지막에 추가 하기 : C:\Program Files\nodejs\

console 객체 사용해보기

- console.log('메시지');
- 파일을 만들어 test 해보자 파일명 : Hello.js



```
1 console.log('Hello Node js');
```



```
C:\Windows\system32\cmd.exe
c:\백현숙자료\nodejs\Example>node Hello.js
Hello Node js

c:\백현숙자료\nodejs\Example>
```

Http 객체 사용하기

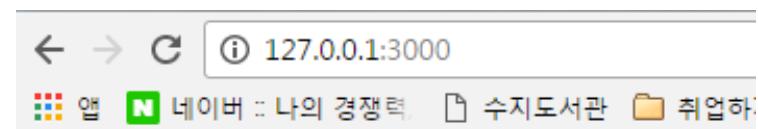
- 파일명 : HelloServer.js

```
1 //http 객체를 알아봅시다 - 웹서버 만들기
2 var http = require('http');
3 http.createServer(function(request, response) {
4     response.writeHead(200, {'Content-Type':'text/html'});
5     response.end('<h1>Hello World!</h1>');
6 }).listen(3000);
▶ 7 |
```

The screenshot shows a Windows command prompt window. The user has navigated to the directory 'C:\백현숙자료\NODEJS\Example'. They first run 'node Hello.js', which outputs 'Hello Node.js'. Then they run 'node HelloServer.js', which starts a server at port 3000.

```
C:\Windows\system32\cmd.exe - node Hello.js
c:\백현숙자료\NODEJS\Example>node Hello.js
Hello Node.js

c:\백현숙자료\NODEJS\Example>node HelloServer.js
```



Hello World!

Node.js Module(모듈)

NPM 사용법

- npm – node.js 라이브러리 설치 프로그램
- npm은 최신 node.js는 설치 시 자동으로 함께 설치가 된다
- **npm install [모듈명] -g**
 - g는 global을 뜻한다. 전역 모듈로 다운받는 것이다. 이 경우는 많이 사용되지 않으나 express, gulp나 grunt 등 커맨드 터미널(Shell)에서 사용되어지는 모듈들은 Global하게 설치하여야 한다.
 - 보통 global하게와 local하게 두번에 걸쳐서 설치하는 경우가 종종 있다.
- **npm install [모듈명] --save**
 - --save는 현재 폴더에 package.json에 집어넣는다는 뜻이다.
 - npm으로 설치를 할 때에, 해당 모듈을 유지 보수하기 위한 정보가 package.json에 저장되어 있다.
 - 따라서 직접 입력해 주거나, --save를 통해서 쉽게 입력할 수 있다.
- **npm install**
 - package.json에 필요한 모듈들을 다 집어넣었을 때에 필요한 모듈들을 한번에 다운로드하기 위하여 사용되어진다.

export

- 사용자가 만든 모듈을 만들때 사용한다
- 모듈 내보내기
 - var app = express();
 - module.exports = app;
- 모듈 사용하기
 - var express = require('../express.js');

Nodejs 동기모드

```
1 //동기모드일때 예외처리
2 var fs = require("fs");
3 ∵try
4 {
5     var data = fs.readFileSync("data.txt", "utf-8");
6     console.log(data);
7 }
8 ∵catch(e)
9 {
10    console.log(e);
11 }
12 console.log("종료.....");
13 |
```

Nodejs 비동기모드

```
1 var fs = require("fs");
2
3 //비동기
4 ∵fs.readFile("./data.txt", "utf8", function(error, data){
5     //error - 에러발생시 에러정보를 가져온다
6     //data - 다 읽은 데이터값 들고오기
7     console.log(data);
8 });
9 console.log("file read end?");
10
11
12 //콜백함수
13 ∵function callback(error, data)
14 {
15     //비동기함수가 동작이 완료되면 시스템에
16     //의해 이 함수가 호출된다
17     //callback - 시스템에 위해 호출되는 함수
18     console.log(data);
19 }
20
21 fs.readFile("data.txt", "utf8", callback);
```

서버1.js

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 4000;

✓ const server = http.createServer((req, res) => {
    res.statusCode = 200;
    //res.setHeader('Content-Type', 'text/plain');
    res.setHeader('Content-Type', 'text/html');
    res.end('<h1 style="color:blue">Hello World</h1>\n');
});

✓ server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

서버2.js

```
var http=require('http');
var fs = require('fs');
var url=require("url");
var url=require("querystring");

http.createServer(function(request, response){

    if(request.method=='GET')
    {
        response.writeHead(200, {'Content-Type':'text/html'});
        var data = fs.readFileSync("./html/input.html", "utf8");
        response.end(data);
    }
    else if(request.method=='POST')
    {
        //request 객체의 data 이벤트 리스너를 붙여야 한다
        request.on('data', function(data){
            response.writeHead(200, {'Content-Type':'text/html'});
            response.end('<h1>' + data + '</h1>');
        });
    }
}).listen(4000);
```

html/input.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="Generator" content="EditPlus®">
    <meta name="Author" content="">
    <meta name="Keywords" content="">
    <meta name="Description" content="">
    <title>Document</title>
  </head>
  <body>
    <form method="post">
      이름 : <input type="text" name="name"><br/>
      나이 : <input type="text" name="age"><br/>
      <input type="submit" value="전송">
    </form>
  </body>
</html>
```

서버2_1.js(get방식 전송처리)

```
var http=require('http');
var fs = require('fs');
var url=require("url");

http.createServer(function(request, response){

    //http://127.0.0.1:4000?name=Tom&age=21

    if(request.method=='GET')
    {
        response.writeHead(200, {'Content-Type':'text/html'});

        var uri = request.url;
        var query = url.parse(uri, true).query;
        console.log(query.name);
        console.log("name : " + query.name);
        console.log("age : " + query.age);

        response.end(JSON.stringify(query));
    }

}).listen(4000, function(){
    console.log("Server start");
});
```

서버2_2.js(포스트방식처리)

```
//포스트방식 전송일때 파라미터 처리하기
var http=require('http');
var fs = require('fs');
var url=require("url");
var querystring=require("querystring");

http.createServer(function(request, response){
    //http://127.0.0.1:4000?name=Tom&age=21
    if(request.method=='POST')
    {
        //request 객체의 data 이벤트 리스너를 붙여야 한다
        request.on('data', function(data){
            var data = querystring.parse(data.toString());

            console.log("name " + data["name"]);
            console.log("name " + data.name);
            console.log("age " + data["age"]);
            console.log("age " + data.age);
            response.writeHead(200, {'Content-Type':'text/html'});
            var result = '<h1>0|름 : '+data.name+'</h1>';
            result += '<h1>나이 : '+data.age+'</h1>';
            response.end(result);
        });
    }
}).listen(4000, function(){
    console.log("Server start");
});
```

postman로 정보보내기

POST http://127.0.0.1:4000

Params Authorization Headers (11) **Body** Pre-req

none form-data x-www-form-urlencoded raw

KEY	VALUE
<input checked="" type="checkbox"/> name	홍길동
<input checked="" type="checkbox"/> age	25
Key	Value

Body Cookies (1) Headers (4) Test Results

Pretty Raw Preview Visualize HTML ▾

1 <h1>이름 : 홍길동</h1>
2 <h1>나이 : 25</h1>

서버3.js

```
var http = require("http"); //http모듈 갖고 오기
var fs = require("fs");    //파일서비스
var url=require("url");

function test(request, response)
{
    //상대방한테 응답을 한다
    var pathName = url.parse(request.url).pathname;
    if( pathName == "/")
    {
        //파일을 불러서 보내기 --비동기로 읽는다
        fs.readFile("./index.html", "utf8",
            function(error, data)
        {
            console.log("data : " + data);
            response.writeHead(200, {'Content-Type':'text/html'});
            response.end(data);
            console.log(request.headers.cookie);
        });
    }
}
```

서버3.js

```
else if(pathName == "/test")
{
    //파라미터 처리
    var query = url.parse(request.url, true).query;

    console.log("name : " + query.name);
    console.log("age : " + query.age);

    console.log(JSON.stringify(query));
    var data = JSON.parse(JSON.stringify(query));
    console.log("name " + data["name"]);
    console.log("name " + data.name);
    console.log("age " + data["age"]);
    console.log("age " + data.name);

}

else{
    //페이지 강제 이동하기
    response.writeHead(302, {"Location": "http://www.daum.net"});
    response.end();
}
```

서버3.js

```
function listenResult()
{
    console.log("Server start http://127.0.0.1:4000");
}
var server = http.createServer(test); //서버 객체 만들기

server.listen(4000, listenResult); //4000번 포트에서 대기, 서버 시작

server.on('connection', function(code){
    console.log('connection on');
});

server.on('request', function(code){
    console.log('request on');
});

server.on('close', function(code){
    console.log('close on');
});
```

서버3.js

```
//다른포트로 서비스 하는 서버를 하나 더 만들기 (이미지 서비스)
http.createServer( function(request, response){
    fs.readFile('./images/Chrysanthemum.jpg',
        function(error, data){
            response.writeHead(200, {'Content-Type':'image/jpeg'});
            response.end(data);
        });
}).listen(4001);

http.createServer( function(request, response){
    fs.readFile('./mp3/Kalimba.mp3',
        function(error, data){
            response.writeHead(200, {'Content-Type':'audio/mp3'});
            response.end(data);
        });
}).listen(4002);
```

html/index.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>Document</title>
  </head>
  <body>
    <h1><font color="#0000ff">Hello MyHome</font></h1>

    <h2>제목2</h2>
    <h3>제목2</h3>
    <h4>제목2</h4>

  </body>
</html>
```

테스트

- node server3.js
- <http://127.0.0.1:4000/>
- <http://127.0.0.1:4000/test?name=홍길동&age=27>
- <http://127.0.0.1:4001>
- <http://127.0.0.1:4002>

서버4.js (ejs 엔진을 이용한 렌더링)

```
var http=require("http");
var fs = require("fs");
var ejs = require("ejs"); //npm install ejs

http.createServer(function(req, res){
    //ejs 페이지를 불러와서 html 페이지를 만든다

    fs.readFile("test.html", "utf8", function(error, data)
    {
        res.writeHead(200, { 'Content-type': 'text/html'});
        res.end(data); //렌더링 안하면 문자 그대로 나옴
        //res.end(ejs.render(data));
        //ejs.render - ejs기반 문서를 html 형태로 전환시켜 반환한다
    });

}).listen(4000, function(){
    console.log("Server Start");
})
```

실행결과

Tom

54

i = 0

i = 1

i = 2

i = 3

i = 4

i = 5

i = 6

i = 7

i = 8

i = 9

서버5.js

```
var http=require("http");
var fs = require("fs");
var ejs = require("ejs"); //npm install ejs

http.createServer(function(req, res){
    //ejs 페이지를 불러와서 html 페이지를 만든다

    fs.readFile("test2.html", "utf8", function(error, data)
    {
        res.writeHead(200, {'Content-type':'text/html'});
        res.end( ejs.render(data,
            {
                name:"홍길동",
                address:"서울시",
                fruits:['사과', '배', '포도', '참외', '수박']
            }
        ));
        //ejs.render - ejs기반 문서를 html 형태로 전환시켜 반환한다
    });
}).listen(4000, function(){
    console.log("Server Start");
})
```

test2.html

```
<html>
    <meta charset="utf-8">
<body>
    <h1>서버로부터 값 받기</h1>
    이름은 <%=name%> 입니다 <br/>
    주소는 <%=address%> 입니다
    <br/><br/>

    <ul>
        <%for(i=0; i<fruits.length; i++){%>
            <li><%=fruits[i]%></li>
        <%}%>
    </ul>

</body>
</html>
```

결과

서버로부터 값 받기

이름은 홍길동 입니다
주소는 서울시 입니다

- 사과
- 배
- 포도
- 참외
- 수박

서버6.js(ejs 엔진) npm install ejs

```
var http = require("http");
var fs = require("fs");
var ejs = require("ejs"); //npm install ejs

http.createServer(function(req, res){
    //ejs 페이지를 불러와서 html 페이지를 만든다 |
    fs.readFile("test3.html", "utf8", function(error, data)
    {
        res.writeHead(200, {'Content-type': 'text/html'});
        res.end( ejs.render(data,
            {
                name: "Trump",
                description: "President of America"
            }
        ));
        //ejs.render - ejs기반 문서를 html 형태로 전환시켜 반환한다
    });
}).listen(4000, function(){
    console.log("Server Start");
})
```

test3.html

```
<h1><%= name %></h1>
<p><%= description %></p>
<hr />
<% for(var i = 0; i < 10; i++) { %>
    <h2>The Square of <%= i %> is <%= i * i %></h2>
<% } %>
```

Trump

President of America

The Square of 0 is 0
The Square of 1 is 1
The Square of 2 is 4
The Square of 3 is 9
The Square of 4 is 16
The Square of 5 is 25
The Square of 6 is 36
The Square of 7 is 49
The Square of 8 is 64
The Square of 9 is 81

서버6.js(jade 엔진) npm install jade

```
var http = require("http");
var fs = require("fs");
var jade = require('jade'); //npm install jade

http.createServer(function(req, res){
    //ejs 페이지를 불러와서 html 페이지를 만든다

    fs.readFile("test1.jade", "utf8", function(error, data)
    {
        res.writeHead(200, { 'Content-type': 'text/html' });
        var fn = jade.compile(data);

        // 출력합니다.
        res.writeHead(200, { 'Content-Type': 'text/html' });
        res.end(fn());
        //ejs.render - ejs기반 문서를 html 형태로 전환시켜 반환한다
    });

}).listen(4000, function(){
    console.log("Server Start");
})
```

test1.jade

```
✓ html
  ✓ head
    title 'This is jade engine'
  ✓ body
    h1 It's very diffcult
    h2 jade is a engine
    hr
    a(href="http://www.daum.net") www.daum.net
```

서버8.js(jade 엔진-값전달)

```
var http = require("http");
var fs = require("fs");
var jade = require('jade');

http.createServer(function(req, res){
    //ejs 페이지를 불러와서 html 페이지를 만든다

    fs.readFile("test2.jade", "utf8", function(error, data){
        {
            res.writeHead(200, { 'Content-type': 'text/html'});
            var fn = jade.compile(data);

            // 출력합니다.
            res.writeHead(200, { 'Content-Type': 'text/html' });
            res.end(fn({
                name: 'Tom',
                description: 'Hello jade With Node.js .. !'
            }));
            //ejs.render - ejs기반 문서를 html 형태로 전환시켜 반환한다
        });
    }).listen(4000, function(){
        console.log("Server Start");
    })
})
```

test2.jade

<https://html2jade.org/> html을 jade로 변환

```
doctype html
html
  head
    title Index Page
  body
    // JADE String
    h1 #{name} .. !
    h2= description
    hr
    - for(var i = 0; i < 10; i++) {
      p
        a(href="http://www.daum.net") Go To Daum #{i}
    - }
```

Express 엔진

express 모듈

- <http://expressjs.com/ko/starter/installing.html>
- 디렉토리 생성후 디렉토리로 이동하자
- \$ mkdir myapp
- \$ cd myapp
- \$ npm init
- \$ npm install express --save
- --save 옵션을 통해 설치된 Node 모듈은 package.json 파일 내의 dependencies 목록에 추가됩니다.
- 이후 app 디렉토리에서 npm install을 실행하면 종속 항목 목록 내의 모듈이 자동으로 설치됩니다.

express 모듈

- express 모듈 - http와 비슷하지만 훨씬 사용이 간편하고 기능도 많은 외부모듈 입니다
- express 모듈에 request 객체와 response 객체에 다양한 기능 추가
 - response.send : 매개변수의 자료형에 따라 적절한 형태로 응답한다
 - response.json : JSON 형태로 응답한다
 - response.jsonp : JSONP 형태로 응답한다
 - response.redirect : 웹페이지 경로를 강제 이동한다
- JSONP 참고자료
 - JSONP란 CORS가 활성화 되기 이전의 데이터 요청 방법으로, 다른 도메인으로부터 데이터를 가져오기 위해 사용하는 방법입니다.
 - 자바스크립트는 서로 다른 도메인에 대한 요청을 보안상 제한하는데, 이 정책은 Same-Origin Policy, SOP라고 합니다.
 - SOP 정책으로 인해 생기는 이슈를 Cross-domain issue라고 하는데 JSONP는 이 이슈를 우회해서 데이터 공유를 가능하게 하였습니다.

외부모듈 이용하기

custom_hello.js

```
var hello=function(){
    console.log('Hello nodejs');
}

module.exports = hello;
```

app.js

```
var hello=require('./custom_hello');
var gb=require('./custom_goodbye');

hello();
gb.goodbye();
```

custom_goodby.js

```
exports.goodbye = function(){
    console.log('good bye');
}
```

app1.js(express 모듈)

```
var express = require('express');
var app = express();

var express = require('express');
var app = express();

//use 함수는 get, post 모두에 응한다, 현재 모든 url을 혼자처리함
app.use(function (request, response){
    response.writeHead(200, {'Content-Type':'text/html'});
    response.end('<h1>Hi Hello</h1>');
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app2.js

```
var express = require('express');
var app = express();

//use-get, post 둘다 처리
app.use("/use", function (req, res){
    res.writeHead(200, {'Content-Type':'text/html'});
    res.end('<h1>Hi Hello</h1>');
});

app.get("/get", function (req, res){
    res.writeHead(200, {'Content-Type':'text/html'});
    res.end('<h1>Get Hello</h1>');
});

app.post("/post", function (req, res){
    res.writeHead(200, {'Content-Type':'text/html'});
    res.end('<h1>PostGet Hello</h1>');
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app3.js

```
var express = require('express');
var app = express();

app.use(function(request, response){
    var output=[]; //배열선언
    for(i=1; i<=10; i++){
        //배열에 count:1 name:name-1 형식으로 10개의 데이터를 추가한다
        output.push( {count:i, name:'name-'+i} );
    }

    response.send( output ); //알아서 출력한다
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app4.js (error 페이지)

```
var express = require('express');
var app = express();

app.use(function(request, response, next){
    response.status(404).send('<h1>ERROR</h1>');
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app5.js (agent 정보)

```
var express = require('express');
var app = express();

app.use(function(request, response, newt){

    var agent = request.header('User-Agent');
    console.log(request.headers);
    console.log(agent);
    response.sendStatus(200);
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app6.js

```
var express = require('express');
var app = express();

app.use(function(request, response, newt){

    var agent = request.header('User-Agent');
    if( agent.toLowerCase().match(/chrome/)){
        response.send("<h1>I am Chrome</h1>");
    }
    else{
        response.send("<h1>I am not Chrome</h1>");
    }

});

app.listen(4000, function () {
  console.log('Example app listening on port 4000!');
});
```

app7.js (request 객체)

```
//http://127.0.0.1:4000/?name=www&age=21&arr=1&arr=2&arr=3
var express = require('express');
var app = express();

app.use(function(request, response, next){
    var name = request.query.name;
    var age = request.query.age;

    response.write('<h1>' + name + '</h1>');
    response.write('<h1>' + age + '</h1>');

    var arr = request.query.arr;

    for (i=0; i<arr.length; i++ )
    {
        response.write('<p>' + arr[i] + '</p>');
    }

});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app8.js(미들웨어)

```
var express = require('express');
var app = express();

// app.use(function(request, response, next){
//   console.log("첫번째 미들웨어");
//   next(); //차례대로 다음 함수(미들웨어)를 호출한다
// });

// app.use(function(request, response, next){
//   console.log("두번째 미들웨어");
//   next();
// });

// app.use(function(request, response, next){
//   console.log("세번째 미들웨어");
//   response.writeHead(200, {'Content-Type':'text/html'});
//   response.end('<h1>Express MiddleWare</h1>');
// });

app.listen(4000, function () {
  console.log('Example app listening on port 4000!');
});
```

app9.js(미들웨어)

```
var express = require('express');
var app = express();
app.use(function(request, response, next){
    request.name="Brown";
    response.name="John";
    console.log("첫번째 미들웨어");
    next(); //차례대로 다음 함수(미들웨어)를 호출한다
});
app.use(function(request, response, next){
    console.log("두번째 미들웨어");
    request.phone="010-0000-0000";
    response.phone="010-1111-1111";
    next();
});
app.use(function(request, response, next){
    console.log("세번째 미들웨어");
    response.writeHead(200, {'Content-Type':'text/html'});
    response.write('<h1>' + request.name + '</h1>');
    response.write('<h1>' + response.name + '</h1>');
    response.write('<h1>' + request.phone + '</h1>');
    response.end('<h1>' + response.phone + '</h1>');
});
app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app10.js(라우팅)

```
var express = require('express');
var app = express();

app.get("/a", function(request, response, next){
    response.send('a 입니다');
});

app.get("/b", function(request, response, next){
    response.send('b 입니다');
});

app.get("/c", function(request, response, next){
    response.send('c 입니다');
});

app.listen(4000, function () {
    console.log('Example app listening on port 4000!');
});
```

app11.js(url, request 객체)

```
var express = require('express');
var app = express();

app.get("/a/:id", function(request, response, next){
    var id = request.params.id;
    response.send('a 입니다' + id );
});

app.get("/b/:page", function(request, response, next){
    var page = request.params.page;
    response.send('b 입니다' + page);
});

app.get("/c", function(request, response, next){
    response.send('c 입니다');
});

app.listen(4000, function () {
  console.log('Example app listening on port 4000!');
});
```

app12.js(미들웨어, 에러처리)

```
// 모듈을 추출합니다.  
var express = require('express');  
  
// 서버를 생성합니다.  
var app = express();  
  
// 라우터를 설정합니다.  
app.get('/index', function (request, response) {  
    response.send('<h1>Index Page</h1>');  
});  
app.all('*', function (request, response) {  
    response.status(404).send('<h1>ERROR - Page Not Found</h1>');  
});  
  
// 서버를 실행합니다.  
app.listen(4000, function () {  
    console.log('Server running at http://127.0.0.1:4000');  
});
```

app13.js(라우터)

```
var express = require('express');
var app = express() // 서버를 생성합니다.

var boardRouter = express.Router() // 라우터를 생성합니다.
var memberRouter = express.Router();

// 라우터를 설정합니다 - 별도의 파일로 작성할 수 있음
boardRouter.get('/', function (request, response) {
  response.send('<h1>게시판입니다</h1>');
});
boardRouter.get('/list', function (request, response) {
  response.send('<h1>게시판입니다</h1>');
});

// 라우터B를 설정합니다.
memberRouter.get('/write', function (request, response) {
  response.send('<h1>회원가입화면입니다</h1>');
});

// 라우터를 설정합니다.
app.use('/board', boardRouter); // http://127.0.0.1:4000/board/list
app.use('/member', memberRouter); // http://127.0.0.1:4000/member/write

// 서버를 실행합니다.
app.listen(4000, function () {
  console.log('Server running at http://127.0.0.1:4000');
});
```

app13_2.js

```
// 모듈을 추출합니다.  
var express = require('express');  
  
// 서버를 생성합니다.  
var app = express();  
  
//외부에 있는 router 를 읽어온다  
var board = require('./routes/board');  
var member = require('./routes/member');  
  
// 라우터를 설정합니다.  
app.use('/board', board); // http://127.0.0.1:4000/board/list  
app.use('/member', member); // http://127.0.0.1:4000/member/write  
  
// 서버를 실행합니다.  
app.listen(4000, function () {  
    console.log('Server running at http://127.0.0.1:4000');  
});|
```

/routes/board.js

```
// 라우터를 생성합니다.  
var express = require('express');  
var router = express.Router();  
  
// 라우터를 설정합니다- 별도의 파일로 작성할 수 있음  
router.get('/', function (request, response) {  
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});  
    response.write('<h1>게시판입니다 목록입니다</h1>');  
    response.end();  
});  
router.get('/list/:page', function (request, response) {  
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});  
    response.write('<h1>게시판입니다 목록입니다</h1>');  
    response.end();  
});  
router.get('/write', function (request, response) {  
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});  
    response.write('<h1>게시판입니다 글쓰기 화면입니다 </h1>');  
    response.end();  
});  
router.get('/view', function (request, response) {  
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});  
    response.write('<h1>게시판입니다 상세보기 화면입니다 </h1>');  
    response.end();  
});  
module.exports=router
```

/routes/member.js

```
// 라우터를 생성합니다.
var express = require('express');
var router = express.Router();

// 라우터를 설정합니다- 별도의 파일로 작성할 수 있음
router.get('/', function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
    response.write('<h1>회원게시판입니다 목록입니다 1111111111</h1>');
    response.end();
});
router.get('/list', function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
    response.write('<h1>회원게시판입니다 목록입니다</h1>');
    response.end();
});
router.get('/write', function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
    response.write('<h1>회원게시판입니다 글쓰기 화면입니다 </h1>');
    response.end();
});
router.get('/view', function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/plain; charset=utf-8'});
    response.write('<h1>회원게시판입니다 상세보기 화면입니다 </h1>');
    response.end();
});
module.exports=router
```

app14.js(image 처리)

```
// 모듈을 추출합니다.  
var express = require('express');  
  
// 서버를 생성합니다.  
var app = express();  
  
// 미들웨어를 설정합니다.  
app.use(express.static(__dirname + '/public'));  
  
app.use(function (request, response) {  
    // 응답합니다.  
    response.writeHead(200, { 'Content-Type': 'text/html' });  
    response.write('<h1>This is Chrysanthemum</h1>');  
    response.end('');  
});  
  
// 서버를 실행합니다.  
app.listen(4000, function () {  
    console.log('Server running at http://127.0.0.1:4000');  
});
```

필요 라이브러리 설치

- npm install cookie-parser 쿠키사용
- npm install morgan 로그
- npm install body-parser 파라미터 처리를 간편하게
- npm install connect-multiparty 파일전송
-

app15.js(미들웨어)

```
// 모듈을 추출합니다.  
var express = require('express');  
var morgan = require('morgan');  
  
// 서버를 생성합니다.  
var app = express();  
  
// 미들웨어를 설정합니다.  
app.use(morgan('combined'));  
app.use(function (request, response) {  
    response.send('<h1>express Basic</h1>');  
});  
  
// 서버를 실행합니다.  
app.listen(4000, function () {  
    console.log('Server running at http://127.0.0.1:4000');  
});
```

npm install morgan --save
morgan : 로그제공

app16.js(쿠키 미들웨어)

```
var express = require('express');
var cookieParser = require('cookie-parser');
var app = express();

// 쿠키 미들웨어를 설정합니다.
app.use(cookieParser());

// 라우터를 설정합니다.
app.get('/getCookie', function (request, response) {
    // 응답합니다.
    response.send(request.cookies);
});

app.get('/setCookie', function (request, response) {
    // 쿠키를 생성합니다.
    response.cookie('string', 'cookie');
    response.cookie('json', {
        name: 'cookie',
        property: 'delicious'
    });
    // 응답합니다.
    response.redirect('/getCookie');
});

// 서버를 실행합니다.
app.listen(4000, function () {
    console.log('Server running at http://127.0.0.1:4000');
});
```

app17.js(bodyParser)

```
var fs = require('fs');
var express = require('express');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

// 서버를 생성합니다.
var app = express();
// 미들웨어를 설정합니다.
app.use(cookieParser());
app.use(bodyParser.urlencoded({ extended: false }));

// 라우터를 설정합니다.
app.get('/', function (request, response) { });
app.get('/login', function (request, response) { });
app.post('/login', function (request, response) { });

// 서버를 실행합니다.
app.listen(4000, function () {
  console.log('Server running at http://127.0.0.1:4000');
});
```

app18.js

```
// 모듈을 주출합니다.
var fs = require('fs');
var express = require('express');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');

// 서버를 생성합니다.
var app = express();
// 미들웨어를 설정합니다.
app.use(cookieParser());
app.use(bodyParser.urlencoded({ extended: true }));

// 라우터를 설정합니다.
app.get('/', function (request, response) {
  if (request.cookies.auth) //쿠키에 auth 값이 true면
  {
    response.send('<h1>Login Success</h1>');
  }
  else |
  {
    response.redirect('/login');// /login url로 무조건 보낸다
  }
});
```

app18.js

```
//html 파일을 읽어서 클라이언트에 전송한다
app.get('/login', function (request, response) {
  fs.readFile('./login.html', function (error, data){
    response.send(data.toString());
  });
});

app.post('/login', function (request, response) {
  // 쿠키를 생성합니다.
  var login = request.body.login;
  var password = request.body.password;
  // 출력합니다.
  console.log(login, password);
  console.log(request.body);
  // 로그인을 확인합니다.
  if (login == 'test' && password == '1234') {// 로그인 성공
    response.cookie('auth', true);
    response.redirect('/');
  }
  else
  {
    // 로그인 실패
    response.redirect('/login');
  }
});|
```

app18.js

```
// 서버를 실행합니다.  
app.listen(4000, function () {  
  console.log('Server running at http://127.0.0.1:4000');  
});
```

login.html

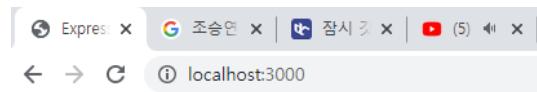
```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8' />
    <title>Login Page</title>
</head>
<body>
    <h1>Login Page</h1>
    <hr />
    <form method="post">
        <table>
            <tr>
                <td><label>Username</label></td>
                <td><input type="text" name="login" /></td>
            </tr>
            <tr>
                <td><label>Password</label></td>
                <td><input type="password" name="password" /></td>
            </tr>
        </table>
        <input type="submit" name="" />
    </form>
</body>
</html>
```

nodemon – 서버 자동 재시작

- nodemon app.js - 자동재시작
- express 생성기 - 기본적인 소스를 모두 만들어준다.
- npm install express-generator -g : 반드시 관리자 권한으로 설정
- #프로젝트 생성하기
- express --view=pug 프로젝트명
- cd 프로젝트명
- npm install
- set DEBUG=프로젝트명:* & npm start

hello 프로젝트 생성하기

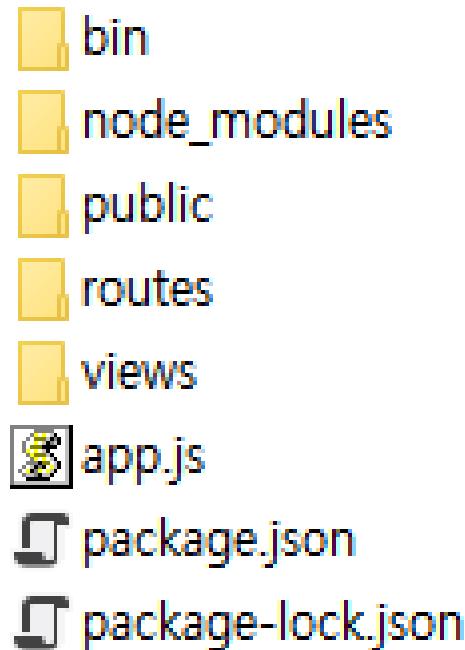
- 주의사항 : 모든 명령분은 콘솔창을 반드시 관리자 권한으로 열어야 한다
- npm install express-generator -g :
- express --view=pug hello
- cd hello
- npm install
- set DEBUG=hello:* & npm start



Express

Welcome to Express

hello2 프로젝트 구조



app.js

- 가장 먼저 실행되는 파일이다
- 자주 사용하는 미들웨어와 기본 라우터가 연결되어 있다

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
```

app.js

```
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', indexRouter);
app.use('/users', usersRouter);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  next(createError(404));
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

crud 만들기(/routes/guestbook.js)

```
var express = require('express');
var router = express.Router();
var mysql = require("mysql");
var pool = mysql.createPool({
  connectionLimit:10,
  host:'localhost',
  user:'user01',
  password:'1234',
  database:'mydb',
  port:5306

});
/* GET home page. */
router.get('/', function(req, res, next) {
  //console.log("dddddd");
  res.redirect('/guestbook/list/1');
});
```

crud 만들기(/routes/guestbook.js)

```
router.get('/list/:page', function(req, res, next) {
  console.log("ddddddddddddd");
  pool.getConnection(function(err, connection){
    var sql ="select id, title, date_format(wdate, '%Y-%m-%d') wdate ";
    sql = sql + " from guestbook limit 0, 5 "; //5개만 가져온다
    connection.query(sql, function(err, rows)
    {
      if(err) console.log("err : " +err);
      //쿼리 실행수 rows 에 데이터 담아온다, 클라이언트에게 rows 보내준다
      res.render('./guestbook/list.ejs', {rows:rows});
      //"rows 라는 이름으로 디비에서 받아온 전체 행을 클라이언트로 보낸다 "
      connection.release();
    });
  });
});
```

crud 만들기(/routes/guestbook.js)

```
✓ router.get('/insert', function(req, res, next) {
  //res.send("dd");
  res.render('./guestbook/insert.ejs');

});

✓ router.post('/insert', function(req, res, next) {
  var body = req.body;
  var title = body.title;
  var contents = body.contents;
  var writer = body.writer
  var datas = [title, writer, contents];

  var sql = "insert into guestbook(title, writer, contents, wdate) ";
  sql = sql + " values(?, ?, ?, now()) ";

  ✓ pool.getConnection(function(err, connection ){
    ✓ connection.query(sql,datas, function(err, rows){
      if(err) console.log("err: " + err);
      res.redirect("/guestbook"); //페이지 이동하기
      connection.release(); //디비 저장 다 되면 연결 객체를 반환한다
    ;})
  });
});
```

crud 만들기(/routes/guestbook.js)

```
router.get('/view/:id', function(req, res, next) {
  //파라미터값 사용하기
  console.log("view : " + req.params.id);
  var id = req.params.id;
  pool.getConnection(function(err, connection){
    if(err) console.log(err);

    var sql ="select id, title, date_format(wdate, '%Y-%m-%d') wdate ";
    sql = sql + ", contents from guestbook where id="+id; //5개만 가져온다
    console.log("쿼리 : " + sql);

    connection.query(sql, function(err, rows)
    {
      if(err) console.log("err : " +err );
      console.log(rows);
      //쿼리 실행수 rows 에 데이터 담아온다, 클라이언트에게 rows 보내준다
      res.render('./guestbook/view.ejs', {rows:rows});
      //"rows 라는 이름으로 디비에서 받아온 전체 행을 클라이언트로 보낸다"
      connection.release();
    });
  });

  module.exports = router;
```

view(/views/guestbook/list.ejs)

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="/stylesheets/style.css" rel="stylesheet">
  </head>
  <body>
    <h1>방명록</h1>
    <table>
      <% for(var i=0; i<rows.length; i++) { %>
        <tr>
          <td><%=rows[i].id%></td>
          <td>
            <a href="/guestbook/view/<%=rows[i].id%>"><%=rows[i].title%></a>
          </td>
          <td><%=rows[i].writer%></td>
          <td><%=rows[i].wdate%></td>
        </tr>
      <% } %>
      <table> <br>
      <a href="/guestbook/insert">글쓰기</a>
    </body>
  </html>
```

view(/views/guestbook/write.ejs)

```
<!DOCTYPE html>
<html >
  <head>
    <title></title>
    <meta charset="UTF-8">
  </head>
  <body>
    <form name="form" method="post">
      제목 : <input type="text" name="title" value=""> <br>
      작성자 : <input type="text" name="writer" value=""> <br>
      내용<br>
      <textarea rows="5" cols="40" name="contents"></textarea>
      <br><br>
      <input type="submit" value="등록">
    </form>
  </body>
</html>
```

view(/views/guestbook/view.ejs)

```
<!DOCTYPE html>
<html>
  <head>
    <title></title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="/stylesheets/style.css" rel="stylesheet">
  </head>
  <body>
    <h1>방명록</h1>
    <table>
      <% for(var i=0; i<rows.length; i++) { %>
        <tr>
          <td><%=rows[i].id%></td>
          <td><%=rows[i].title%></td>
          <td><%=rows[i].writer%></td>
          <td><%=rows[i].contents%></td>
          <td><%=rows[i].wdate%></td>
        </tr>
      <% } %>
      <table> <br>
        <a href="/guestbook/insert">글쓰기</a>
    </body>
  </html>
```

MySQL

MySql 설치하기

- <https://downloads.mariadb.com/MariaDB/mariadb-10.0/winx64-packages/>



← → C <https://downloads.mariadb.com/MariaDB/mariadb-10.0/winx64-packages/>

 MariaDB

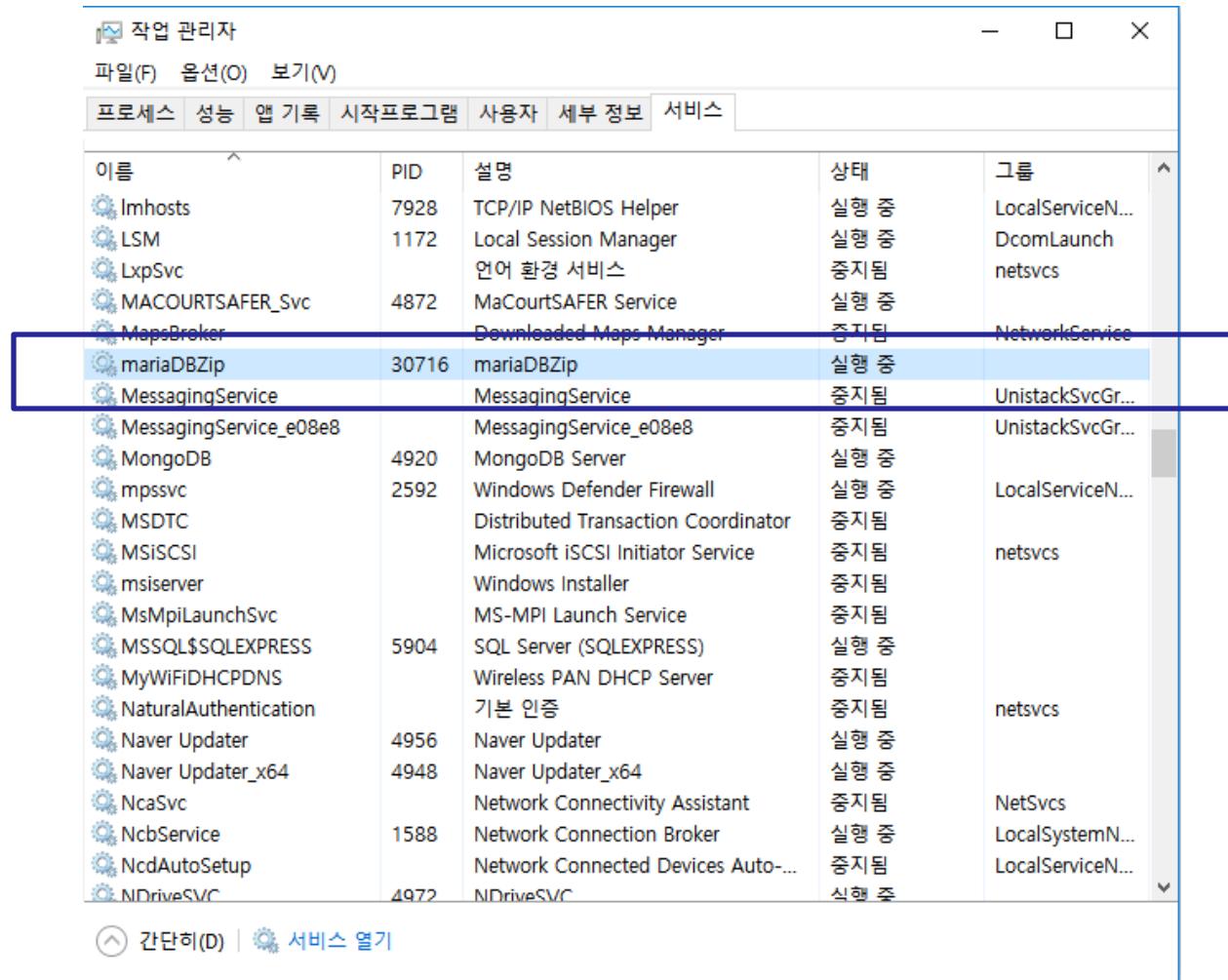
MariaDB/mariadb-10.0/winx64-packages/

- mariadb-10.0.38-winx64.msi
- mariadb-10.0.38-winx64.msi.asc
- mariadb-10.0.38-winx64.zip
- mariadb-10.0.38-winx64.zip.asc
- md5sums.txt
- md5sums.txt.asc
- shalsums.txt
- shalsums.txt.asc
- sha256sums.txt
- sha256sums.txt.asc
- sha512sums.txt
- sha512sums.txt.asc

MySql 설치

- mariadb 실행방법
- c:\mariadb에 압축을 푸다
- cmd 창을 관리자권한으로 실행시킨다.
- cd /mariadb/bin
- mysql_install_db --datadir=C:\mariadb\data --service=mariaDBZip --port=5306 --password=1234

서비스 시작하기



오른쪽
버튼
눌러서
가동을
눌러주자

MySQL과 연동하기

- mysql -u root -p --port=5306
- 패스워드 : 1234

```
C:\mariadb\bin>mysql -u root -p --port=5306
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.3.14-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.003 sec)

MariaDB [(none)]>
```

디비 시작하기

- 디비연결하기

```
mysql -u root -p --port=5306
```

```
Enter your password :1234
```

- 디비 생성

```
CREATE DATABASE mydb default CHARACTER SET UTF8;
```

```
SHOW DATABASES # > #은 mysql에서 주석입니다.
```

- 계정만들기

```
GRANT ALL PRIVILEGES ON mydb.* TO user01@localhost IDENTIFIED BY  
'1234';
```

```
EXIT;
```

- 새로운 계정으로 시작하기

```
mysql -u user01 -p
```

```
use mydb
```

HeidiSql

- <https://www.heidisql.com/>

The screenshot shows the official website for HeidiSQL. At the top, there's a navigation bar with links for Home, Downloads, Screenshots, Forum, Donate, Bugtracker, and Help. A green banner features the HeidiSQL logo and a "What's this?" section. Below the banner, there's a brief introduction about the software, download links, and two screenshots of the HeidiSQL interface. The main content area includes news items and a forum section.

https://www.heidisql.com

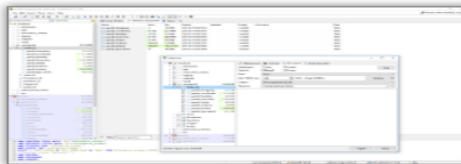
HeidiSQL

Home Downloads Screenshots Forum Donate Bugtracker Help

What's this?

HeidiSQL is free software, and has the aim to be easy to learn. "Heidi" lets you see and edit data and structures from computers running one of the database systems MariaDB, MySQL, Microsoft SQL or PostgreSQL. Invented in 2002 by Ansgar, with a development peak between 2009 and 2013, HeidiSQL belongs to the most popular tools for MariaDB and MySQL worldwide.

Download HeidiSQL, read further about [features](#), take part in [discussions](#) or see some [screenshots](#).

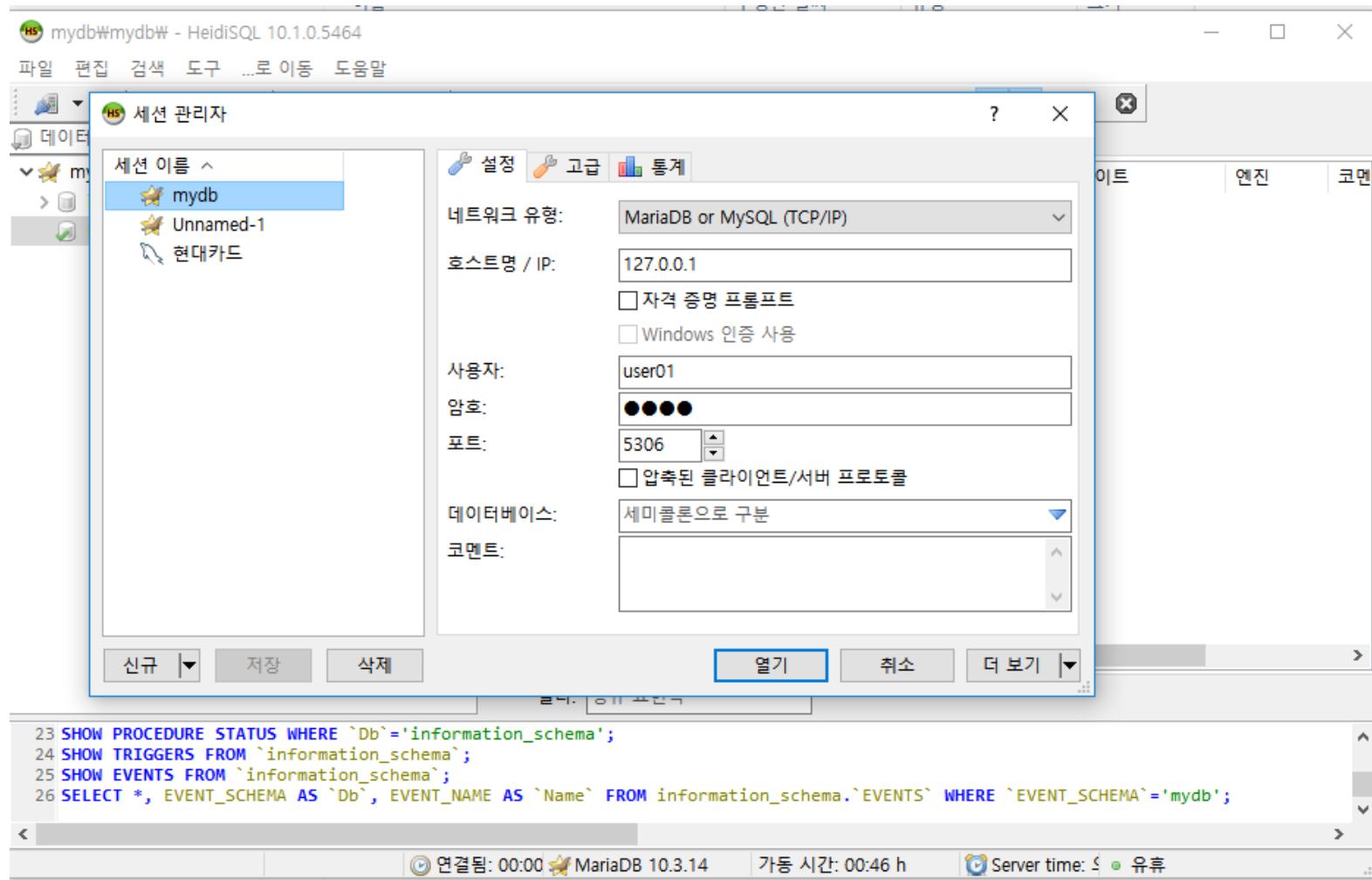


News

07 Apr Filter box for snippets, functions etc.
I often found myself searching in the many snippets I stored over the years. To make it quick and easy to find anything in the query helpers box, I just added a filter box on top of it. The tree wi ...

26 Jan HeidiSQL 10.1 bugfix release
This is a new release which mainly fixes the installer, which named the executable in your program files folder "heidisql32.exe", not "heidisql.exe". Sounds minor, but I suppose there are quite a f ...

HeidiSql



guestbook table 작성

```
CREATE TABLE guestbook (
    id INT(11) NOT NULL AUTO_INCREMENT,
    title VARCHAR(300) NULL DEFAULT NULL,
    contents LONGTEXT NULL DEFAULT NULL,
    writer VARCHAR(50) NULL DEFAULT NULL,
    wdate DATE NULL DEFAULT NULL,
    PRIMARY KEY (id)
);
```

데이터추가

```
insert into guestbook(title, contents, writer, wdate) values
('제목1', '내용1', '홍길동1', now());
insert into guestbook(title, contents, writer, wdate) values
('제목2', '내용2', '홍길동2', now());
insert into guestbook(title, contents, writer, wdate) values
('제목3', '내용3', '홍길동3', now());
insert into guestbook(title, contents, writer, wdate) values
('제목4', '내용4', '홍길동4', now());
insert into guestbook(title, contents, writer, wdate) values
('제목5', '내용5', '홍길동5', now());

select * from guestbook;
```

mysql연동

```
import pymysql

conn = pymysql.connect(host = 'localhost', user = 'user01', password =
'1234' ,db = 'mydb', port=5306)
curs = conn.cursor()

sql = "SELECT * FROM guestbook" # 실행 할 쿼리문 입력
curs.execute(sql) # 쿼리문 실행

rows = curs.fetchall() # 데이터 패치

for row in rows :
    print(row)

conn.close()
```

mysql연동

```
sql = """
insert into guestbook(title, contents, writer, wdate)
values (%s, %s, %s, now())
"""

curs.execute(sql, ('제목입니다.', '내용입니다.', '작성자'))
curs.execute(sql, ('제목입니다.', '내용입니다.', '작성자'))
```

```
sql = """update guestbook
set title = '제목을 수정합니다.'
where id=1
"""

curs.execute(sql)
```

```
sql = "delete from guestbook where id=%s"
curs.execute(sql, 6)
```

MongoDB

MongoDB 설치하기

- 몽고디비 다운로드 및 설치
- <https://www.mongodb.com/download-center?jmp=nav#community>
- 설치 후 사용하기
- 사용 방법
- 환경 변수 path 에 다음 경로 추가하기
- C:\Program Files\MongoDB\Server\4.0\bin

MongoDB 사용하기

- 몽고디비 서버 작동하기
- mongod
- 데이터 들어갈 경로 없다고 함
- 경로 만들기 c:/data/db
- 데이터베이스
- c:/data/db
- 몽고디비 서버 작동하기
- mongod

MongoDB 사용하기

- 서버와 통신하기
- mongo
- 명령어
- >db
- test
- 디비를 모두 보여줘라
- >show dbs
- admin 0.000GB
- config 0.000GB
- local 0.000GB

MongoDB 사용하기

디비 사용 명령어 - 없으면 알아서 만들어준다

>use mydb

디비 삭제 명령어

>db.dropDatabase()

MongoDB에서는 table도 record도 없다.

대신 비슷한 개념으로 각각 collection과 document가 존재한다.

RDBMS로 따지면 table과 record와 비슷한개념이다.

> show collections

> db.createCollection('person')

{ "ok" : 1 }

> show collections

person

MongoDB 사용하기

- 데이터 추가하기 형식
- db.<컬렉션명>.insert(<json>)
 - db.person.insert({'name':'홍길동', 'age':26, 'gender':'m'})
 - db.person.insert({'name':'장길산', 'age':62, 'gender':'m'})
 - db.person.insert({'name':'임꺽정', 'age':28, 'gender':'m'})
 - db.person.find()

MongoDB 사용하기

```
db.createCollection("member");
db.person.remove();
db.member.insert({'member_id':'test1', 'password':'1234',
  'username':'홍길동', 'phone':'010-0000-0000'});
db.member.insert({'member_id':'test2', 'password':'1234',
  'username':'김길동', 'phone':'010-1111-1111'});
db.member.insert({'member_id':'test3', 'password':'1234',
  'username':'이길동', 'phone':'010-2222-2222'});
db.member.insert({'member_id':'test4', 'password':'1234',
  'username':'장길동', 'phone':'010-3333-3333'});
db.member.insert({'member_id':'test5', 'password':'1234',
  'username':'고길동', 'phone':'010-4444-4444'});

db.member.find();
```

MongoDB 사용하기

- 데이터 검색하기

1. 모든 데이터 출력

db.<컬렉션명>.find()

db.person.find()

2. 조건 출력

db.<컬렉션명>.find(<json>)

db.person.find({'name':'홍길동'})

find내에 조건을 넣지 않는다면 현재 컬렉션 내의 모든 데이터를 보여준다.

만약 조건을 건다면 해당 조건에 일치하는 데이터만 보여준다.

만약 해당조건에 RDBMS처럼 LIKE(부분일치)로 데이터를 보고싶다면 정규표현식을 조건으로 사용하면된다.

MongoDB 사용하기

- 데이터수정하기

```
db.<컬렉션명>.update(<json1>, <json2>)
```

```
db.person.update({'name':'홍길동'}, {$set:{'age':40}})
```

```
db.person.find()
```

- 데이터 삭제하기

```
db.<컬렉션명>.remove(<json>)
```

```
db.person.remove({'name':'홍길동'})
```

```
db.person.find()
```

MongoDB 사용하기

- 조건 부여하기
- \$eq (equals) 주어진 값과 일치하는 값
- \$gt (greater than) 주어진 값보다 큰 값
- \$gte (greather than or equals) 주어진 값보다 크거나 같은 값
- \$lt (less than) 주어진 값보다 작은 값
- \$lte (less than or equals) 주어진 값보다 작거나 같은 값
- \$ne (not equal) 주어진 값과 일치하지 않는 값
- \$in 주어진 배열 안에 속하는 값
- \$nin 주어진 배열 안에 속하지 않는 값

```
db.person.find({age:{$gte:40, $lt:60}})
```

```
db.person.find({age:{$gte:40}})
```

```
db.person.find({name:{$in:["장", "임"]}})
```

MongoDB 사용하기

- 자동 시퀀스

```
db.customSequences.insert(  
    {  
        _id: "hero", // 시퀀스 이름  
        seq: 0      // 초기 값  
    }  
)
```

MongoDB 사용하기

```
db.system.js.save(  
{  
    "_id" : "getNextSequence",  
    "value" : function(name) {  
        var ret = db.customSequences.findAndModify(  
            {  
                query: { _id:name},  
                update: {$inc: { seq:1}},  
                new: true  
            });  
        return ret.seq;  
    }  
});
```

```
db.loadServerScripts()  
getNextSequence("hero");
```

node.js 와 몽고디비 연동하기

```
npm install --save express mongoose body-parser
```

```
id, title, writer, date_format(wdate, '%Y-%m-%d') wdate
```

```
use mydb
db.createCollection("guestbook");
db.guestbook.remove({});
db.guestbook.insert({'id':1, 'title':'제목1', 'contents':'내용1', 'writer':'홍길동',
'wdate':'2019-03-15'});
db.guestbook.insert({'id':2, 'title':'제목2', 'contents':'내용2', 'writer':'임꺽정',
'wdate':'2019-03-16'});
db.guestbook.insert({'id':3, 'title':'제목3', 'contents':'내용3', 'writer':'장길산',
'wdate':'2019-03-17'});
db.guestbook.insert({'id':4, 'title':'제목4', 'contents':'내용4', 'writer':'홍경래',
'wdate':'2019-03-18'});
db.guestbook.insert({'id':5, 'title':'제목5', 'contents':'내용5', 'writer':'장승업', 'wdate':new
Date()});
db.guestbook.find();
```

데이터 전부보기

```
db.guestbook.find();
```

restful api

```
router.get('/list/:page', function(req, res, next) {
  console.log("guestbook3");
  pool.getConnection(function(err, connection){
    var sql ="select id, title, writer, contents, date_format(wdate, '%Y-%m-%d') wdate ";
    sql = sql + " from guestbook limit 0, 10 "; //5개만 가져온다
    connection.query(sql, function(err, rows)
    {
      if(err) console.log("err : " +err);
      //쿼리 실행후 rows 에 데이터 담아온다, 클라이언트에게 rows 보내준다
      //test= JSON.stringify( {data:rows})
      res.send( {data:rows});
      //"rows 라는 이름으로 디비에서 받아온 전체 행을 클라이언트로 보낸다 "
      connection.release();
    });
  });
});
```

restful api

POST ▼ http://127.0.0.1:3000/guestbook3/insert **Send** ▼

Params Authorization Headers (11) **Body** ● Pre-request Script Tests Settings C

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

KEY	VALUE	DESCRIPTION	...
<input checked="" type="checkbox"/> title	제목1		
<input checked="" type="checkbox"/> writer	홍길동1		
<input checked="" type="checkbox"/> contents	내용1		