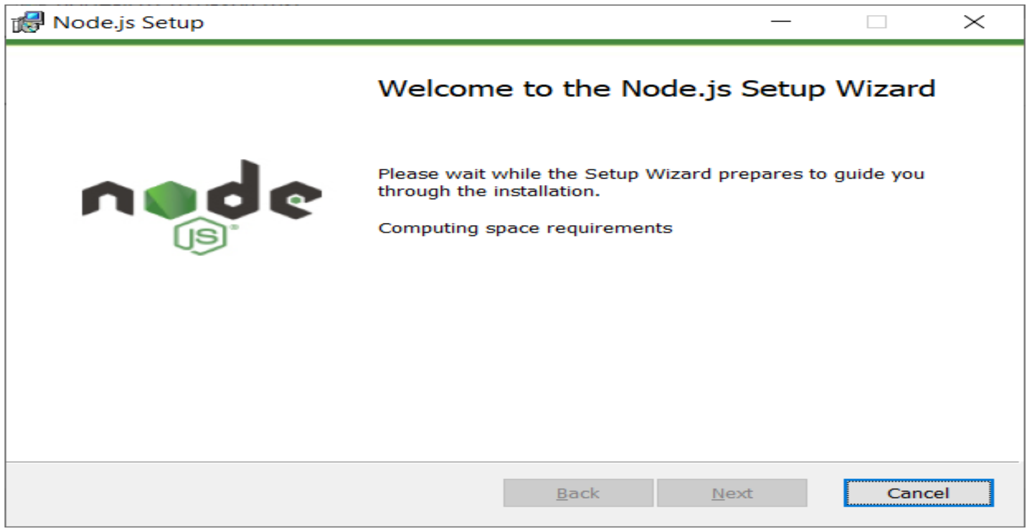
Introduction to TypeScript

Installation

- Node.js is an open source cross-platform runtime environment for server-side java script without a browser support
  - https://nodejs.org/en/download/



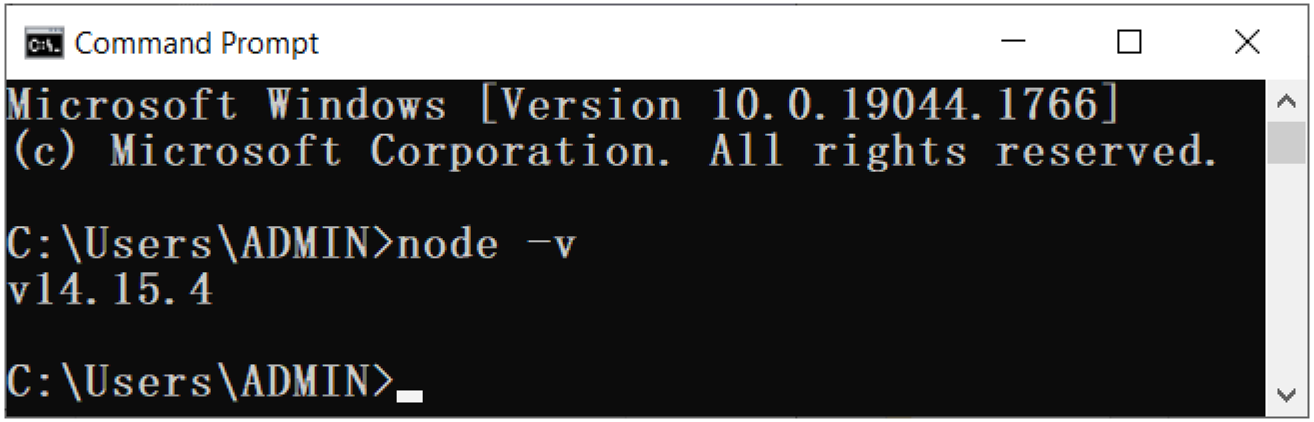
Welcome to the Node.js Setup Wizard

Please wait while the Setup Wizard prepares to guide you through the installation.

Computing space requirements

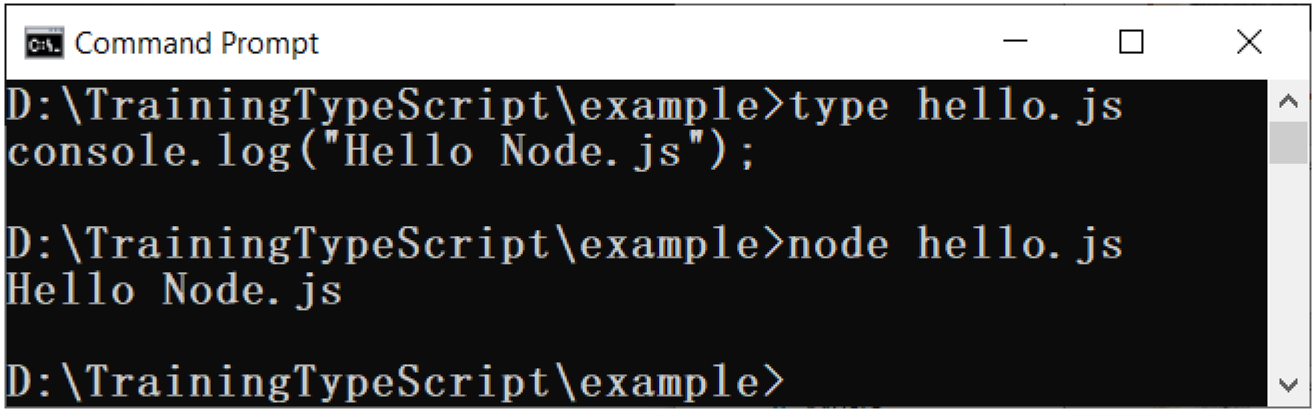
Installation

- node.js version



```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>node -v
v14.15.4

C:\Users\ADMIN>_
```

- hello.js

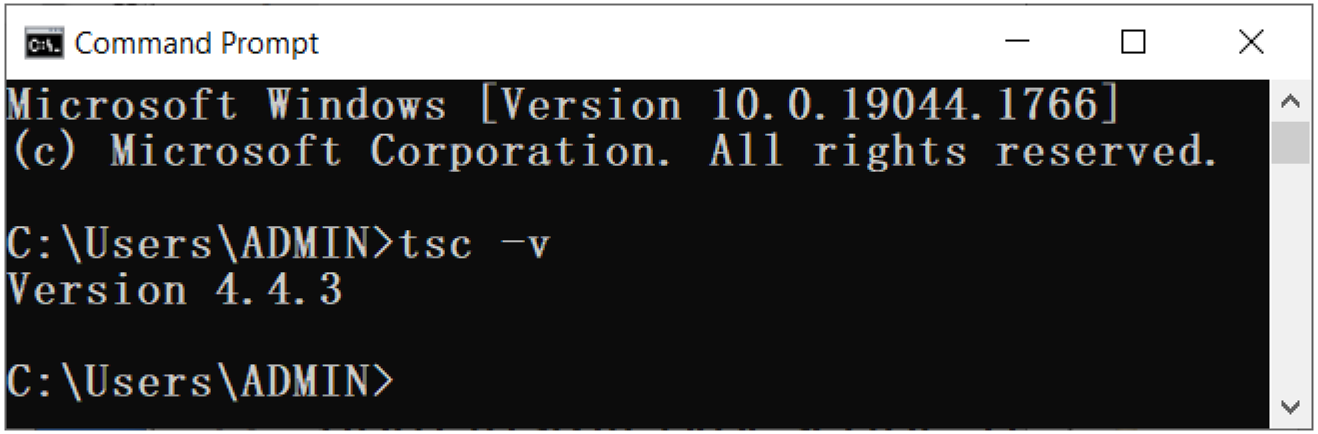


```
D:\TrainingTypeScript\example>type hello.js
console.log("Hello Node.js");

D:\TrainingTypeScript\example>node hello.js
Hello Node.js

D:\TrainingTypeScript\example>
```

Installation

- typescript

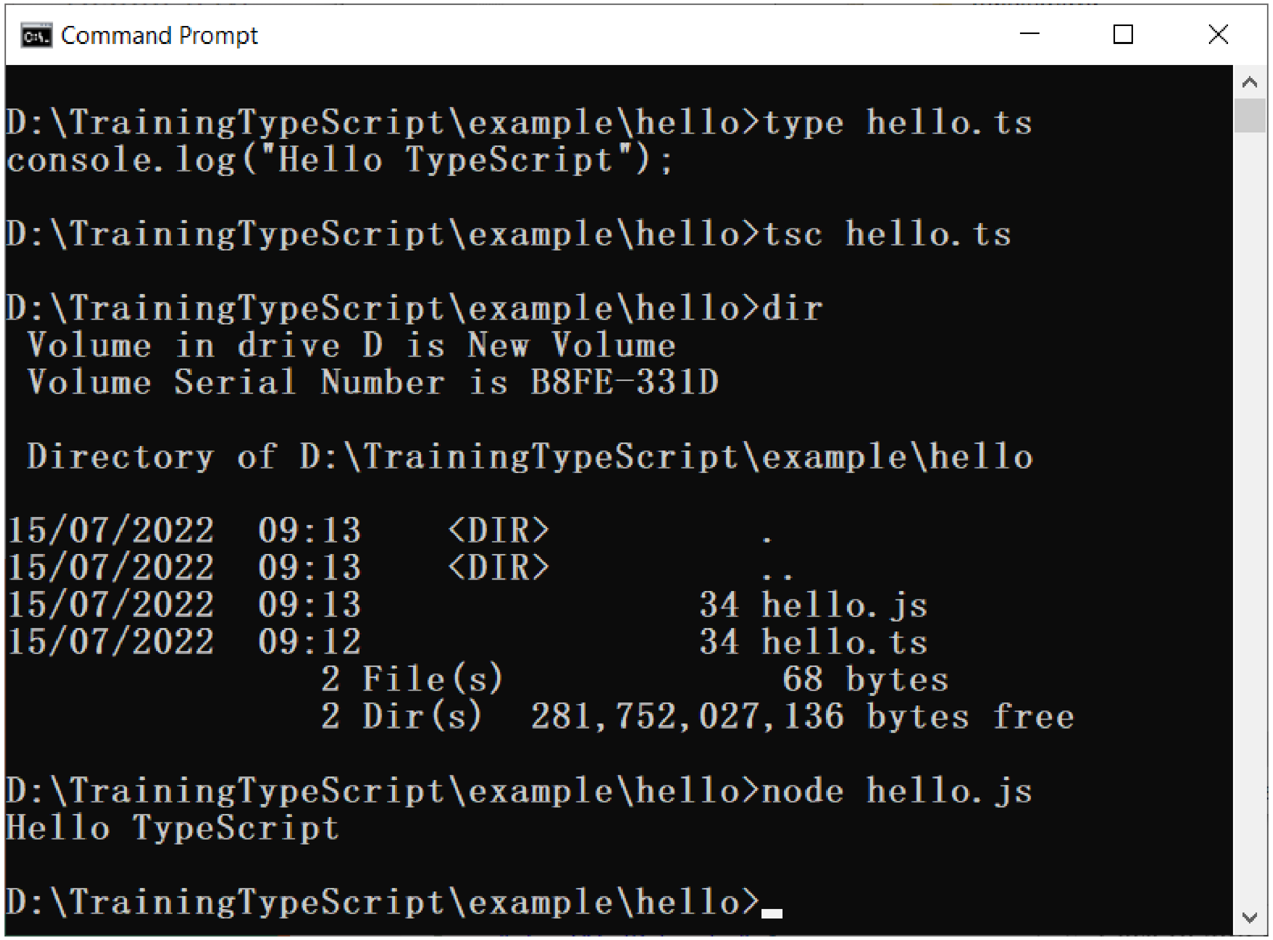
```
npm install typescript --save-dev    //As dev dependency
npm install typescript -g            //Install as a global module
npm install typescript@latest -g     //Install latest if you have an older version
npm install typescript@4.4.3         //Specify version
```



```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADMIN>tsc -v
Version 4.4.3

C:\Users\ADMIN>
```

Compiler

- tsc



Command Prompt

```
D:\TrainingTypeScript\example\hello>type hello.ts
console.log("Hello TypeScript");

D:\TrainingTypeScript\example\hello>tsc hello.ts

D:\TrainingTypeScript\example\hello>dir
 Volume in drive D is New Volume
 Volume Serial Number is B8FE-331D

 Directory of D:\TrainingTypeScript\example\hello

15/07/2022  09:13    <DIR>          .
15/07/2022  09:13    <DIR>          ..
15/07/2022  09:13                34 hello.js
15/07/2022  09:12                34 hello.ts
               2 File(s)             68 bytes
               2 Dir(s)  281,752,027,136 bytes free

D:\TrainingTypeScript\example\hello>node hello.js
Hello TypeScript

D:\TrainingTypeScript\example\hello>_
```

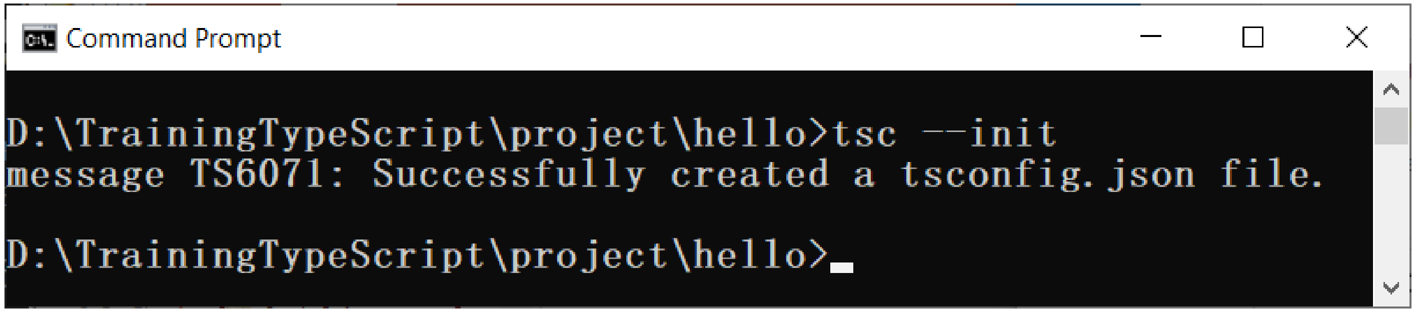
Compiler Flag

- tsc

Flag	Compiler flag & Description
--help	Displays the help manual
--module	Load external modules
--target	Set the target ECMA version
--declaration	Generates an additional .d.ts file
--removeComments	Removes all comments from the output file
--out	Compile multiple files into a single output file
--sourcemap	Generate a sourcemap (.map) files
--module noImplicitAny	Disallows the compiler from inferring the any type
--watch	Watch for file changes and recompile them on the fly

Compiler Configuration

- tsconfig.json
  - tsc --init



```
D:\TrainingTypeScript\project\hello>tsc --init
message TS6071: Successfully created a tsconfig.json file.

D:\TrainingTypeScript\project\hello>_
```

Compiler Configuration

- tsconfig.json - I

```
{
  "compilerOptions": {
    // สั่งให้คอมไพล์ออกมาเป็น JavaScript ES6
    "target": "es6",
    // ชื่อโฟลเดอร์ที่ output ไฟล์ JavaScript ที่คอมไพล์แล้ว
   "outDir": "./dist",
    // ชื่อโฟลเดอร์ sourcecode ไฟล์ TypeScript
    "rootDir": "./src",
    // หากใช้งานกับ React คือมีไฟล์ .tsx ให้คอมไพล์เป็น .jsx ด้วย
   "jsx": "react",
    // หากใช้กับ node
    "moduleResolution": "node",
  },
```

Compiler Configuration

- tsconfig.json - II

```
// กำหนดขอบเขตของไฟล์ที่จะให้คอมไพล์
 // เช่น ทุกไฟล์ที่นามสกุล .ts ในโฟลเดอร์ไหนก็ได้ใต้ /src
  "include": [
      "src/**/*.ts"
  ],

  // กำหนดไฟล์และโฟลเดอร์ที่ไม่ต้องคอมไพล์
 // เช่นโฟลเดอร์ node_modules หรือไฟล์ spec
  "exclude": [
      "node_modules",
      "**/*.spec.ts"
  ]
}
```

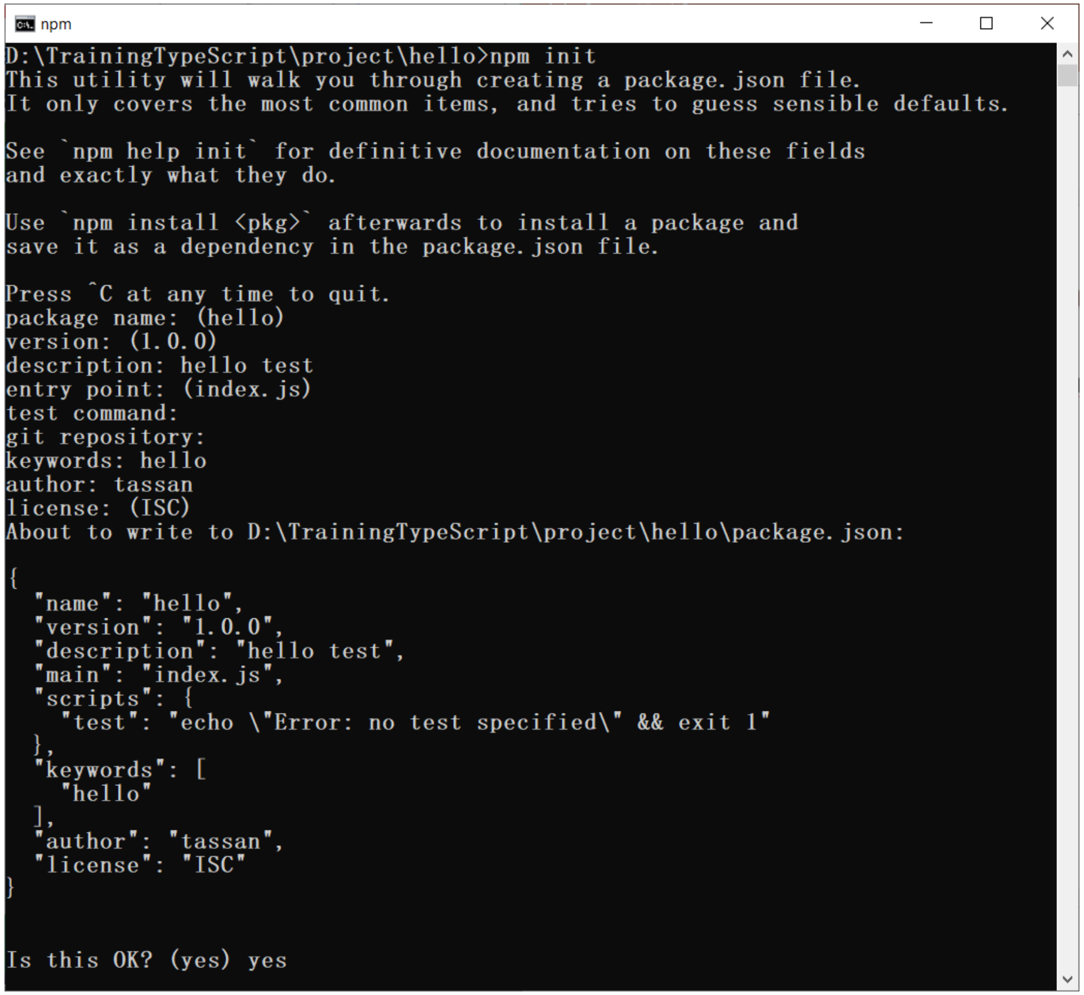
Compiler Configuration

- tsconfig.json

```
{
  "compilerOptions": {
              "target": "es5",
              "outDir": "./dist",
              "rootDir": "./src",
              "module": "commonjs",
              "esModuleInterop": true,
              "forceConsistentCasingInFileNames": true,
              "strict": true,
              "skipLibCheck": true
  },
```

Create Project

- package.json
  - npm init
  - npm init -y



```
D:\TrainingTypeScript\project\hello>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hello)
version: (1.0.0)
description: hello test
entry point: (index.js)
test command:
git repository:
keywords: hello
author: tassan
license: (ISC)
About to write to D:\TrainingTypeScript\project\hello\package.json:

{
  "name": "hello",
  "version": "1.0.0",
  "description": "hello test",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "hello"
  ],
  "author": "tassan",
  "license": "ISC"
}


Is this OK? (yes) yes
```

Create Project

- package.json

```
npm install typescript --save-dev
npm install ts-node --save-dev
npm install ts-node-dev --save-dev
npm install @types/node --save-dev
```

```
 "devDependencies": {
                "@types/node": "^18.0.4",
                "ts-node": "^10.9.1",
                "ts-node-dev": "^2.0.0",
                "typescript": "^4.7.4"
}
```

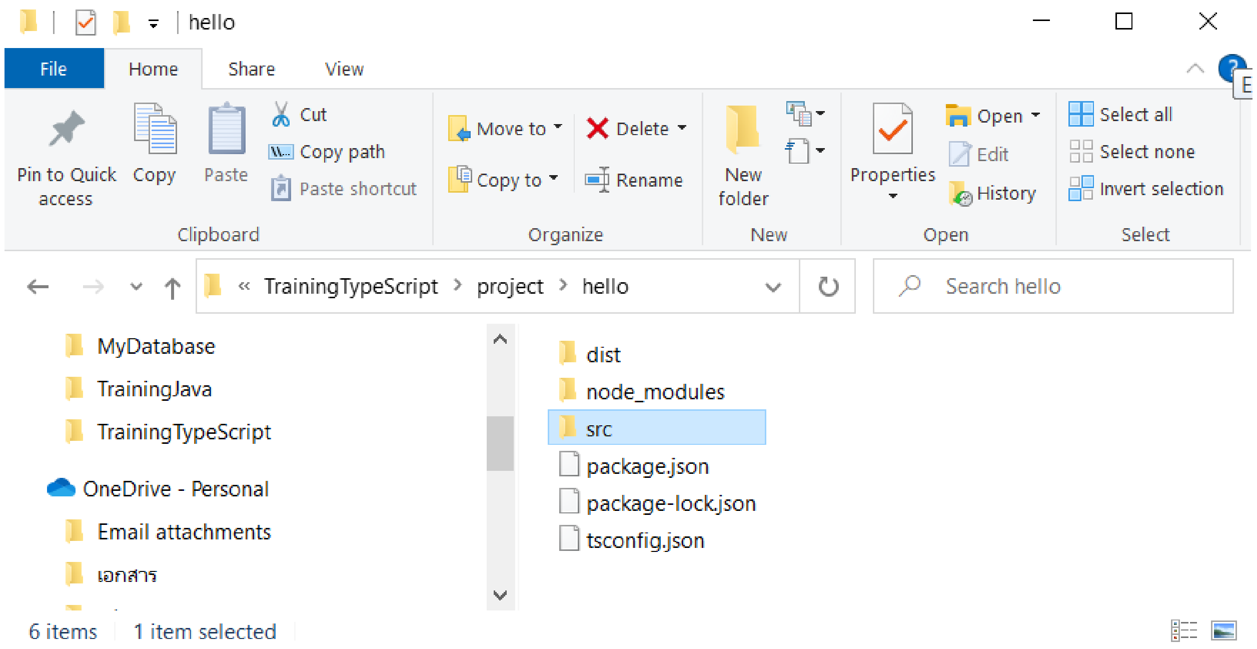
Create Project

- package.json

```
{
  "name": "hello",
  "version": "1.0.0",
  "description": "hello test",
  "main": "index.js",
  "scripts": {
    "build": "tsc --project ./",
    "dev": "ts-node-dev src/index.ts",
    "prod": "node dist/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "hello"
  ],
  "author": "tassan",
  "license": "ISC",
  "devDependencies": {
    "@types/node": "^18.0.4",
    "ts-node": "^10.9.1",
    "ts-node-dev": "^2.0.0",
    "typescript": "^4.7.4"
  }
}
```

Create Project

- folder structure



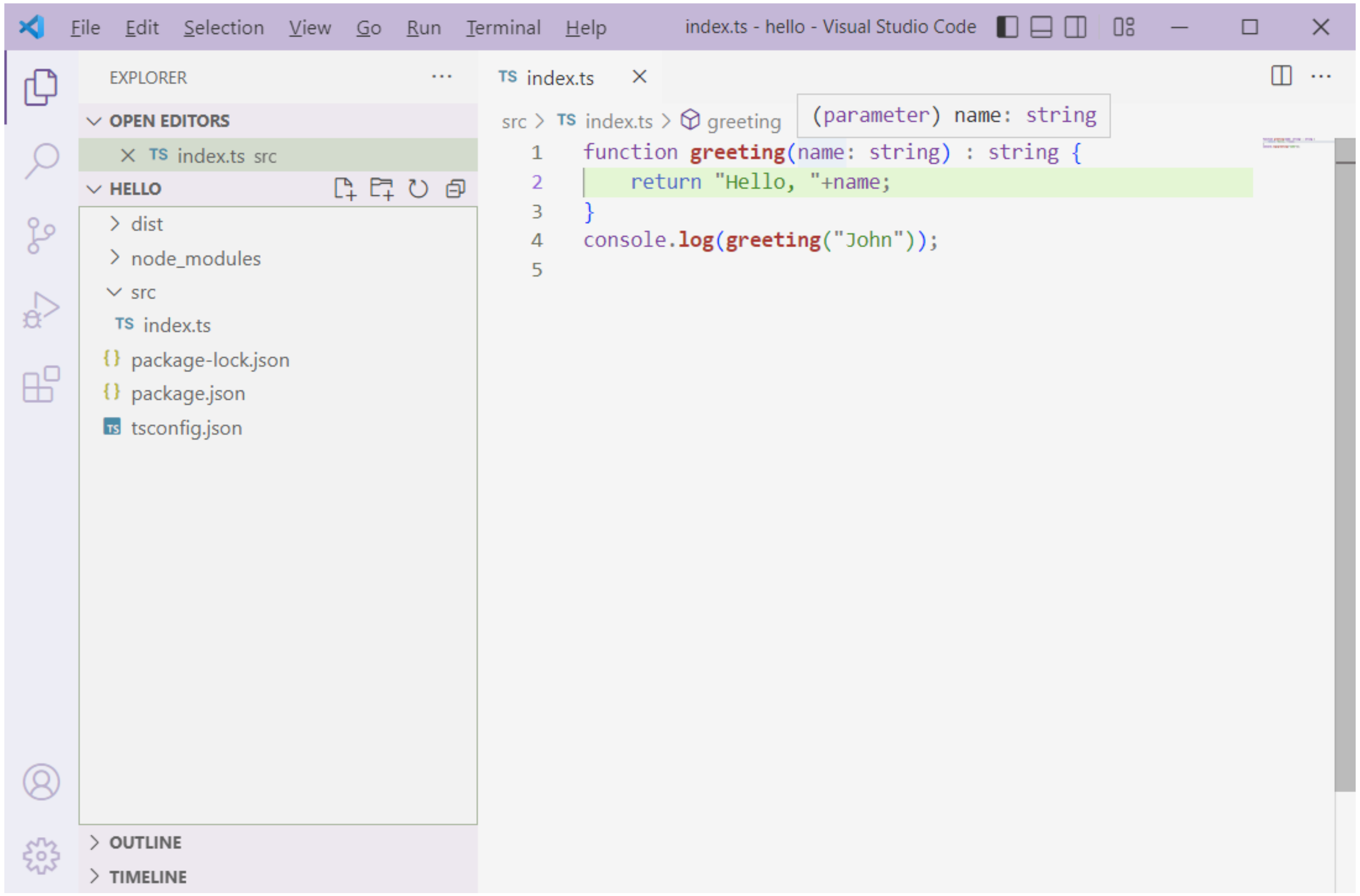
« TrainingTypeScript › project › hello

dist
node_modules
src
package.json
package-lock.json
tsconfig.json

6 items | 1 item selected

Create Project

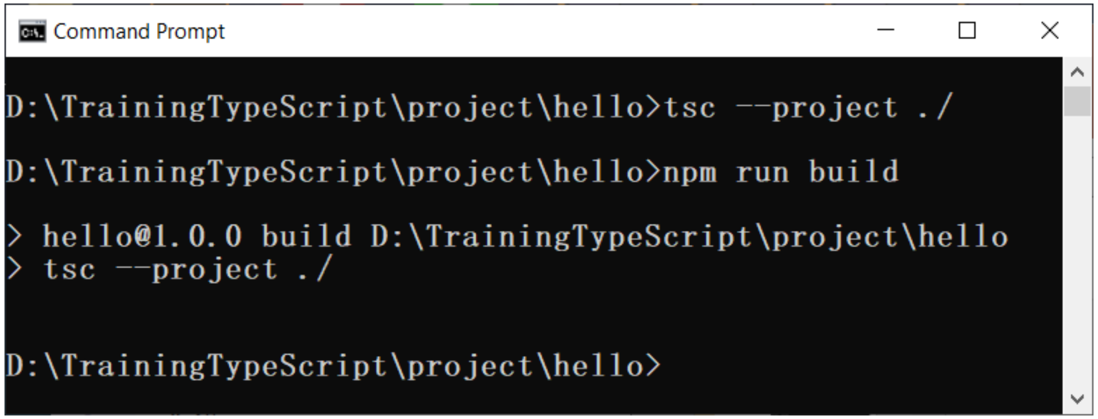
- IDE



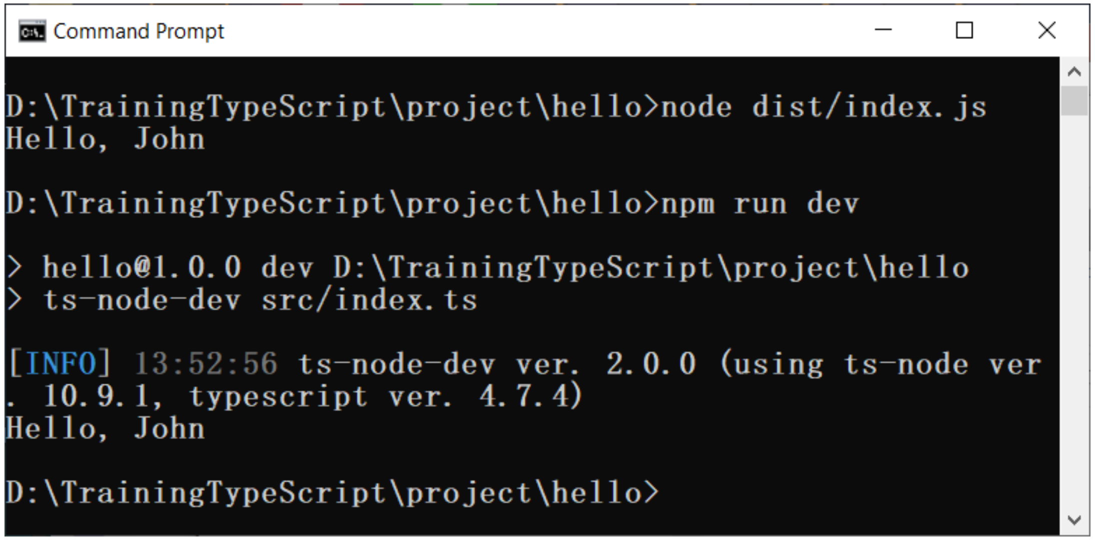
```
function greeting(name: string) : string {
    return "Hello, "+name;
}
console.log(greeting("John"));
```

Create Project

- build and run



```
D:\TrainingTypeScript\project\hello>tsc --project ./

D:\TrainingTypeScript\project\hello>npm run build

> hello@1.0.0 build D:\TrainingTypeScript\project\hello
> tsc --project ./


D:\TrainingTypeScript\project\hello>
```



```
D:\TrainingTypeScript\project\hello>node dist/index.js
Hello, John

D:\TrainingTypeScript\project\hello>npm run dev

> hello@1.0.0 dev D:\TrainingTypeScript\project\hello
> ts-node-dev src/index.ts

[INFO] 13:52:56 ts-node-dev ver. 2.0.0 (using ts-node ver
. 10.9.1, typescript ver. 4.7.4)
Hello, John

D:\TrainingTypeScript\project\hello>
```

Application

- arguments
  - process.argv

```
import { Arguments } from "./utils/Arguments";

process.argv.forEach(function (arg, index, array) {
    console.log(index + ': ' + arg);
});
let args = process.argv.slice(2);
console.log("args = "+args);
let user = Arguments.getString(args,'','-user');
let email =
Arguments.getString(args,'tassun_oro@hotmail.com','-
email');
console.log("Hello "+user+" : "+email);
```

args.ts

Application

- arguments – Arguments.ts

```
class Arguments {
    private static isParameterOption(argument?: string) : boolean {
        if(argument!=null && argument.length>0 && argument.charAt(0)=='-') {
            return true;
        }
        return false;
    }

    public static getString(args?: string[],defaultValue?: string,...options:
string[]) : string | undefined {
        if(args!=null && args.length>0) {
            for(let i = 0,isz=args.length; i<isz; i++) {
                let para = args[i];
                for(let j=0; j<options.length; j++) {
                    if(para == options[j] && (args.length>(i+1))
                    && !this.isParameterOption(args[i+1])) {
                        return args[i+1];
                    }
                }
            }
        }
        return defaultValue;
    }
```

Application

- arguments

```
    public static getDate(args?: string[],defaultValue?:
Date,...options: string[]) : Date | undefined {
        if(args!=null && args.length>0) {
            for(let i = 0,isz=args.length; i<isz; i++) {
                let para = args[i];
                for(let j=0; j<options.length; j++) {
                    if(para == options[j] && (args.length>(i+1))
                    && !this.isParameterOption(args[i+1])) {
                        //date in format : yyyy-MM-dd or yyyy-MM-
ddTHH:mm:ss
                        return new Date(args[i+1]);
                    }
                }
            }
        }
        return defaultValue;
    }
```

Application

- arguments

```
    public static getInteger(args?: string[],defaultValue?:
number,...options: string[]) : number | undefined {
        if(args!=null && args.length>0) {
            for(let i = 0,isz=args.length; i<isz; i++) {
                let para = args[i];
                for(let j=0; j<options.length; j++) {
                    if(para == options[j] && (args.length>(i+1))
                    && !this.isParameterOption(args[i+1])) {
                        return parseInt(args[i+1]);
                    }
                }
            }
        }
        return defaultValue;
    }

}
export {
    Arguments
}
```

Application

- Environment
  - process.env

```
import { HTTP_PORT, RESOURCES_PATH } from
"./utils/EnvironmentVariable";

console.log("USERNAME",process.env.USERNAME);
console.log("HTTP_PORT",HTTP_PORT);
console.log("RESOURCES_PATH",RESOURCES_PATH);
```

envs.ts

Application

- environment - EnvironmentVariable.ts

```
import os from "os";
export const DB_URL = process.env.DB_URL ||
"mysql://root:root@localhost:3306/accessdb?charset=utf8&conn
ectionLimit=10";
export const DB_HOST = process.env.DB_HOST || "localhost";
export const DB_USER = process.env.DB_USER || "root";
export const DB_PASSWORD = process.env.DB_PASSWORD ||
"root";
export const DB_DATABASE = process.env.DB_DATABASE ||
"accessdb";
export const DB_PORT = parseInt(process.env.DB_PORT ||
"3306") || 3306;
export const HTTP_PORT = parseInt(process.env.HTTP_PORT ||
"8080") || 8080;
export const RESOURCES_PATH = process.env.RESOURCES_PATH ||
os.tmpdir();
```

Application

- environment
  - .env
    - npm install dotenv --save

```
import 'dotenv/config';
import { HTTP_PORT, RESOURCES_PATH } from
"./utils/EnvironmentVariable";

console.log("USERNAME",process.env.USERNAME);
console.log("HTTP_PORT",HTTP_PORT);
console.log("RESOURCES_PATH",RESOURCES_PATH);
```

```
// .env file
HTTP_PORT=8088
RESOURCES_PATH=c:\temp
```

dotenvs.ts

Application

- configuration
  - npm install config --save
  - npm install @types/config –save-dev

```
import config from 'config';

console.log("DB_URL",config.get('DB_URL'));
console.log('NODE_CONFIG_DIR: ' +
config.util.getEnv('NODE_CONFIG_DIR'));

if(config.has("authentications")) {
    let authenlist = config.get("authentications") as any;
    console.log("authentications",authenlist);
    for(let i=0,isz=authenlist.length;i<isz;i++) {
        console.log(JSON.stringify(authenlist[i]));
    }
}
```

conf.ts

Application

- configuration
  - /config/default.json

```
{
    "DB_URL":"mysql://root:root@localhost:3306/accessdb?charset=utf8
&connectionLimit=10",
    "authentications": [
        { "authtype":"WOW", "domainname":"freewillsolutions.com",
"tenanturl":"https://rm.ezwow.io/ezwow-gateway/api/login",
"basedn":"" , "enabled": true },
        { "authtype":"NEWS", "domainname":"freewillsolutions.com",
"tenanturl":"https://notify-
devvoffice.freewillsolutions.com/ezlogin", "basedn":"" , "enabled":
true },
        { "authtype":"AD", "domainname":"freewillgroup.com",
"tenanturl":"ldap://10.22.91.24:389",
"basedn":"DC=freewillgroup,DC=com" , "enabled": true }
    ]
}
```

Database

- mysql
  - npm install mysql –save
  - npm install @types/mysql --save-dev
  - connection

```
import { DB_URL } from "./utils/EnvironmentVariable";
import { Connection, MysqlError } from 'mysql';
import mysql from 'mysql';

const conn: Connection = mysql.createConnection(DB_URL);
```

my_conn.ts

Database

- mysql
  - connection - retrieve

```
conn.connect((cerr: MysqlError) => {
    if(cerr) {
        console.error(cerr);
    }
    let bookid = "100";
    let sql = "select * from book where bookid = ? ";
    conn.query(sql,[bookid],(qerr, rows, fields) => {
        if(qerr) {
            console.error(qerr);
            return;
        }
        if(rows && rows.length>0) {
            let row = rows[0];
            console.log("row",row);
        }
    });
    conn.end();
});
```

my_conn.ts

Database

- mysql
  - connection - update

```
import { DB_URL } from "./utils/EnvironmentVariable";
import { Connection, MysqlError } from 'mysql';
import mysql from 'mysql';

const conn: Connection = mysql.createConnection(DB_URL);
conn.connect((cerr: MysqlError) => {
    if(cerr) {
        console.error(cerr);
    }
    let bookid = "100";
    let title = "Docter Sleep";
    let sql = "update book set title = ? where bookid = ? ";
    conn.query(sql,[title,bookid],(qerr, rows, fields) => {
        if(qerr) {
            console.error(qerr);
            return;
        }
        console.log("affected "+rows.affectedRows+" rows.");
    });
    conn.end();
});
```

my_conn_update.ts

Database

- mysql
  - connection pool

```
import { DB_URL } from "./utils/EnvironmentVariable";
import { Connection, Pool, PoolConnection, MysqlError } from
'mysql';
import mysql from 'mysql’;

const pool: Pool = mysql.createPool(DB_URL);
```

my_pool.ts

Database

- mysql
  - connection pool - retrieve

```
pool.getConnection((cerr: MysqlError, conn: Connection) => {
    if(cerr) { console.error(cerr); }
    let bookid = "100";
    let sql = "select * from book where bookid = ? ";
    conn.query(sql,[bookid],(qerr, rows, fields) => {
        if(qerr) {
            console.error(qerr);
            return;
        }
        if(rows && rows.length>0) {
            let row = rows[0];
            console.log("row",row);
        }
    });
    let pconn : PoolConnection = conn as PoolConnection;
    pconn.release();
    pool.end();
});
```

my_pool.ts

Database

- mongodb
  - npm install mongodb –save
  - npm install @types/mongodb --save-dev

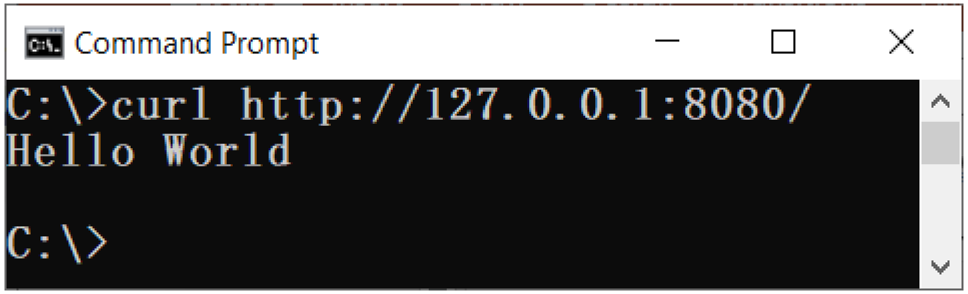
```
import { MongoClient } from 'mongodb';

const uri =
"mongodb+srv://tsodb:tsopassword@cluster0.8mht0.mongodb.net/mydb?ret
ryWrites=true&w=majority";
const client = new MongoClient(uri);
client.connect(err => {
  const collection = client.db("mydb").collection("tso");
  collection.find({}).toArray(function(err, result) {
    if (err) throw err;
    console.log(result);
    client.close();
  });
});
```

mongo.ts

Web Application

- http server

```
import http from 'http';

http.createServer(function handler(req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World');
}).listen(8080);
console.log('Server running at http://127.0.0.1:8080/');
```



```
C:\>curl http://127.0.0.1:8080/
Hello World

C:\>
```

http1.ts

Web Application

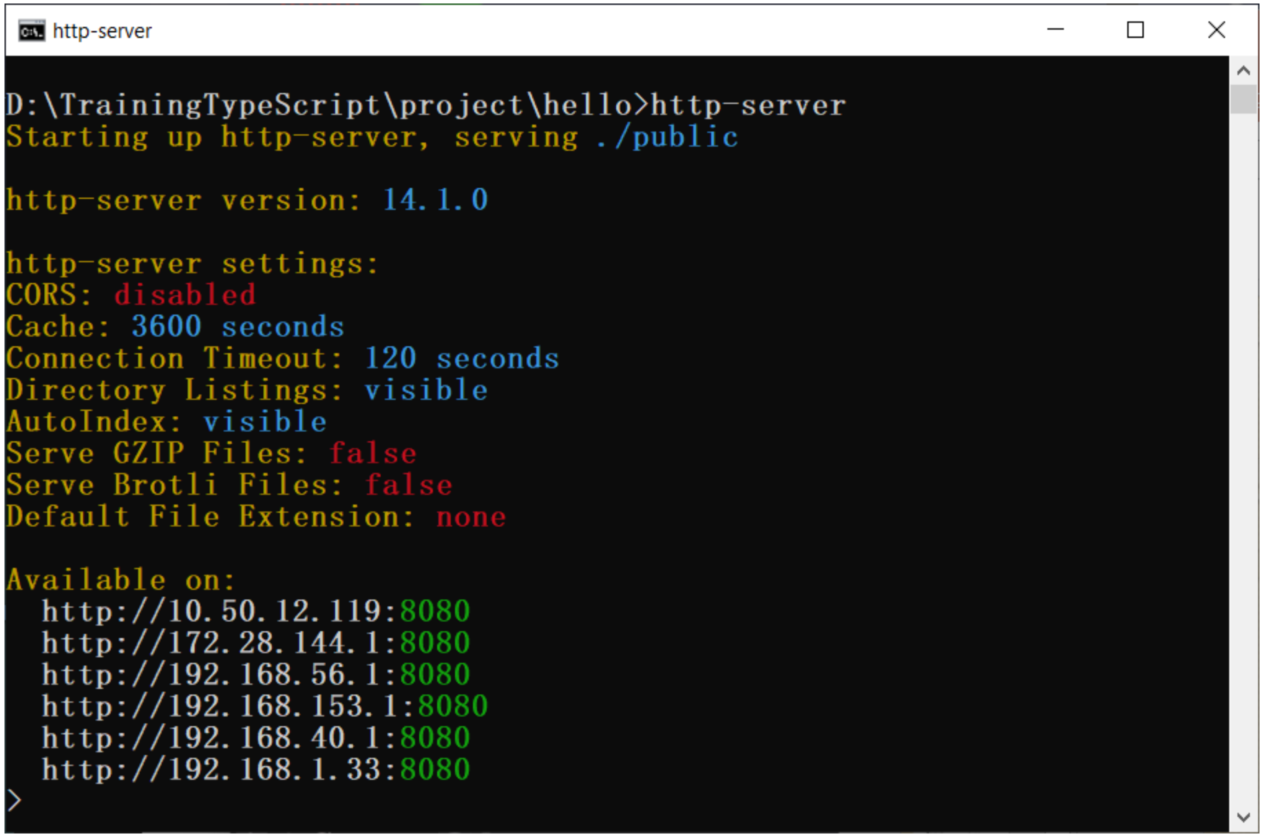
- http server
  - route

```
import http from 'http';
import url from 'url';
http.createServer(function handler(req, res) {
    if(req.url == '/') {
        res.writeHead(200, {'Content-Type': 'text/plain'});
        res.end('Hello World');
    } else if(req.url == '/hello') {
        res.writeHead(200, {'Content-Type': 'application/json'});
        let response = {"message" : "Hello World"};
        res.end(JSON.stringify(response));
    } else if(req.url) {
        console.log("request",req.url);
        let q = url.parse(req.url, true).query;
        res.writeHead(200, {'Content-Type': 'application/json'});
        let response = {"message" : "Hello, "+q.name};
        res.end(JSON.stringify(response));
    }
}).listen(8080);
console.log('Server running at http://127.0.0.1:8080/');
```

http2.ts

Web Application

- http server
  - npm install http-server



http-server

```
D:\TrainingTypeScript\project\hello>http-server
Starting up http-server, serving ./public

http-server version: 14.1.0

http-server settings:
CORS: disabled
Cache: 3600 seconds
Connection Timeout: 120 seconds
Directory Listings: visible
AutoIndex: visible
Serve GZIP Files: false
Serve Brotli Files: false
Default File Extension: none

Available on:
  http://10.50.12.119:8080
  http://172.28.144.1:8080
  http://192.168.56.1:8080
  http://192.168.153.1:8080
  http://192.168.40.1:8080
  http://192.168.1.33:8080
>
```

Web Application

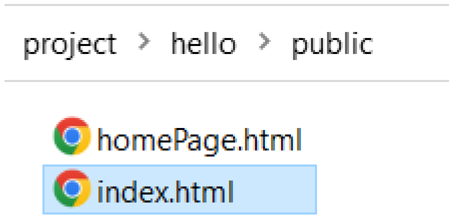
- express
  - npm install express –save
  - npm install @types/express --save-dev

```
import express from 'express';

const app = express();

app.use(express.static("public"));

app.get("/home", function(request, response)  {
    response.sendFile(__dirname + '/public/homePage.html');
});

app.listen(8080);
```

server1.ts

Web Application

- express
  - public folder



project › hello › public

homePage.html
index.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Home</title>
  </head>
  <body>

     <h1>Hello World</h1>

  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Home</title>
  </head>
  <body>

     <h1>This is Home Page</h1>

  </body>
</html>
```

Web Application

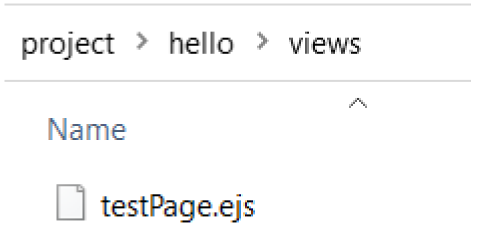
- express
  - ejs (embedded javascript template)
  - npm install ejs --save

```
import express from 'express';

const app = express();

app.set("view engine", "ejs");
app.set("views", "./views");

app.get("/test", function(request, response)  {
    response.render("testPage",{username: "John"});
});

app.listen(8080);
```

server2.ts

Web Application

- express
  - ejs - views folder



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Test</title>
  </head>
  <body>

     <h1>This is Test Page</h1>
     <h2>Wellcome, <%=username%></h2>
  </body>
</html>
```

Web Application

- express - route
  - npm install cors –save
  - npm install @types/cors --save-dev

```
import { HTTP_PORT } from "./utils/EnvironmentVariable";
import { Application } from 'express';
import { Server } from 'http';
import { AddressInfo } from 'net';
import bodyparser from 'body-parser';
import express from 'express';
import cors from 'cors';

const app : Application = express();

app.set('view engine','ejs');
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded());
app.use(cors({
    credentials: true,
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH']
}));
```

server3.ts

Web Application

- express - route

```
var response = {
    type: "result",
    status: "ok",
    message: "",
    body: ""
};

app.get('/hello', function (req, res) {
    res.contentType('application/json');
    //using query direct access string parameter
    //ex. curl http://localhost:8080/hello?name=test
    var pname = req.query.name;
    console.log("do get : " + req.originalUrl + ", path=" +
req.path + ", name = " + pname);
    response.status = "ok";
    response.message = "hello " + (pname == null ? "world" :
pname);
    console.log(response);
    res.send(JSON.stringify(response));
});
```

server3.ts

Web Application

- express - route

```
const urlencodedparser = bodyparser.urlencoded({ extended: false
});
app.post('/hello', urlencodedparser, function (req, res) {
    res.contentType('application/json');
    //using body parser www-url-encoded as parameters
    //ex. curl -X POST http://localhost:8080/hello -d name=test
    var pname = req.body.name;
    console.log("do post : " + req.originalUrl + ", name = " +
pname);
    response.status = "ok";
    response.message = "hello " + (pname == null ? "world" :
pname);
    console.log(response);
    res.end(JSON.stringify(response));
});
```

server3.ts

Web Application

- express - route

```
app.get('/hi/:name', function (req, res) {
    res.contentType('application/json');
    //using params direct access path parameter
    //ex. curl http://localhost:8080/hi/test
    var pname = req.params.name;
    console.log("do get : " + req.originalUrl + ", name = " +
pname);
    response.status = "ok";
    response.message = "hi " + (pname == null ? "world" : pname);
    console.log(response);
    res.json(response);
});
```

server3.ts

Web Application

- express - route

```
app.get('/error', function (req, res) {
    res.contentType('application/json');
    //using status code to defined error
    //ex. curl http://localhost:8080/error
    console.log("do get : " + req.originalUrl);
    response.status = "error";
    response.message = "test error";
    console.log(response);
    res.status(400).json(response);
});

const server : Server = app.listen(HTTP_PORT, function () {
    let addr = server.address() as AddressInfo;
    let host = addr.address;
    let port = addr.port;
    console.log("working directory : "+__dirname);
    console.log("Server running at http://%s:%s", host, port);
});
```

server3.ts

Web Application

- express - router
  - npm install moment --save

```
import { HTTP_PORT } from "./utils/EnvironmentVariable";
import { Application } from 'express';
import { Server } from 'http';
import { AddressInfo } from 'net';
import express from 'express';
import cors from 'cors';
import fetchrouter from './routers/FetchRouter';

const app : Application = express();

app.set('view engine','ejs');
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded());
app.use(cors({
    credentials: true,
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH']
}));
```

server4.ts

Web Application

- express - router

```
app.use("/fetch",fetchrouter);

const server : Server = app.listen(HTTP_PORT, function () {
    let addr = server.address() as AddressInfo;
    let host = addr.address;
    let port = addr.port;
    console.log("working directory : "+__dirname);
    console.log("Server running at http://%s:%s", host, port);
});
```

server4.ts

Web Application

- express - router
  - FetchRouter.ts

```
import { JSONReply } from "../model/JSONReply";
import { Request, Response } from 'express';
import express from 'express';
import moment from 'moment';

const router = express.Router();

//using params direct access path parameter
//ex. curl -X POST http://localhost:8080/fetch/time/current
router.post('/time/:name', function(req: Request, res: Response)
{
    doFetch(req,res);
});

//using params direct access path parameter
//ex. curl http://localhost:8080/fetch/time/current
router.get('/time/:name', function(req: Request, res: Response) {
    doFetch(req,res);
});
```

Web Application

- express - router

```
function doFetch(req: Request, res: Response) : void {
    res.contentType('application/json');
    let pname = req.params.name;
    console.log("do fetch : "+req.originalUrl+", name = "+pname);
    let response: JSONReply = new JSONReply();
    response.head.modeling("hello","fetch");
    response.head.composeNoError();
    let body : Map<String,String> = new Map();
    let d = new Date();
    let m = moment(d);
    body.set("datetime", m.format('DD-MMM-YYYY HH:mm:ss'));
    if(pname && pname=="current") {
        body.set("result", m.format('HH:mm:ss'));
    } else if(pname && pname=="date") {
        body.set("result", m.format('DD/MM/YYYY'));
    } else if(pname && pname=="time") {
        body.set("result", m.format('HH:mm:ss'));
    } else if(pname && pname=="datetime") {
        body.set("result", m.format('DD/MM/yyyy HH:mm:ss'));
    }
    response.body = Object.fromEntries(body);
    console.log(response);
    res.json(response);
}
export default router;
```

Web Application

- express - router
  - JSONReply.ts - 1

```
class JSONHeader {
    public model: String = '';
    public method: String = '';
    public errorcode: String = '';
    public errorflag: String = 'N';
    public errordesc: String = '';
    protected composeFailure(errorflag: String, errorcode: String, errordesc:
String) : void {
        this.errorflag = errorflag;
        this.errorcode = errorcode;
        this.errordesc = errordesc;
    }
    public composeError(errorcode: String, errordesc: String) : void {
        this.composeFailure("Y",errorcode,errordesc);
    }
    public composeNoError() : void {
        this.composeFailure("N", "0", "");
    }
    public modeling(model: String, method: String) : void {
        this.model = model;
        this.method = method;
    }
}
```

Web Application

- express - router
  - JSONReply.ts - II

```
class JSONReply {
    public head: JSONHeader = new JSONHeader();
    public body: Object = { };
}

export {
    JSONHeader,
    JSONReply
}
```

Web Application

- express application

```
import { HTTP_PORT } from "./utils/EnvironmentVariable";
import { Application } from 'express';
import { Server } from 'http';
import { AddressInfo } from 'net';
import express from 'express';
import cors from 'cors';
import bookrouter from './routers/BookRouter';

const app : Application = express();

app.set('view engine','ejs');
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded());
app.use(cors({
    credentials: true,
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH']
}));

app.use("/book",bookrouter);
```

server5.ts

Web Application

- express application

```
const server : Server = app.listen(HTTP_PORT, function () {
    let addr = server.address() as AddressInfo;
    let host = addr.address;
    let port = addr.port;
    console.log("working directory : "+__dirname);
    console.log("Server running at http://%s:%s", host, port);
});
```

server5.ts

Web Application

- express application
  - BookRouter.ts - 1

```
import { Request, Response } from 'express';
import express from 'express';
import moment from 'moment';
import { DBConnection } from "../db/DBConnection";

const router = express.Router();

router.get('/:bookid', function(req: Request, res: Response) {
    doGetBook(req,res);
});
```

Web Application

- express application - BookRouter.ts - II

```
async function doGetBook(req: Request, res: Response) {
    res.contentType('text/html');
    let bookid = req.params.bookid;
    let conn = null;
    try {
        conn = await DBConnection.getConnection();
        let sql = "select * from book where bookid = ? ";
        conn.query(sql,[bookid],(qerr, rows, fields) => {
            if(qerr) {
                console.error(qerr);
                res.render('errorPage',{ errormessage: qerr.sqlMessage?qerr.sqlMessage:qerr.message });
                return;
            }
            if(rows && rows.length>0) {
                let row = rows[0];
                console.log("row",row);
                let m = moment(row.publishdate);
                row.publishdate = m.format('DD/MM/YYYY');
                res.render('bookPage',{ record: row });
                return;
            }
            res.render('errorPage',{ errormessage: "Record not found" });
        });
    } catch(cerr: any) {
        res.render('errorPage',{ errormessage: cerr.sqlMessage?cerr.sqlMessage:cerr.message });
    } finally {
        if(conn) DBConnection.releaseConnection(conn);
    }

}

export default router;
```

Web Application

- express application
  - bookPage.ejs

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8">
    <title>Book Information</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <td>Title</td><td><%=record.title%></td>
      </tr>
      <tr>
        <td>Author</td><td><%=record.author%></td>
      </tr>
      <tr>
        <td>Publisher</th><td><%=record.publisher%></td>
      </tr>
      <tr>
        <td>Price</th><td><%=record.price%></td>
      </tr>
      <tr>
        <td>Publish Date</th><td><%=record.publishdate%></td>
      </tr>
    </table>
  </body>
</html>
```

Web Application

- express application
  - DBConnection.ts - 1

```
import { DB_URL } from "../utils/EnvironmentVariable";
import { Connection, Pool, PoolConnection, MysqlError } from 'mysql';
import mysql from 'mysql';

const pool: Pool = mysql.createPool(DB_URL);

export class DBConnection {

    public static getConnection() : Promise<Connection> {
        return new Promise<Connection>((resolve, reject) => {
            pool.getConnection((cerr: MysqlError, conn: Connection) => {
                if(cerr) {
                    if(conn) DBConnection.releaseConnection(conn);
                    reject(cerr);
                } else {
                    resolve(conn);
                }
            });
        });
    }
```

Web Application

- express application
  - DBConnection.ts - II

```
    public static getConnectionAsync(callback: Function) {
        pool.getConnection((cerr: MysqlError, conn: Connection) => {
            if(cerr) {
                if(conn) DBConnection.releaseConnection(conn);
                callback(cerr, null);
            } else {
                callback(null, conn);
            }
        });
    }

    public static releaseConnection(conn: Connection) {
        try {
            let pconn : PoolConnection = conn as PoolConnection;
            pconn.release();
        } catch(ex) {
            console.error(ex);
        }
    }
}
```

Web Application

- express session application
  - npm install express-session –save
  - npm install @types/express-session --save-dev

Web Application

- express session application

```
import { HTTP_PORT } from "./utils/EnvironmentVariable";
import { Application } from 'express';
import { Server } from 'http';
import { AddressInfo } from 'net';
import express from 'express';
import cors from 'cors';
import session from 'express-session';
import loginrouter from './routers/LoginRouter';

const app : Application = express();

app.set('view engine','ejs');
app.use(express.static('public'));
app.use(express.json());
app.use(express.urlencoded());
app.use(cors({
    credentials: true,
    methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH']
}));
```

server6.ts

Web Application

- express session application

```
app.use(session({
      secret: 'SomeSuperLongHardToGuessSecretString',
      resave: true,
      saveUninitialized: true,
      cookie: {
          maxAge: 10*60*1000, //10s expired
      },
    })
  );

app.use("/",loginrouter);

const server : Server = app.listen(HTTP_PORT, function () {
    let addr = server.address() as AddressInfo;
    let host = addr.address;
    let port = addr.port;
    console.log("working directory : "+__dirname);
    console.log("Server running at http://%s:%s", host, port);
});
```

server6.ts

Web Application

- express session application
  - LoginRouter.ts - I

```
import path from 'path';
import express from 'express';
const router = express.Router();

router.get('/in',(req, res) => {
    let sess : any = req.session;
    if(sess.email) {
        return res.redirect('/admin');
    }
    let parent = path.dirname(__dirname);
    res.sendFile(path.dirname(parent)+'/public/login.html');
});

router.post('/login',(req, res) => {
    let sess : any = req.session;
    sess.email = req.body.email;
    res.end('done');
});
```

Web Application

- express session application
  - LoginRouter.ts - II

```
router.get('/admin',(req, res) => {
    let sess : any = req.session;
    res.contentType('text/html');
    console.log("session",sess);
    if(sess.email) {
        res.write(`<h2>Hello ${sess.email}</h2>`);
        res.write(`<h2>Session ID ${sess.id}</h2><br/>`);
        res.end('<a href="/logout">Logout</a>');
    } else {
        res.write('Please login first.<br/>');
        res.end('<a href="/">Login');
    }
});
```

Web Application

- express session application
  - LoginRouter.ts - III

```
router.get('/logout',(req, res) => {
    req.session.destroy((err) => {
        if(err) {
            return console.log(err);
        }
        res.redirect('/in');
    });

});

export default router;
```

Web Application

- express session application
  - login.html - 1

```
<html>
<head>
<title>Session Management in NodeJS using Node and Express</title>
<script src="./js/jquery-1.11.1-min.js"></script>
<script>
$(document).ready(function(){
    $("#submit").click(function(){
        var email = $("#email").val();
        var pass = $("#password").val();
        $.post("/login",{email:email,pass:pass},function(data){
            if(data==='done') {
                window.location.href="/admin";
            }
        });
    });
});
</script>
</head>
```

Web Application

- express session application
  - login.html - II

```
<body>
<input type="text" size="40" placeholder="Type your email"
id="email"><br />
<input type="password" size="40" placeholder="Type your password"
id="password"><br />
<input type="button" value="Submit" id="submit">
</body>
</html>
```

Q & A