let message: string = ‘Hello, World!’;

console.log(message);

// <1>

// <2>

Listing 1.1

$ ls app.ts

$ tsc app.ts

$ ls

app.js app.ts

$ node app.js Hello, World!

$

// <1>

// <2>

// <3>

// <4>

Penggunaan masing-masing dapat dilihat pada Listing 1.2.

var a = 10; console.log(a); var a = 20; console.log(a); let b = 30; console.log(b); b = b + 5;

// let b = b + 10;

const c = “softwareku”; console.log(c);

// c = “softwareku - versi 1.0”;

// <1>

// <2>

// <3>

// <4>

Listing 1.2 Deklarasi variabel

let isFinished: boolean = false;

console.log(isFinished, typeof isFinished); // <1> let price: number = 150.34;

console.log(price, typeof price);

let numOfEmployees: number = 25; console.log(numOfEmployees, typeof numOfEmployees); let progLang: string = “TypeScript”; console.log(progLang, typeof progLang)

let university: string[] = [‘UT’, ‘UGM’, ‘ITB’]; console.log(university, typeof university)

let employee: [number, string, boolean, number, string]; employee = [1, “Zaky Aditya”, true, 20, “Engineer”]; console.log(employee, typeof employee);

enum Color {

Black = 2,

Blue, Yellow, Green = 3,

Red = 3 \* 3

}

console.log(Color, typeof Color); let code: string | number; console.log(code, typeof code); code = ‘my code’; console.log(code, typeof code); code = 21;

console.log(code, typeof code); let valueNull = null

console.log(valueNull, typeof valueNull)

let valueUndefined = undefined console.log(valueUndefined, typeof valueUndefined) let valueAny: any;

console.log(valueAny, typeof valueAny) valueAny = true;

console.log(valueAny, typeof valueAny) valueAny = 42;

console.log(valueAny, typeof valueAny) valueAny = “TypeScript”; console.log(valueAny, typeof valueAny) valueAny = []

console.log(valueAny, typeof valueAny) valueAny = {};

console.log(valueAny, typeof valueAny) valueAny = Math.random; console.log(valueAny, typeof valueAny) valueAny = null;

console.log(valueAny, typeof valueAny) valueAny = undefined; console.log(valueAny, typeof valueAny) let valueUnknown: unknown;

console.log(valueUnknown, typeof valueUnknown); valueUnknown = true;

console.log(valueUnknown, typeof valueUnknown); valueUnknown = 42;

console.log(valueUnknown, typeof valueUnknown); valueUnknown = “TypeScript”; console.log(valueUnknown, typeof valueUnknown); valueUnknown = [];

console.log(valueUnknown, typeof valueUnknown); valueUnknown = {};

console.log(valueUnknown, typeof valueUnknown); valueUnknown = Math.random; console.log(valueUnknown, typeof valueUnknown); valueUnknown = null;

console.log(valueUnknown, typeof valueUnknown); valueUnknown = undefined; console.log(valueUnknown, typeof valueUnknown);

Listing 1.3

Tipe data dasar di TypeScript

let a: number = 21; let b: number = 3;

let c:number = a / b; if (c > 5) {

console.log(“lebih besar dari 5”);

} else if (c > 3) { console.log(“antar 3 - 5”);

} else {

console.log(“di bawah 3”);

}

Listing 1.4 Pencabangan dengan if

let hari: number = 5; let hariStr: string;

switch (hari) { case 0:

hariStr = “Minggu”; break;

case 1:

hariStr = “Senin”; break;

case 2:

hariStr = “Selasa”; break;

case 3:

hariStr = “Rabu”; break;

case 4:

hariStr = “Kamis”; break;

case 5:

hariStr = “Jum’at”; break;

case 6:

hariStr = “Sabtu”; break;

default:

hariStr = “Tidak ada hari tersebut”; break;

}

console.log(hariStr);

Listing 1.5

Seleksi kondisi dengan switch

for (let i = 0; i < 3; i++) { console.log(“Looping ke “ + i);

}

let arr = [“nilai 1”, “nilai 2”, “nilai 3”, “nilai 4”]; for (var nilai of arr) {

console.log(nilai);

}

for (var index1 in arr) { console.log(index1);

}

console.log(index1);

// <1>

// <2>

// <3>

// <4>

for (let index2 in arr) { console.log(index2);

}

// <5>

// console.log(index2)

// <6>

let str = “Universitas Terbuka”;

for (var huruf of str) { console.log(huruf);

}

// <7>

Listing 1.6 Perulangan menggunakan for

let nilai1: number = 5;

while (nilai1 < 10) { console.log( nilai1 ) nilai1++;

}

// <1>

let nilai2: number = 5;

while (nilai2 < 5) { console.log( nilai2 ) nilai2++;

}

// <2>

let nilai3: number = 5;

do { // <3>

console.log( nilai3 ) nilai3++;

} while ( nilai3 < 10) let nilai4: number = 5;

do { // <4>

console.log( nilai4 ) nilai4++;

} while ( nilai4 < 5)

Listing 1.7 Perulangan menggunakan *while*

$ npm init -y

...

...

“author”: “”,

“license”: “ISC”

}

// <1>

$ ls package.json

$ npm install @types/node --save-dev

...

...

+ @types/node@14.14.17

added 1 package from 44 contributors and audited 1 package in 5.597s found 0 vulnerabilities

// <2>

// <3>

$ ts-node

* import \* as os from ‘os’;

{}

* of.userInfo()

{

uid: 1000,

gid: 1000, username: ‘bpdp’,

homedir: ‘/home/bpdp’,

shell: ‘/usr/bin/fish’

}

>

// <4>

// <5>

// <6>

Listing 1.8 Fungsi Pustaka Standar

console.log(add(32,12));

// <1>

function add(x: number, y: number): number { return x + y;

}

// <2>

// <3>

console.log(add(21,12));

// <4>

Listing 1.9 Mendefinisikan Fungsi

//console.log(2, 3);

// <1>

let add = function (x: number, y: number): number { // <2> return x + y; // <3>

};

console.log(add(2, 3));

// <4>

Listing 1.10

Membuat fungsi menggunakan *function expression*

function tampilkan(arg1: any): void { console.log(arg1)

}

tampilkan(“Halo!”);

Listing 1.11 Nilai kembalian *void*

nonStop();

function nonStop(): never { while (true) { console.log(‘tulisan non-stop’);

}

}

Listing 1.12 Nilai kembalian *never*

let add1 = (a: number, b: number): number => { return a + b }; // <1> let add2 = (a: number, b: number): number => a + b; // <2> let lenStr1 = (s: string): number => s.length; // <3>

let lenStr2 = s => s.length; // <4>

// Penggunaan arrow func console.log(add1(2,3));

console.log(add2(2,3)); console.log(lenStr1(‘abcdefg’)); console.log(lenStr2(‘abcdefg’));

// <5>

Listing 1.13 Penggunaan \_arrow function\_ (=>)

((str: string, idx: number) => { console.log(str[idx])

})(‘Universitas Terbuka’, 4);

// <1>

// <2>

// <3>

let idxStr = function(str: string, idx: number): void { // <4> console.log(str[idx]); // <5>

}

console.log(idxStr(‘Universitas Terbuka’, 4));

// <6>

Listing 1.14 Fungsi Anonymous

let scoping = function(input: any) {

// <1>

let angka1 = 100;

if (typeof input == ‘number’) { let angka2 = angka1 + input; var angka3 = angka1 - input;

}

// <2>

// <3>

// Error: Cannot find name ‘angka2’

// return angka2;

// karena definisi dengan var, maka

// dapat diakses return angka3;

// <4>

// <5>

}

console.log(scoping(1));

Listing 1.15

Ruang lingkup *let* dan *var*

interface IPerson { nik: string;

nama: string; alamat: string; menikah: boolean;

}

// <1>

interface IPegawai extends IPerson { readonly npp: string;

jabatan: string; gaji: number; email?: string;

}

// <2>

let peg01: IPegawai = { nik: ‘012345’,

nama: ‘Donal’,

alamat: ‘Jl. Awan Biru 21’, menikah: true,

npp: ‘98123’,

jabatan: ‘Manager SDM’, gaji: 15000000

}

// <3>

console.log(peg01.nama, peg01.jabatan); // <4>

// error: Cannot assign to ‘npp’ because it is a read-only property

// peg01.npp = ‘981234’;

interface IKamusList { [index:string]:string

}

// <5>

let strKamus: IKamusList = {}; strKamus[“university”] = “universitas”;

strKamus[“freedom”] = “merdeka”;

// <6>

console.log(strKamus[“university”]);

interface IPemrosesNilai { (kunci: number,

nilai: string ): void

}

// <7>

function tambahNilai (kunci: number, nilai: string): void { console.log(‘Menambah ‘, kunci, nilai);

}

// <8>

function perbaruiNilai (kunci: number, nilaiBaru: string): void { // <9> console.log(‘Memperbarui ‘, kunci, nilaiBaru);

}

let pemrosesTambah: IPemrosesNilai = tambahNilai;

pemrosesTambah(123, ‘Nilai 123’);

// <10>

let pemrosesPerbarui: IPemrosesNilai = perbaruiNilai; pemrosesPerbarui(123, ‘Nilai baru 123’);

Listing 1.16 Penggunaan *Interface*

class Person { perNik: string;

protected perNama: string;

perAlamat: string;

// <1>

// <2>

constructor(nik: string, nama: string) { this.perNik = nik;

this.perNama = nama;

}

// <3>

}

class Pegawai extends Person { pegNpp: string;

private \_pegJmlTanggungan: number; readonly dept: string;

gaji: number;

// <4>

// <5>

static potongPajak = 10;

// <6>

constructor(nik: string, npp: string, // <7> nama: string, dept: string) {

super(nik, nama); this.pegNpp = npp; this.dept = dept;

}

getGaji(): number { return this.gaji;

}

// <8>

setGaji(gajiBaru: number): void { this.gaji = gajiBaru;

}

// <9>

getPotonganPajak(): number {

return this.gaji \* (Pegawai.potongPajak / 100);

}

// <10>

presensi(): void {

let dateTime = new Date(); console.log(“Presensi pada “ +

dateTime.toLocaleTimeString() + ‘ - ‘ + dateTime.toDateString());

}

// <11>

}

let pakBambang = new Pegawai(‘nik1122’, ‘npp123’, // <12> ‘Bambang Purnomosidi’, ‘IT’);

console.log(pakBambang.setGaji(15750500)); // <13> console.log(pakBambang.getGaji()); console.log(pakBambang.presensi()); console.log(pakBambang.getPotonganPajak());

// error:

// Property ‘\_pegJmlTanggungan’ is private and only

// accessible within class ‘Pegawai’

// console.log(pakBambang.\_pegJmlTanggungan);

Listing 1.17 Definisi *Class* dan *Instance*

// tanpa generics

function funN(argN: number): number { return argN;

}

// <1>

function funS(argS: string): string { return argS;

}

console.log(funN(23)); console.log(funS(‘TypeScript’));

// tanpa generics - menggunakan any

// tidak type safe, karena menerima tipe

// data apa saja.

function funA(argA: any): any{ return argA;

}

// <2>

console.log(funA(true)); console.log(funA([1,2,3]));

// menggunakan generics function funG1<T>(argG1:T):T { return argG1;

}

// <3>

function funGn<T, U>(argGn1:T, argGn2: U): U { return argGn2;

}

// <4>

console.log(funG1<string>(‘TypeScript’)); console.log(funG1<number>(23));

console.log(funGn<string, number>(‘TypeScript’, 23)); interface argGenConstraint {

length: number;

}

// <5>

function panjang<T extends argGenConstraint>(argGC:T): number { // <6> return argGC.length;

}

let hasil1 = panjang({ length: 23, name: ‘TypeScript’});

let hasil2 = panjang(‘TypeScript’);

// <7>

console.log(hasil1);

console.log(hasil2);

Listing 1.18 Penggunaan *Generics*

function getAngkaAcak(max: number): number {

return Math.floor(Math.random() \* Math.floor(max));

}

// <1>

let p = new Promise<unknown>((resolve, reject) => {

// <2>

let acak = getAngkaAcak(1000);

if (acak > 500) { resolve(true); return;

}

// <3>

reject(“Hasil < = 500”);

// <4>

});

console.log(p);

p.catch(err => console.log(“ERROR - “, err));

// Berbagai kemungkinan saat run:

// Promise { true }

// -> jika angka random lebih besar dari 500

// Promise { <rejected> ‘Hasil < = 500’ }

// ERROR - Hasil < = 500

// -> jika yang terjadi adalah reject

// <5>

// <6>

Listing 1.19

Penggunaan *Promise* Untuk *Asynchronous Programming*.

function getAngkaAcak(max: number): number {

return Math.floor(Math.random() \* Math.floor(max));

}

let p = new Promise<number>((resolve, reject) => { let acak = getAngkaAcak(20);

let hasil: number; // <1>

if (acak > 10) { resolve(acak); return;

}

reject(“Hasil < = 10”);

}).then(function(hasil) { console.log(hasil); return hasil\*2;

}).then(function(hasil) { console.log(hasil); return hasil\*2;

// <2>

// <3>

// <4>

// <5>

// <6>

}).then(function(hasil) { console.log(hasil); return hasil\*2;

});

console.log(p);

p.catch(err => console.log(“ERROR - “, err));

// run (dapat berbeda-beda):

// pertama (jika resolve):

// Promise { <pending> }

// 12

// 24

// 48

// kedua (jika reject):

// Promise { <pending> }

// ERROR - Hasil < = 10

Listing 1.20

*Promise Chaining*

function getAngkaAcak(max: number): number {

return Math.floor(Math.random() \* Math.floor(max));

}

// <1>

function lebihDari(max: number, angka: number): boolean | number { // <2> if (angka > max) {

throw “ERR: arg 1 harus lebih besar daripada arg 2”

}

let acak = getAngkaAcak(max); if (acak > angka) {

return true;

} else { return acak;

}

}

let p = async function (): Promise<boolean | number> {

// <3>

try {

//let hasilOK: boolean | number = await lebihDari(100, 500); let hasilOK: boolean | number = await lebihDari(1000, 500); return hasilOK;

// <4>

// <5>

} catch(error) {

// <6>

return error;

}

};

(async () => { console.log(await p())

})()

// <7>

// jika <4> di uncomment, maka hasil:

// ERR: arg 1 harus lebih besar daripada arg 2

// jika menggunakan <5>, maka hasil kemungkinan:

// true

// 134 (atau angka lain)

Listing 1.21 Penggunaan *Async/Await*

$ mkdir restful-api

$ cd restful-api

$ npm init -y

$ npm install -g nodemon concurrently

$ npm install @types/node @types/express express --save-dev

$ mkdir src dist

$ tsc --init

// <1>

// <2>

// <3>

// <4>

// <5>

// <6>

// <7>

Listing 1.22 Penggunaan *Resful-API*

{

“name”: “restful-api”,

“version”: “1.0.0”,

“description”: “”,

“main”: “index.js”, “scripts”: {

“start:dev”: “nodemon dist/index.js”,

“build:dev”: “tsc --watch --preserveWatchOutput”, “dev”: “concurrently \”npm:build:dev\” \”npm:start:dev\””

},

“keywords”: [],

“author”: “”,

“license”: “ISC”, “devDependencies”: { “@types/express”: “^4.17.10”, “express”: “^4.17.1”

}

}

Listing 1.23 Penggunaan *Paket JSON*

{

“compilerOptions”: { “target”: “es5”, “module”: “commonjs”, “outDir”: “./dist”,

“rootDir”: “./src”, “strict”: true,

“esModuleInterop”: true, “skipLibCheck”: true,

“forceConsistentCasingInFileNames”: true, “resolveJsonModule”: true

}

}

Listing 1.24 Konfigurasi pada *file* tsconfig.json

export const PORT = process.env.PORT || 4000;

Listing 1.25

File *config/constants.ts*

*File config/constants.ts* digunakan untuk menampung konfigurasi yang diperlukan, dalam hal ini adalah konfigurasi untuk *port* yaitu 4000.

[

{

“balance”: “$3,946.45”,

“picture”: [“http://placehold.it/32x32”,](http://placehold.it/32x32) “age”: 23,

“name”: “Bird Ramsey”, “gender”: “male”,

“company”: “NIMON”,

“email”: [“birdramsey@nimon.com”](mailto:birdramsey@nimon.com)

},

{

“balance”: “$2,499.49”,

“picture”: [“http://placehold.it/32x32”,](http://placehold.it/32x32) “age”: 31,

“name”: “Lillian Burgess”, “gender”: “female”, “company”: “LUXURIA”,

“email”: [“lillianburgess@luxuria.com”](mailto:lillianburgess@luxuria.com)

},

{

“balance”: “$2,820.18”,

“picture”: [“http://placehold.it/32x32”,](http://placehold.it/32x32) “age”: 34,

“name”: “Kristie Cole”, “gender”: “female”, “company”: “QUADEEBO”,

“email”: [“kristiecole@quadeebo.com”](mailto:kristiecole@quadeebo.com)

},

{

“balance”: “$3,277.32”,

“picture”: [“http://placehold.it/32x32”,](http://placehold.it/32x32) “age”: 30,

“name”: “Leonor Cross”, “gender”: “female”,

“company”: “GRONK”,

“email”: [“leonorcross@gronk.com”](mailto:leonorcross@gronk.com)

},

{

“balance”: “$1,972.47”,

“picture”: [“http://placehold.it/32x32”,](http://placehold.it/32x32) “age”: 28,

“name”: “Marsh Mccall”, “gender”: “male”, “company”: “ULTRIMAX”,

“email”: [“marshmccall@ultrimax.com”](mailto:marshmccall@ultrimax.com)

}

]

Listing 1.26 resources/users.json

import { Request, Response } from ‘express’;

export abstract class CrudController {

public abstract create(req: Request, res: Response): void; public abstract read(req: Request, res: Response): void; public abstract update(req: Request, res: Response): void; public abstract delete(req: Request, res: Response): void;

}

Listing 1.27 controllers/CrudController.ts

import { Request, Response } from ‘express’; import { CrudController } from ‘../CrudController’; import usersjson from ‘../../resources/users.json’;

export class UserController extends CrudController {

Public create(req: Request<import(“express-serve-static-core”).ParamsDictionary>, res: Response): void {

throw new Error(“Belum diimplementasikan”);

}

public read(req: Request<import(“express-serve-static-core”).ParamsDictionary>, res: Response): void {

res.json(usersjson);

}

public update(req: Request<import(“express-serve-static-core”).ParamsDictionary>, res: Response): void {

throw new Error(“Belum diimplementasikan”);

}

public delete(req: Request<import(“express-serve-static-core”).ParamsDictionary>, res: Response): void {

throw new Error(“Belum diimplementasikan”);

}

}

Listing 1.28 controllers/User/User.ts

import { UserController } from ‘./User/User’; const userController = new UserController();

export { userController

};

Listing 1.29 controllers/index.ts

import express, { Request, Response } from ‘express’; import { userController } from ‘../../controllers’;

export const router = express.Router({ strict: true

});

router.post(‘/’, (req: Request, res: Response) => { userController.create(req, res);

});

router.get(‘/’, (req: Request, res: Response) => { userController.read(req, res);

});

router.patch(‘/’, (req: Request, res: Response) => { userController.update(req, res);

});

router.delete(‘/’, (req: Request, res: Response) => { userController.delete(req, res);

});

Listing 1.30 routes/User/User.ts

import { router as userRouter } from ‘./User/User’;

export { userRouter

};

Listing 1.31

*routes/index.ts*

import express from ‘express’;

import { PORT } from ‘./config/constants’;

import { userRouter } from ‘./routes’;

const app = express(); app.use(express.json());

app.get(‘/’, (req, res) => {

res.send(‘Selamat datang di RESTful API gateway’);

});

app.use(‘/users’, userRouter);

app.listen(PORT, () => {

console.log(`Endpoint sudah siap dan dapat diakses di port ${PORT}`);

});

Listing 1.32

*File* Utama: *index.ts*