

# Lekcija 4 - Podešavanje okruženja sa alatima

- Uz TS dolazi i nekoliko osobina koje nam olakšavaju rad sa TS-om
- Neke od njih smo upoznali preko **tsc** komande tj. korišćenjem **tsconfig.json** fajla sa opcionim parametrima za podešavanje rada transpajlera
- Upoznajemo još neke opcije u tsconfig fajlu.

# Lekcija 4 - Podešavanje okruženja sa alatima

- strict:true – omogućava sve strict opcije provere tipova
- Ako je properti bez tipa prilikom transpajlovanja će javiti grešku

```
class Person {  
  lastName;  
  firstName;  
}
```

```
test.ts(2,5): error TS7008: Member 'lastName' implicitly has an 'any' type.  
test.ts(3,5): error TS7008: Member 'firstName' implicitly has an 'any' type.
```

- outDir:"js/" – lokacija izlaznog fajla gde će se nalaziti transpajlovani JS fajlovi

## Lekcija 4 - Podešavanje okruženja sa alatima

- strictNullChecks:true - proverava se da li postoji null vrednost ako postoji javiće grešku
- Ako se desi greška ona će u svakom slučaju izgenerisati JS fajl
- Šta ako ne želimo generisanje JS fajla nakon pojave greške u kodu ? Koristimo noEmitError properti da onemogućimo generisanje JS fajlova
- noEmitOnError:true; - ne prikazuj JS fajlove u izlaznom folderu ako ima grešaka u kodu

# Lekcija 4 - Podešavanje okruženja sa alatima

- outfile: "./scripts/app.js" – ispiši sve JS fajlove u jedan izlazni fajl na lokaciju scripts i nazovi ga app.js
- noImplicitAny: true - ne može se koristiti implicitni any mora se eksplicitno definisati
- inlineSourceMap: true - napravi jedan fajl sa source mapama umesto da se generiše u svakom fajlu posebno

```
"use strict";  
function func(s, n, b) {  
    if (b) {  
        console.log(s + n);  
    }  
    else {  
        console.log('Boolean vrednost je false');  
    }  
}  
  
func('ey bro', 404, true);  
///  
sourceMappingURL=data:application/json;base64,  
2FwcC50cyJdLCJuYW1lcyI6W10sIm1hcHBpbmdzIjo7i00FB  
LENBQUMsQ0FBQztRQU5KLE9BQU8sQ0FBQyxHQUFHLENBQU  
BQUMsQ0FBQyxHQUFHLENBQUMsMkJBQTJCLENBQUMsQ0FBQ  
yxDQUFDIn0=
```

# Lekcija 4 - Podešavanje okruženja sa alatima

- **NPM projekat**
- Ako hoćemo da pripremimo naše okruženje za ozbiljnije projekte koristićemo **package.json** za instalaciju paketa potrebnih za rad kao i komandi za pokretanje projekta
- Pokrenimo komandu **npm init**
- **Ispratimo sve potrebne korake do kraja**

# Lekcija 4 - Podešavanje okruženja sa alatima

- **NPM projekat**
- Ako hoćemo da pripremimo naše okruženje za ozbiljnije projekte koristićemo **package.json** za instalaciju paketa potrebnih za rad kao i komandi za pokretanje projekta
- Pokrenimo komandu **npm init**
- **Ispratimo sve potrebne korake do kraja**

```
D:\Kursevi\NSZ\TypeScript\Radni folder\Projekat>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.
```

# Lekcija 4 - Podešavanje okruženja sa alatima

- Rezultat nakon npm init

▲ Projekat

{ } package.json

```
{  
  "name": "projekat",  
  "version": "1.0.0",  
  "description": "Projekat TS",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "Nenad Bosanac",  
  "license": "ISC"  
}
```

- Pogledajmo u main properti gde je definisan root ili startni fajl, a to je index.js
- Napravimo ovaj fajl u root folderu

# Lekcija 4 - Podešavanje okruženja sa alatima

- Napravimo **person.ts** fajl
- Dopunimo **index.ts** i importujmo **person.ts** u njega
- Kopirajmo **package.json** iz prethodnog projekta i kompajlirajmo projekat

```
export class Person{  
  firstName: string;  
  lastName: string;  
}
```

```
import {Person} from './person';  
  
let foo = new Person();  
foo.firstName = "Pera";  
foo.lastName = "Peric";  
console.log(foo);
```



# Lekcija 4 - Podešavanje okruženja sa alatima

- **Workflow**
- Napravimo workflow gde ćemo sa jednom komandom da kompajliramo i pokrenemo ceo kod tj. pokrenuti index.js fajl
- Kompajliranje znamo da radimo sa **tsc** komandom
- Pokretanje znamo da radimo sa **node** komandom
- Objedinimo ove dve komande u **package.json** skriptu u **scripts** properti objekat i dodajmo **start** atribut sa sledećim komandama

# Lekcija 4 - Podešavanje okruženja sa alatima

- Komande u start atributu:

```
"scripts": {  
  "start": "tsc -p tsconfig.json && node js/index.js",  
  "test": "echo \"Error: no test specified\" && exit 1",  
},
```

Komanda za transpilovanje

Komanda za pokretanje index.js

Oznaka za razdvajanje komandi

## Lekcija 4 - Podešavanje okruženja sa alatima

- Pokrenimo komandu `npm start` i dobićemo pokrenutu i jednu i drugu komandu :

```
D:\Kursevi\NSZ\TypeScript\Radni folder\Projekat>npm start
```

```
> projekat@1.0.0 start D:\Kursevi\NSZ\TypeScript\Radni folder\Projekat
```

```
> tsc -p tsconfig.json && node js/index.js
```

```
Person { firstName: 'Pera', lastName: 'Peric' }
```

# Lekcija 4 - Podešavanje okruženja sa alatima

- Instalacija paketa
- Podsetimo se kako instaliramo pakete i instalirajmo jedan paket lodash:

```
D:\Kursevi\NSZ\TypeScript\Radni folder\Projekat>npm install lodash --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN projekat@1.0.0 No repository field.

+ lodash@4.17.11
added 1 package in 6.149s
```

- U okviru package.json dobijemo novi paket

```
"author": "Nenad Bosanac",
"license": "ISC",
"dependencies": {
  "lodash": "^4.17.11"
}
```

## Lekcija 4 - Podešavanje okruženja sa alatima

- Tako importovanu biblioteku možemo da koristimo ako ju importujemo u index.ts .
- **Lodash** se nalazi u node\_modules folderu
- Putanja ovde nije relativna kao sa Person klasom
- Moramo takođe da pišemo alias as nakon \*
- \* znači da importujemo sve funkcije iz biblioteke lodash

# Lekcija 4 - Podešavanje okruženja sa alatima

- Primetite grešku koja se pojavljuje u kodu

```
import {Person} from './person';  
import * as _lodash from 'lodash';
```

```
let foo = new Person();  
foo.firstName = "Pera";  
foo.lastName = "Peric";  
console.log(foo);
```

[ts]

Could not find a declaration file for module 'lodash'. 'd:/Kurs evi/NSZ/TypeScript/Radni folder/Projekat/node\_modules/lodash/lodash.js' implicitly has an 'any' type.

Try `npm install @types/lodash` if it exists or add a new declaration (.d.ts) file containing `declare module 'lodash';`

- Ona kaže da je potrebno da koristimo @types budući da je on optimizovan za TS

# Lekcija 4 - Podešavanje okruženja sa alatima

- Mi smo sa npm instalirali samo JS kod, a JS kod ne koristi types za TS kod
- Zato koristimo **type definitions** – ili **definicije tipova**
- **On omogućava definisanje tipova na osnovu izvornog koda**
- **Omogućava deklamaciju**
- Na taj način možemo da koristimo i autocomplete ove biblioteke za TS

# Lekcija 4 - Podešavanje okruženja sa alatima

- Instalaciju type definition radimo na sledeći način:

```
D:\Kursevi\NSZ\TypeScript\Radni folder\Projekat>npm install @types/lodash --save-dev  
  
npm WARN projekat@1.0.0 No repository field.  
  
+ @types/lodash@4.14.121  
added 1 package in 108.879s
```

- Instalacija @types se vrši u poseban folder u node\_modules

```
└─ node_modules  
  └─ @types  
    └─ lodash
```

```
"devDependencies": {  
  "@types/lodash": "^4.14.121"  
}
```



# Lekcija 4 - Podešavanje okruženja sa alatima

- Pogledajmo sada kako izgleda kod bez greške i sa autocomplete funkcijama u TS
- Izaberimo **reverse** funkciju iz lodash biblioteke




```
import {Person} from './person';
import * as _lodash from 'lodash';

let foo = new Person();
foo.firstName = "Pera";
foo.lastName = "Peric";
console.log(foo);

var array = [1,2,3,4,5];
```

\_lodash.r

- random (method) LoDashSta
- range
- rangeRight
- rearg



```
console.log(_lodash.reverse(array));
```



# Lekcija 5 - Zadatak

- Setovanje projekta