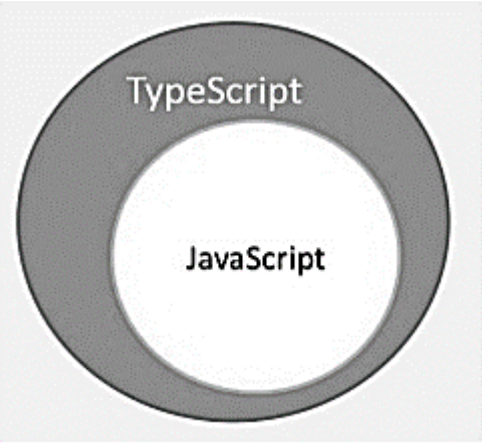
# Uvod u TypeScript

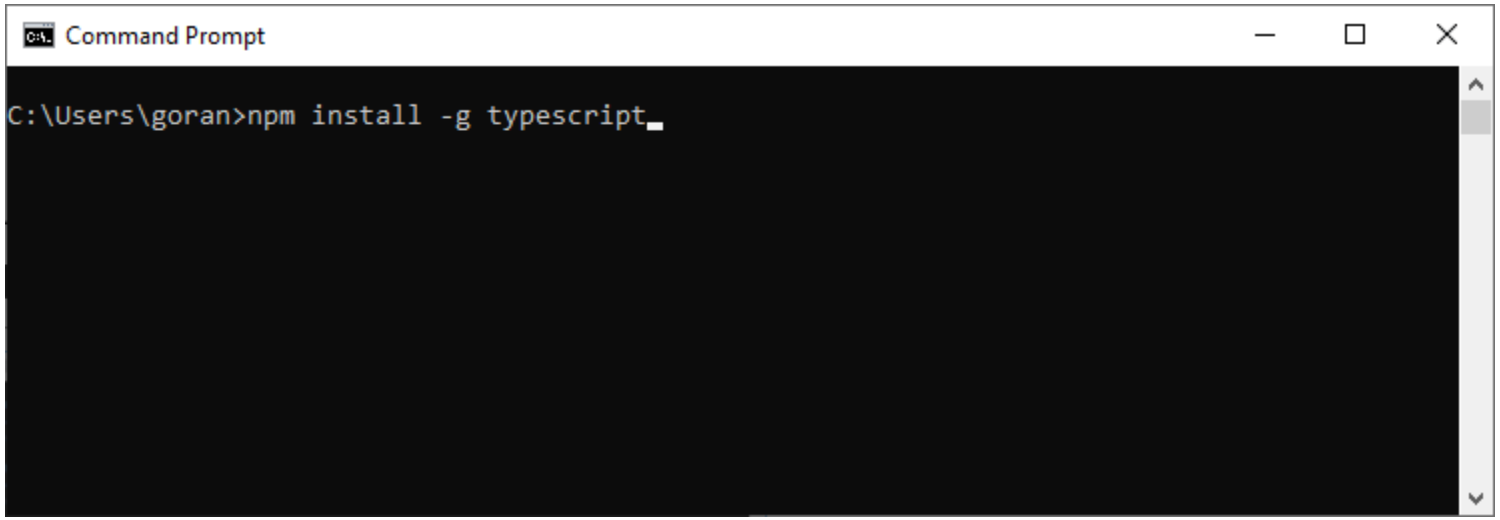
# TypeScript

- TypeScript je objektno orijentisan programski  jezik
- Typescript je tipizirani nadskup javascripta
- Kod napisan u TypeScript-u ne može se izvršiti direktno, već se prevodi u JavaScript



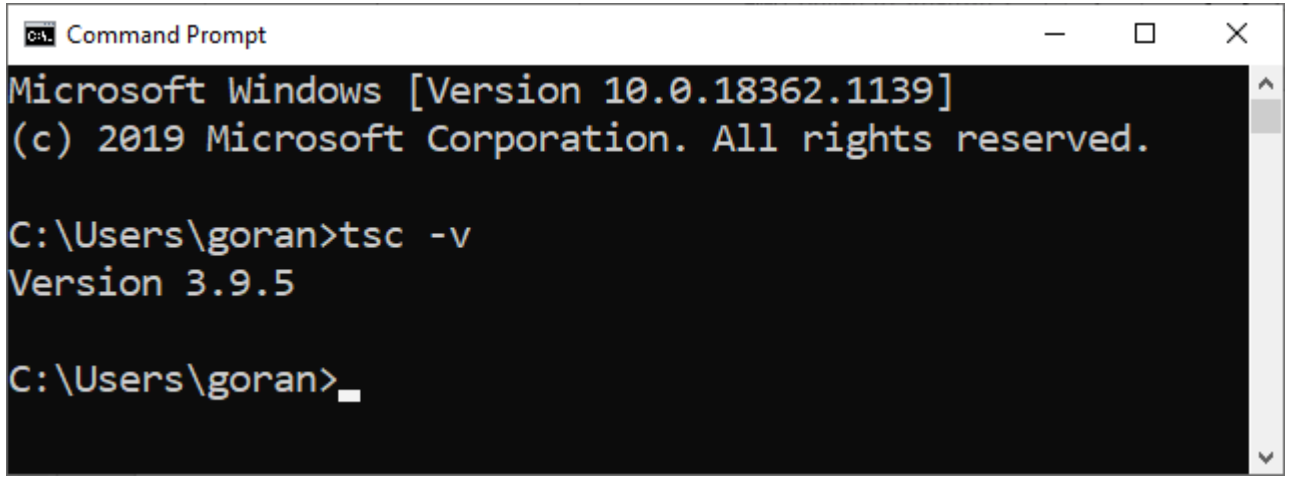

# Globalno  instaliranje TypeScripta

```
npm install -g typescript
```



# Provera TypeScript verzije

```
tsc -v
```



# Instaliranje TypeScripta kao zavisnog paketa u projektu

```
npm install typescript --save-dev
```

# Kreiranje konfiguracionog fajla

**npm init –yes**

**----------**

**npm init -y**

Kreira package.json

```
goran@DESKTOP-ESB9HU8 MINGW64 ~/Desktop/IT05
$ npm init -y
Wrote to C:\Users\goran\Desktop\IT05\package.json:

{
  "name": "IT05",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



goran@DESKTOP-ESB9HU8 MINGW64 ~/Desktop/IT05
$ tsc -init
message TS6071: Successfully created a tsconfig.json file.
```

# Konfigurisanje Typscript opcija

**npx tsc --init**

Kreira tsconfig.json

npx se koristi za izvršavanje paketa koji nisu globalno instalirani

# Tipovi podataka

- **any -** osnovni dinamički tip
- **number**
  - celi i realni brojevi
- **string**
  - sekvenca unicode karaktera
- **boolean**
  - logička vrednost true ili false
- **void**
  - označava da funkcija ne vraća vrednost
- **null**
  - objekat koji nema vrednost
- **undefined**
  - vrednost dodeljena neinicijalizovanoj promenljivoj

# Upotreba okruženja VS Code

- Kreira se folder ts projekta
- Kreira se **src** folder gde smeštamo ts fajlove
- Unutar foldera src se kreira fajl sa ekstenzijom ts npr. **primer01.ts**
- Unutar fajla se napiše ts kod
- Fajl se kompajlira  u terminalu korišćenjem komande: **npx  tsc primer01**
- Korišćenjem noda izvršava se dobijeni javascript: **node primer01**

# Typescript moduli

- Promenljive, funkcije, klase i sl. podrazumevano imaju globalni opseg
- Promenljiva koja je definisanu u fajlu primer01.ts je vidljiva u fajlu primer02.ts
- Moduli omogućavaju definisanje promenljivih, funkcija, klasa vezanih za fajl
- Modul se kreira korišćenjem ključne reči export
- Modul se može referencirati iz drugog modula korišćenjem ključne reči import
- Svaki fajl koji na početku ima export ili import komandu se tretira kao modul

# Kompajliranje i pozivanje fajla

```
C:\Users\goran\Desktop\IT05\src>npx tsc primer01
```

```
C:\Users\goran\Desktop\IT05\src>node primer01
```

```
C:\Users\goran\Desktop\IT05\src>npx tsc primer01.ts

C:\Users\goran\Desktop\IT05\src>node primer01
Ime: Marko
Prvi broj: 50
Drugi broj: 42.5
Suma je: 92.5

C:\Users\goran\Desktop\IT05\src>
```

# TypeScript promenljive

```
export{}; // Označava fajl kao modul, ali ne izvozi ništa
const ime:string = 'Marko';
const a:number = 50;
const b:number = 42.50;
const suma = a + b ;

console.log('Ime:',ime);
console.log('Prvi broj:',a);
console.log('Drugi broj:',b);
console.log('Suma je:',suma);
```

# TS operatori

- Aritmetički operatori (+, -, *, /, %,++,--)
- Operatori poređenja (>,<,>=,<=,==,!=)
- Logički operatori (&&,||,!)
- Operatori dodeljivanja (=,+=,-=,*=,/=)

# Aritmetički operatori

```
// Fajl1.ts
export {}; // Označava fajl kao modul, ali ne izvozi ništa

const a: number = 10;
const b: number = 2;
let rezultat: number = 0; // let ima vidljivost na nivou bloka

rezultat = a + b;
console.log('Zbir:', rezultat);

rezultat = a - b;
console.log('Razlika:', rezultat);

rezultat = a * b;
console.log('Proizvod:', rezultat);

rezultat = a / b;
console.log('Količnik:', rezultat);
```

# Operator konkatenacije

```
export {};

const s1: string = 'prvi';
const s2: string = 'drugi';
const s3: string = s1 + s2;

console.log('Konkatenacija:', s3);
```

```
Konkatenacija: prvidrugi
```

# Uslovni operator

```
export {};

const a: number = 5 * Math.random() - 2;
const b: string = a > 0 ? 'pozitivan' : 'negativan';

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

```
goran@Goran-HP MINGW64 ~/Desktop/IT05
$ node primer04
a: 2.052247998571545
b: pozitivan
```

# TS funkcije

```
export {};

function saberi1(x: number, y: number): number
{
    return x + y;
}

function saberi2(x: number, y: number): void {
    console.log('Rezultat je:', x + y);
}

const rezultat: number = saberi1(5, 6.1);
console.log(rezultat);
saberi2(6.1, 7.8);
```

# Funkcija sa opcionim parametrom

```
export {};

function prikazi(ime: string, prezime: string, email?: string): void {
    console.log('Ime:', ime);
    console.log('Prezime:', prezime);

    if (email !== undefined) {
        console.log('Email:', email);
    }

    console.log('----------------');
}

prikazi('Marko', 'Markovic');
prikazi('Jovan', 'Jovanovic', 'jovan@gmail.com');
```

```
Ime: Marko
Prezime: Markovic
________________
Ime: Jovan
Prezime: Jovanovic
Email: jovan@gmail.com
________________
```

# Podrazumevana vrednost parametra funkcije

```
export {};

function racunaj(cena: number, popust: number = 0.2): void {
    const konacnaCena = cena * (1 - popust);
    console.log('Konačna cena:', konacnaCena.toFixed(2)); // Zaokružujemo na dve decimale
}

racunaj(1000);
racunaj(1000, 0);
racunaj(1000, 0.3);
```

```
goran@DESKTOP-ESB9HU8 MINGW64 ~/Desktop/IT05
$ node primer07
Konacna cena: 800
Konacna cena: 1000
Konacna cena: 700
```

# Anonimna funkcija

```
export{};
let zbir = function (a: number, b: number): void {
    console.log('Zbir je:', a + b);
};
zbir(5, 6);
```

# Lambda izrazi za definisanje anonimnih funkcija

```
export {};

const prikaziVreme = () => new Date().toLocaleTimeString();
const vreme = prikaziVreme();
console.log(vreme);

const uvecaj10 = (x: number) => x + 10;
const r: number = uvecaj10(2);
console.log('Rezultat:', r);
```

```
goran@DESKTOP-ESB9HU8 MINGW64 ~/Desktop/IT05
$ tsc primer09

goran@DESKTOP-ESB9HU8 MINGW64 ~/Desktop/IT05
$ node primer09
20:34:15
rezultat 12
```

# Nizovi

```
const brojevi: Array<number> = [1,3,5,7,9];
```

```
export {};

const brojevi: number[] = [1, 3, 5, 7, 9];

// Korišćenje "for" petlje
console.log('Korišćenje "for" petlje:');
for (let i = 0; i < brojevi.length; i++) {
    console.log(brojevi[i]);
}
console.log('--------------');

// Korišćenje "for...in" petlje
console.log('Korišćenje "for...in" petlje:');
for (const i in brojevi) {
    console.log(i, brojevi[i]);
}
console.log('--------------');

// Korišćenje "for...of" petlje
console.log('Korišćenje "for...of" petlje:');
for (const i of brojevi) {
    console.log(i);
}
console.log('--------------');

// Korišćenje "forEach" metode
console.log('Korišćenje "forEach" metode:');
brojevi.forEach(function (x: number) {
    console.log(x);
});
```

# Filtriranje niza – funkcija filter

```
export{};

const a:number[] = [12,5,8,130,44];
const b = a.filter(x=>x>20);

for (const i of b) {
    console.log(i);
}
```

```
$ node primer11
130
44
```

# Transformacija niza – funkcija map

```
export{};
const a:number[] = [12,5,8,130,44];
const c = a.map(x=>x*x);

for (const i in c) {
    console.log(i,c[i]);
}
```

```
$ node primer12
0 144
1 25
2 64
3 16900
4 1936
```

# Typescript klase

- Typescript koristi klase da bi se iskoristile prednosti OOP kao što su abstrakcija i enkapsulacija
- Klasa se sastoji iz polja, metoda i konstruktora
- Podrazumevano polja klase imaju public modifikator pristupa
- Klase u typescriptu se kompajliraju u javascript funkcije

# Klase

```
export{};
class Osoba {
        ime:string;
        prezime: string;
        starost:number;
        constructor(ime:string, prezime: string, starost:number ) {
            this.ime = ime;
            this.prezime = prezime;
            this.starost = starost;
        }

        stampaj1():void
        {
            console.log('Ime',this.ime);
            console.log('Prezime', this.prezime);
            console.log('Starost',this.starost);
        }

        stampaj2():string
        {
            return this.ime + " " + this.prezime;
        }
      }
    const os1 = new Osoba('Marko','Markovic',25);
    os1.stampaj1();
    console.log(os1.stampaj2());
```

```
$ node primer13
Ime Marko
Prezime Markovic
Starost 25
Marko Markovic
```

# Automatsko dodeljivanje parametara konstruktora poljima klase

```
export { };
class Osoba {
    constructor(public ime: string, public prezime: string, public starost: number) {
    }
    stampaj(): void {
        console.log(this.ime, this.prezime, this.starost);
    }
}
const os1 = new Osoba('Marko', 'Markovic', 25);
os1.stampaj();
```

# Klasa sa privatnim poljima

primer15.ts

```
export class Osoba {
    constructor(private ime: string, private prezime: string, private starost: number) {
    }
    stampaj(): void {
        console.log(this.ime, this.prezime, this.starost);
    }
}
```

primer16.ts

```
import { Osoba } from "./primer15";

const os1 = new Osoba('Marko','Markovic',25);
os1.stampaj();
```

# Definisanje gettera/settera

```
export { };
class Osoba {
    private _ime: string = '';
    //getter
    public get ime(): string {

        return this._ime;
    }

    //setter
    public set ime(novoIme: string) {
        if (novoIme.length > 10) {
            console.log('Ime ne moze imati vise od 10 karaktera');
            return;
        }
        this._ime = novoIme;
    }
}
const os = new Osoba();
os.ime = 'Marko231334353465754 74';
console.log(os.ime);
```

# Interfejsi

- Interfejs je struktura koja definiše ugovor u aplikaciji
- Može se iskoristiti za definisanje tipa promenljive
- Definiše sintaksu koju klase treba da slede
- Interfejs definiše samo potpise svojstva i metoda
- Klasa koja implementira interfejs mora da implementira sve što je definisano u interfejsu
- TypeScript kompajler ne konvertuje interfejs u javascript on samo koristi interfejs da proveri tip (duck typing)

# Interfejs - primer

```ts
export{};
interface IOsoba {
    id: number;
    ime: string;
    prezime: string;
}
var os1: IOsoba = {
    id: 1,
    ime: "Marko",
    prezime: "Markovic"
};
```

tsc

```js
var os1: IOsoba = {
    id: 1,
    ime: "Marko",
    prezime: "Markovic"
};
```

.ts fajl

.js fajl

# Interfejs kao parametar funkcije

```
export{};

interface IOsoba {
    id: number;
    ime: string;
    prezime: string;
}

function Stampaj(os: IOsoba) {
    console.log(os.id, os.ime, os.prezime);
}

const os1= {
    id: 1,
    ime: "Marko",
    prezime: "Markovic",
    adresa: "Cara Dusana 22"
};

Stampaj(os1);
```

# Klasa bazirana na interfejsu

```
export {};

interface IOsoba {
    ime: string;
    prezime: string;
}

class Student implements IOsoba {
    constructor(public ime: string, public prezime: string, public smer: string) {}

    stampaj(): void {
        console.log('Ime:', this.ime);
        console.log('Prezime:', this.prezime);
        console.log('Smer:', this.smer);
    }
}

const st = new Student("Marko", "Markovic", "Informacioni sistemi");
st.stampaj();
```

# Pitanje 1

Prevodjenje typescript koda koji se nalazi u fajlu primer01.ts u javascript fajl primer01.js  vrši se korišćenjem komande:

a. compile primer01
b. npx tsc primer01
c. npx node primer01

Odgovor: b

# Pitanje 2

Polja typescript klase imaju podrazumevano sledeći modifikator pristupa:

a. public
b. private
c. protected

Odgovor: a

# Pitanje 3

Konstruktor typescript klase je funkcija koja  ima:

a. isto ime kao i klasa
b. ime constructor
c. ime get

Odgovor: b

# Pitanje 4

Unutar osoba.ts fajla definisan je samo interfejs IOsoba. Šta se dobija izvršavanjem komande: tsc osoba

a. osoba.js fajl koji je prazan
b. osoba.js fajl koji sadrži interfejs
c. osoba.js fajl koji sadrži klasu

Odgovor: a