Predavanje 6, TypeScript, 2025./2026.

```
mkdir typescript_primer
npm install tsx @types/node
```

Staviti u package.json scripts:

```
"watch": "tsx watch index.ts"
```

index.ts

```ts
// jednostavni tipovi
const x: number = 5.5;
const b: boolean = false;
const s: string = "Jurica";
const o: object = { a: 1, b: 2 };

// tipizacija funkcije
const f: (o: object) => void = (o: object) => {
    console.log(o);
}

// polja jednostavnih tipova
const arr: number[] = [1, 2, 3, 4];

// TS jednostavno ne dopušta ovo ako je n: string
function elOfArrDummy(n: number): void {
    console.log(n);
}

// tipizacija ovoga
const printArr = (arr: number[]) => {
    for (let el of arr)
        elOfArrDummy(el);
}

// printArr(arr);
```

```
// na kraju tipizirajmo polje
// polje više tipova
const arr2: (number | boolean)[] = [1, 0, false, false, 1, false, 0, true];
// opet će mi trebati printArr pa napravimo generički printArr
function genPrintArr<T>(arr: T[]): void {
    for (let i in arr) {
        let el: T = arr[i];
        console.log(el);
    }
}

genPrintArr(arr);
genPrintArr(arr2);

// vratimo se tipizaciji arr2


// složeni tipovi
interface Car {
    brand: string,
    nSeats: number,
    isUsed: boolean,
    drive: () => void
};

const car: Car = {
    brand: "VW",
    nSeats: 5,
    isUsed: true,
    drive: () => { console.log("reeeevvin'") }
};

// tip možemo deklarirati i inline
const motorcycle: { brand: string, cc: number } = {
    brand: "Harley-Davidson",
    cc: 1000
};

// ofc možemo imati polje složenih tipova ili više njih
// ali ne možemo assignati vrijednosti 1, false, {a: 1, b: 2}
const vehicles: (Car | { brand: string, cc: number })[] = [
    // 1, false, { a: 1, b: 2 }
    car, motorcycle
];
```

```
// callb
const multiplyAndPrint = (arr: number[], printCallb: (arr: number[]) => void) =>
{
    let arr2 = arr.map((el: number) => 2 * el);
    printCallb(arr2);
}
multiplyAndPrint([1, 2, 3], genPrintArr);


// opcionalni argumenti
// export const optionalArgumentsFun = (n: number, callb?: (n: number) => void)
=> {
//     if (callb)
//         callb(n);
//     else console.log("No callb");
// }
// optionalArgumentsFun(128, (n) => { console.log(n) });
// optionalArgumentsFun(129);

// importanje i eksportanje (bilo čega, npr. funkcija ili tipova)
import { optionalArgumentsFun } from "./some_fun"; // ne treba nastavak .ts
optionalArgumentsFun(128, (n) => { console.log(n) });
optionalArgumentsFun(129);
```

stvoriti some_fun.ts

```
import { t, ct } from './some_type';

export const optionalArgumentsFun: ct = (n: number, callb?: t) => {
    if (callb)
        callb(n);
    else console.log("No callb");
}
```

stvoriti some_type.ts

```
export type t = (n: number) => void;
export type ct = (n: number, callb?: t) => void;
```