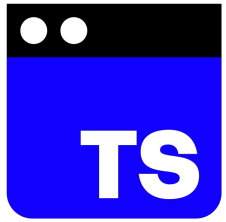
Code Academy

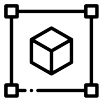


Tipų aprašymas

TypeScript

2 paskaita

Paskaitos eiga



Objekto tipų aprašymas

Masyvo tipo aprašymas

Funkcijų tipų aprašymas

Objekto tipų aprašymas

Objektų tipų aprašymas naudojant type alias

Objekto tipui apibūdinti galime kurti `type alias`.

Tipo aprašymas:

```
type Person = {
  name: string,
  surname: string,
  age: number,
  height: number,
  married: boolean
};
```

Person objektas 1:

```
const person: Person = {
  name: 'Serbentautas',
  surname: 'Bordiūras',
  age: 25,
  height: 180,
  married: false
};
```

Person objektas 2:

```
const person: Person = {
  name: 'Manora',
  surname: 'Svantaitė',
  age: 27,
  height: 170,
  married: true
};
```

Objektų tipų aprašymas naudojant interfaces

Objekto tipui apibūdinti galime kurti `interfaces`.

Tipo aprašymas:

```
interface Person {
  name: string,
  surname: string,
  age: number,
  height: number,
  married: boolean
};
```

Person objektas 1:

```
const person: Person = {
  name: 'Serbentautas',
  surname: 'Bordiūras',
  age: 25,
  height: 180,
  married: false
};
```

Person objektas 2:

```
const person: Person = {
  name: 'Manora',
  surname: 'Svantaitė',
  age: 27,
  height: 170,
  married: true
};
```

Koks skirtumas tarp interface ir type alias?

Praktikoje naudojami tiek interface tiek type alias.

Dažniausiai programuotojų komanda nusprendžia kokia stilistika aprašinės objektus. Visgi yra keli skirtumai.

Koks skirtumas tarp interface ir type alias?

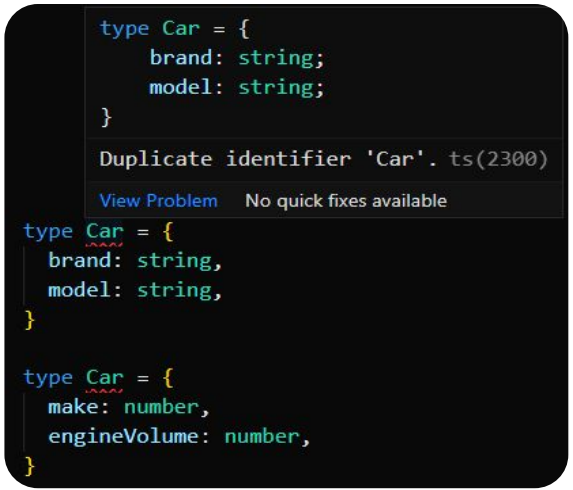
1. `type alias` gali būti ne tik objektas, o interface privalo būti objektas

```
GERAI
type Id = number;

BLOGAI
Interface Identification = number;
```

Koks skirtumas tarp interface ir type alias?

2. `interface` gali būti papildomas po deklaravimo, o type alias negali būti papildomas



Klausimai?

Užduotis 1

Sukurkite 4 objektų tipus

1. Sukurkite 2 objekto tipus naudodami `type alias`
    - Animal - 4 savybės
    - Flat - 5 savybės
2. Sukurkite 2 objekto tipus naudodami `interface`
    - Flower - 4 savybės
    - Student - 5 savybės

Masyvo tipo aprašymas

Masyvo tipo aprašymas naudojant type[ ]

```
interface Address {
country: string;
city: string;
street: string;
number: string;
}

const numbers: number[] = [1, 2, 3, 4];
const words: string[] = ["Don't", "come", "easy", "to", "me"];

const addresses: Address[] = [
  { country: 'USA', city: 'Washington', street: 'Halaway str.', number: '17-202a'  },
  { country: 'Lithuania', city: 'Vilnius', street: 'Kauno g.', number: '1-17'  },
  { country: 'UK', city: 'London', street: 'Bridge aven.', number: '15'  },
];
```

Masyvo tipo aprašymas naudojant Array<type>

```
type Address = {
  country: string,
  city: string,
  street: string,
  number: string,
}

const numbers: Array<number> = [1, 2, 3, 4];
const words: Array<string> = ["Don't", "come", "easy", "to", "me"];

const addresses: Array<Address> = [
  { country: 'USA', city: 'Washington', street: 'Halaway str.', number: '17-202a' },
  { country: 'Lithuania', city: 'Vilnius', street: 'Kauno g.', number: '1-17' },
  { country: 'UK', city: 'London', street: 'Bridge aven.', number: '15' },
];
```

type[ ] VS Array<type>

Masyvo tipą galima aprašyti kaip tik norite, nėra jokio funkcinio skirtumo.

Visgi praktikoje dažniau naudojama laužtinių skliaustų sintaksė.

```
const arrayOfStrings: string[] = ['a', 'b', 'c'];
const arrayOfNumbers: number[] = [1, 2, 3];
const arrayOfBooleans: boolean[] = [true, false];
```

Klausimai?

Užduotis 2

Duomenų formavimo užduotis naudojant masyvus

1. Sukurkite studento tipą:
    - vardas: string
    - pavardė: string
    - kursas: number
    - vidurkis: number
2. Sukurkite studentų masyvą iš 6 studentų

Duomenų formavimo užduotis naudojant masyvus

3. Panaudokite studentų masyvą, kad sukurtumėte:
   - pilnų vardų masyvą:

```
[
    ‘Skara Blauzdaitė’,
    ‘Makaronas Bomžpakis’,
    ‘Frakas Skveras’
]
```

   - pirmojo kurso studentų masyvą
   - visų studentų vidurkį
4. Visiems gautiems duomenims aprašykite tipus!

Funkcijų tipų aprašymas

Funkcijos tipo aprašymas naudojant type alias

```
type Add = (a: number, b: number) => number;
const add: Add = (a, b) => a + b;

const sum = add(5, 4);
console.log(sum); 9
```

```
type Add = (a: number, b: number) => number;
const add: Add = function (a, b) {
    return a + b;
}

const sum = add(5, 4);
console.log(sum); 9
```

Funkcijų tipo aprašymas naudojant interface

```
interface Add {
   (a: number, b: number): number
}
const add: Add = (a, b) => a + b;

const sum = add(5, 4);
console.log(sum); 9
```

```
interface Add {
   (a: number, b: number): number
}
const add: Add = function (a, b) {
   return a + b;
}

const sum = add(5, 4);
console.log(sum); 9
```

Funkcijų tipo aprašymas deklaravimo metu

```
const add = (a: number, b: number): number =>
a + b;

const sum = add(5, 4);
console.log(sum); 9
```

```
const add = function (a: number, b: number):
number {
    return a + b;
}

const sum = add(5, 4);
console.log(sum); 9
```

type alias vs interface vs funkcijos deklaravimo metu

- `interface` naudojamas funkcijoms kuomet reikalinga kurti hibridinį tipą. Pusiau funkcija, pusiau objektas
  - Hybrid type

- `type alias` skirtas aprašyti funkcijos tipui, kuomet tipas nėra hibridinis, bet naudojamas daugiau nei vienai funkcijai

- Funkcijos deklaravimo metu tipus aprašome, kuomet toks funkcijos tipas nebus naudojamas kitose projekto vietose ir yra nesudėtingas

type alias praktinis pavyzdys

```
type BinaryNumericFunction = (a: number, b: number) => number;

const add: BinaryNumericFunction = function (a, b) {
  return a + b
}

const subtract: BinaryNumericFunction = (a, b) => a / b;

let multiply: BinaryNumericFunction = function (a, b) {
  return a * b;
}

let divide: BinaryNumericFunction = (a, b) => a / b;
```

interfaces praktinis pavyzdys

```
interface RectangleAreaCalculator {
  (a: number, b: number): number,
  square(a: number): number
}

const rectangleAreaCalculator: RectangleAreaCalculator = Object.assign(
  (a, b) => a * b,
  {
    square: (a) => a ** 2
  }
);

const rectArea = rectangleAreaCalculator(4, 7);
console.log(rectArea); 28

const squareArea = rectangleAreaCalculator.square(5);
console.log(squareArea); 25
```

funkcijos deklaravimo metu praktinis pavyzdys

```
const printDate = function (date: Date): void {
  console.log(date);
}

const calcArrSum = (arr: number[]): number => {
  return arr.reduce((sum, x) => sum + x);
}

function capitalize(str: string): string {
  return str[0].toUpperCase() + str.slice(1);
}
```

Klausimai?

Užduotis 3

Funkcijų tipų aprašymas

1. Sukurkite funkcijas ir aprašykite jas tipais:
2. Funkciją, kuri grąžina skaičių padauginta iš 2:
    - Panaudokite šią funkciją sudauginti vienai reikšmei
    - Panaudokite šią funkciją sukurti naują dvigubų reikšmių masyvą iteruojant per skaičių masyvą, naudojant `Array.prototype.map`
3. Funkciją, kuri tarpus sakinyje pakeičia brūkšneliais (labas rytas → labas-rytas)
4. Funkciją, kuri skaičiuoja skaičių masyvo vidurkį

Iki kito karto!