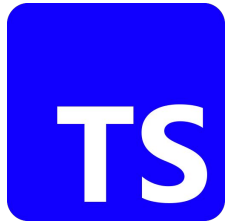
Code Academy



Bendriniai tipai

TypeScript

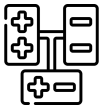
6 paskaita

Paskaitos eiga

Bendrinio tipo apibūdinimas

Bendrinių klasių aprašymas

Bendrinių funkcijų aprašymas



Bendrinio tipo paveldimumas

Bendrinio tipo apibūdinimas

Kas yra bendrinis tipas - generic type?

Skaitydami medžiagą galite žodį “bendrinis” pakeisti - “bendros paskirties” ar “bendro naudojimo”.

Bendrinis tipas - tai kintamojo, funkcijos parametro arba klasės konstruktoriaus tipas - kuris perduodamas vykdymo metu.

Šis tipas nėra konkretus tipo, funkcijos ar klasės konstruktoriaus deklaravimo metu. Tipas perduodamas Bendrinio tipo naudojimo metu.

Programavimo konstruktai yra aprašomi bendros paskirties tipais tuomet, kuomet jų panaudojimas/paskirtis yra nepriklausoma nuo tipo - yra bendro naudojimo.

Bendrinių tipų panaudojimo pavyzdys

```
const sort = <T>( arr: T[], cmpFunction: (a: T, b: T) => number): T[] => {
  return [...arr].sort(cmpFunction);
}

const numbers = [1, -2, 3, -4, 5, -6];
const words = ['a', 'bb', 'ac', 'dc', 'cb'];

      const numbersAsc: number[]
const numbersAsc = sort(numbers, (a, b) => a - b);
numbersAsc;  [ -6, -4, -2, 1, 3, 5 ]
      const wordsDesc: string[]
const wordsDesc = sort(words, (a, b) => b.localeCompare(a));
wordsDesc;  [ 'dc', 'cb', 'bb', 'ac', 'a' ]
```

Dažniausi bendrinių tipų panaudojimo atvejai

- Aprašant bendrinius algoritmus:
  - Rikiavimas
  - Rūšiavimas
  - Paieška
  - Trynimas

- Aprašant duomenų struktūras
  - Sąrašas (List)
  - Medis (Tree)
  - Eilė (Queue)
  - Maišos lentelės (Hash tables)

- Aprašant Bendrines klases
  - Formos
  - Validacijos klasės

Klausimai?

Bendrinių funkcijų aprašymas

Bendrinių funkcijų aprašymas

Dažnai funkcijos atlieka veiksmus nepriklausomus nuo pačių kintamųjų tipo:

- Spausdina
- Talpina į kitą struktūrą
- Rikiuoja
- Filtruoja
- Kopijuoja ir t.t.

Tokiais atvejais yra patogu aprašyti funkcijas naudojant bendriniu tipus, išvengiant kodo dubliavimo.

Bendrinius algoritmus būtų galima aprašyti ir su `any` tipu, tačiau naudojant `any`, funkcijos grąžinimo tipas nebūtų susietas su parametrų tipu/-ais.

Funkcija naudojanti any VS Bendrinė funkcija

```ts
const copyValue = (original: any) => {
  if (original instanceof Object) {
    return JSON.parse(JSON.stringify(original));
  }

  return original;
}

const number: any
const number = copyValue(7);

const arr: any
const arr = copyValue([1, 2, 3]);

const obj: any
const obj = copyValue({
  name: 'Serbenth',
  surname: 'Le Bordouir'
});
```

```ts
const copyValue = <Type>(original: Type): Type => {
  if (original instanceof Object) {
    return JSON.parse(JSON.stringify(original));
  }

  return original;
}

const number: 7
const number = copyValue(7);

const arr: number[]
const arr = copyValue([1, 2, 3]);

const obj: {
    name: string;
    surname: string;
}
const obj = copyValue({
  name: 'Serbenth',
  surname: 'Le Bordouir'
})
```

Klausimai?

Bendrinio tipo paveldimumas

Bendrinio tipo paveldimumas - aibės nurodymas

Kartais norime aprašyti bendrines funkcijas arba struktūras tik tam tikrai tipų aibei. Pvz. :

- Tik iteruojamiems kintamiesiems
- Tik prototipo objektams, kurie turi metodą `move`
- ir t.t.

Tokias atvejais mes turime aprašyti bendrinius tipus taip, kad jie priklausytų reikalingai tipų aibei. Tam ir naudojamas bendrinių tipų paveldimumas.

`Type` yra objektas turintis savybes `name` ir `surname`

```
<Type extends { name: string, surname: string }>
```

Tipų paveldimumo pavyzdžiai

```
conts getFullname = <Person extends { name: string, surname: string }>(
      { name, surname }: Person): string => {
      Return name[0].toUpperCase() + name.slice(1) + ' ' + surname[0].toUpperCase() +
surname.slice(1)}

console.lo(getFullname({ name: 'Dziuvesis', surname: 'gardesis' }));  Dziuvesis Gardesis
console.lo(getFullname({ name: 'Kempe', surname: 'Grybe', age: 17  }));  Kempe Grybe

const isNotEmpty = <Type extends { length: number }>(iter: Type): boolean => {
      Return iter.lenght > 0;
}

console.log(isNotEmpty([1, 5, 2, 5, 6]));  true
console.log(isNotEmpty([]));  false
console.log(isNotEmpty('labas'));  true
console.log(isNotEmpty([' ']));  false
```

Bendrinių klasių aprašymas

Bendrinės klasės

Kartais norime sukurti ne tik bendrinę funkciją, bet visą prototipą, kuris atliks bendrinę logiką.

Geriausias, jau egzistuojantis pavyzdys yra masyvas (Array). Nepriklausomai nuo masyvo elementų tipo, galima pridėti elementą, pašalinti, rikiuoti, filtruoti ir t.t.

Norint perpanaudoti kodą ir padaryti jį *lankstų*, programuotojai neretai naudoja bendrinius tipus kurti klasėms arba duomenų struktūroms.

Bendrinės klasės pavyzdys - Esybės

```
interface IFeddable {
 weight: number;
 energy: number;
 eat(food: number): void;
}
```

```
class Cat implements IFeedable {
 constructor(
  public weight: number,
  public energy: number,
  ) { }

 eat(food: number): void {
  this.energy += food * 0.1;
  this.weight += food * 0.0001;
  }
}
```

```
class Person implements IFeedable {
 constructor(
  public weight: number,
  public energy: number,
 ) { }

 eat(food: number): void {
  this.energy += food * 0.15;
  this.weight += food * 0.00005;
  }
}
```

Bendrinės klasės pavyzdys - Bendrinė klasė

```
class Shelter<Guest extends IFeedable> {
  constructor(private guests: Guest[] = []) {}

  takeIn(guest: Guest): void {
    this.guests.push(guest);
  }

  feedGuests(allFood: number) {
    const allWeight = this.guests.reduce(prev, guest) => prev + guest.weight, 0);
    const foodPerWeight = allFood / allWeight;

    this.guests.forEach((guest) => {
      guest.eat(foodPerWeight * guest.weight);
    });
  }

  viewGuestVitals(): void {
    console.table(
      this.guests.map(({ weight, energy }) => ({
        weight: weight.toFixed(),
        Energy: energy.toFixed(),
      }))
    );
  }
}
```

Bendrinės klasės pavyzdys - Žmonės

```
const person1 = new Person(80, 89000);
const person2 = new Person(75, 92000);
const peopleShelter = new Shelter<Person>();
peopleShelter.takeIn(person1);
peopleShelter.takeIn(person2);
```

```
(method) Shelter<Person>.viewGuestVitals(): void
```

```
peopleShelter.viewGuestVitals();
```

(index)	weight	energy
0	'80.00'	'89000.00'
1	'75.00'	'92000.00'

```
(method) Shelter<Person>.feedGuests(allFood: number): void
```

```
peopleShelter.feedGuests(1200);
```

```
(method) Shelter<Person>.viewGuestVitals(): void
```

```
peopleShelter.viewGuestVitals();
```

(index)	weight	energy
0	'80.65'	'89193.55'
1	'75.60'	'92181.45'

Bendrinės klasės pavyzdys - Katės

```
const cat1 = new Cat(3, 1400);
const cat2 = new Cat(3.5, 1300);
const catShelter = new Shelter<Cat>();
catShelter.takeIn(cat1);
catShelter.takeIn(cat2);

            (method) Shelter<Cat>.viewGuestVitals(): void
catShelter.viewGuestVitals();
```

(index)	weight	energy
0	'3.00'	'1400.00'
1	'3.50'	'1300.00'

```
            (method) Shelter<Cat>.feedGuests(allFood: number): void
catShelter.feedGuests(450);

            (method) Shelter<Cat>.viewGuestVitals(): void
catShelter.viewGuestVitals();
```

(index)	weight	energy
0	'3.21'	'1420.77'
1	'3.74'	'1324.23'

Klausimai?

Paskaitos darbas

Paskaitos darbas

Paskaitoje atliksime užduotis, tokia eiga:

1. Sprendžiame užduotis savarankiškai
2. Po savarankiško sprendimo laiko (10-30 min.) dėstytojas išsprendžia 1 užduotį argumentuodamas sprendimą
3. Studentai užduoda klausimus apie sprendimą
4. Sprendimų palyginimas
5. Atliekama sekanti užduotis

Jeigu išsprendėte užduotį anksčiau nei kiti, spręskite sekančias užduotis.

Užduoties aptarimo metu, nesidrovėkite klausti kuo daugiau klausimų.
Nebūtinai jūsų sprendimas yra prastesnis. Galbūt net geresnis?

Iki kito karto!