



**Code
Academy**



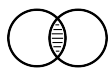
Išplėstiniai tipai

TypeScript

4 paskaita



Paskaitos eiga



Tipų sąjungos operatorius -
unions



Tipų rinkiniai - tuples



Tipų taikymo prielaidos -
assertions



Konkretūs tipai - literal types



Išvardijimai - enums

Sajungos operatorius - union





Sąjungos operatorius

Sąjungos operatorius - leidžia apibūdinti tipą, kaip vieną iš daugelio išvardintų tipų atskirtų operatoriu `|`.

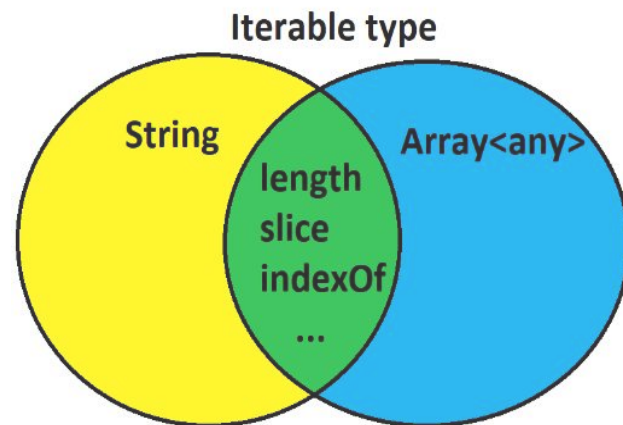
Tokiu atveju galite aprašyti kintamojo arba funkcijos parametro tipą lanksčiai. Galime sakyti, kad tipas sudarytas iš daugelio tipų yra tipų aibė - arba tipų sąjunga.

```
type ID = string | number;  
  
const a: ID = 'Labas';  
const b: ID = 7;
```



Panaudojimo atvejai

Dažniausiai jungos operatorius naudojamas apibūdinti tipui, kuris pasižymi bendrais bruožais. Tuomet galime naudoti tipų sąjungą aprašyti logikai, kuri remiasi tais bendrais bruožais.



```
type IterableType = string | Array<any>;

const validateMaxLength = (value: IterableType, max: number): boolean => {
  return value.length <= max;
};
```

Tipų rinkiniai - tuples





Tipų rinkiniai - tuples

Tipų rinkiniai, tai konkretaus dydžio masyvo tipas. Kiekvienas masyvo elementas aprašomas tipu, nebūtinai tokiu pačiu.

```
type Entry = [string, any];  
  
type Coordinate2D = [number, number];  
  
type Setter<Type> = (val: Type) => void;  
type Getter<Type> = () => Type;  
type EncapsulationMethods<T> = [Setter<T>, Getter<T>];
```



Panaudojimo atvejai

Kuomet norime, kad funkcija grąžintų kelis kintamuosius, patogų juos grąžinti masyve.

Naudodami tuples galime nurodyti grąžinamų reikšmių konkretų kiekį ir tipus.

Grąžinant tuple reikšmes, patogų naudoti masyvo destrukūrizaciją.

```
const getCircleInfo = (radius: number): [string, string] => {  
  const perimeter = 2 * Math.PI * radius;  
  const area = Math.PI * radius ** 2;  
  
  return [perimeter.toFixed(2), area.toFixed(2)];  
}  
  
const circleInfo = getCircleInfo(4);  
const [perimeter2, area2] = getCircleInfo(4);  
const [perimeter3, area3] = getCircleInfo(6);  
console.log(circleInfo[0], circleInfo[1]); 25.13 50.27  
console.log(perimeter2, area2); 25.13 50.27  
console.log(perimeter3, area3); 37.70 113.10
```


Tipų prielaidos - assertions





Tipų prielaidos - assertions

Tipų prielaidos, tai nurodymas TypeScript serveriui (kuris tikrina jūsų TS kodo korektiškumą), kad tam tikrą tipą jis suvoktų taip, kaip nurodo programuotojas.

Tai daroma tuomet, kai programuotojas žino daugiau negu gali žinoti TypeScript serveris ir *pagelbėja* TypeScript serveriui *suvokti*, tipus konkrečiau.



Tipų prielaidos - assertions

Darant tipų prielaidas, reikia būti labai atidiems, nes TypeScript serveris programuotoju besąlygiškai *tiki*.

Tipų prielaidų reikėtų vengti kiek tik įmanoma labiau, nes anksčiau ar vėliau programuotojai suklysta ir programa tampa nekorektiška.



Panaudojimo atvejai

```
const contactForm = document.querySelector('.js-contact-form');
```

```
const contactForm: Element | null
```

```
Object is possibly 'null'. ts(2531)
```

[View Problem](#) No quick fixes available

```
contactForm.addEventListener('submit', (e) => {  
  e.preventDefault();  
});
```

```
const contactForm = document.querySelector('.js-contact-form') as HTMLFormElement;
```

```
const contactForm: HTMLFormElement
```

```
contactForm.addEventListener('submit', (e) => {  
  e.preventDefault();  
});
```

Konkretūs tipai - literal types





Konkretūs tipai - literal types

Konkretūs tipai skirti sumažinti primityvių kintamųjų aibėms.

Galite konkretinti tik `string`, `number` ir `boolean` tipo reikšmes.

Tipų konkrešinimui dažniausiai naudojami apibūdinti, funkcijų parametrus ar objektų savybes.

```
function buildChart(type: "line" | "column" | "area") {  
  // build chart  
  return "chart has been build";  
}
```

```
buildChart("")
```

area	area
column	
line	



Panaudojimo atvejai

```
type Category = 'Food' | 'Clothes' | 'Other';
```

```
const milkCategory: Category = 'Food';
```

```
const breadCategory: Category
```

Type '"food"' is not assignable to type 'Category'. Did you mean '"Food"'? ts(2820)

[View Problem](#) No quick fixes available

```
const breadCategory: Category = 'food';
```

```
type Weekday = 1 | 2 | 3 | 4 | 5 | 6 | 7;
```

```
const monday: Weekday = 1;
```

```
const tuesday: Weekday
```

Type '0' is not assignable to type 'Weekday'. ts(2322)

[View Problem](#) No quick fixes available

```
const tuesday: Weekday = 0;
```

Išvardijimai - enums





Išvardinimai - enums

Išvardinimai, tai reikšmė ir tipas vienu metu.

Aprašant išvardijimą (enum) yra sukuriamas tipas, skirtas apibūdinti konstantų grupei, ir tuo pat metu sukuriamas JavaScript objektas, kurį naudojant galite pasiekti tas konstantas.

Išvardijimai naudojami apibūdinti reikšmių, pasirinkimų ar kategorijų variacijas.

Aprašant išvardijimo konstantas, jos automatiškai numeruojamos nuo 0, tačiau konstantom galima nurodyti konkrečias reikšmes dėl aiškumo, skaitomumo.

Išvardijimo tikslas - išvengti programuotojo rašymo klaidų.



Panaudojimo atvejai

```
const authReducer: Reducer<AuthState, AuthAction> = (state = initialState, action) => {  
  switch (action.type) {  
    case AuthActionType.AUTH_LOADING: { ...  
    }  
    case AuthActionType.AUTH_SUCCESS: { ...  
    }  
    case AuthActionType.AUTH_FAILURE: { ...  
    }  
    case AuthActionType.AUTH_LOGOUT: { ...  
    }  
    case AuthActionType.AUTH_CLEAR_ERROR: { ...  
    }  
    default: return state;  
  }  
};
```

```
export enum AuthActionType {  
  AUTH_LOADING = 'AUTH_LOADING',  
  AUTH_LOGOUT = 'AUTH_LOGOUT',  
  AUTH_CLEAR_ERROR = 'AUTH_CLEAR_ERROR',  
  AUTH_SUCCESS = 'AUTH_SUCCESS',  
  AUTH_FAILURE = 'AUTH_FAILURE',  
}
```

Klausimai?



Paskaitos darbas





Paskaitos darbas

Paskaitoje atliksime užduotis, tokia eiga:

1. Sprendžiame užduotis savarankiškai
2. Po savarankiško sprendimo laiko (~10 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!

