TypeScript Funkcie, podmienky

kurz Vývoj progresívnych webových aplikácií

Eduard Kuric

STU FIIT

Funkcie

- Deklarácia funkcie

```
function name(parameter: type, …): returnType  {
  // rob nieco
}

function add(a: number, b: number)
: number {
  return a + b;
}
let sum = add('10', '20’); // error
```

Funkcie : void

```
function echo(message: string): void {
 console.log(message.toUpperCase());
}
function add(a: number, b: number) {
  return a + b;
}
```

- Ak neurčíme návratový typ explicitne, transpilátor odvodí typ (v tomto prípade number)
- Ak má funkcia rôzne vetvy, ktorých nábratové hodnoty majú rozne typy, transpilátor odvodí typ `union` alebo `any`
  - **preto pridávajme explicitné typy čo najviac ako je možné**

Typ funkcie

```
//  definícia typu
let add: (x: number, y: number) => number;

// priradenie premennej, ok typy sedia
add = function (x: number, y: number) {
  return x + y;
};
```

Typ funkcie /2

```
// naraz deklaracia + priradenie
let add: (a: number, b: number) => number =
    function (x: number, y: number) {
      return x + y; }
    ;

// error, ine typy parametrov
add = function (x: string, y: string): number
{
     return x.concat(y).length;
};
```

Voliteľné parametre

```
function multiply(a: number, b: number,
                  c?: number): number {
  // kontrolu, ci je parameter odovzdany
  // robime takto:
  // typeof c !== 'undefined'
}
```

- **Voliteľné parametre musia nasledovať až za povinnými parametrami.**

Predvolená hodnota parametra

```
function applyDiscount(price, discount = 0.05) {
  return price * (1 - discount);
}
```

```
function applyDiscount(price: number,
                       discount: number = 0.05): number {
  return price * (1 - discount);
}
```

```
// error, v definicia typu nemoze obsahovat predv. param.
let promotion: (price: number,
        discount: number = 0.05) => number;
```

Rest parameter

- Umožňuje funkcii akceptovať žiaden alebo niekoľko parametrov určitého typu.
- Platia tieto pravidlá:
  - Funkcia má iba rest parameter
  - Ak má viac parametrov, rest parameter musí byť posledný
  - Typ rest parametra musí byť `array type`

```
function fn(...rest: type[]) {
   ...
}
```

Rest parameter /2

```
function getTotal(...numbers: number[]): number {
  let total = 0;
  numbers.forEach((num) => total += num);
  return total;
}

console.log(getTotal()); // 0
console.log(getTotal(10, 20)); // 30
console.log(getTotal(10, 20, 30)); // 60
```

Preťaženie (overloading) funkcií

```
// definicia (signatura) funkcie
function add(a: number, b: number): number;

// definicia (signatura) funkcie
function add(a: string, b: string): string;

// implementacia funkcie
function add(a: any, b: any): any { return a + b; }
```

Preťaženie (overloading) funkcií /2

- **Preťaženie funkcií s rôznym počtom parametrov a typmi nie je podporované**

```
// signatura
function sum(a: number, b: number): number;

// signatura
function sum(a: number, b: number, c: number): number;

// implemetacia, ak by nebol parem. c volitelny,
// transpilator by vypisal error
function sum(a: number, b: number, c?: number): number {
  if (c) return a + b + c; return a + b;
}
```

Preťaženie (overloading) funkcií /3

- `function display(a:string, b:string):void {}`
- `function display(a:number): void {}`

- `// error, Compiler Error: Duplicate function implementatio`

Preťaženie metód (v triede)

```
class Counter {
  private current: number = 0;
  count(): number; // sig
  count(target: number): number[];  // sig
  // implementacia
  count(target?: number): number | number[]
  { … }
}
```

if, else if, else

```
let itemCount = 11;

if (itemCount > 0 && itemCount <= 5) {
    …
} else if (itemCount > 5 && itemCount <= 10) {
    …
} else {
    …
}
```

ternárny operátor

```
const max = 100;
let counter = 100;

counter < max ? counter++ : counter = 1;
```

switch case

```
let targetId = 'btnDelete';

switch (targetId) {
    case 'btnUpdate':
        console.log('Update');
        break;
    case 'btnDelete':
        console.log('Delete');
        break;
    case 'btnNew':
        console.log('New');
        break;
    default:
        console.log('DEFAULT');
}
```

for, while

```
for (let i = 0; i < 10; i++) {
    if (i % 2)
        continue;
    console.log(i);
}

let i = 0;
while (i < 10) {
    i++
    if(i == 5)
        break;
}
```

do while

```
let i = 0;

do {
    console.log(i);
    i++
} while (i < 10);
```