TypeScript

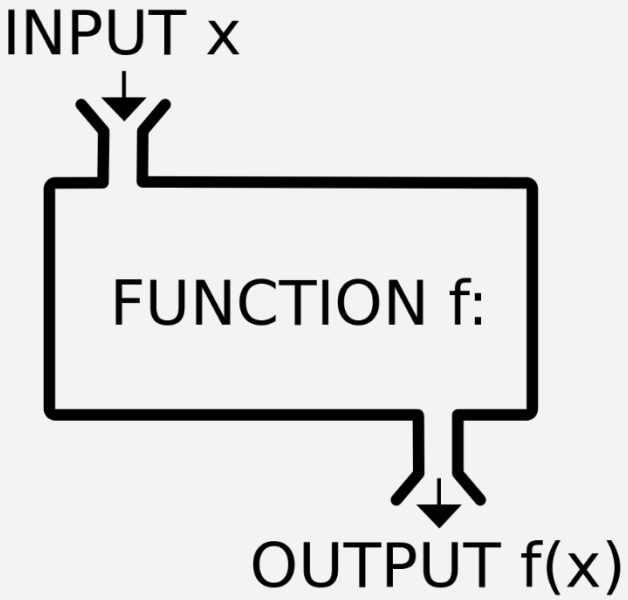
Functions

Content

1. Functions
2. Arrow functions
3. Optional Parameters
4. Default Parameters
5. Rest Parameters
6. Function overloadings

1. Functions

- Functions are the building blocks of readable, maintainable, and reusable code.




```
function name(parameter: type, parameter:type,...): returnType {
    // do something
}
```

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

```ts
function echo(message: string): void {
    console.log(message.toUpperCase());
}
```

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

2. Arrow functions

- Fat arrow notations are used for anonymous functions, It allows a short syntax for writing function expressions. They are also called lambda functions in other languages.

```
const squareOld = function(number) {
  return number * number;
};
const squareNew = number => {
  return number * number;
};
// for 1 parameter -> can ommit parentheses
const squareNew1 = number => {
  return number * number;
};
// for single statement -> remove return keyword & curly braces
const squareNew2 = number => number * number;

console.log(squareOld(5), squareNew(5), squareNew1(5), squareNew2(5));
```

3. Optional Parameters

```
function multiply(a: number, b: number, c?: number): number {

    if (typeof c !== 'undefined') {
        return a * b * c;
    }
    return a * b;
}
```

```
function multiply(a: number, b?: number, c: number): number {

    if (typeof c !== 'undefined') {
        return a * b * c;
    }
    return a * b;
}
```

4. Default Parameters

```
function name(parameter1=defaultValue1,...) {
   // do something
}
```

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

console.log(applyDiscount(100)); // 95
```

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

console.log(applyDiscount(100)); // 95
```

5. Rest Parameters

- A rest parameter allows a function to accept zero or more arguments of the specified type. In TypeScript, the rest parameters follow these rules:
  - A function has only one rest parameter.
  - The rest parameter appears last in the parameter list.
  - The type of the rest parameter is an array type.

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

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

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

6. Function overloadings

```
function addNumbers(a: number, b: number): number {
    return a + b;
}

function addStrings(a: string, b: string): string {
    return a + b;
}
```

```
function add(a: number | string, b: number | string): number | string {
    if (typeof a === 'number' && typeof b === 'number')
        return a + b;

    if (typeof a === 'string' && typeof b === 'string')
        return a + b;
}
```

```
function add(a: number, b: number): number;
function add(a: string, b: string): string;
function add(a: any, b: any): any {
    return a + b;
}
```

```
function sum(a: number, b: number): number;
function sum(a: number, b: number, c: number): number;
function sum(a: number, b: number, c?: number): number {
    if (c) return a + b + c;
    return a + b;
}
```

THE END