TypeScript Functions

1. Named Functions

A **named function** has a specific name and can be reused multiple times in the code.

Syntax

```
function functionName(parameters): returnType {
  // function body
}
```

Example

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

console.log(add(5, 10)); // Output: 15
```

Key Points

- The function name is add.
- It takes two parameters (a and b), both of type number.
- It returns a number.

2. Anonymous Functions

An **anonymous function** does not have a name. It is usually assigned to a variable.

Syntax

```
let variableName = function(parameters): returnType {
  // function body
};
```

Example

```
let multiply = function(x: number, y: number): number {
  return x * y;
};

console.log(multiply(4, 5)); // Output: 20
```

Key Points

- The function is stored in a variable (multiply).
- It does not have a function name.
- It behaves like a regular function.

3. Arrow Functions (Lambda Functions)

Arrow functions provide a shorter syntax for writing functions.

Syntax

```
        let functionName = (parameters): returnType => expression;
```

Example

```
      let square = (num: number): number => num * num;

      console.log(square(6)); // Output: 36
```

Key Points

- Uses => (fat arrow) instead of function keyword.
- **Single-line functions** don't need {} or return keyword.
- **Multi-line functions** require {} and return.

Multi-line Example

```
      let greet = (name: string): string => {
        return `Hello, ${name}!`;
      };

  console.log(greet("Pavan")); // Output: Hello, Pavan!
```

Summary Table

Type	Syntax Example	Key Features
Named Function	function sum(a, b) { return a + b; }	Has a name, reusable, traditional syntax
Anonymous Function	let multiply = function(x, y) { return x * y; };	No name, stored in a variable
Arrow Function	let square = (x) => x * x;	Shorter syntax, uses =>