Here are some code examples illustrating the usage of generics in TypeScript:

1. Generic Functions:

```typescript
function identity<T>(arg: T): T {
    return arg;
}

let result = identity<number>(10); // result is of type number
```

2. Generic Classes:

```typescript
class Box<T> {
    private value: T;

    constructor(value: T) {
        this.value = value;
    }

    getValue(): T {
        return this.value;
    }
}

let box = new Box<string>("Hello");
console.log(box.getValue()); // Output: Hello
```

3. Using Constraints with Generics:

```typescript
interface Printable {
    print(): void;
}

function printValue<T extends Printable>(obj: T): void {
    obj.print();
}

class MyClass implements Printable {
    print(): void {
```

```
        console.log("Printing from MyClass");
    }
}

printValue(new MyClass()); // Output: Printing from MyClass
```

4. Generic Functions with Arrays:

```typescript
function printArray<T>(arr: T[]): void {
    arr.forEach(item => console.log(item));
}

printArray<number>([1, 2, 3]); // Output: 1 2 3
printArray<string>(["a", "b", "c"]); // Output: a b c
```

5. Using Multiple Type Parameters:

```typescript
function combine<T, U>(a: T, b: U): [T, U] {
    return [a, b];
}

let result = combine<number, string>(10, "Hello");
// result is inferred as [number, string]
console.log(result); // Output: [10, "Hello"]
```

These examples demonstrate various ways to use generics in TypeScript, including generic functions, classes, constraints, and working with multiple type parameters. They illustrate how generics provide flexibility and type safety in writing reusable code.