# TypeScript Generic Classes



**Summary**: in this tutorial, you will learn how to develop TypeScript generic classes.

## Introduction to TypeScript generic classes

A generic class has a generic type parameter list in angle brackets `<>` that follows the name of the class:

```
class className<T>{
    //...
}
```

TypeScript allows you to have multiple generic types in the type parameter list. For example:

```
class className<K,T>{
    //...
}
```

The generic constraints are also applied to the generic types in the class:

```
class className<T extends TypeA>{
    //...
}
```

Placing the type parameter on the class allows you to develop methods and properties that work with the same type.

## TypeScript generic classes example

In this example, we will develop a generic `Stack` class.

A stack is a data structure that works on the last-in-first-out (or LIFO) principle. It means that the first element you place into the stack is the last element you can get from the stack.

Typically, a stack has a size. By default, it is empty. The stack has two main operations:

- Push: push an element into the stack.
- Pop: pop an element from the stack.

The following shows a complete generic Stack class called `Stack<T>`:

```
class Stack<T> {
    private elements: T[] = [];

    constructor(private size: number) {
    }
    isEmpty(): boolean {
        return this.elements.length === 0;
    }
    isFull(): boolean {
        return this.elements.length === this.size;
    }
    push(element: T): void {
        if (this.elements.length === this.size) {
            throw new Error('The stack is overflow!');
        }
        this.elements.push(element);

    }
    pop(): T {
        if (this.elements.length == 0) {
            throw new Error('The stack is empty!');
        }
        return this.elements.pop();
    }
}
```

The following creates a new stack of numbers:

```
let numbers = new Stack<number>(5);
```

This function returns a random number between two numbers, `low` and `high`:

```
function randBetween(low: number, high: number): number {
    return Math.floor(Math.random() * (high - low + 1) + low);
}
```

Now, you can use the `randBetween()` function to generate random numbers for pushing into the `numbers` stack:

```
let numbers = new Stack<number>(5);

while (!numbers.isFull()) {
    let n = randBetween(1, 10);
    console.log(`Push ${n} into the stack.`)
    numbers.push(n);
}
```

Output:

```
Push 3 into the stack.
Push 2 into the stack.
Push 1 into the stack.
Push 8 into the stack.
Push 9 into the stack.
```

The following shows how to pop elements from the stack until it is empty:

```
while (!numbers.isEmpty()) {
    let n = numbers.pop();
    console.log(`Pop ${n} from the stack.`);
}
```

Output:

```
Pop 9 from the stack.
Pop 8 from the stack.
Pop 1 from the stack.
Pop 2 from the stack.
Pop 3 from the stack.
```

Similarly, you can create a stack of strings. For example:

```
let words = 'The quick brown fox jumps over the lazy dog'.split(' ');

let wordStack = new Stack<string>(words.length);

// push words into the stack
words.forEach(word => wordStack.push(word));

// pop words from the stack
while (!wordStack.isEmpty()) {
    console.log(wordStack.pop());
}
```

How it works:

- First, split the sentences into words.
- Second, create a stack whose size is equal to the number of words in the `words` array.
- Third, push elements of the `words` array into the stack.
- Finally, pop words from the stack until it is empty.

In this tutorial, you have learned how to develop generic classes in TypeScript.