- **Модуль:** Typescript. Generics and type manipulations
- **Урок:** Generics functions

TypeScript Handbook
from
Иван Петриченко
udemy

# ФУНКЦИИ-ОБОБЩЕНИЯ

Для создания второго (и следующих) аргумента, который тоже будет определен во время вызова функции, достаточно прописать его через запятую:

```ts
function processingData<T, S>(data: T, options: S): T {
    return data;
}

const res1 = processingData(1, "fast");
const res2 = processingData("1", "slow");
const res3 = processingData<number, string>(10, "slow");
```

Для выполнения разных операций с разными типами данных используем **сужение типов**:

```ts
function processingData<T, S>(data: T, options: S): string {
    switch (typeof data) {
        case "string":
            return `${data}, speed: ${options}`;
        case "number":
            return `${data.toFixed()}, speed: ${options}`;
        default:
            return "Not valid";
    }
}
```

Из заглушки можно сформировать **массив** и им типизировать **аргументы**. В таком случае передаем массив нужных типов:

```ts
function processingData<T, S>(data: T[], options: S): T[] {
    return data;
}
let res1 = processingData([1.004], "fast");
let res2 = processingData(["1"], "slow");
const res3 = processingData<number, string>([10], "slow");
```

Для методов точно такие же правила, как и для функций. А если они описаны в интерфейсе, то есть три варианта синтаксиса:

```ts
interface DataSaver {
    processing: <T>(data: T) => T;
}

const saver: DataSaver = {
    processing: <T>(data: T) => {
        // ... Some actions
        console.log(data);
        return data;
    },
};

saver.processing(5);
```

```ts
interface DataSaver {
    processing: <T>(data: T) => T;
}

const saver: DataSaver = {
    processing(data) {
        // ... Some actions
        console.log(data);
        return data;
    }
};

saver.processing(5);
```

```ts
interface DataSaver {
    processing: <T>(data: T) => T;
}

const saver: DataSaver = {
    processing: (data) => {
        // ... Some actions
        console.log(data);
        return data;
    }
};

saver.processing(5);
```

- **Модуль:** Typescript. Generics and type manipulations
- **Урок:** Generics functions

TypeScript Handbook
from
Иван Петриченко
udemy

# ФУНКЦИИ-ОБОБЩЕНИЯ КАК АННОТАЦИИ

Функции-обобщения можно использовать в качестве аннотаций. Например, создаем отдельную функцию-шаблон, а потом используем в нужных местах:

```
1  function processing<T>(data: T): T {
2      return data;
3  }
4
5  let newFunc: <T>(data: T) => T = processing;
6
7  interface DataSaver {
8      processing: <T>(data: T) => T;
9  }
10
11 const saver: DataSaver = {
12     processing: processing
13 };
```

Если мы хотим оптимизировать саму аннотацию, то можно вынести её в отдельный интерфейс:

```
1  interface ProcessingFn {
2      <T>(data: T): T
3  }
4
5  function processing<T>(data: T): T {
6      return data;
7  }
8
9  let newFunc: ProcessingFn = processing;
10
11 interface DataSaver {
12     processing: ProcessingFn;
13 }
```