- **Модуль:** Typescript. Базовые знания
- **Урок:** Использование системы типов в функциях

TypeScript Handbook
from
Иван Петриченко
udemy

ТИПИЗАЦИЯ АРГУМЕНТОВ ФУНКЦИЙ

Первый ключевой момент любой функции - это её **аргументы**. При стандартной настройке, если не указать тип каждого аргумента - TS подскажет об ошибке. Это значит, что аргумент может быть чем угодно

А это позволяет выполнить с этим аргументом любую операцию внутри функции, а это большая вероятность получить ошибку.

Два неправильных способа её исправить:

- 👉 Аргументам назначить тип any (что угодно)
- 👉 В конфигурации TS поставить **noImplicitAny** в позицию false (tsconfig.json)

Правильный способ - указать каждому аргументу чем он может быть:

```
function logBrtMsg(isBirthday: boolean, userName: string, age: number) {
    if (isBirthday) {
        return `Congrats ${userName.toUpperCase()}, age: ${age + 1}`;
    } else {
        return "Error";
    }
}
```

```
const logBrtMsg = (
    isBirthday: boolean,
    userName: string,
    age: number
) => {}
```

Теперь при вызове функции TS будет подсказывать нам, какие аргументы нужно передать и в каком виде:

```
logBrtMsg(true, "John", 40);
```

Такая аннотация типов аргументов у функции позволяет выполнять только доступные операции в теле функции, позволяет получать подсказки при её вызове и предупреждения, если вы сделали что-то не так

Это особенно полезно, когда вы работаете с чужими функциями или с теми, которые вы создали когда-то давно 🙂

- **Модуль:** Typescript. Базовые знания
- **Урок:** Использование системы типов в функциях

TypeScript Handbook
from
Иван Петриченко
udemy

ТИПИЗАЦИЯ ВОЗВРАЩАЕМОГО ЗНАЧЕНИЯ

Второй ключевой момент любой функции - это **возвращаемое значение**. Если TS будет знать, что именно функция может вернуть, то он сможет вам подсказывать об ошибках в будущем, при её использовании

Указывать тип возвращаемого значения нужно после аргументов:

```
1  function logBrtMsg(isBirthday: boolean, userName: string, age: number): string {
2      if (isBirthday) {
3          return `Congrats ${userName.toUpperCase()}, age: ${age + 1}`;
4      } else {
5          return "Error";
6      }
7  }
```

Например, эта функция всегда возвращает строку, а значит, к её результату можно применить только методы, которые есть у строк.

Таким образом любой разработчик сразу видит, что он получит в итоге, не копаясь во внутренностях функции. А TS будет предупреждать вас об ошибках. Даже если вы захотите изменить логику внутри

Если функция ничего не возвращает (вывод в консоль, отправка данных на сервер, работа с DOM-деревом...), то возвращаемый тип будет `void`

Void – это не только отсутствие возвращаемого значения, но и его полное игнорирование.

Нужно ли ставить аннотацию типа возвращаемого значения, если TS и сам это может сделать?

! Да, это позволит намного быстрее понимать логику любой функции и отлавливать ошибки, когда у вас несколько возвращаемых значений