

# КАК НЕ ПОТЕРЯТЬ КОНТЕКСТ

При усложнении функционала всегда можно потерять контекст в классах. В TS существует возможность четко сказать, чем должен быть контекст и не получать такие ситуации:

```
1 class Player {
2   #login: string;
3
4   constructor(login: string) {
5     this.#login = login;
6   }
7
8   login() {
9     return `Player ${this.#login} online!`;
10  }
11 }
12
13 const player = new Player("Test");
14 const test = player.login;
15 test(); // Error: Cannot read properties of undefined (reading '#login')
```

Проблему можно решить при помощи **bind** или **стрелочной функции**. Но до запуска кода вы не узнаете об ошибке. Поэтому в TS есть вариант сразу сказать, **чем будет контекст** прямо первым аргументом в функции. И вы увидите ошибку на этапе разработки:

```
1 class Player {
2   #login: string;
3
4   constructor(login: string) {
5     this.#login = login;
6   }
7
8   login(this: Player) { // this type
9     return `Player ${this.#login} online!`;
10  }
11 }
12
13 const player = new Player("Test");
14 const test = player.login;
15 test(); // Error: The 'this' context of type 'void' is not assignable to method's 'this' of type 'Player'
```

Иногда мы из метода возвращаем контекст, то есть ссылку на экземпляр объекта. В таких случаях **не стоит жестко типизировать возвращаемое значение**, так как оно может сломать логику:

```
1 connect(): Player {
2   // Do stmh
3   return this;
4 }
```

**Неправильно:** даже в наследуемых классах будет возвращать Player, что может привести к ошибке

```
1 connect() {
2   // Do stmh
3   return this;
4 }
```

**Правильно:** всегда будет возвращаться текущий экземпляр класса, даже в потомках

При помощи контекста мы можем проверить, к какому классу относится экземпляр и написать свой **защитник типа**:

```
1 isPro(): this is CompetitivePlayer {
2   return this instanceof CompetitivePlayer;
3 }
4
5 const somePlayer: Player | CompetitivePlayer = new CompetitivePlayer("Test");
6 somePlayer.isPro() ? console.log(somePlayer) : console.log(somePlayer);
```