

Раздел

[Продвинутая работа с функциями](#)

Навигация по уроку

У стрелочных функций нет «this»

Стрелочные функции не имеют «arguments»

Итого

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)
[→ Продвинутая работа с функциями](#) 16-го мая 2020

Повторяем стрелочные функции

Давайте вернёмся к стрелочным функциям.

Стрелочные функции – это не просто «сокращение», чтобы меньше писать. У них есть ряд других полезных особенностей.

При написании JavaScript-кода часто возникают ситуации, когда нам нужно написать небольшую функцию, которая будет выполнена где-то ещё.

Например:

- `arr.forEach(func)` – `func` выполняется `forEach` для каждого элемента массива.
- `setTimeout(func)` – `func` выполняется встроенным планировщиком.
- ...и так далее.

Это очень в духе JavaScript – создать функцию и передать её куда-нибудь.

И в таких функциях мы обычно не хотим выходить из текущего контекста. Здесь как раз и полезны стрелочные функции.

У стрелочных функций нет «this»

Как мы помним из главы [Методы объекта, "this"](#), у стрелочных функций нет `this`. Если происходит обращение к `this`, его значение берётся снаружи.

Например, мы можем использовать это для итерации внутри метода объекта:

```
1 let group = {
2   title: "Our Group",
3   students: ["John", "Pete", "Alice"],
4
5   showList() {
6     this.students.forEach(
7       student => alert(this.title + ': ' + student)
8     );
9   }
10 };
11
12 group.showList();
```

Здесь внутри `forEach` использована стрелочная функция, таким образом `this.title` в ней будет иметь точно такое же значение, как в методе `showList: group.title`.

Если бы мы использовали «обычную» функцию, была бы ошибка:

```
1 let group = {
2   title: "Our Group",
3   students: ["John", "Pete", "Alice"],
4
5   showList() {
6     this.students.forEach(function(student) {
7       // Error: Cannot read property 'title' of undefined
8       alert(this.title + ': ' + student)
9     });
10  }
11 };
12
13 group.showList();
```

Раздел

[Продвинутая работа с функциями](#)

Навигация по уроку

У стрелочных функций нет «this»

Стрелочные функции не имеют «arguments»

Итого

Комментарии

Поделиться



Редактировать на GitHub



Ошибка возникает потому, что `forEach` по умолчанию выполняет функции с `this`, равным `undefined`, и в итоге мы пытаемся обратиться к `undefined.title`.

Это не влияет на стрелочные функции, потому что у них просто нет `this`.

⚠ Стрелочные функции нельзя использовать с `new`

Отсутствие `this` естественным образом ведёт к другому ограничению: стрелочные функции не могут быть использованы как конструкторы. Они не могут быть вызваны с `new`.

ℹ Стрелочные функции VS `bind`

Существует тонкая разница между стрелочной функцией `=>` и обычной функцией, вызванной с `.bind(this)`:

- `.bind(this)` создаёт «связанную версию» функции.
- Стрелка `=>` ничего не привязывает. У функции просто нет `this`. При получении значения `this` — оно, как обычная переменная, берётся из внешнего лексического окружения.

Стрелочные функции не имеют «arguments»

У стрелочных функций также нет переменной `arguments`.

Это отлично подходит для декораторов, когда нам нужно пробросить вызов с текущими `this` и `arguments`.

Например, `defer(f, ms)` принимает функцию и возвращает обёртку над ней, которая откладывает вызов на `ms` миллисекунд:

```
1 function defer(f, ms) {
2   return function() {
3     setTimeout(() => f.apply(this, arguments), ms)
4   };
5 }
6
7 function sayHi(who) {
8   alert('Hello, ' + who);
9 }
10
11 let sayHiDeferred = defer(sayHi, 2000);
12 sayHiDeferred("John"); // выводит "Hello, John" через 2
```

То же самое без стрелочной функции выглядело бы так:

```
1 function defer(f, ms) {
2   return function(...args) {
3     let ctx = this;
4     setTimeout(function() {
5       return f.apply(ctx, args);
6     }, ms);
7   };
8 }
```

Здесь мы были вынуждены создать дополнительные переменные `args` и `ctx`, чтобы функция внутри `setTimeout` могла получить их.

Итого

Стрелочные функции:

- Не имеют `this`.
- Не имеют `arguments`.

Раздел

[Продвинутая работа с функциями](#)

Навигация по уроку

У стрелочных функций нет «this»

Стрелочные функции не имеют «arguments»

Итого

Комментарии

Поделиться



Редактировать на GitHub

- Не могут быть вызваны с `new`.
- (У них также нет `super`, но мы про это не говорили. Про это будет в главе [Наследование классов](#)).



Всё это потому, что они предназначены для небольшого кода, который не имеет своего «контекста», выполняясь в текущем. И они отлично справляются с этой задачей!

Проводим [курсы по JavaScript и фреймворкам](#). ✕

💬 Комментарии

перед тем как писать...

