

 → [Язык JavaScript](#) → [Продвинутая работа с функциями](#)

 16 мая 2020 г.

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();
```

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`.



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 секунды
```

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`.
- 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!



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

Комментарии

- Если вам кажется, что в статье что-то не так - вместо комментария напишите [на GitHub](#).
- Для одной строки кода используйте тег `<code>`, для нескольких строк кода — тег `<pre>`, если больше 10 строк — ссылку на песочницу ([plnkr](#), [JSBin](#), [codepen...](#))
- Если что-то непонятно в статье — пишите, что именно и с какого места.

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ

ИЛИ ЧЕРЕЗ DISQUS ?

Имя

13

Поделиться

Лучшие Новые Старые

1

1.sohr.1

25 дней назад

Не можешь решать задачки на codewars, Leetcode и много вопросов в процессе обучения
Тогда переходи в Тг чат @communityforprogg

0 0 Ответить



Юрий

месяц назад

⚡ Хочешь повторить стрелочные функции?

Тогда переходи в ТГ-блог «Джун на фронте»!

👤 Автор - системный администратор, который с декабря 2021 года освоил HTML, CSS, JS, Vue, Nuxt, React Native, MongoDB и Node.js.

Следи за моим путем в мир разработки: от новичка до создателя 🤖 Телеграм-бота для автоматической отправки откликов!

💡 Ежедневно ты найдешь полезные ответы на насущные вопросы, анализ рынка вакансий и советы от человека, прошедшего этот путь.

Вбивайте «Джун на фронте» и присоединяйтесь к нам!

0 0 Ответить

A

Achmed

4 месяца назад edited

В плане контекста интересна ещё такая конструкция `(0,a.f)()`
по отношению к стрелочным функциям они все равны: `a.f() = (a.f)() = (0,a.f)()`
а вот по отношению к методам, определённым через `function`, `(a.f)()` и `(0,a.f)()` ведут себя по-разному. При вызове такой функции `this` должен быть равен объекту, которому она принадлежит. Но при вызове `(0,a.f)()` `this` равен либо `undefined` в строгом режиме, либо `Window`. При этом такая конструкция часто встречается в обфусцированных файлах.

```
let b = {
  bb: function(){
    'use strict';
    let a={
      f:()=>console.log(this),
      ff:function(){
        console.log(this)
      }
    }
  }
}
```

```

    },
    (a.f)(); /* b */
    (a.ff)(); /* a */
    (0,a.f)(); /* b */
    (0,a.ff)(); /* undefined */
  }
}
b.bb();

```

Также this в function внутри метода обращается в undefined/Window

[показать больше](#)

0 0 Ответить [↗](#)

S **Sosockol** → Achmed
3 месяца назад edited

кста, чот не нашёл где пишут про самовызывающиеся функции (()=>{})() Может, плохо искал?

В статье про устаревшее ключевое слово VAR, раздел IIFE.

1 0 Ответить [↗](#)

H **Никита Степанов**
4 месяца назад

В некоторых задачах применение **apply** никак не обосновывается. И без передачи контекста все прекрасно работает. Например в последней если строку:

```
setTimeout(() => f.apply(this, arguments), ms)
```

...заменить на:

```
setTimeout(() => f(...arguments), ms)
```

....то ошибок не будет. И код будет работать точно так же, как и в первом случае.

Чем мой пример хуже предлагаемого авторами. Почему в примерах применяют передачу this, я не понимаю. Может кто-нибудь объяснит)

0 0 Ответить [↗](#)


A **Achmed** → Никита Степанов
4 месяца назад

Обосновывается) Там же написано "Это отлично подходит для декораторов, когда нам нужно пробросить вызов с текущими this и arguments." И тут же показывают как это сделать.

Другое дело, что пример некорректный и контекст никак не влияет на этот конкретный sayHi

1 0 Ответить [↗](#)

ДН Этот комментарий ожидает проверки. [Показать комментарий.](#)

 **FL1MO** → Джун на фронте
10 месяцев назад

Какой же ты жалкий... Я разблокировал одного из заблокированных и это оказывается ты:)

можно вопрос? Почему ты ноешь именно тут?

0 0 Ответить

D

devmen

→ Джун на фронте

год назад

исчезни, хуила

6 0 Ответить

B

Валерий Катрук

год назад

Не совсем понял последний пример

```
function defer(f, ms) {  
  return function(...args) {  
    let ctx = this;  
    setTimeout(function() {  
      return f.apply(ctx, args);  
    }, ms);  
  };  
}
```

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

Зачем объявлять переменную let ctx = this; ? Разве не достаточно передать в аргументы return f.apply(this, args); ?

0 0 Ответить



Tarasov Alexander

→ Валерий Катрук

год назад

в анонимной функции function() внутри setTimeout() this другой и равен undefined, поэтому нужно передать контекст из defer

3 0 Ответить

A

Anatolii Tarasov (Tarasov.Fron

→ Tarasov Alexander

10 месяцев назад edited

Поправьте, если не прав. Контекст в стрелочную функцию передается от анонимной возвращаемой функции, которая впоследствии может быть вызвана как метод объекта (соответственно this будет установлен на объект), а не из defer.

0 0 Ответить



Василий Васильевич

→ Anatolii Tarasov (Tarasov.Fron

9 месяцев назад

В первом случае стрелочная функция берет переменную this из своей области видимости, потому что по умолчанию в стрелочной функции не создается this как в методе или обычной функции. Во втором случае, используется замыкание, что бы пробросить нужный параметр this

0 0 Ответить

B

Валерий Катрук

→ Tarasov Alexander

год назад

Спасибо тебе, Добрый человек! Теперь понял!

0 0 Ответить 

J

JabaDushnila

→ Валерий Катрук

год назад

так или иначе используется это же слово `this`.
логика уходит в безвременный отпуск

1 0 Ответить 



Egor Demeshko

→ JabaDushnila

год назад

в самой функции, которая передается в качестве аргумента в вызов функции `setTimeout`, `this` будет пустой(если мы говорим про стрелочную функцию), он должен будет браться из контекста выполнения более глубокого, или предыдущего.

поэтому строкой выше создают переменную `ctx` куда сохраняют ссылку на `this`, но этот `this` уже взят из вызова функции `function(...args)`.

`this` это универсальный индикатор-переменная, которая ссылается на текущее скажем так внутреннее значение контекста исполнения в момент вызова функции.

при вызове функции у вас создается каждый раз новый контекст исполнения и значение `this` оценивается именно в этот момент, в момент вызова и старта исполнения функции.

у стрелочной функции в принципе нет такой ссылки, поэтому оно всегда берется снаружи.

а сам `setTimeout` метод глобальный, поэтому вы там цепляете `this = window`.

как я понимаю, потому что фактически идет вызов `window.setTimeout()`;

БУДУ РАД ЕСЛИ МЕНЯ ПОПРАВЯТ или дополнят.

[показать больше](#)

2 0 Ответить 

A

Anatolii Tarasov (Tarasov.Fron

год назад edited

Нужно отметить, что вывод "Стрелочные функции не имеют `arguments`." неточен.

Несмотря на отсутствие `arguments` стрелочные функции могут запоминать и иметь доступ аргументам с помощью записи `(...args)`.

Например, `return (...args) => setTimeout(this , ms, ...args);`

0 3 Ответить 

F

Fedor Karasev

→ Anatolii Tarasov (Tarasov.Fron

год назад

Вы путаете остаточные параметры и псевдомассив `arguments`.

5 0 Ответить 



Anatolii Tarasov (Tarasov.Fron

→ Fedor Karasev

год назад

Спасибо за уточнение. Но мой посыл был в том, чтобы автор отразил в тексте, что стрелочные функции могут принимать аргументы посредством остаточных параметров.

Это позволяет избавиться от пары лишних строчек в коде при присваивании переменной значения this внешней функции.

0 2 Ответить



Maxim

→ Anatolii Tarasov (Tarasov.Fron

год назад

Ваше невнимательное чтение вгоняет читателей в заблуждение. Автор четко отразил конкретную сущность и эта сущность экзотический объект arguments, про переменные, аргументы или параметры речь не шла. Следовательно все это отлично работает.

P.S Зачем придумывать то чего не написано на деле?

3 0 Ответить



Anatolii Tarasov (Tarasov.Fron

→ Maxim

год назад

А вы вообще программист? Как у вас с логикой? "про ... речь не шла. Следовательно все это отлично работает."

Подобную чушь я слышал пару раз. От людей у которых с логикой прямо беда. Теперь и вы добавились.

1 2 Ответить



Maxim

→ Anatolii Tarasov (Tarasov.Fron

год назад edited

Про стрелочные функции в двух главах объяснялось, следовательно про что речь не шла вероятно работает также как у обычных функций. Нетрудно сложить 2 и 2 верно? Или нет?

Как я вижу вы невероятно оскорблены тем что вас поправили, видимо вы из тех людей, которые считают, что абсолютно правы во всем. Увы это не так, просыпайтесь!

P.S Попытка оскорбить такая жалкая, когда найдете за что можно прицепиться ко мне по делу - пишите, пока молча обтекайте :)

6 1 Ответить



Anatolii Tarasov (Tarasov.Fron

→ Maxim

год назад

А вы кроме 2 + 2 больше ничего складывать не умеете?

0 4 Ответить



Albert M

→ Anatolii Tarasov (Tarasov.Fron

7 месяцев назад

Зря Максимку обижаешь, это единственный толкователь смыслов спецификации на этом ресурсе.

0 0 Ответить

А **Александр Катков**
2 года назад



Самое прикольное, что в разделе "У стрелочных функций нет this", если во втором примере код записать вот так:

```
let group = {
  title: "Our Group",
  students: ["John", "Pete", "Alice"],

  showList() {
    this.students.forEach(function(student) {
      alert(group.title + ': ' + student)
    });
  }
};
```

group.showList();

то результат будет точно таким же, как и с использованием стрелочной функции. И это объяснимо. На самом деле стрелочная функция делает тоже самое. Или я ошибаюсь? Вопрос к автору.

1 2 Ответить

И **Илья Лыков** → Александр Катков
2 года назад



Потому что ты создал метод объекта students(контекст this объект group). Метод apply привязывает контекст к функции.

0 1 Ответить

А **Александр Катков** → Илья Лыков
год назад



Если быть точным, то students - это массив. И никакой метод этого массива я не создавал. forEach - это встроенный метод любого массива. Просто если в функции showList "напрямую" взять свойство group.title объекта group, то результат будет точно таким же, если использовать стрелочную функцию. Потому что стрелочная функция делает тоже самое. Автор пишет "Ошибка возникает потому, что forEach по умолчанию выполняет функции с this, равным undefined, и в итоге мы пытаемся обратиться к undefined.title". Это происходит потому что, с моей точки зрения, теряется контекст. Т.е. в this.students.forEach(function(student) this еще равен group, а в this.title он уже равен undefined. Т.е. так работает javascript. Вопрос на самом деле в том, зачем были изобретены стрелочные функции. Хотелось бы услышать мнение автора.

0 2 Ответить

С **Сергей Красносельский** → Александр Катков
год назад



Так суть же в том, что мы используем this для контекста. Попробуй запустить свой код с group.title в таком случае:

```
let clone = group
clone.showList() // работает как нужно
```

```
group = null
clone.showList() // будет ошибка, потому что группа теперь равна 0
```

Убирая `this` и меняя его на `group` ты сам лишаешь смысла такого рода запись, если оставить `this`, то контекстом для `showList` станет `clone`, в тоже время твоя запись через `function declaration` просто не может получить этот контекст, потому что у неё есть свой `this`, чего нет у стрелочной функции

0 0 Ответить ↗

A

Александр Катков

→ Сергей Красносельский

год назад

Понятно, что мы используем `this` для контекста. Я на самом деле писал о другом. О том, что в определенный момент контекст теряется при использовании нами `this`. В этом состоит "особенность" javascripta. Очевидно разработчики языка, изобретая стрелочные функции, пытались устранить эту "особенность".

0 0 Ответить ↗

M

Михаил Виноградов

2 года назад edited

⚡ Ну чё, народ, погнали? ⚡

Реально ли **изучить javascript за 7 месяцев** и трудоустроиться?

Вот я решил проверить и веду свой блог **Джаваскриптизёр** на ютубе, где буду выкладывать видео каждую неделю на протяжении 7 месяцев.

💪💪 Вторая неделя обучения завершена, второй видос тоже готов 💪💪

Если тебе тоже интересен джаваскрипт, присоединяйся:

Ютуб: **Джаваскриптизёр**

ТГ: **@javascriptizerr**

🔥 Удачи всем нам 🔥

0 3 Ответить ↗

A

Александр Катков

→ Михаил Виноградов

2 года назад

Мне кажется, нереально.

1 0 Ответить ↗

P

pipisasa

→ Александр Катков

год назад

Я справился за 4 месяца

0 0 Ответить ↗

A

Александр Катков

→ pipisasa

год назад

Поздравляю. Вместе с трудоустройством?

0 0 Ответить ↗

... ..

```
function defer(f, ms) {  
  return function(...args) {  
    let ctx = this; // (1)  
    setTimeout(function() {  
      return f.apply(ctx, args);  
    }, ms);  
  };  
}
```

Чему равно значение this в строчке 1? Если перед этой строчкой написать alert(this) выведет undefined. Так чему равно значение this??

0 0 Ответить

**Igor Українець**

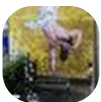
→ incrediblesilent

2 года назад

Потому, что у Вас кусок кода вырван из контекста...
Вот полная реализация, где можно посмотреть this:

```
let john = {  
  name: "John",  
  sayHi() {  
    alert('Hello, ' + this.name);  
  }  
};  
  
function defer(f, ms) {  
  return function(...args) {  
    let ctx = this;  
    alert("ctx: " + ctx);           /* ctx: [object Object] */  
    setTimeout(function() {  
      return f.apply(ctx, args);  
    }, ms);  
  };  
}  
  
john.sayHi = defer(john.sayHi, 2000);  
john.sayHi();                      // выводит "Hello, John" через 2 секунды
```

2 0 Ответить

**Хоц**

→ Igor Українець

год назад edited

немного изменил для большей ясности, от куда именно контекст берется

```
const john = {  
  name: "John",  
  sayHi() {  
    alert('Hello, ' + this.name);  
  }  
}
```

```
};

const sem = {
  name: "Sem"
}

function defer(f, ms) {
  return function() { // ...args - в данном случае не нужны
    let ctx = this;
    console.log(ctx); // {name: 'John', sayHi: f} или {name: 'Sem', sayHi: f} в зависимости от ко
    setTimeout(function() {
```

показать больше

1 0 Ответить



Эдгар

→ incrediblesilent

2 года назад

this определяется в момент выполнения функции, а не её декларирования. Это объект перед точкой в момент вызова функции.

alert(this) выведет [object Window] т.к. вызывая функцию alert вы вызываете метод alert глобального объекта Window браузера.

4 0 Ответить

A

Achmed

→ Эдгар

4 месяца назад

В твоём примере `this` в качестве аргумента подставляется в момент вызова функции. И алерт вернёт контекст того места, откуда его вызвали, а не потому, что это метод глобального объекта.

```
(new class classA {f(){alert(this.constructor.name)}}).f() // classA
```

Здесь `this` определяется в процессе выполнения f() и передаётся глобальному alert, но он не выведет объект Window)

0 0 Ответить

R

risonka

2 года назад

18.07.2022

1 2 Ответить



Yuri

2 года назад

Наглядно о this и стрелках:

```
'use strict';

let admin = {
  name: 'John',
  /* 1 */
  arrow: arg => console.log(arg.name),
  /* 2 */
```

```

    fToArrow() { this.arrow(this) },
    /* 3 */
    fWrapArrow() {
        const wrappedArrow = () => console.log(this.name);
        wrappedArrow();
    },
}

admin.arrow(admin);           /* вообще без this */
admin.fToArrow();             /* this как аргумент */
admin.fWrapArrow();           /* this из функции-обертки */

```

4 0 Ответить

A

Andrey Kulebyakin

2 года назад

Когда выполняешь этот код в консоли

```

let group = {
  title: "Our Group",
  students: ["John", "Pete", "Alice"],

  showList() {
    this.students.forEach(function(student) {
      // Error: Cannot read property 'title' of undefined
      alert(this.title + ': ' + student)
    });
  }
};

group.showList();

```

Никакой ошибки не происходит, просто выводится "undefined John". Почему так?

1 0 Ответить

K

Kron1410

→ Andrey Kulebyakin

2 года назад

Значит у Вас не строгий режим. Сделайте 'use strict' и будет ошибка.

1 0 Ответить



Комментарий был удален.



Wyse

→ Guest

2 года назад

Как успехи?

0 0 Ответить



Комментарий был удален.



Tory495

→ Guest

2 года назад

Надеюсь фронтендером

0 0 Ответить



Комментарий был удален.



Е

Evgeniy АСТЕСК

→ Guest

2 года назад



Не могу найти в телеге сообщество...

0 0 Ответить

Д

Дарья Иванова

2 года назад



Как вызываются стрелочные функции?
Обычные функции вызываются просто name()
а как стрелочные?

2 2 Ответить



Асланов Шамиль

→ Дарья Иванова

2 года назад



стрелочные тоже по имени.

```
let sayHi = (name, phrase) => alert(name + ': ' + phrase)
sayHi('John', 'Hi')
```

6 0 Ответить



AN

→ Дарья Иванова

2 года назад



Есть глава про стрелочные функции. Там как раз про вызовы и отличия от обычной функции.

3 0 Ответить

Р

Роман Носов

3 года назад



```
let user = {
  nam: 'Name',
```

```
  sayHi: () => {
    return this.nam + 'Hello!'
  }
}
```

```
console.log(user.sayHi());
```

не могу разобраться, я же вызываю как метод, но внутри this все равно не определен!

3 0 Ответить

К

Кирилл

→ Роман Носов

2 года назад edited



Так как стрелочная функция не имеет своего контекста (this), то значение this берется из внешнего окружения (другой функции), НО выше стрелочной функции нет другой функции, поэтому значение this = undefined (взято из глобального окружения)

Правильный вариант:

```
let user = {
  nam: 'Name',

  sayHi() {
    let func = () => {
      return this.nam + 'Hello!';
    };

    return func();
  }
}

console.log(user.sayHi());
```

2 0 Ответить



Эдгар

→ Кирилл

2 года назад

Понятнее сказать, что вызывая `user.sayHi()` вызывается стрелочная функция, т.к именно её возвращает этот метод объекта `user`. А у неё нет `this`, она не видит, что вызывается метод на объекте `user`

1 0 Ответить

L

legeida

→ Роман Носов

2 года назад

Здесь `this` привязан к глобальному окружению, а не к объекту

1 0 Ответить

A

Алексей Туренко

→ Роман Носов

3 года назад

```
let user = {
  nam:'Name',

  sayHi: function () {
    return this.nam + 'Hello!'
  }
}
```

1 0 Ответить

M

Mikhail Makarov

3 года назад edited

Ребята зачем в предпоследнем примере используется `apply`?
Если можно написать так:

```
function defer(f, ms) {
  return function() {
    setTimeout(() => f(...arguments), ms)
  };
}
function sayHi(who) {
  alert('Hello, ' + who);
}

let sayHiDeferred = defer(sayHi, 2000);
sayHiDeferred("John"); // выводит "Hello, John" через 2 секунды
```

3 0 Ответить

К

Кирилл

→ Mikhail Makarov

2 года назад edited

Если контекст важен, то контекст обязательно нужно передавать. Например, в коде ниже (1) `this.text = undefined`

```
function defer(f, ms) {

  function func() {
    setTimeout(() => f(...arguments), ms)
  };

  func.text = "How are you?";

  return func;
}

function sayHi(who, text) {
  alert('Hello, ' + who + ", " + this.text); // 1
}

let sayHiDeferred = defer(sayHi, 2000);
sayHiDeferred("John"); // выводит "Hello, John" через 2 секунды
```

0 0 Ответить



Андрей Коляда

→ Mikhail Makarov

3 года назад

Насколько я понимаю, для того чтобы контекст выполнения функции был подвязан и ты смог спокойно применить эту функцию в будущем к любой другой

5 0 Ответить

E

Ernst Laik

3 года назад

Не понял, почему тут написано, что стрелочные функции не имеют "arguments". Как же не имеют, если в главе "Стрелочные функции - основы" утверждалось обратное. `let sum = (a, b) => a + b`; Либо я чего-то не понял, но, простите, что есть (a, b), как не аргументы? Значит, когда необходимо, стрелочные функции таки имеют "arguments"?

3 0 Ответить



Yuri Bil

→ Ernst Laik

3 года назад

arguments - это объект псевдомассив всех параметров, использованных при вызове функции, даже если мы не указывали никакие аргументы при объявлении функции

9 0 Ответить

B

Boom

→ Ernst Laik

3 года назад

Имеется ввиду ...args

0 4 Ответить

A

Aleksandr Volkov

→ Boom

2 года назад

нет, arguments - это такой встроенный объект

1 0 Ответить

A

Азиз Гайсин

→ Ernst Laik

3 года назад

У стрелочных функций также нет **переменной** arguments. Не аргументов, а переменной arguments

12 0 Ответить

E

Eneier

3 года назад edited

```
let group = {
  title: "Cheaken Team",
  students: ["John", "Pete", "Alice"],

  showList() {
    this.students.forEach(function(student) {
      let title = group.title;
      alert(title + ': ' + student)
    });
  }
};

group.showList();
```

кто мешает добавить 1 строку, чтоб работала обычная функция? =) easy

с другой стороны, чем короче код тем лучше в некоторых моментах.

1 0 Ответить



sadaie

→ Eneier

2 года назад edited

Без контекста теряется смысл всей конструкции.

В данном примере видно, что при удалении свойства "title" из объекта "group" оно пропадает из объекта "newGroup"

group.showList();

Это происходит из-за того, что свойство ссылается на объект "group", а не на клонированный объект.

```
let group = {
  title: "Check Team",
  students: ["John", "Pete", "Alice"],
  showList() {
    this.students.forEach(function (student) {
      let title = group.title;
      console.log(title + ': ' + student);
    });
  }
};

let {...newGroup} = group; // Clone object
delete group.title; // Remove property from object
newGroup.showList();
```

2 0 Ответить

E

Ed

→ Eneier

3 года назад

```
let newGroup = group;
group = null;
```

```
newGroup.showList();
// whoops! An error!
```

1 0 Ответить

Д

Давид

→ Eneier

3 года назад

Тем, что это подойдёт только для одного объекта, если у тебя будет описан класс, ты не сможешь обратиться к его экземпляру никак кроме this

0 0 Ответить



Mafi0zy

→ Eneier

3 года назад edited

не стоит цепляться за внешнее название объектов

1 0 Ответить

Г

Георгий Новицкий

3 года назад edited

С вашего позволения внесу уточнение, чтобы не вводить в заблуждения и раздумья).

Стрелочные функции не имеют собственного контекста выполнения, из-за этого они наследуют сущности this и arguments от родительской функции.

20 0 Ответить

К

Klishe

→ Георгий Новицкий

5 месяцев назад

```
let bar = (i) => {
  if (i < 0) return;
  console.log("begin: " + i);
```

```
bar(i - 1);
console.log("end: " + i);
};

console.log(bar(3));
```

Почему тогда здесь создаётся стек контекстов выполнения, если у стрелочных функций их нет?

0 0 Ответить



Георгий Новицкий

→ Klishe

2 месяца назад

У стрелочных функций нет собственного контекста выполнения.

0 0 Ответить



Иван

→ Георгий Новицкий

2 года назад

this sounds great - i'll try to remember! Thanks

0 0 Ответить



TM

3 года назад

может кто объяснить зачем в `setTimeout(() => f.apply(this, arguments), ms)` нужна стрелочная функция? `setTimeout((f.apply(this, arguments), ms)` тоже работает

0 0 Ответить



PFS-WEB

→ TM

3 года назад edited

Ваш код не работает. Если написать как у вас, вот так: `setTimeout(f.apply(this, arguments), ms);` то функция вызывается моментально, а не через указанное время, в данном случае 2 секунды. Дело в том, что вы передаете в `setTimeout` не функцию, а вызов функции. К примеру, попробуйте этот код:

```
setTimeout(sayHi(), 2000);

function sayHi() {
  alert("Hello");
}
```

`sayHi` это функция, которая просто выводит "Hello", запустите этот код и вы увидите, что функция выполнится моментально, т.к мы передали туда вызов функции, то есть, она сразу и вызывается.

А теперь на этот код:

```
setTimeout(sayHi, 2000);
```

показать больше

6 0 Ответить

K**Kichnal Dance**[→ PFS-WEB](#)

2 года назад



Спасибо за подробное разъяснение. Встали некоторые моменты на свои места)

0 0 Ответить

D**Dmitriy**

3 года назад



Уважаемый автор, фраза "У стрелочных функций нет «this»" некорректна. У них есть this и этим this в теле функции можно активно пользоваться.

3 0 Ответить

A**Archie McAlister**

3 года назад



ну смысл немного некорректно передан - this у стрелочной функции есть, но создается он в момент объявления функции и принимает значение либо объекта window, либо, что важнее, внешней функции, в которой наша стрелочная функция было объявлена.

5 0 Ответить

C**Ca Reem**[→ Archie McAlister](#)

3 года назад



Почему тогда нельзя использовать в конструкторе?

0 0 Ответить

C**Слава Переверзев**

3 года назад



Всем привет! не подскажете как переписать такой код `keys.forEach((key) => result[key] = (typeof (obj[key]) !== "object") ? deepCopy(obj[key]) : obj[key])` в обычную функцию без стрелки
Спасибо)

0 0 Ответить

[Загрузить ещё комментарии](#)[Подписаться](#)[О защите персональных данных](#)[Не продавайте мои данные](#)

