

 → [Язык JavaScript](#) → [Основы JavaScript](#)

 9 июля 2022 г.

Стрелочные функции, основы

Существует ещё один очень простой и лаконичный синтаксис для создания функций, который часто лучше, чем Function Expression.

Он называется «функции-стрелки» или «стрелочные функции» (arrow functions), т.к. выглядит следующим образом:

```
1 let func = (arg1, arg2, ...argN) => expression;
```

Это создаёт функцию `func`, которая принимает аргументы `arg1..argN`, затем вычисляет `expression` в правой части с их использованием и возвращает результат.

Другими словами, это сокращённая версия:

```
1 let func = function(arg1, arg2, ...argN) {
2   return expression;
3 };
```

Давайте рассмотрим конкретный пример:

```
1 let sum = (a, b) => a + b;
2
3 /* Эта стрелочная функция представляет собой более короткую форму:
4
5 let sum = function(a, b) {
6   return a + b;
7 };
8 */
9
10 alert( sum(1, 2) ); // 3
```



Как вы можете видеть, `(a, b) => a + b` задаёт функцию, которая принимает два аргумента с именами `a` и `b`. И при выполнении она вычисляет выражение `a + b` и возвращает результат.

- Если у нас только один аргумент, то круглые скобки вокруг параметров можно опустить, сделав запись ещё короче:

```
1 let double = n => n * 2;
2 // примерно тоже что и: let double = function(n) { return n * 2 }
3
4
```



```
alert( double(3) ); // 6
```

- Если аргументов нет, круглые скобки будут пустыми, но они должны присутствовать:

```
1 let sayHi = () => alert("Hello!");
2
3 sayHi();
```



Стрелочные функции можно использовать так же, как и Function Expression.

Например, для динамического создания функции:

```
1 let age = prompt("Сколько Вам лет?", 18);
2
3 let welcome = (age < 18) ?
4   () => alert('Привет!') :
5   () => alert("Здравствуй!");
6
7 welcome();
```



Поначалу стрелочные функции могут показаться необычными и даже трудночитаемыми, но это быстро пройдет по мере того, как глаза привыкнут к этим конструкциям.

Они очень удобны для простых однострочных действий, когда лень писать много слов.

Многострочные стрелочные функции

Стрелочные функции, которые мы видели до этого, были очень простыми. Они брали аргументы слева от `=>` и вычисляли и возвращали выражение справа.

Иногда нам нужна более сложная функция, с несколькими выражениями и инструкциями. Это также возможно, нужно лишь заключить их в фигурные скобки. При этом важное отличие – в том, что в таких скобках для возврата значения нужно использовать `return` (как в обычных функциях).

Вроде этого:

```
1 let sum = (a, b) => { // фигурная скобка, открывающая тело многострочной функ
2   let result = a + b;
3   return result; // если мы используем фигурные скобки, то нам нужно явно указ
4 };
5
6 alert( sum(1, 2) ); // 3
```



Дальше – больше

Здесь мы представили главной целью стрелочных функций краткость. Но это ещё не всё!

Стрелочные функции обладают и другими интересными возможностями.

Чтобы изучить их более подробно, нам сначала нужно познакомиться с некоторыми другими аспектами JavaScript, поэтому мы вернёмся к стрелочным функциям позже, в главе [Повторяем стрелочные функции](#).

А пока мы можем использовать их для простых однострочных действий и колбэков.

Итого

Стрелочные функции очень удобны для простых действий, особенно для однострочных.

Они бывают двух типов:

1. Без фигурных скобок: `(...args) => expression` – правая сторона выражения: функция вычисляет его и возвращает результат. Скобки можно не ставить, если аргумент только один: `n => n * 2`.
2. С фигурными скобками: `(...args) => { body }` – скобки позволяют нам писать несколько инструкций внутри функции, но при этом необходимо явно вызывать `return`, чтобы вернуть значение.

Задачи

Перепишите с использованием функции-стрелки

Замените код Function Expression стрелочной функцией:

```
1 function ask(question, yes, no) {
2   if (confirm(question)) yes()
3   else no();
4 }
5
6 ask(
7   "Вы согласны?",
8   function() { alert("Вы согласились."); },
9   function() { alert("Вы отменили выполнение."); }
10 );
```

решение

```
1 function ask(question, yes, no) {
2   if (confirm(question)) yes()
3   else no();
4 }
5
6 ask(
7   "Вы согласны?",
8
9
10
```

```
() => alert("Вы согласились."),  
() => alert("Вы отменили выполнение.")  
);
```

Выглядит короче и понятней, правда?



Предыдущий урок

Следующий урок



Поделиться



Карта учебника

Проводим курсы по JavaScript и фреймворкам.



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

36

Share

Лучшие

Новые

Старые

Н

Headgehog

🕒 10 часов назад

— 🚩

```
"use strict";
let fun = (question, yes, no) => { return ((question) ? yes() : no()) };
console.log(fun("U soglasen?", () => "YES", () => "NO"));
```

0

0

Ответить

АС

Andrei Chekhanadski

🕒 11 дней назад

— 🚩

```
let ask = (question, yes, no) =>
confirm(question) ? yes() : no();

ask(
'Вы согласны?',
function() { alert("Вы согласились."); },
function() { alert("Вы отменили выполнение."); }
);
```

0

0

Ответить

НП

Никита Покора

🕒 11 дней назад

— 🚩

```
let ask = (question, yes, no) => {
if (confirm(question)) yes()
else no();
}
ask(
"Вы согласны?",
function() { alert("Вы согласились."); },
function() { alert("Вы отменили выполнение."); }
);
```

0

0

Ответить

U

Unnn

🕒 13 дней назад

— 🚩

let ask = confirm('Вы согласны?') ?

© 2007—2023 Илья Кантор [проект](#) [связаться с нами](#) [пользовательское соглашение](#) [политика конфиденциальности](#)

0 1 Ответить

M

Maklamar

🕒 14 дней назад

— 🚩

```
const ask = (question, yes, no) => confirm(question) ? yes(alert('Yes')) :  
no(alert("No"))  
ask('OK?')
```

1 0 Ответить

RD

Regina Dautova

🕒 16 дней назад

— 🚩

```
let question = confirm('Вы согласны?')  
  
const ask = (question) ?  
() => alert("Вы согласились.") :  
() => alert("Вы отменили выполнение.")  
  
ask()
```

0 0 Ответить



Denis Butyrskiy

🕒 19 дней назад

— 🚩

Ребят, всем привет! Изучаю JavaScript.

Главная цель ⚡ - «С нуля до Junior Frontend Developer за 7 месяцев».

На протяжении этих 7 месяцев буду изучать JS и записывать еженедельные **видеоотчёты** об этом.

Всё здесь:

YouTube по названию канала **Denis Butyrskiy**

ТГ @itway_chat - обсуждаем всё, что касается обучения

Давайте объединяться, общаться, делиться друг с другом успехами и неудачами. Это сильно помогает пройти сложные моменты, уже проверено. Под каждым видео есть вся нужная информация 🔥

0 1 Ответить

AM

Арман Минасян

🕒 22 дня назад

— 🚩

```
let ask = (question, yes, no) => {  
  if (confirm(question)) yes()  
  else no();  
}
```

```
ask(  
  "Вы согласны?",  
  () => { alert("Да, согласен."); },  
  () => { alert("Нет, не согласен."); }  
);
```

```
() => { alert( вы согласились. ); },  
() => { alert("Вы отменили выполнение."); }  
);
```

0 0 Ответить

К

Кирилл

→ Арман Минасян

🕒 18 дней назад edited

— 🚩

Лишние скобки для стрелочных функций, когда однострочные действия

```
ask(  
  "Вы согласны?",  
  () => alert("Вы согласились."),  
  () => alert("Вы отменили выполнение.")  
);
```

2 0 Ответить

К

kamytt

🕒 22 дня назад edited

— 🚩

```
let ask = (question, yes, no) => {  
  if (confirm(question)) yes()  
  else no();  
};
```

```
ask('Вы согласны?', () => alert('Вы согласились'), () => alert('Вы  
отменили выполнение'));
```

0 0 Ответить

КК

Katya Klep

🕒 25 дней назад

— 🚩

Разве function ask(...) Это тоже самое что и let ask=(..)?

0 0 Ответить

К

Кирилл

→ Katya Klep

🕒 18 дней назад edited

— 🚩

Нет, есть существенное отличие: вызвать объявленную функцию function ask() { } можно из любого места кода, а function expression (let ask = function() { }) только после объявления (не выше)

1 0 Ответить



Dima Kim

🕒 месяц назад

— 🚩

```
let ask=(question, yes, no) => confirm(question)?yes():no();
```

```
ask(  
  "Вы согласны?",  
  () => alert("Вы согласились."),  
  () => alert("Вы отменили выполнение.")  
);
```

1 0 0

**Алексей Данилюк**

🕒 месяц назад edited



```
let ask = (question, yes, no) =>{
  if (confirm(question)) yes()
  else no();
}
ask("Вы согласны?",
()=>alert('YES'),
()=>alert('No')
)
```

1 0 Ответить

**Road to WEB 2023**

🕒 месяц назад



стрелочные функции

давайте вместе ботать, пытаюсь вкатиться в веб-разработку за январь, roadtoweb2023

0 1 Ответить

**Саня Шар**

🕒 месяц назад



```
let question = confirm ('Вы согласны?');
```

```
let answer = question?
```

```
() => alert ('Вы согласились!');
```

```
() => alert ('Вы не согласились!');
```

```
answer ();
```

0 0 Ответить

**Сергей М.**

🕒 месяц назад



```
let a, b, c;
```

```
let ur = '5x - x2 - 12 = 0';
```

```
let ar = ur.split(' ');
```

```
getCoefs(a, (value.includes('x2'))); // Не пойму как передать услови
```

```
getCoefs(b, (value.endsWith('x'))); // использовать this тоже не по
```

```
getCoefs(c, isFinite(it));
```

```
function getCoefs(it, condition) {
```

```
  let arr = new Array();
```

```
  for (let value of ar) {
```

```
    arr.push(value);
```

```
    if (condition) break;
```

```
  };
```

```
  alert(arr);
```

```
  arr.reverse();
```

```
  if (arr[0] === undefined || 0) {
```


it = 0; // Уравнения бывают неполными

[показать больше](#)

0 0 Ответить

PP

Paul Pumphut

🕒 2 месяца назад

— 🚩

const ask = q => alert(confirm(q) ? 'Вы согласились.' : 'Вы отменили выполнение.');

2 1 Ответить



Алексей Обухов

🕒 2 месяца назад

— 🚩

может кому будет полезно, в однострочной стрелочной функции чтобы вернуть объект, его надо поместить в круглые скобки:

() => ({prop: "value"})

без круглых скобок JS подумает, что это не объект, а многострочная стрелочная функция

6 0 Ответить

Н

HymeraDoma

🕒 2 месяца назад

— 🚩

а так можно в задаче?

let question = ask => ask? alert(`вы согласились`) : alert(`вы не согласились`);

question (confirm(`вы согласны?`));

1 0 Ответить

Загрузить ещё комментарии

[Подписаться](#)

[Privacy](#)

[Не продавайте мои данные](#)