let func = new Function([arg1, arg2, ...argN], functionBody);

Функция создаётся с заданными аргументами arg1...argN и телом functionBody.

Например

let sum = new Function('a', 'b', 'return a + b');

alert( sum(1, 2) ); // 3

А вот функция без аргументов, в этом случае достаточно указать только тело:

let sayHi = new Function('alert("Hello")');

sayHi(); // Hello

Главное отличие от других способов объявления функции, которые были рассмотрены ранее, заключается в том, что функция создаётся полностью «на лету» из строки, переданной во время выполнения.

Все предыдущие объявления требовали от нас, программистов, писать объявление функции в скрипте.

Но new Function позволяет превратить любую строку в функцию. Например, можно получить новую функцию с сервера и затем выполнить её:

let str = ... код, полученный с сервера динамически ...

let func = new Function(str);

func();

Это используется в очень специфических случаях, например, когда мы получаем код с сервера для динамической компиляции функции из шаблона, в сложных веб-приложениях.

**[Замыкание](https://learn.javascript.ru/new-function" \l "zamykanie)**

Обычно функция запоминает, где родилась, в специальном свойстве [[Environment]]. Это ссылка на лексическое окружение (Lexical Environment), в котором она создана (мы разбирали это в главе [Область видимости переменных, замыкание](https://learn.javascript.ru/closure)).

Но когда функция создаётся с использованием new Function, в её [[Environment]] записывается ссылка не на внешнее лексическое окружение, в котором она была создана, а на глобальное. Поэтому такая функция имеет доступ только к глобальным переменным.

function getFunc() {

let value = "test";

let func = new Function('alert(value)');

return func;

}

getFunc()(); // ошибка: value не определено

Сравним это с обычным объявлением:

function getFunc() {

let value = "test";

let func = function() { alert(value); };

return func;

}

getFunc()(); // "test", из лексического окружения функции getFunc

Эта особенность new Function выглядит странно, но оказывается очень полезной на практике.

Представьте, что нужно создать функцию из строки. Код этой функции неизвестен во время написания скрипта (поэтому не используем обычные функции), а будет определён только в процессе выполнения. Мы можем получить код с сервера или с другого ресурса.

Наша новая функция должна взаимодействовать с основным скриптом.

Что если бы она имела доступ к внешним переменным?

Проблема в том, что перед отправкой JavaScript-кода на реальные работающие проекты код сжимается с помощью минификатора – специальной программы, которая уменьшает размер кода, удаляя комментарии, лишние пробелы, и, что самое главное, локальным переменным даются укороченные имена.

Например, если в функции объявляется переменная let userName, то минификатор изменяет её на let a (или другую букву, если она не занята) и изменяет её везде. Обычно так делать безопасно, потому что переменная является локальной, и никто снаружи не имеет к ней доступ. И внутри функции минификатор заменяет каждое её упоминание. Минификаторы достаточно умные. Они не просто осуществляют «тупой» поиск-замену, они анализируют структуру кода, и поэтому ничего не ломается.

Так что если бы даже new Function и имела доступ к внешним переменным, она не смогла бы найти переименованную userName.

**Если бы new Function имела доступ к внешним переменным, при этом были бы проблемы с минификаторами.**

Кроме того, такой код был бы архитектурно хуже и более подвержен ошибкам.

Чтобы передать что-то в функцию, созданную как new Function, можно использовать её аргументы.

## [Итого](https://learn.javascript.ru/new-function" \l "itogo)

Синтаксис:

let func = new Function ([arg1, arg2, ...argN], functionBody);

По историческим причинам аргументы также могут быть объявлены через запятую в одной строке.

Эти 3 объявления ниже эквивалентны:

new Function('a', 'b', 'return a + b'); // стандартный синтаксис

new Function('a,b', 'return a + b'); // через запятую в одной строке

new Function('a , b', 'return a + b'); // через запятую с пробелами в одной строке

Функции, объявленные через new Function, имеют [[Environment]], ссылающийся на глобальное лексическое окружение, а не на родительское. Поэтому они не могут использовать внешние локальные переменные. Но это очень хорошо, потому что страхует нас от ошибок. Переданные явно параметры – гораздо лучшее архитектурное решение, которое не вызывает проблем у минификаторов.