


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

 30 декабря 2022 г.

Синтаксис "new Function"

Существует ещё один вариант объявления функции. Он используется крайне редко, но иногда другого решения не найти.

Синтаксис

Синтаксис для объявления функции:

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

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

Это проще понять на конкретном примере. Здесь объявлена функция с двумя аргументами:

```
1 let sum = new Function('a', 'b', 'return a + b');
2
3 alert( sum(1, 2) ); // 3
```



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

```
1 let sayHi = new Function('alert("Hello")');
2
3 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:

```
1 let str = ... код, полученный с сервера динамически ...
2
3 let func = new Function(str);
4 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

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](#)).

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.



```
1 function getFunc() {
2   let value = "test";
3
4   let func = new Function('alert(value)');
5
6   return func;
7 }
8
9 getFunc(); // ошибка: value не определено
```

Сравним это с обычным объявлением:



```
1 function getFunc() {
2   let value = "test";
3
4   let func = function() { alert(value); };
5
6   return func;
7 }
8
9 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

Синтаксис:

```
1 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:

```
1 new Function('a', 'b', 'return a + b'); // стандартный синтаксис
2 new Function('a,b', 'return a + b'); // через запятую в одной строке
3 new Function('a , b', 'return a + b'); // через запятую с пробелами в одной ст
```

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.



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



8

Поделиться

Лучшие

Новые

Старые

E

Easy Madi_BRO

13 дней назад

Судя по комментариям я понял что я лох.

0

0

Ответить



Юрий

14 дней назад

⚡ Хотите понять синтаксис "new Function"?

Тогда гоу в мой блог 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.

Изучи мой путь в мире разработки: от новичка до создателя 🤖 **Telegram-бота для автоматической отправки** откликов!

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

Вбивайте «Джун на фронте» и присоединяйтесь.

0

0

Ответить



Y

Yurii Vintoniak

8 месяцев назад

Какая используется Тема в примерах кода?

Кто-то может сказать?

0

0

Ответить



Rewwoken → Yurii Vintoniak

7 месяцев назад

Dracula, вероятнее всего

1

0

Ответить



M

Молодой var

10 месяцев назад

Как новичку(3 месяца изучаю js, полтора на этом сайте) после замыкания приятно читать такие легкие уроки)

12

0

Ответить 

N

namediablo

→ Молодой var

месяц назад

— 

замыкания легко дались, а вот рекурсия...

0

0

Ответить 

E

Easy Madi_BRO

→ namediablo

13 дней назад

— 

Не знаю мне рекурсия легко далось. Но сложно было запоминать все эти методы массивов.

0

0

Ответить 



Sergey RJS

год назад edited

— 

Первое, что приходит в голову, размышляя о применении синтаксиса new function - json преобразования. Мы не можем просто взять и скопировать методы таким образом: `JSON.parse(JSON.stringify(obj))`. Конечно, объект можно клонировать и другими способами (пример: `structuredClone(obj)`), но поддерживается он значительно хуже, чем json методы)

Задача: написать функции для `JSON.parse` и `JSON.stringify`, которые позволят клонировать объект вместе с его методами. Начните с простых методов без аргументов.

Пример:

```
let obj = {
  name: "John",
  surname: "Smith",
  fullName() {
    return this.name + ' ' + this.surname;
  },
};
```

```
function stringify(key, value) {
  ''
```

[показать больше](#)

2

0

Ответить 

S

Sentrin

→ Sergey RJS

год назад edited

— 

```
const functionLable = "$function$"
```

```
gify(key, value) {
  if (typeof value == "function") {
    return functionLable + value
  }
  return value
}
```

```
function parse(key, value) {  
  if (value.toString().startsWith(functionLabel))  
    return new Function(value.match(/\{(.\+)\}/s).at(1))  
  return value;  
}
```

0 0 Ответить



Sergey RJS

→ Sentrin

год назад edited

хорошее решение через регулярные выражения, только вместо gify следует написать function stringify чтобы код работал.

Осталось добавить поддержку параметров функции, например:

```
fullName(age) {  
  return `${this.name} ${this.surname} is ${age} years old`;  
},
```

0 0 Ответить



Egor Demeshko

→ Sergey RJS

год назад

насколько я понимаю, примерно об этом и шла речь в статье. спасибо, что поделились, интересно встречается ли потребность в таком на практике. мне как новичку пока сложно представить. пока сталкивался скорее с импортом уже готовых, не строковых, функций методов.

0 0 Ответить



Sergey RJS

→ Egor Demeshko

год назад

На практике вряд ли. Для сохранения данных пользователя, например, в localStorage (либо в базе данных на стороне сервера) достаточно их записать в объект и сохранить, преобразовав в JSON. Функции передавать нет необходимости.

1 0 Ответить

J

John Galt

год назад

Конструктор Function () лучше всего воспринимать как глобально ограниченную версию функции eval(), которая определяет переменные и функции в собственной закрытой области видимости. Возможно вам никогда не придется применять конструктор Function() в своем коде.

Дэвид Флэнаган JavaScript Полное руководство

6 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 5 Ответить



Ігор Українець

2 года назад edited

Експеримент с калькулятором. В калькуляторе вынуждены сохранять значения аргументов для new Function в области глобального лексического окружения (т.к. ни локальные переменные, ни объектные свойства "функцией из строки" не читаются)

```
let a;
let b;
function Calculator() {

    this.multiply = prompt( 'Введите выражение: ', "a * b" );
    a = prompt( 'Введите a: ', 2 );
    b = prompt( 'Введите b: ', 3 );

    return new Function( "a = 1", "b = 1", `return ${this.multiply}` );
}

let calculator = new Calculator();

alert( calculator(a, b) ); // a * b      2      3      6
alert( calculator() );    // 1
alert( calculator(6, 8) ); // 48
```

4 0 Ответить



Ігор Українець

→ [Ігор Українець](#)

год назад edited

Слегка подправил код для для корректной работы с суммированием (+prompt)

```
let a;
let b;
function Calculator() {

    this.multiply = prompt( 'Введите выражение: ', "a * b" );
    a = +prompt( 'Введите a: ', 2 );
    b = +prompt( 'Введите b: ', 3 );

    return new Function( "a = 1", "b = 1", `return ${this.multiply}` );
}

let calculator = new Calculator();

alert( calculator(a, b) ); // a * b      2      3      6
alert( calculator() );    // 1
alert( calculator(6, 8) ); // 48
```

Наглядная демонстрация, что new Function имеет доступ только к глобальным переменным. (здесь

можно легко попробовать сохранять а, б как локальные переменные, или объектные свойства и убедиться, что new Function, ввиду записанной в специальное свойство [[Environment]] ссылки НЕ на внешнее лексическое окружение, в котором она была создана, а на ГЛОБАЛЬНОЕ, не имеет доступа к своему родному внешнему лексическому окружению)

показать больше

1 0 Ответить



Sam Gonzalez

→ Igor Українець

год назад

на русском пожалуйста, тогда получишь ответ

1 3 Ответить



Igor Українець

→ Sam Gonzalez

год назад edited

можете не отвечать)))

Мне бы был интересен Ваш ответ, если бы Вы умели культурно общаться и знали русский язык.

Вот Ваше "грамотное общение" на русском:

"Написал такой код, в консоли выводится верный результат, но как выполненную работу не считает)"

Выучите, как следует русский, тогда, возможно, будет желание общаться с Вами на русском)

1 2 Ответить



Maxim

→ Igor Українець

9 месяцев назад edited

Я вижу ты нормальный парень, но я тебе проясню несколько моментов чтобы было предельно понятно.

1) Сайт русскоязычный, никто не будет лезть в переводчик чтобы перевести местечковый язык/диалект языка.

2) С учетом конфликта между бывшими республиками СССР, люди будут отстаивать свой сложившийся режим, какой бы он плохой не был.

3) Учитывая что твои собратья по несчастью любят людоедство и поливание желчью всех вокруг, то реакция с противоположной стороны соответственная.

Учитывая все-то что я перечислил, то наезды могут быть по любой из причин и не надо удивляться, так как это обратная реакция, а с учетом длительности конфликта и предыстории повстанцев Петлюры, а потом ОУН и УПА, то все то что мы имеем сейчас закономерно с развалом СССР.

Подводя итог, не провоцируй людей и не получишь грубой реакции в ответ (иногда это будет несправедливо).

Удачи тебе.

показать больше

3 3 Ответить



mashush

→ Maxim

19 дней назад

Я, конечно, не лингвист и сайт действительно русскоязычный, поэтому изначально им был написан текст на русском языке, но на украинской раскладке, о каком переводчике речь? Вы не понимаете о чем он написал? К чему Ваш огромный комментарий на тему текущих событий?

0 0 Ответить 

Загрузить ещё комментарии

Подписаться

О защите персональных данных

Не продавайте мои данные

