Eval: выполнение строки кода

Встроенная функция `eval` позволяет выполнять строку кода.

Синтаксис:

```
let result = eval(code);
```

Например:

```
let code = 'alert("Привет")';
eval(code); // Привет
```

Строка кода может быть большой, содержать переводы строк, объявления функций, переменные и т.п.

Результатом `eval` будет результат выполнения последней инструкции.

Например:

```
let value = eval('1+1');
alert(value); // 2
```

```
let value = eval('let i = 0; ++i');
alert(value); // 1
```

Код в `eval` выполняется в текущем лексическом окружении, поэтому ему доступны внешние переменные:

```
let a = 1;

function f() {
  let a = 2;

  eval('alert(a)'); // 2
}

f();
```

Значения внешних переменных можно изменять:

```
let x = 5;
eval("x = 10");
alert(x); // 10, значение изменено
```

В строгом режиме у `eval` имеется своё лексическое окружение. Поэтому функции и переменные, объявленные внутри `eval`, нельзя увидеть снаружи:

```
// напоминание: режим 'use strict' включён по умолчанию

eval("let x = 5; function f() {}");

alert(typeof x); // undefined (нет такой переменной)
// функция f тоже невидима
```

Без `use strict` у `eval` не будет отдельного лексического окружения, поэтому `x` и `f` будут видны из внешнего кода.

Использование «eval»

В современной разработке на JavaScript `eval` используется весьма редко. Есть даже известное выражение – «eval is evil» («eval – это зло»).

Причина такого отношения достаточно проста: давным-давно JavaScript был не очень развитым языком, и многие вещи можно было сделать только с помощью `eval`. Но та эпоха закончилась более десяти лет назад.

На данный момент нет никаких причин, чтобы продолжать использовать `eval`. Если кто-то всё ещё делает это, то очень вероятно, что они легко смогут заменить `eval` более современными конструкциями или JavaScript-модулями.

Пожалуйста, имейте в виду, что код в `eval` способен получать доступ к внешним переменным, и это может иметь побочные эффекты.

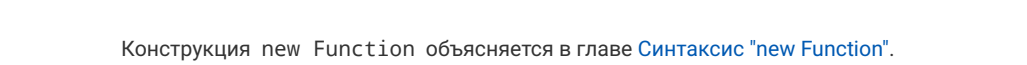
Минификаторы кода (инструменты, используемые для сжатия JS-кода перед тем, как отправить его конечным пользователям) заменяют локальные переменные на другие с более короткими именами для оптимизации. Обычно это безопасная манипуляция, но не тогда, когда в коде используется `eval`, так как код из `eval` может изменять значения локальных переменных. Поэтому минификаторы не трогают имена переменных, которые могут быть доступны из `eval`. Это ухудшает степень сжатия кода.

Использование внутри `eval` локальных переменных из внешнего кода считается плохим решением, так как это усложняет задачу по поддержке такого кода.

Существует два пути, как гарантированно избежать подобных проблем.

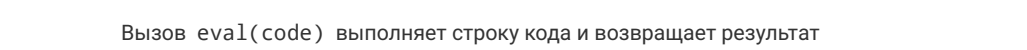
Если код внутри `eval` не использует внешние переменные, то вызывайте его так – `window.eval(...)`:

В этом случае код выполняется в глобальной области видимости:



```
let x = 1;
{
  let x = 5;
  window.eval('alert(x)'); // 1 (глобальная переменная)
}
```

Если коду внутри `eval` нужны локальные переменные, поменяйте `eval` на `new Function` и передавайте необходимые данные как аргументы:



```
let f = new Function('a', 'alert(a)');

f(5); // 5
```

Конструкция `new Function` объясняется в главе Синтаксис "new Function". Она создаёт функцию из строки в глобальной области видимости. Так что локальные переменные для неё невидимы, но всегда можно передать их как аргументы. Получается очень аккуратный код, как в примере выше.

Итого

Вызов `eval(code)` выполняет строку кода и возвращает результат последней инструкции.

- Это редко используется в современном JavaScript, так как в этом обычно нет необходимости.
- Возможен доступ к внешним локальным переменным. Это считается плохой практикой.
- Чтобы выполнить строку кода с помощью `eval` в глобальной области видимости, используйте `window.eval(code)`.
- Или же, если ваш код нуждается в каких-то данных из внешней области видимости, то используйте `new Function`, передав эти данные в качестве аргументов.

Редактировать на GitHub

Задачи

Eval-калькулятор

важность: 4

Создайте калькулятор, который запрашивает ввод какого-нибудь арифметического выражения и возвращает результат его вычисления.

В этой задаче нет необходимости проверять полученное выражение на корректность, просто вычислить и вернуть результат.

Запустить демо

решение

Комментарии

перед тем как писать...