8 августа 2023 г.

Устаревшее ключевое слово "var"

Эта статья предназначена для понимания старых скриптов

Информация, приведенная в этой статье, полезна для понимания старых скриптов.

Мы не пишем современный код таким образом.

В самой первой главе про переменные мы ознакомились с тремя способами объявления переменных:

1. `let`
2. `const`
3. `var`

`let` и `const` ведут себя одинаково по отношению к лексическому окружению, области видимости.

Но `var` – это совершенно другой зверь, берущий своё начало с давних времён. Обычно `var` не используется в современных скриптах, но всё ещё может скрываться в старых.

Если в данный момент вы не работаете с подобными скриптами, вы можете пропустить или отложить прочтение данной главы, однако, есть шанс, что вы столкнётесь с `var` в будущем.

На первый взгляд, поведение `var` похоже на `let`. Например, объявление переменной:

```
function sayHi() {
  var phrase = "Привет"; // локальная переменная, "var" вместо "let"

  alert(phrase); // Привет
}

sayHi();

alert(phrase); // Ошибка: phrase не определена
```

...Однако, отличия всё же есть.

Для «var» не существует блочной области видимости

Область видимости переменных `var` ограничивается либо функцией, либо, если переменная глобальная, то скриптом. Такие переменные доступны за пределами блока.

Например:

```
if (true) {
  var test = true; // используем var вместо let
}

alert(test); // true, переменная существует вне блока if
```

Так как `var` игнорирует блоки, мы получили глобальную переменную `test`.

А если бы мы использовали `let test` вместо `var test`, тогда переменная была бы видна только внутри `if`:

```
if (true) {
  let test = true; // используем let
}

alert(test); // Error: test is not defined
```

Аналогично для циклов: `var` не может быть блочной или локальной внутри цикла:

```
for (var i = 0; i < 10; i++) {
  // ...
}

alert(i); // 10, переменная i доступна вне цикла, т.к. является глобальной пер
```

Если блок кода находится внутри функции, то `var` становится локальной переменной в этой функции:

```
function sayHi() {
  if (true) {
    var phrase = "Привет";
  }

  alert(phrase); // срабатывает и выводит "Привет"
}

sayHi();
alert(phrase); // Ошибка: phrase не определена (видна в консоли разработчика)
```

Как мы видим, `var` выходит за пределы блоков `if`, `for` и подобных. Это происходит потому, что на заре развития JavaScript блоки кода не имели лексического окружения. Поэтому можно сказать, что `var` – это пережиток прошлого.

«var» допускает повторное объявление

Если в блоке кода дважды объявить одну и ту же переменную `let`, будет ошибка:

```
let user;

```

```
let user; // SyntaxError: 'user' has already been declared
```

Используя `var`, можно переобъявлять переменную сколько угодно раз. Повторные `var` игнорируются:

```
var user = "Pete";

var user; // ничего не делает, переменная объявлена раньше
// ...нет ошибки

alert(user); // Pete
```

Если дополнительно присвоить значение, то переменная примет новое значение:

```
var user = "Pete";

var user = "John";

alert(user); // John
```

«var» обрабатываются в начале запуска функции

Объявления переменных `var` обрабатываются в начале выполнения функции (или запуска скрипта, если переменная является глобальной).

Другими словами, переменные `var` считаются объявленными с самого начала исполнения функции вне зависимости от того, в каком месте функции реально находятся их объявления (при условии, что они не находятся во вложенной функции).

Т.е. этот код:

```
function sayHi() {
  phrase = "Привет";

  alert(phrase);

  var phrase;
}
sayHi();
```

...Технически полностью эквивалентен следующему (объявление переменной `var phrase` перемещено в начало функции):

```
function sayHi() {
  var phrase;

  phrase = "Привет";

  alert(phrase);

```

```
}
sayHi();
```

...И даже коду ниже (как вы помните, блочная область видимости игнорируется):

```
function sayHi() {
  phrase = "Привет"; // (*)

  if (false) {
    var phrase;
  }

  alert(phrase);
}
sayHi();
```

Это поведение называется «hoisting» (всплытие, поднятие), потому что все объявления переменных `var` «всплывают» в самый верх функции.

В примере выше `if (false)` условие никогда не выполнится. Но это никаким образом не препятствует созданию переменной `var phrase` , которая находится внутри него, поскольку объявления `var` «всплывают» в начало функции. Т.е. в момент присвоения значения `(*)` переменная уже существует.

Объявления переменных «всплывают», но присваивания значений – нет.

Это проще всего продемонстрировать на примере:

```
function sayHi() {
  alert(phrase);

  var phrase = "Привет";
}

sayHi();
```

Строка `var phrase = "Привет"` состоит из двух действий:

1. Объявление переменной `var`
2. Присвоение значения в переменную `=` .

Объявление переменной обрабатывается в начале выполнения функции («всплывает»), однако присвоение значения всегда происходит в той строке кода, где оно указано. Т.е. код выполняется по следующему сценарию:

```
function sayHi() {
  var phrase; // объявление переменной срабатывает вначале...

  alert(phrase); // undefined

  phrase = "Привет"; // ...присвоение - в момент, когда исполнится данная стро
}


```

```
sayHi();
```

Поскольку все объявления переменных `var` обрабатываются в начале функции, мы можем ссылаться на них в любом месте. Однако, переменные имеют значение `undefined` до строки с присвоением значения.

В обоих примерах выше вызов `alert` происходил без ошибки, потому что переменная `phrase` уже существовала. Но её значение ещё не было присвоено, поэтому мы получали `undefined`.

IIFE

В прошлом, поскольку существовал только `var`, а он не имел блочной области видимости, программисты придумали способ её эмулировать. Этот способ получил название «Immediately-invoked function expressions» (сокращенно IIFE).

Это не то, что мы должны использовать сегодня, но, так как вы можете встретить это в старых скриптах, полезно понимать принцип работы.

IIFE выглядит следующим образом:

```
(function() {

  var message = "Hello";

  alert(message); // Hello

})();
```

Здесь создаётся и немедленно вызывается Function Expression. Так что код выполняется сразу же и у него есть свои локальные переменные.

Function Expression обёрнуто в скобки `(function {...})`, потому что, когда JavaScript встречает `"function"` в основном потоке кода, он воспринимает это как начало Function Declaration. Но у Function Declaration должно быть имя, так что такой код вызовет ошибку:

```
// Пробуем объявить и сразу же вызвать функцию
function() { // <-- SyntaxError: Function statements require a function name

  var message = "Hello";

  alert(message); // Hello

}();
```

Даже если мы скажем: «хорошо, давайте добавим имя», – это не сработает, потому что JavaScript не позволяет вызывать Function Declaration немедленно.

```
// ошибка синтаксиса из-за скобок ниже
function go() {

}(); // <-- нельзя вызывать Function Declaration немедленно
```

Так что скобки вокруг функции – это трюк, который позволяет объяснить JavaScript, что функция была создана в контексте другого выражения, а значит, что это Function Expression: ей не нужно имя и её можно вызвать немедленно.

Помимо круглых скобок существуют и другие способы сообщить JavaScript, что мы имеем в виду Function Expression:

```
// Способы создания IIFE

(function() {
  alert("Круглые скобки вокруг функции");
})();

(function() {
  alert("Круглые скобки вокруг всего выражения");
}());

!function() {
  alert("Выражение начинается с логического оператора НЕ");
}();

+function() {
  alert("Выражение начинается с унарного плюса");
}();
```

Во всех перечисленных случаях мы объявляем Function Expression и немедленно запускаем его. Ещё раз отметим: в настоящее время необходимости писать подобный код нет.

Итого

Существует 2 основных отличия `var` от `let/const`:

1. Переменные `var` не имеют блочной области видимости, они ограничены, как минимум, телом функции.
2. Объявления (инициализация) переменных `var` производится в начале исполнения функции (или скрипта для глобальных переменных).

Есть ещё одно небольшое отличие, относящееся к глобальному объекту, мы рассмотрим его в следующей главе.

Эти особенности, как правило, не очень хорошо влияют на код. Блочная область видимости – это удобно. Поэтому много лет назад `let` и `const` были введены в стандарт и сейчас являются основным способом объявления переменных.

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.

157 Комментариев

Войти

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ ИЛИ ЧЕРЕЗ DISQUS ?

Имя

18 Поделиться

Лучшие Новые Старые

Джун на фронте

месяц назад

Уже владеете JavaScript? Что ж, куда двигаться дальше?

Карьерный путь — продолжение для веб-разработчика. Я делюсь личным опытом в IT: от старта с нуля до момента, когда я получил свой первый job offer, за эти 700 дней полны открытий и практического опыта!

У меня был шанс поработать верстальщиком, был редактором в издании, конкурирующем с Хабром, а теперь я нахожусь в активном поиске позиции фронтенд-разработчика.

Подписывайтесь на мой **телеграм-канал «Джун на фронте»**, где мы вместе будем откликаться на вакансии, решать тестовые задания и нацелимся на поиск работы мечты!

0 0 Ответить • Поделиться ›

frontforme

2 месяца назад

Привет, хочешь стать веб-разработчиком и устроиться на работу?

Опытный frontend-разработчик из крупной компании рассказывает как попасть в frontend-IT самым коротким путем в бесплатном блоге.

Давай к нам! ТГ: @davay_v_frontend

0 0 Ответить • Поделиться ›

Джун в Телеграме — @divatoz

3 месяца назад

Мечтаешь о карьере веб-разработчика, но не знаешь что изучать?

Тогда подписывайся на мой **ТГ Джун на фронте**, где я делюсь своим опытом от новичка до 655 дня в IT. В моем блоге ты узнаешь:

— Как эффективно и интересно изучать веб-разработку 🚀

— Обзоры интересных книг и курсов 📚

— Демонстрацию моих проектов и кода 💻

Ответы на вопросы подписчиков и обратную связь 💬

Мой блог — это пространство для обмена знаниями и вдохновением между новичками и опытными разработчиками **БЕЗ РЕКЛАМЫ И ПЛАТНЫХ КУРСОВ** для новичков и профи!

0 1 Ответить • Поделиться ›

Рахим
4 месяца назад

A var в jquery работает как в джс?

1 0 Ответить • Поделиться ›

Lust1k ↗ Рахим
4 месяца назад

да

1 0 Ответить • Поделиться ›

Svyatoslav Olshanechky
5 месяцев назад

Не так давно смотрел работу одной исследовательской группы, которые заморачивались над performance. Так вот, эти ребята многие переменные определяли как var, так как если использовать const везде без разбора, то происходит проверка на уникальность и все это как оказалось жрет ресурсы. Поэтому считаю, что статья полезна, а все новые чудо учителя с своими курсами утверждающими, что нужно забыть про var не правы.

4 1 Ответить • Поделиться ›

Maxim ↗ Svyatoslav Olshanechky
4 месяца назад edited

В основном var используют в производительном коде, например mobx (скачал себе с CDN последний релиз). И там я хочу сказать все написано в устаревшем стиле со всеми var и функциональном стиле без spread, rest и прочего. Понятно что такой подход завязан на совместимости, но параллельно это также преследует цели быстродействия, так как цель библиотеки: "оперативная работа с реактивными данными". И вот тут использование var оправдано на все 1000%. Конечно человек или компания, которая будет использовать данную библиотеку применит классический подход со всеми новомодными ключевыми словами, чем и ухудшит производительность, но не значительно.

Я вообще к чему это? Я к тому что для приложения лучше держаться классического подхода с современным веянием стиля кода, а вот там где создается серьезный технический бойлерплейт то там лучше выбрать быстродействие несмотря на все тенденции программирования.

Но по итогу выбор всегда за вами.

1 0 Ответить • Поделиться ›

Artur Oganesian ↗ Svyatoslav Olshanechky
4 месяца назад

друг вар не нужен нигде никак потому что она является ошибкой сам по себе

0 2 Ответить • Поделиться ›

Станислав Я ↗ Svyatoslav Olshanechky

4 месяца назад

Основное слово "заморачивались". Чтобы заморочиться и ничего не испортить нужен опыт. И если новичкам говорить "Используйте var вместо let или const", то ничего хорошего из этого не выйдет. Потенциальный ущерб от "побочных действий" var страшнее, чем потери производительности, которые в обычных условиях можно и не заметить. А вот, например, всплытие переменной и её случайное переопределение может испортить жизнь в большем количестве проектов. Если у вас есть измеренные проблемы с производительностью, и var тут может помочь - это одно, но в общем случае "забыть про var" мне кажется хорошей идеей.

3 2 Ответить • Поделиться ›

С **Сергей Блиничкин**
8 месяцев назад edited

не пойму, для чего эта статья? Если не показано принципиальное отличие var и let. Где механизмы? как они работают с памятью? Почему var всплывает? Как let не выходит из области видимости? Что не даёт всплывать? Единственное, что понятно: есть [[Environment]] и let с ним не плохо работает - всё! яЯ чуть не забыл прошлую тему...

0 0 Ответить • Поделиться ›

A **Artur Oganesian** → Сергей Блиничкин
4 месяца назад

про hoisting читай узнаешь что как

0 0 Ответить • Поделиться ›

V **Veh Gerli** → Сергей Блиничкин
8 месяцев назад

Это учебник для чайников, а не для самоваров.

11 0 Ответить • Поделиться ›

MB Этот комментарий ожидает проверки. Показать комментарий.

T **Troll Troll** → Михаил Виноградов
9 месяцев назад

Кринж🤡🤡🤡

2 0 Ответить • Поделиться ›

З **Запрещаю писать даты**
год назад

Весь бы яваскрипт был такой понятный, как тема с var

9 1 Ответить • Поделиться ›

A **Artur Oganesian** → Запрещаю писать даты
4 месяца назад

новый язык яваскрипт :D :D :D

2 0 Ответить • Поделиться ›

T **Troll Troll** Запрещаю писать даты
9 месяцев назад

Все становится понятно при 3-5 перечитывании этого учебника. В первый раз, с полного 0, всем непонятно.

5 0 Ответить • Поделиться ›

P **poniapolla** Запрещаю писать даты
год назад

ахах, улыбнуло :)

2 1 Ответить • Поделиться ›

C **Северная жиза** Запрещаю писать даты
год назад

Аахахахаха, согласен, особенно после замыкания так приятно var изучать)

19 0 Ответить • Поделиться ›

B **Вася Беднов** Северная жиза
год назад

И после рекурсии)

8 1 Ответить • Поделиться ›

Загрузить ещё комментарии