Устаревшее ключевое слово "var"

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"

  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 i
```

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.
```

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
```

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» 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 внач

  alert(phrase); // undefined

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

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`.

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.

Комментарии

перед тем как писать...