

Раздел

[Продвинутая работа с функциями](#)


Навигация по уроку

Использование для полифилов

Итого

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)
[→ Продвинутая работа с функциями](#) 28-го августа 2019

Глобальный объект

Глобальный объект предоставляет переменные и функции, доступные в любом месте программы. По умолчанию это те, что встроены в язык или среду исполнения.

В браузере он называется `window`, в Node.js — `global`, в другой среде исполнения может называться иначе.

Недавно `globalThis` был добавлен в язык как стандартизированное имя для глобального объекта, которое должно поддерживаться в любом окружении. В некоторых браузерах, например Edge не на Chromium, `globalThis` ещё не поддерживается, но легко реализуется с помощью полифила.

Далее мы будем использовать `window`, полагая, что наша среда — браузер. Если скрипт может выполняться и в другом окружении, лучше будет `globalThis`.

Ко всем свойствам глобального объекта можно обращаться напрямую:

```
1 alert("Привет");
2 // это то же самое, что и
3 window.alert("Привет");
```

В браузере глобальные функции и переменные, объявленные с помощью `var` (не `let/const`!), становятся свойствами глобального объекта:

```
1 var gVar = 5;
2
3 alert(window.gVar); // 5 (становится свойством глобального объекта)
```

Пожалуйста, не полагайтесь на это. Такое поведение поддерживается для совместимости. В современных проектах, использующих [JavaScript-модули](#), такого не происходит.

Если бы мы объявили переменную при помощи `let`, то такого бы не произошло:

```
1 let gLet = 5;
2
3 alert(window.gLet); // undefined (не становится свойством глобального объекта)
```

Если свойство настолько важное, что вы хотите сделать его доступным для всей программы, запишите его в глобальный объект напрямую:

```
1 // сделать информацию о текущем пользователе глобальной
2 window.currentUser = {
3   name: "John"
4 };
5
6 // где угодно в коде
7 alert(currentUser.name); // John
8
9 // или, если у нас есть локальная переменная с именем "
10 // получим её из window явно (безопасно!)
11 alert(window.currentUser.name); // John
```

Раздел

[Продвинутая работа с функциями](#)

Навигация по уроку

Использование для полифилов

Итого

Комментарии

Поделиться



[Редактировать на GitHub](#)



При этом обычно не рекомендуется использовать глобальные переменные. Следует применять их как можно реже. Дизайн кода, при котором функция получает входные параметры и выдаёт определённый результат, чище, надёжнее и удобнее для тестирования, чем когда используются внешние, а тем более глобальные переменные.

Использование для полифилов

Глобальный объект можно использовать, чтобы проверить поддержку современных возможностей языка.

Например, проверить наличие встроенного объекта Promise (такая поддержка отсутствует в очень старых браузерах):

```
1 if (!window.Promise) {
2   alert("Ваш браузер очень старый!");
3 }
```



Если нет (скажем, используется старый браузер), мы можем создать полифил: добавить функции, которые не поддерживаются окружением, но существуют в современном стандарте.

```
1 if (!window.Promise) {
2   window.Promise = ... // собственная реализация современ
3 }
```



Итого

- Глобальный объект хранит переменные, которые должны быть доступны в любом месте программы.

Это включает в себя как встроенные объекты, например, `Array`, так и характерные для окружения свойства, например, `window.innerHeight` – высота окна браузера.

- Глобальный объект имеет универсальное имя – `globalThis`.

...Но чаще на него ссылаются по-старому, используя имя, характерное для данного окружения, такое как `window` (браузер) и `global` (Node.js). Так как `globalThis` появился недавно, он не поддерживается в IE и Edge (не-Chromium версия), но можно использовать полифил.

- Следует хранить значения в глобальном объекте, только если они действительно глобальны для нашего проекта. И стараться свести их количество к минимуму.
- В браузерах, если только мы не используем [модули](#), глобальные функции и переменные, объявленные с помощью `var`, становятся свойствами глобального объекта.
- Для того, чтобы код был проще и в будущем его легче было поддерживать, следует обращаться к свойствам глобального объекта напрямую, как `window.x`.

Проводим [курсы по JavaScript и фреймворкам](#).



Комментарии

перед тем как писать...