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. Он поддерживается во всех основных браузерах, и в ноде.

Ко всем свойствам глобального объекта можно обращаться напрямую:

alert("Привет");

// это то же самое, что и

window.alert("Привет");

В браузере глобальные функции и переменные, объявленные с помощью var (не let/const!), становятся свойствами глобального объекта (В Node.js это не так.):

var gVar = 5;

alert(window.gVar); // 5 (становится свойством глобального объекта)

То же самое касается функций, объявленных с помощью синтаксиса Function Declaration (выражения с ключевым словом function в основном потоке кода, не Function Expression)

Пожалуйста, не полагайтесь на это. Такое поведение поддерживается для совместимости. В современных проектах, использующих [JavaScript-модули](https://learn.javascript.ru/modules), такого не происходит.

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

let gLet = 5;

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:

// сделать информацию о текущем пользователе глобальной, для предоставления доступа всем скриптам

window.currentUser = {

name: "John"

};

// где угодно в коде

alert(currentUser.name); // John

// или, если у нас есть локальная переменная с именем "currentUser",

// получим её из window явно (безопасно!)

alert(window.currentUser.name); // John

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.

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

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

if (!window.Promise) {

alert("Ваш браузер очень старый!");

}

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

if (!window.Promise) {

window.Promise = ... // собственная реализация современной возможности языка

}