29 октября 2022 г.

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. Он поддерживается во всех основных браузерах.

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)
```

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, 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:

```
// сделать информацию о текущем пользователе глобальной, для предоставления до
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.

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):

```
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
}
```

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).

- 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` .

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.

328 Комментариев

1 Войти

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ ИЛИ ЧЕРЕЗ DISQUS ?

Имя

62 Поделиться

Лучшие Новые Старые

Юрий

14 дней назад

⚡ **Хочешь понять зачем нужен глобальный объект?**
Тогда гоу в мой блог ТГ-блог «Джун на фронте»!

👨‍💻 Автор - системный администратор, который с декабря 2021 года освоил HTML, CSS, JS, Vue, Nuxt, React Native, MongoDB и Node.js.

Изучи мой путь в мире разработки: от новичка до создателя 🤖 **Телеграм-бота для автоматической отправки** откликов!

💡 Тут ты найдешь полезные ответы на насущные вопросы, анализ рынка вакансий и советы от человека, прошедшего этот путь.

Вбивайте «**Джун на фронте**» и присоединяйтесь.

0 2 Ответить

ДН

Этот комментарий ожидает проверки. Показать комментарий.

А

Андриян Соколенко → Джун на фронте

год назад

Бросай это дело, грузчики стране нужнее! Там всё получится!

3 0 Ответить

Джун на фронте → Андриян Соколенко

год назад

По себе людей не судят

0 12 Ответить

D

devmen → Джун на фронте

год назад

удались, пожалуйста, ты только негатив тут создаешь
Программирование это не твое

4 3 Ответить

4 0 Ответить

Андриян Соколенко → Джун на фронте
год назад

Я по тебе сужу

3 0 Ответить

Flyress
2 года назад

Тут сказано: "То же самое касается функций, объявленных с помощью синтаксиса Function Declaration", то есть Function Delcaration на уровне global lexical environment попадают также в глобальный обьект. Но этот код вызывает ошибку, и не видит функцию в обьекте (видит undefined).

```
/* script.js */

function globalFunc(a) {
console.log(a + 100);
var c = 6;
}

console.log(globalThis.globalFunc(-1)); // тут ошибка TypeError, 'is not a function'
console.log(globalThis.globalFunc); // следовательно, тут undefined? почему?
```

0 0 Ответить

Louis - Nicola d'Avout → Flyress
год назад edited

Ваша функция ничего не возвращает, поэтому если записать в console.log() она вернёт undefined.
вместо console.log(a + 100) напишите:
return a + 100 И функция заработает

0 0 Ответить

Сергей М. → Flyress
год назад

Нет там ошибки. Первый раз дает 99 из console.log(a + 100);, потом твоя функция не возвращает значения из console.log(globalThis.globalFunc(-1)); и дальше собственно сама функция из console.log(globalThis.globalFunc);.
Бессмысленно много выдачи в консоль. Если в твоей функции прописать, чтобы она возвращала значение return a + 100;, то будет нормально работать

4 0 Ответить

Andrei Khotko
2 года назад

Из текущей статьи:

> 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.

Тогда такой вопрос в теме "Замыкания" так ли так, плохо и неправо, использовать внешние

Тогда такой вопрос: в теме "Замыкания" только так, налево и направо, использовались внешние переменные в замыкающих функциях. Так стоит ли писать такие функции, использующие переменные из внешнего лексического окружения, или стоит писать "чистые функции", принимающие все необходимые значения извне?

1 1 Ответить

Дима Andrei Khotko
2 месяца назад

внешние переменные != глобальные переменные

0 0 Ответить

Michael Shomoroff Andrei Khotko
год назад edited

Да простят меня джаваскриптоносцы за такой питонообразный код.
В замыкания можно использовать для написания функций обёрток для других функций, которые выполняются вместо них, делают что-то а потом возвращают оборачиваемую функцию.

```
function counter(func) {
let counter = 1;
function wrapper(...args) {
console.log(`вызвана ${counter++} раз`);
return func(...args);
}
return wrapper;
}

let say_hello = function () {
console.log("Hello!");
}

say_hello = counter(say_hello);
```

показать больше

2 0 Ответить

Anna Andrei Khotko
2 года назад

так там суть во вложенности функций и основная функция часто ничего просто не делает
так что не думаю что к этому относится данный коммент

0 0 Ответить

Вадим Andrei Khotko
2 года назад

Ну, в главе "Замыкания" объясняется внутренняя структура и как работают замыкания. Автор дает понять, что эта возможность используется очень часто, но никаких советов по поводу стоит или не стоит, там нет.

2 1 Ответить

Viktor

V 2 года назад

03.08.2022 (()

0 3 Ответить

M **Mr.BigMac** → Viktor
2 года назад

А ты решаешь задачи , практикуешься на других ресурсах или чисто материал читаешь?

Не знаю как у кого но я могу и по несколько часов над одной дупля гонять

2 0 Ответить

V **Viktor** → Mr.BigMac
2 года назад

Ну пока материал читаю, ещё пытаюсь задачи решать на других сайтах, и в Ютубе смотрю видео, пока что туго даётся с задачами.

0 1 Ответить

З **Запрещаю писать даты** → Viktor
2 года назад

Не пиши даты, ты все страницы засрал. мало того не понимаешь что читаешь, так еще и флудишь

5 1 Ответить

V **Viktor** → Запрещаю писать даты
2 года назад

Прошу прощения ваше величество,что мешаю вам читать очень нужные комментарии,и простите нас всех на этом сайте кто пишет даты

5 3 Ответить

T **Tiny** → Viktor
2 года назад

Не прощу!!

0 1 Ответить

Загрузить ещё комментарии