

Раздел

[Свойства объекта, их конфигурация](#)

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство


Метод Object.defineProperties

Object.getOwnPropertyDescriptors

Глобальное запечатывание объекта

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Язык программирования JavaScript](#)  
[→ Свойства объекта, их конфигурация](#) 14-го сентября 2020

## Флаги и дескрипторы свойств

Как мы знаем, объекты могут содержать свойства.

До этого момента мы рассматривали свойство только как пару «ключ-значение». Но на самом деле свойство объекта гораздо мощнее и гибче.

В этой главе мы изучим дополнительные флаги конфигурации для свойств, а в следующей – увидим, как можно незаметно превратить их в специальные функции – геттеры и сеттеры.

### Флаги свойств

Помимо значения **value**, свойства объекта имеют три специальных атрибута (так называемые «флаги»).

- **writable** – если `true`, свойство можно изменить, иначе оно только для чтения.
- **enumerable** – если `true`, свойство перечисляется в циклах, в противном случае циклы его игнорируют.
- **configurable** – если `true`, свойство можно удалить, а эти атрибуты можно изменять, иначе этого делать нельзя.

Мы ещё не встречали эти атрибуты, потому что обычно они скрыты. Когда мы создаём свойство «обычным способом», все они имеют значение `true`. Но мы можем изменить их в любое время.

Сначала посмотрим, как получить их текущие значения.



Метод [Object.getOwnPropertyDescriptor](#) позволяет получить полную информацию о свойстве.



Его синтаксис:

```
1 let descriptor = Object.getOwnPropertyDescriptor(obj, p
```

**obj**

Объект, из которого мы получаем информацию.

**propertyName**

Имя свойства.

Возвращаемое значение – это объект, так называемый «дескриптор свойства»: он содержит значение свойства и все его флаги.

Например:

```
1 let user = {  
2   name: "John"  
3 };  
4  
5 let descriptor = Object.getOwnPropertyDescriptor(user,  
6  
7 alert( JSON.stringify(descriptor, null, 2) );  
8 /* дескриптор свойства:  
9 {  
10  "value": "John",  
11  "writable": true,  
12  "enumerable": true,  
13  "configurable": true  
14 }  
15 */
```



Раздел

Свойства объекта, их конфигурация

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство

Метод Object.defineProperty

Object.getOwnPropertyDescriptors

Глобальное запечатывание объекта

Комментарии

Поделиться



Редактировать на GitHub



Чтобы изменить флаги, мы можем использовать метод `Object.defineProperty`.

Его синтаксис:

```
1 Object.defineProperty(obj, propertyName, descriptor)
```

**obj, propertyName**

Объект и его свойство, для которого нужно применить дескриптор.

**descriptor**

Применяемый дескриптор.

Если свойство существует, `defineProperty` обновит его флаги. В противном случае метод создаёт новое свойство с указанным значением и флагами; если какой-либо флаг не указан явно, ему присваивается значение `false`.

Например, здесь создаётся свойство `name`, все флаги которого имеют значение `false`:

```
1 let user = {};  
2  
3 Object.defineProperty(user, "name", {  
4   value: "John"  
5 });  
6  
7 let descriptor = Object.getOwnPropertyDescriptor(user,  
8  
9 alert( JSON.stringify(descriptor, null, 2) );  
10 /*  
11 {  
12   "value": "John",  
13   "writable": false,  
14   "enumerable": false,  
15   "configurable": false  
16 }  
17 */
```

Сравните это с предыдущим примером, в котором мы создали свойство `user.name` «обычным способом»: в этот раз все флаги имеют значение `false`. Если это не то, что нам нужно, надо присвоить им значения `true` в параметре `descriptor`.

Теперь давайте рассмотрим на примерах, что нам даёт использование флагов.

## Только для чтения

Сделаем свойство `user.name` доступным только для чтения. Для этого изменим флаг `writable`:

```
1 let user = {  
2   name: "John"  
3 };  
4  
5 Object.defineProperty(user, "name", {  
6   writable: false  
7 });  
8  
9 user.name = "Pete"; // Ошибка: Невозможно изменить дост
```

Теперь никто не сможет изменить имя пользователя, если только не обновит соответствующий флаг новым вызовом `defineProperty`.

Раздел

Свойства объекта, их конфигурация

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство

Метод Object.defineProperties

Object.getOwnPropertyDescriptors

Глобальное запечатывание объекта

Комментарии

Поделиться



Редактировать на GitHub



### Ошибки появляются только в строгом режиме

В нестрогом режиме, без `use strict`, мы не увидим никаких ошибок при записи в свойства «только для чтения» и т.п. Но эти операции всё равно не будут выполнены успешно. Действия, нарушающие ограничения флагов, в нестрогом режиме просто молча игнорируются.

Вот тот же пример, но свойство создано «с нуля»:

```
1 let user = { };
2
3 Object.defineProperty(user, "name", {
4   value: "John",
5   // для нового свойства необходимо явно указывать все
6   enumerable: true,
7   configurable: true
8 });
9
10 alert(user.name); // John
11 user.name = "Pete"; // Ошибка
```

## Неперечислимое свойство

Теперь добавим собственный метод `toString` к объекту `user`.

Встроенный метод `toString` в объектах – неперечислимый, его не видно в цикле `for...in`. Но если мы напишем свой собственный метод `toString`, цикл `for...in` будет выводить его по умолчанию:

```
1 let user = {
2   name: "John",
3   toString() {
4     return this.name;
5   }
6 };
7
8 // По умолчанию оба свойства выведутся:
9 for (let key in user) alert(key); // name, toString
```

Если мы этого не хотим, можно установить для свойства `enumerable: false`. Тогда оно перестанет появляться в цикле `for...in` аналогично встроенному `toString`:

```
1 let user = {
2   name: "John",
3   toString() {
4     return this.name;
5   }
6 };
7
8 Object.defineProperty(user, "toString", {
9   enumerable: false
10 });
11
12 // Теперь наше свойство toString пропало из цикла:
13 for (let key in user) alert(key); // name
```

Неперечислимые свойства также не возвращаются `Object.keys`:

```
1 alert(Object.keys(user)); // name
```

## Неконфигурируемое свойство

Раздел

Свойства объекта, их конфигурация

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство

Метод Object.defineProperty

Object.getOwnPropertyDescriptors

Глобальное запечатывание объекта

Комментарии

Поделиться



Редактировать на GitHub



Флаг неконфигурируемого свойства (configurable: false) иногда предустановлен для некоторых встроенных объектов и свойств.

Неконфигурируемое свойство не может быть удалено.

Например, свойство Math.PI – только для чтения, неперечислимое и неконфигурируемое:

```
1 let descriptor = Object.getOwnPropertyDescriptor(Math,
2
3 alert( JSON.stringify(descriptor, null, 2) );
4 /*
5 {
6   "value": 3.141592653589793,
7   "writable": false,
8   "enumerable": false,
9   "configurable": false
10 }
11 */
```

То есть программист не сможет изменить значение Math.PI или перезаписать его.

```
1 Math.PI = 3; // Ошибка
2
3 // delete Math.PI тоже не работает
```

Определение свойства как неконфигурируемого – это дорога в один конец. Мы не сможем отменить это действие, потому что defineProperty не работает с неконфигурируемыми свойствами.

В коде ниже мы делаем user.name «навечно запечатанной» константой:

```
1 let user = { };
2
3 Object.defineProperty(user, "name", {
4   value: "John",
5   writable: false,
6   configurable: false
7 });
8
9 // теперь невозможно изменить user.name или его флаги
10 // всё это не будет работать:
11 //   user.name = "Pete"
12 //   delete user.name
13 //   defineProperty(user, "name", ...)
14 Object.defineProperty(user, "name", {writable: true});
```

#### Ошибки отображаются только в строгом режиме

В нестрогом режиме мы не увидим никаких ошибок при записи в свойства «только для чтения» и т.п. Эти операции всё равно не будут выполнены успешно. Действия, нарушающие ограничения флагов, в нестрогом режиме просто молча игнорируются.

## Метод Object.defineProperty

Существует метод `Object.defineProperty(obj, descriptors)`, который позволяет определять множество свойств сразу.

Его синтаксис:

```
1 Object.defineProperty(obj, {
2   prop1: descriptor1,
3   prop2: descriptor2
```

Раздел

[Свойства объекта, их конфигурация](#)

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство

Метод `Object.defineProperty`

`Object.getOwnPropertyDescriptors`

Глобальное запечатывание объекта

Комментарии

Поделиться



[Редактировать на GitHub](#)

```
4    // ...
5  });
```



Например:



```
1  Object.defineProperty(user, {
2    name: { value: "John", writable: false },
3    surname: { value: "Smith", writable: false },
4    // ...
5  });
```

Таким образом, мы можем определить множество свойств одной операцией.

## Object.getOwnPropertyDescriptors

Чтобы получить все дескрипторы свойств сразу, можно воспользоваться методом `Object.getOwnPropertyDescriptors(obj)`.

Вместе с `Object.defineProperty` этот метод можно использовать для клонирования объекта вместе с его флагами:

```
1  let clone = Object.defineProperty({}, Object.getOwnPr
```

Обычно при клонировании объекта мы используем присваивание, чтобы скопировать его свойства:

```
1  for (let key in user) {
2    clone[key] = user[key]
3  }
```



...Но это не копирует флаги. Так что если нам нужен клон «получше», предпочтительнее использовать `Object.defineProperty`.

Другое отличие в том, что `for...in` игнорирует символьные свойства, а `Object.getOwnPropertyDescriptors` возвращает дескрипторы *всех* свойств, включая свойства-символы.



## Глобальное запечатывание объекта

Дескрипторы свойств работают на уровне конкретных свойств.

Но ещё есть методы, которые ограничивают доступ ко *всему* объекту:

### Object.preventExtensions(obj)

Запрещает добавлять новые свойства в объект.

### Object.seal(obj)

Запрещает добавлять/удалять свойства. Устанавливает `configurable: false` для всех существующих свойств.

### Object.freeze(obj)

Запрещает добавлять/удалять/изменять свойства. Устанавливает `configurable: false`, `writable: false` для всех существующих свойств.

А также есть методы для их проверки:

### Object.isExtensible(obj)

Возвращает `false`, если добавление свойств запрещено, иначе `true`.

### Object.isSealed(obj)

Возвращает `true`, если добавление/удаление свойств запрещено и для всех существующих свойств установлено `configurable: false`.

Раздел

[Свойства объекта, их конфигурация](#)

Навигация по уроку

Флаги свойств

Только для чтения

Неперечислимое свойство

Неконфигурируемое свойство

Метод `Object.defineProperties`

`Object.getOwnPropertyDescriptors`

Глобальное запечатывание объекта

Комментарии

Поделиться



Редактировать на GitHub



## Object.isFrozen(obj)

Возвращает `true`, если добавление/удаление/изменение свойств запрещено, и для всех текущих свойств установлено `configurable: false, writable: false`.

На практике эти методы используются редко.

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



## Комментарии

перед тем как писать...

© 2007–2020 Илья Кантор | [о проекте](#) | [связаться с нами](#) | [пользовательское соглашение](#) | [политика конфи](#)

