Флаги и дескрипторы 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.

true - везде по умолчанию.

Метод **Object.getOwnPropertyDescriptor** позволяет получить полную информацию о свойстве. Его синтаксис:

```
let descriptor = Object.getOwnPropertyDescriptor(obj, propertyName);
```

где 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:

```
let user = {
  name: "John"
}
let descriptor = Object.getOwnPropertyDescriptor(user, 'name');
alert( JSON.stringify(descriptor, null, 2 ) );
/* дескриптор свойства:
{
  "value": "John",
  "writable": true,
  "enumerable": true,
  "configurable": true
}
*/
```

Чтобы изменить флаги, мы можем использовать метод **Object.defineProperty**. Его синтаксис:

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

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**.

```
let user = {};
Object.defineProperty(user, "name", {
  value: "John"
});
let descriptor = Object.getOwnPropertyDescriptor(user, 'name');
alert( JSON.stringify(descriptor, null, 2 ) );
/*
{
  "value": "John",
  "writable": false,
  "enumerable": false,
  "configurable": false
}
 */
```

Если у свойства **configurable: false,** то с этим свойством Object.defineProperty уже не работает, отменить этот флаг мы уже не сможем.

Существует метод **Object.defineProperties**(obj, descriptors), который позволяет определять множество свойств сразу. Его синтаксис:

```
Object.defineProperties(obj, {
  prop1: descriptor1,
  prop2: descriptor2
  // ...
});
```

Например:

```
Object.defineProperties(user, {
```

```
  name: { value: "John", writable: false },
  surname: { value: "Smith", writable: false },
  // ...
});
```

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

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

```
let clone = Object.defineProperties({}, Object.getOwnPropertyDescriptors(obj));
```

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: На практике эти методы используются редко.

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.

Object.isFrozen(obj) - Возвращает true, если добавление/удаление/изменение свойств запрещено, и для всех текущих свойств установлено configurable: false, writable: false.