18 августа 2022 г.

Свойства - геттеры и сеттеры

Есть два типа свойств объекта.

Первый тип это *свойства-данные (data properties)*. Мы уже знаем, как работать с ними. Все свойства, которые мы использовали до текущего момента, были свойствами-данными.

Второй тип свойств мы ещё не рассматривали. Это *свойства-аксессоры (accessor properties)*. По своей сути это функции, которые используются для присвоения и получения значения, но во внешнем коде они выглядят как обычные свойства объекта.

Геттеры и сеттеры

Свойства-аксессоры представлены методами: «геттер» – для чтения и «сеттер» – для записи. При литеральном объявлении объекта они обозначаются `get` и `set`:

```
let obj = {
  get propName() {
    // геттер, срабатывает при чтении obj.propName
  },

  set propName(value) {
    // сеттер, срабатывает при записи obj.propName = value
  }
};
```

Геттер срабатывает, когда `obj.propName` читается, сеттер – когда значение присваивается.

Например, у нас есть объект `user` со свойствами `name` и `surname`:

```
let user = {
  name: "John",
  surname: "Smith"
};
```

А теперь добавим свойство объекта `fullName` для полного имени, которое в нашем случае `"John Smith"`. Само собой, мы не хотим дублировать уже имеющуюся информацию, так что реализуем его при помощи аксессора:

```
let user = {
  name: "John",
  surname: "Smith",

```

```
  get fullName() {
    return `${this.name} ${this.surname}`;
  }
};

alert(user.fullName); // John Smith
```

Снаружи свойство-аксессор выглядит как обычное свойство. В этом и заключается смысл свойств-аксессоров. Мы не *вызываем* `user.fullName` как функцию, а *читаем* как обычное свойство: геттер выполнит всю работу за кулисами.

На данный момент у `fullName` есть только геттер. Если мы попытаемся назначить `user.fullName=`, произойдёт ошибка:

```
let user = {
  get fullName() {
    return `...`;
  }
};

user.fullName = "Тест"; // Ошибка (у свойства есть только геттер)
```

Давайте исправим это, добавив сеттер для `user.fullName`:

```
let user = {
  name: "John",
  surname: "Smith",

  get fullName() {
    return `${this.name} ${this.surname}`;
  },

  set fullName(value) {
    [this.name, this.surname] = value.split(" ");
  }
};

// set fullName запустится с данным значением
user.fullName = "Alice Cooper";

alert(user.name); // Alice
alert(user.surname); // Cooper
```

В итоге мы получили «виртуальное» свойство `fullName`. Его можно прочитать и изменить.

Дескрипторы свойств доступа

Дескрипторы свойств-аксессоров отличаются от «обычных» свойств-данных.

Свойства-аксессоры не имеют `value` и `writable`, но взамен предлагают функции `get` и `set`.

То есть, дескриптор аксессора может иметь:

- `get` – функция без аргументов, которая сработает при чтении свойства,
- `set` – функция, принимающая один аргумент, вызываемая при присвоении свойства,
- `enumerable` – то же самое, что и для свойств-данных,
- `configurable` – то же самое, что и для свойств-данных.

Например, для создания аксессора `fullName` при помощи `defineProperty` мы можем передать дескриптор с использованием `get` и `set`:

```
let user = {
  name: "John",
  surname: "Smith"
};

Object.defineProperty(user, 'fullName', {
  get() {
    return `${this.name} ${this.surname}`;
  },

  set(value) {
    [this.name, this.surname] = value.split(" ");
  }
});

alert(user.fullName); // John Smith

for(let key in user) alert(key); // name, surname
```

Ещё раз заметим, что свойство объекта может быть либо свойством-аксессором (с методами `get/set`), либо свойством-данным (со значением `value`).

При попытке указать и `get`, и `value` в одном дескрипторе будет ошибка:

```
// Error: Invalid property descriptor.
Object.defineProperty({}, 'prop', {
  get() {
    return 1
  },

  value: 2
});
```

Умные геттеры/сеттеры

Геттеры/сеттеры можно использовать как обёртки над «реальными» значениями свойств, чтобы получить больше контроля над операциями с ними.

Например, если мы хотим запретить устанавливать короткое имя для `user`, мы можем использовать сеттер `name` для проверки, а само значение хранить в отдельном свойстве `_name`:

```
let user = {
  get name() {
```

```
    return this._name;
  },

  set name(value) {
    if (value.length < 4) {
      alert("Имя слишком короткое, должно быть более 4 символов");
      return;
    }
    this._name = value;
  }
};

user.name = "Pete";
alert(user.name); // Pete

user.name = ""; // Имя слишком короткое...
```

Таким образом, само имя хранится в `_name`, доступ к которому производится через геттер и сеттер.

Технически, внешний код всё ещё может получить доступ к имени напрямую с помощью `user._name`, но существует широко известное соглашение о том, что свойства, которые начинаются с символа `"_"`, являются внутренними, и к ним не следует обращаться из-за пределов объекта.

Использование для совместимости

У аксессоров есть интересная область применения – они позволяют в любой момент взять «обычное» свойство и изменить его поведение, поменяв на геттер и сеттер.

Например, представим, что мы начали реализовывать объект `user`, используя свойства-данные имя `name` и возраст `age`:

```
function User(name, age) {
  this.name = name;
  this.age = age;
}

let john = new User("John", 25);

alert( john.age ); // 25
```

...Но рано или поздно всё может измениться. Взамен возраста `age` мы можем решить хранить дату рождения `birthday`, потому что так более точно и удобно:

```
function User(name, birthday) {
  this.name = name;
  this.birthday = birthday;
}

let john = new User("John", new Date(1992, 6, 1));
```

Что нам делать со старым кодом, который использует свойство `age`?

Мы можем попытаться найти все такие места и изменить их, но это отнимает время и может быть невыполнимо, если код используется другими людьми. И кроме того, `age` – это отличное свойство для `user`, верно?

Давайте его сохраним.

Добавление геттера для `age` решит проблему:

```
function User(name, birthday) {
  this.name = name;
  this.birthday = birthday;

  // возраст рассчитывается из текущей даты и дня рождения
  Object.defineProperty(this, "age", {
    get() {
      let todayYear = new Date().getFullYear();
      return todayYear - this.birthday.getFullYear();
    }
  });
}

let john = new User("John", new Date(1992, 6, 1));

alert( john.birthday ); // доступен как день рождения
alert( john.age );      // ...так и возраст
```

Теперь старый код тоже работает, и у нас есть отличное дополнительное свойство!

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.

166 Комментариев

Войти

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ ИЛИ ЧЕРЕЗ DISQUS ?

Имя

16 Поделиться

Лучшие Новые Старые

Юрий

месяц назад

⚡ **Хочешь разобраться зачем нужны геттеры и сеттеры?**
Тогда переходи в 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.

Следи за моим путем в мир разработки: от новичка до создателя 🤖 Телеграм-бота для автоматической отправки откликов!

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

Вбивайте «Джун на фронте» и присоединяйтесь к нам!

0 2 Ответить

I

Ibragim

6 месяцев назад

Сегодня я узнал какой настоящий возраст у John`a

10 0 Ответить

S

Sergey Gonchar

7 месяцев назад

Возможно я слишком стар для всего этого дерьма плохо соображаю, но мне остается неясным, зачем функции маскировать под свойства? Разве это не создает путаницу в чтении кода?

0 0 Ответить

A

Андрей Введенский Sergey Gonchar

6 месяцев назад

Не стоит потокать своему СПГС. Решили вот такую фичу добавить в язык, по аналогии со всеми ООП языками, вот и все.

0 0 Ответить

Этот комментарий ожидает проверки. Показать комментарий.

дн

devmen → Джун на фронте
год назад

опять ты хуила:D
черт ты, а не джун)

11 0 Ответить

Егор Храпунов → devmen
год назад

добра тебе, поскуда))

1 1 Ответить

Сергей Веденяпин → devmen
год назад

я тоже бесился от этих бесконечных простыней в комментах, пока не открыл для себя возможность блокировать пользователя. И теперь не вижу этих комментариев.

1 0 Ответить

Егор Храпунов → Сергей Веденяпин
год назад

толку гореть от человека из инета) ???

0 0 Ответить

Сергей Веденяпин → Егор Храпунов
год назад

согласен, но что поделать?))) я же не могу приказать себе быть хорошим)

0 0 Ответить

Арест Хачатрян → devmen
год назад

откуда такая агрессия

1 0 Ответить

miksa
год назад

и для чего это нужно, если можно использовать обычные методы..

2 6 Ответить

Сергей Веденяпин → miksa
год назад

это синтаксический сахар)

0 0 Ответить

ilowen → miksa
год назад edited

Самая простая причина: если, например fullName() определен как геттер, мы можем получить доступ к его значению обратившись к нему как к простому свойству объекта (не вызывая его как функцию), что делает код более "читабельным". Но это только верхушка.

0 0 Ответить

Sleepy Asura
год назад edited

Вообще, мне будет просто запомнить гет и сет, т.к они очень похожи на декораторы для функции. Как декораторы расширяют и что-то делают с получаемой функцией, так и геттеры и сеттеры расширяют возможности над свойствами объекта, и оба делают это за кулисами.

2 1 Ответить

С **Северная жиза**
2 года назад

А где практические задачки? Вообще маловато практики конечно, что б одну тему изучил и задачек 20 на неё решить, вот тогда точно тему усвоил

8 0 Ответить

eternal_elers → Северная жиза
2 года назад

Ну здесь тема крайне легкая в усвоении и, уж темболее, понимании. А вот без задачек в Замыканиях или Регулярных выражениях было бы посложнее.
Главное - всегда пишите от самостоятельно код из примеров. Так вы будете блок за блоком писать код и смотреть на него как на отдельные элементы, попутно анализируя их работу, абстрагируясь от конечного варианта.
А задачки - тут Вам нужны полет фантазии и немного гугления. Последнему нужно учиться сразу, а не ждать ответов.

6 2 Ответить

MB Этот комментарий ожидает проверки. Показать комментарий.

D **devmen** → Михаил Виноградов
год назад

исчезни хуила

3 0 Ответить

M **Мария**
2 года назад

Хм, последний пример про возраст выходит с небольшим багом)
если бы Джон родился в декабре 1992, то сегодня 30 августа, ответ был бы не "30")

2 1 Ответить

Gascaynae → Мария
год назад

Да, верно, там имеется баг. Но тут сделано как обычный пример и каждый может дописать программу с проверкой всех условий :)

0 0 Ответить

Andrei Khotko → Мария
2 года назад

Так ведь верно, 30 августа 2022 года ему будет 29, а 30 лет ему станет в декабре 2022 года...

0 0 Ответить

Сергей Веденяпин → Andrei Khotko
2 года назад

удивительно, правда?))))

0 0 Ответить

Александр Вельможко → Мария
2 года назад

для начала, где там декабрь нашли?

1 0 Ответить

Владимир Филиппов
2 года назад

> Например, если мы хотим запретить устанавливать короткое имя для user, мы можем использовать сеттер name для проверки, а само значение хранить в отдельном свойстве _name:

Это зачем ?

Правильно ли я понял, что Pete из примера будет хранится, но в сам объект не попадёт ? Не понимаю зачем здесь _

0 1 Ответить

Савелий Жадан → Владимир Филиппов
2 года назад

Pete из примера будет храниться в обьекте в свойстве _name, так как мы задаем его в set(value) в строчке this._name = value
Идея состоит в том, что к таким свойствам не принято обращаться напрямую и нужны они просто, чтобы хранить значение

2 0 Ответить

Владимир Филиппов → Савелий Жадан
2 года назад

Спасибо!

Спасибо!

0 0 Ответить

К **Кирилл**

2 года назад edited

Почему дескриптор свойства-аксессора (name) отображается как свойство-данные?

```
"{
  'name': {
    'value': 'Pete',
    'writable': true,
    'enumerable': true,
    'configurable': true
  },
  '_name': {
    'value': 'Pete',
    'writable': true,
    'enumerable': true,
    'configurable': true
  }
}"
```

```
let user = {
  get name() {
```

показать больше

1 0 Ответить

A **Aleksandr Volkov** → Кирилл

2 года назад

Да, почему-то при использовании метода Object.assign копируется как свойство-данные. При копировании через метод из прошлого урока `let clone = Object.defineProperties({}, Object.getOwnPropertyDescriptors(user)` такого не происходит и свойство-аксессор копируется как свойство-аксессор. Судя по всему, у методов заложено разное поведение, нужно углубляться в спецификацию

2 0 Ответить

S **Shira**

2 года назад

Такс, почитал вроде сегодня немного можно и отдохнуть,лето же! Ребят не забрасывайте,но и не забывайте что нельзя сидеть целыми днями на этом сайте, саморазвитие это хорошо, но и отдых тоже

13 2 Ответить

 **Дима Дим** → Shira

2 года назад

отдых? если для тебя тяжело читать текст который всем приносит удовольствие то тебе нужно обернуться на свой путь жизни и посмотреть где ты свернул не туда ,чел. Всем интересно читать как школьникам смотреть тик ток

0 9 Ответить

S **Shira** → Дима Дим
2 года назад

Мне приносит удовольствие чтение , но у меня нагорало от дз , которое иногда сделать сложно.Многим не интересно читать , а интереснее делать практику , как и мне))

4 0 Ответить

E **Евгений**
2 года назад

Не совсем понял фразу "При попытке указать и get, и value в одном дескрипторе будет ошибка". Имеется в виду, что мы не можем в одном defineProperty и добавить геттер/сеттер и установить значение данного свойства?

2 0 Ответить

K **Константин Дубовцев** → Евгений
2 года назад

Что здесь может быть непонятного? При создании нового свойства с помощью defineProperty может быть созданы либо свойства-данные, либо свойства-аксессоры. Либо то, либо другое, но не оба вместе.

2 0 Ответить

D **dockwall** → Евгений
2 года назад

Смотри, свойства-аксессоры не имеют стандартного value, это по сути функции, обернутые в свойство. Ты можешь их прописывать так, что при обращении к "свойству" get будет что-то возвращать, а при присваивании set - что-то устанавливать

0 0 Ответить

E **Евгений** → dockwall
2 года назад

Это я понимаю. Только не понимаю, как читать эту мудреную фразу.
Геттер в любом случае не позволяет устанавливать значения, а только получать их из свойства (не важно, "реального" или аксессора).

1 0 Ответить

D **dockwall** → Евгений
2 года назад

Обрати внимание, что ты еще и writable не можешь поставить вместе с get/set.

А все потому что у тебя свойство может быть одним из двух вариантов:

1)Свойство-значение: просто обычное свойство с value
2)Свойство-аксессор: оно по сути работает "посредником", не имеет своего value, его соответственно невозможно перезаписать (ведь все присваивания идут через сеттер), поэтому для него и writable не работает

2 0 Ответить

A **Artemiusz Kuzniecow**
2 года назад

Понимаю, что не в тему, да и мало кому это будет полезным, но я рад, что не я один слушаю Элиса Купера

6 1 Ответить

I **IsGris** Artemiusz Kuzniecow
2 года назад

некоторые люди радуются жизни,некоторые богатству,но этот человек обошел всех их стороной,он рад что слушает Элиса Купера

1 1 Ответить

A **Artemiusz Kuzniecow** IsGris
2 года назад

Я же написал, что радуюсь, что слушаю его не один. Всегда же приятно, когда кто-то ещё кроме тебя слушает Элиса Купера.

3 0 Ответить

Загрузить ещё комментарии