4 апреля 2022 г.

# Опциональная цепочка '?.'

**Новая возможность**

Эта возможность была добавлена в язык недавно. В старых браузерах может понадобиться полифил.

Опциональная цепочка `?.` — это безопасный способ доступа к свойствам вложенных объектов, даже если какое-либо из промежуточных свойств не существует.

## Проблема «несуществующего свойства»

Если вы только начали читать учебник и изучать JavaScript, то, возможно, проблема вас ещё не коснулась, но она довольно распространена.

В качестве примера предположим, что у нас есть объекты `user`, которые содержат информацию о наших пользователях.

У большинства наших пользователей есть адреса в свойстве `user.address` с улицей `user.address.street`, но некоторые из них их не указали.

В таком случае, когда мы попытаемся получить `user.address.street`, а пользователь окажется без адреса, мы получим ошибку:

```
let user = {}; // пользователь без свойства "address"

alert(user.address.street); // Ошибка!
```

Это ожидаемый результат. JavaScript работает следующим образом. Поскольку `user.address` имеет значение `undefined`, попытка получить `user.address.street` завершается ошибкой.

Во многих практических случаях мы бы предпочли получить здесь `undefined` вместо ошибки (что означало бы «улицы нет»).

...Или ещё один пример. В веб-разработке мы можем получить объект, соответствующий элементу веб-страницы, с помощью специального вызова метода, такого как `document.querySelector('.elem')`, и он возвращает `null`, когда такого элемента нет.

```
// document.querySelector('.elem') равен null, если элемента нет
let html = document.querySelector('.elem').innerHTML; // ошибка, если он равен
```

Ещё раз, если элемент не существует, мы получим сообщение об ошибке доступа к свойству `.innerHTML` у `null`. И в некоторых случаях, когда отсутствие элемента является нормальным, мы хотели бы избежать ошибки и просто принять `html = null` в качестве результата.

Как мы можем это сделать?

Очевидным решением было бы проверить значение с помощью `if` или условного оператора `?`, прежде чем обращаться к его свойству, вот так:

```
let user = {};

alert(user.address ? user.address.street : undefined);
```

Это работает, тут нет ошибки... Но это довольно неэлегантно. Как вы можете видеть, `"user.address"` появляется в коде дважды.

Вот как то же самое выглядело бы для `document.querySelector`:

```
let html = document.querySelector('.elem') ? document.querySelector('.elem').i
```

Как видно, поиск элемента `document.querySelector('.elem')` здесь вызывается дважды, что не очень хорошо.

Для более глубоко вложенных свойств это ещё менее красиво, поскольку потребуется больше повторений.

К примеру, давайте аналогично вычислим `user.address.street.name`.

Нам нужно проверить как `user.address`, так и `user.address.street`:

```
let user = {}; // у пользователя нет адреса

alert(user.address ? user.address.street ? user.address.street.name : null : r
```

Это просто ужасно, у кого-то могут даже возникнуть проблемы с пониманием такого кода.

Есть немного лучший способ написать это, используя оператор `&&`:

```
let user = {}; // пользователь без адреса

alert( user.address && user.address.street && user.address.street.name ); // u
```

Проход при помощи логического оператора И `&&` через весь путь к свойству гарантирует, что все компоненты существуют (если нет, вычисление прекращается), но также не является идеальным.

Как вы можете видеть, имена свойств по-прежнему дублируются в коде. Например, в приведённом выше коде `user.address` появляется три раза.

Вот почему в язык была добавлена опциональная цепочка `?.`. Чтобы решить эту проблему – раз и навсегда!

## Опциональная цепочка

Опциональная цепочка `?.` останавливает вычисление и возвращает `undefined`, если значение перед `?.` равно `undefined` или `null`.

**Далее в этой статье, для краткости, мы будем говорить, что что-то «существует», если оно не является `null` и не `undefined`.**

Другими словами, `value?.prop` :

- работает как `value.prop` , если значение `value` существует,
- в противном случае (когда `value` равно `undefined/null` ) он возвращает `undefined` .

Вот безопасный способ получить доступ к `user.address.street` , используя `?.` :

```
let user = {}; // пользователь без адреса

alert( user?.address?.street ); // undefined (без ошибки)
```

Код лаконичный и понятный, в нем вообще нет дублирования.

А вот пример с `document.querySelector` :

```
let html = document.querySelector('.elem')?.innerHTML; // будет undefined, ec
```

Считывание адреса с помощью `user?.address` работает, даже если объект `user` не существует:

```
let user = null;

alert( user?.address ); // undefined
alert( user?.address.street ); // undefined
```

Обратите внимание: синтаксис `?.` делает необязательным значение перед ним, но не какое-либо последующее.

Так например, в записи `user?.address.street.name ?.` позволяет `user` безопасно быть `null/undefined` (и в этом случае возвращает `undefined` ), но это так только для `user` . Доступ к последующим свойствам осуществляется обычным способом. Если мы хотим, чтобы некоторые из них были необязательными, тогда нам нужно будет заменить больше `.` на `?.` .

**Не злоупотребляйте опциональной цепочкой**

Нам следует использовать `?.` только там, где нормально, что чего-то не существует.

К примеру, если, в соответствии с логикой нашего кода, объект `user` должен существовать, но `address` является необязательным, то нам следует писать `user.address?.street` , но не `user?.address?.street` .

В этом случае, если вдруг `user` окажется `undefined` , мы увидим программную ошибку по этому поводу и исправим её. В противном случае, если слишком часто использовать `?.` , ошибки могут замалчиваться там, где это неуместно, и их будет сложнее отлаживать.

**Переменная перед `?.` должна быть объявлена**

Если переменной `user` вообще нет, то `user?.anything` приведёт к ошибке:

```
// ReferenceError: user is not defined
user?.address;
```

Переменная должна быть объявлена (к примеру, как `let/const/var user` или как параметр функции). Опциональная цепочка работает только с объявленными переменными.

## Сокращённое вычисление

Как было сказано ранее, `?.` немедленно останавливает вычисление, если левая часть не существует.

Так что если после `?.` есть какие-то вызовы функций или операции, то они не произойдут.

Например:

```
let user = null;
let x = 0;

user?.sayHi(x++); // нет "user", поэтому выполнение не достигает вызова sayHi

alert(x); // 0, значение не увеличилось
```

## Другие варианты применения: ?.(), ?.[]

Опциональная цепочка `?.` — это не оператор, а специальная синтаксическая конструкция, которая также работает с функциями и квадратными скобками.

Например, `?.()` используется для вызова функции, которая может не существовать.

В приведённом ниже коде у некоторых наших пользователей есть метод `admin`, а у некоторых его нет:

```
let userAdmin = {
  admin() {
    alert("Я админ");
  }
};

let userGuest = {};

userAdmin.admin?.(); // Я админ

userGuest.admin?.(); // ничего не произойдет (такого метода нет)
```

Здесь в обеих строках мы сначала используем точку (`userAdmin.admin`), чтобы получить свойство `admin`, потому что мы предполагаем, что объект `user` существует, так что читать из него безопасно.

Затем `?.()` проверяет левую часть: если функция `admin` существует, то она запускается (это так для `userAdmin`). В противном случае (для `userGuest`) вычисление остановится без ошибок.

Синтаксис `?.[]` также работает, если мы хотим использовать скобки `[]` для доступа к свойствам вместо точки `.`. Как и в предыдущих случаях, он позволяет безопасно считывать свойство из объекта, который может не существовать.

```
let key = "firstName";

let user1 = {
  firstName: "John"
};

let user2 = null;

alert( user1?.[key] ); // John
alert( user2?.[key] ); // undefined
```

Также мы можем использовать `?.` с `delete`:

```
delete user?.name; // удаляет user.name если пользователь существует
```

**Мы можем использовать `?.` для безопасного чтения и удаления, но не для записи**

Опциональная цепочка `?.` не имеет смысла в левой части присваивания.

Например:

```
let user = null;

user?.name = "John"; // Ошибка, не работает
// то же самое что написать undefined = "John"
```

## Итого

Синтаксис опциональной цепочки `?.` имеет три формы:

1. `obj?.prop` – возвращает `obj.prop` если `obj` существует, в противном случае `undefined`.
2. `obj?.[prop]` – возвращает `obj[prop]` если `obj` существует, в противном случае `undefined`.
3. `obj.method?.()` – вызывает `obj.method()`, если `obj.method` существует, в противном случае возвращает `undefined`.

Как мы видим, все они просты и понятны в использовании. `?.` проверяет левую часть на `null/undefined` и позволяет продолжить вычисление, если это не так.

Цепочка `?.` позволяет безопасно получать доступ к вложенным свойствам.

Тем не менее, мы должны использовать `?.` осторожно, только там, где по логике кода допустимо, что левая часть не существует. Чтобы он не скрывал от нас ошибки программирования, если они возникнут.

Предыдущий урок Следующий урок

Проводим курсы по JavaScript и фреймворкам.

## 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.

110 Комментариев

1 Войти

Присоединиться к обсуждению...

ВОЙТИ С ПОМОЩЬЮ ИЛИ ЧЕРЕЗ DISQUS ?

Имя

11 Поделиться

Лучшие Новые Старые

**Джун на фронте**

день назад

Приглашаю в **блог JS-прогресса** !

В **тг Джун на фронте** я врываюсь в веб! Учусь делать динамичные сайты, без банальных конструкторов 🤮

Мы вместе пройдем собесы, отправим первый отклик и выполним тестовые!

0 0 Ответить • Поделиться ›

**KingNut Academy**

9 дней назад

Разбор задач, тестов и всего что поможет при прохождении собеседования на должность frontend developer (html, css, js, ts, react). Заходи в ТГ @interview_masters

0 0 Ответить • Поделиться ›

**Alexey Kazakov**

21 день назад

Авторы, пожалуйста, уберите неразбериху с фразой "перед ?." - У вас перед то слева, то справа, так не должно быть.

0 0 Ответить • Поделиться ›

**хлорка**

3 месяца назад

14.12.2022 16:18

0 0 Ответить • Поделиться ›

**Kei Presley**

4 месяца назад

У меня delete user?.name; из примера с урока выдает ошибку, что с ?., что без. Бесполезен с delete...

0 1 Ответить • Поделиться ›

© 2007—2023 Илья Канторо проектесвязаться с намипользовательское соглашение политика конфиденциальности

Получается синтаксис ?. не проверяет перед удалением наличие переменной user, что делает его бесполезным в операциях удаления. Если я ошибаюсь, поправьте, пожалуйста

1 1 Ответить • Поделиться ›

**Player_50B** → Виталий Цыганок
2 месяца назад

Получается - да.

0 0 Ответить • Поделиться ›

X

**хлорка** → Виталий Цыганок
3 месяца назад

если переменная не объявлена он выдаст ошибку

0 0 Ответить • Поделиться ›

**Виталий Цыганок** → хлорка
3 месяца назад

Именно поэтому в операциях удаления оператор ?. бесполезен, так он возвращает ошибку, если переменная не существует. Хотя мог бы просто возвращать undefined, а не ошибку

1 1 Ответить • Поделиться ›

**Андрій Білецький** → Виталий Цыганок
месяц назад

Увеличьте вложенность свойств
let user={
name:{
firstName: 'John'}}

и теперь проверьте конструкцию ?. с delete на свойстве name - все прекрасно работает.

1 0 Ответить • Поделиться ›

A

**Александр** → Виталий Цыганок
2 месяца назад

Вам уже 3 раза объяснили, при чем тут delete?
Внимательно читайте статью: "Если переменной user вообще нет, то user?.anything приведёт к ошибке..."

1 0 Ответить • Поделиться ›

**CyberFalcon** → Kei Presley
4 месяца назад

Может переменная 'user' не объявлена. Что пишет в ошибке?

7 0 Ответить • Поделиться ›

**D** **Denise**
5 месяцев назад

Hehe What?..

0 0 Ответить • Поделиться ›

 **Илья Александров**
7 месяцев назад

Разьяснение о том что такое queryselector, innerHTML и тд через +100500 тем, но сделаем вид, что все всем понятно :)

12 1 Ответить • Поделиться ›

 **eternal_elers** → Илья Александров
7 месяцев назад

"В веб-разработке мы можем получить объект, соответствующий элементу веб-страницы, с помощью специального вызова метода, такого как document.querySelector('.elem')"
Прямо из текста "вырываю", комментарии излишни...

3 0 Ответить • Поделиться ›

**R** **RvR By** → Илья Александров
7 месяцев назад

А какая разница, что делает queryselector? Это никак к теме не относится :)

6 0 Ответить • Поделиться ›

**S** **S1m** → Илья Александров
7 месяцев назад

сейчас это не так важно, тут скорее просто показали как это будет выглядеть(куда лучше и читабельнее)

2 0 Ответить • Поделиться ›

 **Alex Safronov** → Илья Александров
7 месяцев назад

Это уже работа с DOM. queryselector : выбираешь элемент в html тот же class или id. innerHTML выбор HTML документа.

0 0 Ответить • Поделиться ›

**C** **Северная жиза**
7 месяцев назад

Если выводить не alert а в консоль через console.log, выдаёт ошибку

0 0 Ответить • Поделиться ›

**З** **Запрещаю писать даты**
7 месяцев назад

Зачем вы пишете даты прочтения debilы

10 3 Ответить • Поделиться ›

Загрузить ещё комментарии

Подписаться Privacy Не продавайте мои данные