Mehdi Bouchachi

THE COMPLETE FRONT-END DEVELOPMENT

SECTION

JAVASCRIPT REVEIW

LECTURE

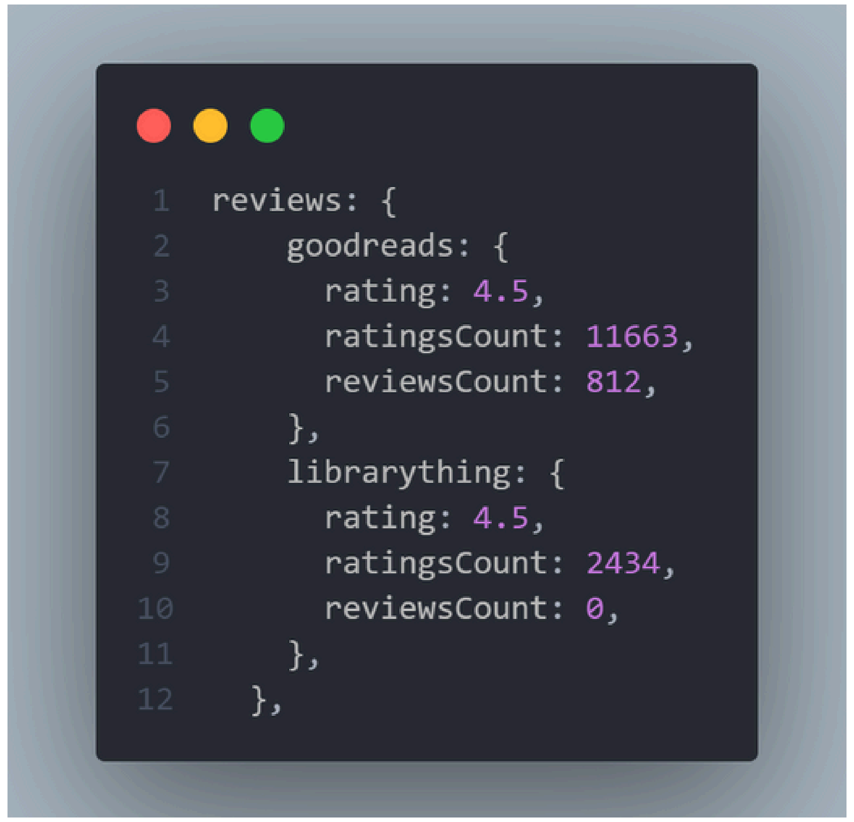
OPTIONAL CHAINING

@Mehdibouchachi

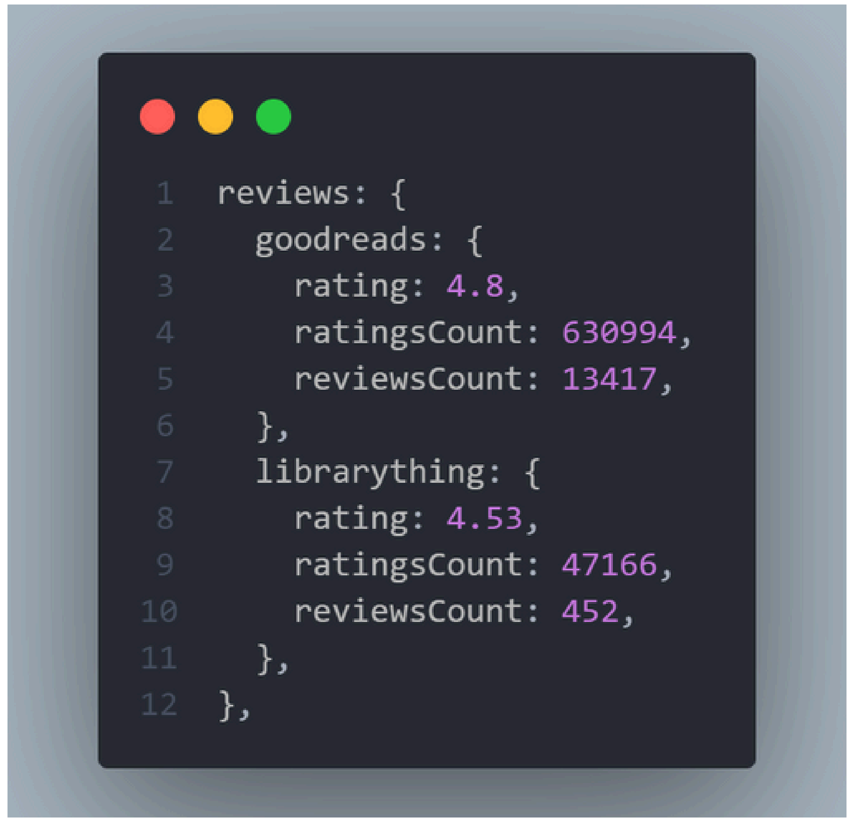
OPTIONAL CHAINING

BEFORE WE START

👉 Each movie has **the review field** and it has **reviews** from **Goodreads source** and also from **the libraryanything** And notice how each of them has a different reviews count.



```
reviews: {
    goodreads: {
      rating: 4.5,
      ratingsCount: 11663,
      reviewsCount: 812,
    },
    librarything: {
      rating: 4.5,
      ratingsCount: 2434,
      reviewsCount: 0,
    },
  },
```

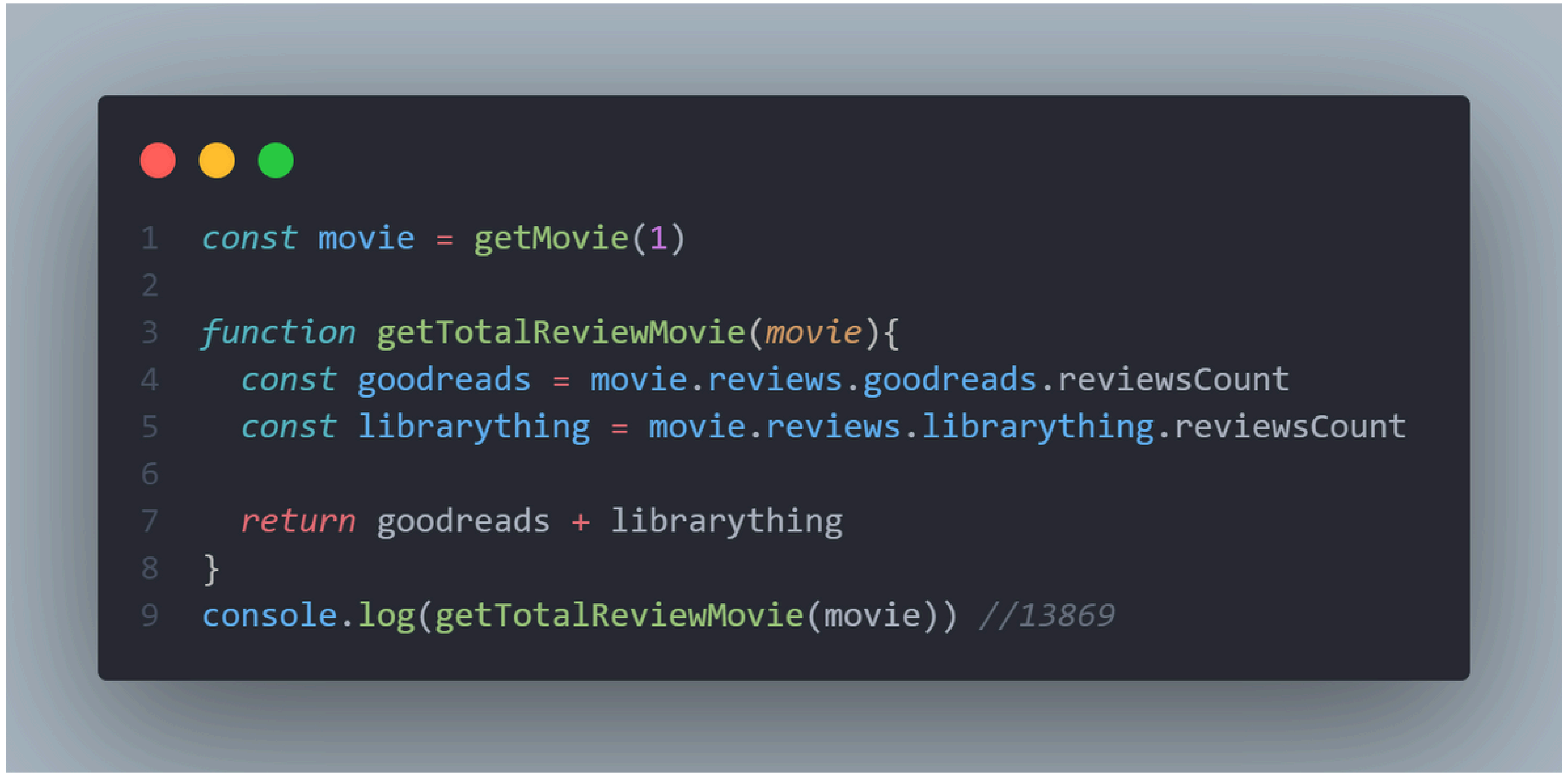


```
reviews: {
  goodreads: {
    rating: 4.8,
    ratingsCount: 630994,
    reviewsCount: 13417,
  },
  librarything: {
    rating: 4.53,
    ratingsCount: 47166,
    reviewsCount: 452,
  },
},
```

👉 We want to create a **function** that gives us the total reviewsCount for each movie.

OPTIONAL CHAINING

FUNCTION

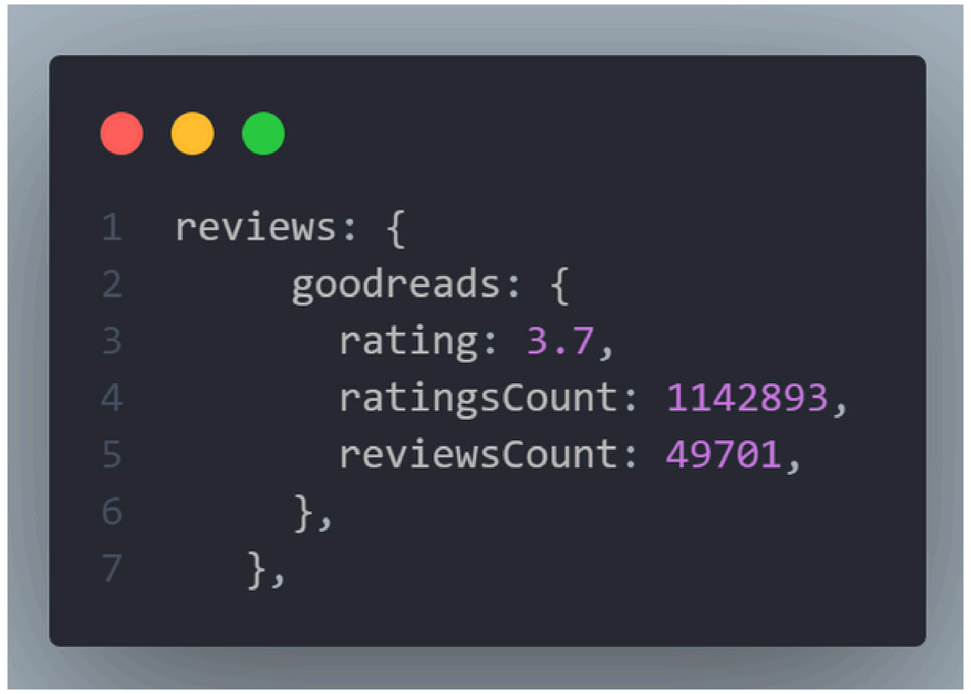


```
const movie = getMovie(1)

function getTotalReviewMovie(movie){
  const goodreads = movie.reviews.goodreads.reviewsCount
  const librarything = movie.reviews.librarything.reviewsCount

  return goodreads + librarything
}
console.log(getTotalReviewMovie(movie)) //13869
```

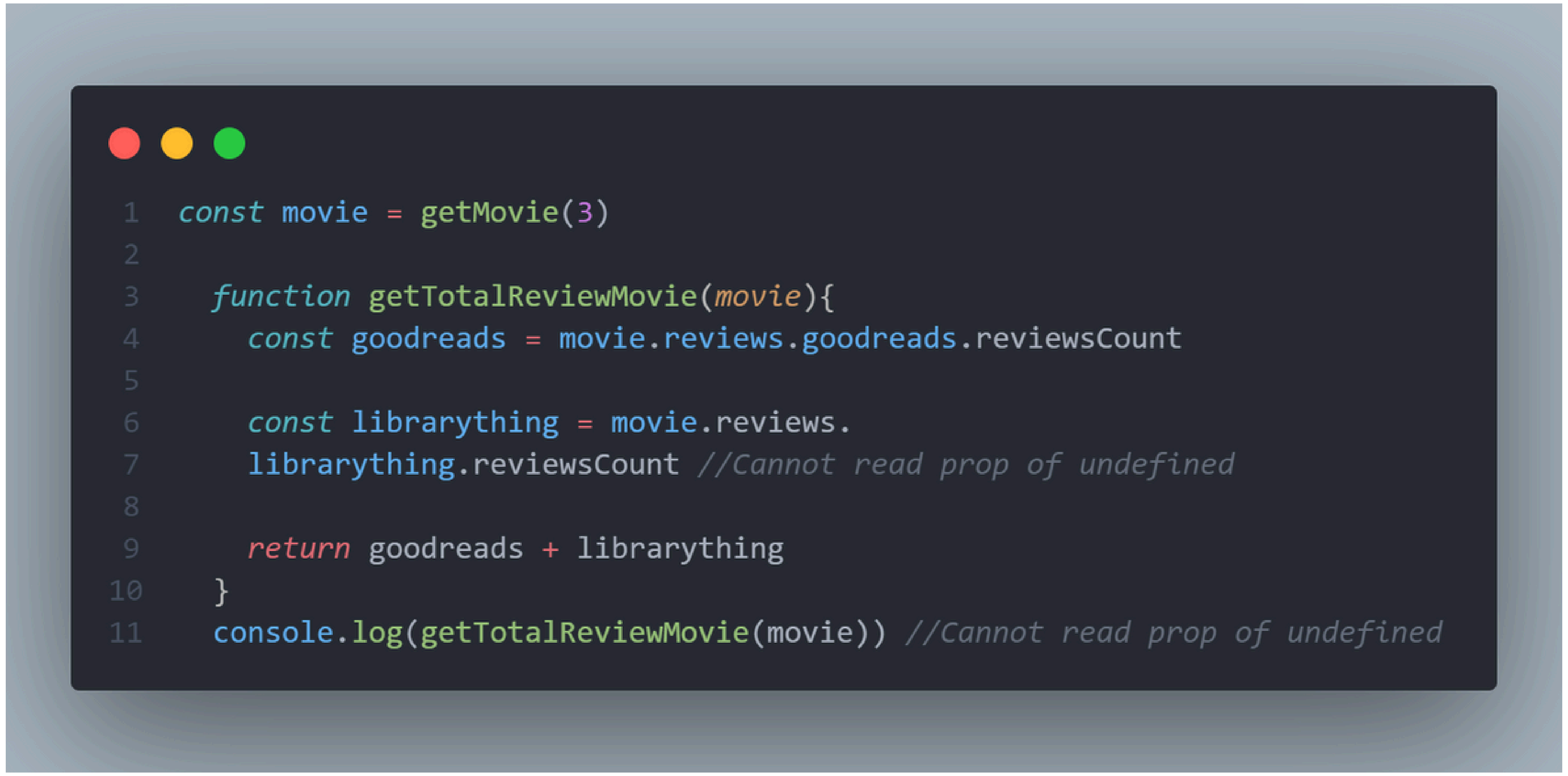
OPTIONAL CHAINING

👉 everything is normal right?

👉 let’s change to movie number 3 (librarything -> undefined)



```
reviews: {
     goodreads: {
       rating: 3.7,
       ratingsCount: 1142893,
       reviewsCount: 49701,
     },
   },
```



```
const movie = getMovie(3)

  function getTotalReviewMovie(movie){
    const goodreads = movie.reviews.goodreads.reviewsCount

    const librarything = movie.reviews.
    librarything.reviewsCount //Cannot read prop of undefined

    return goodreads + librarything
  }
  console.log(getTotalReviewMovie(movie)) //Cannot read prop of undefined
```

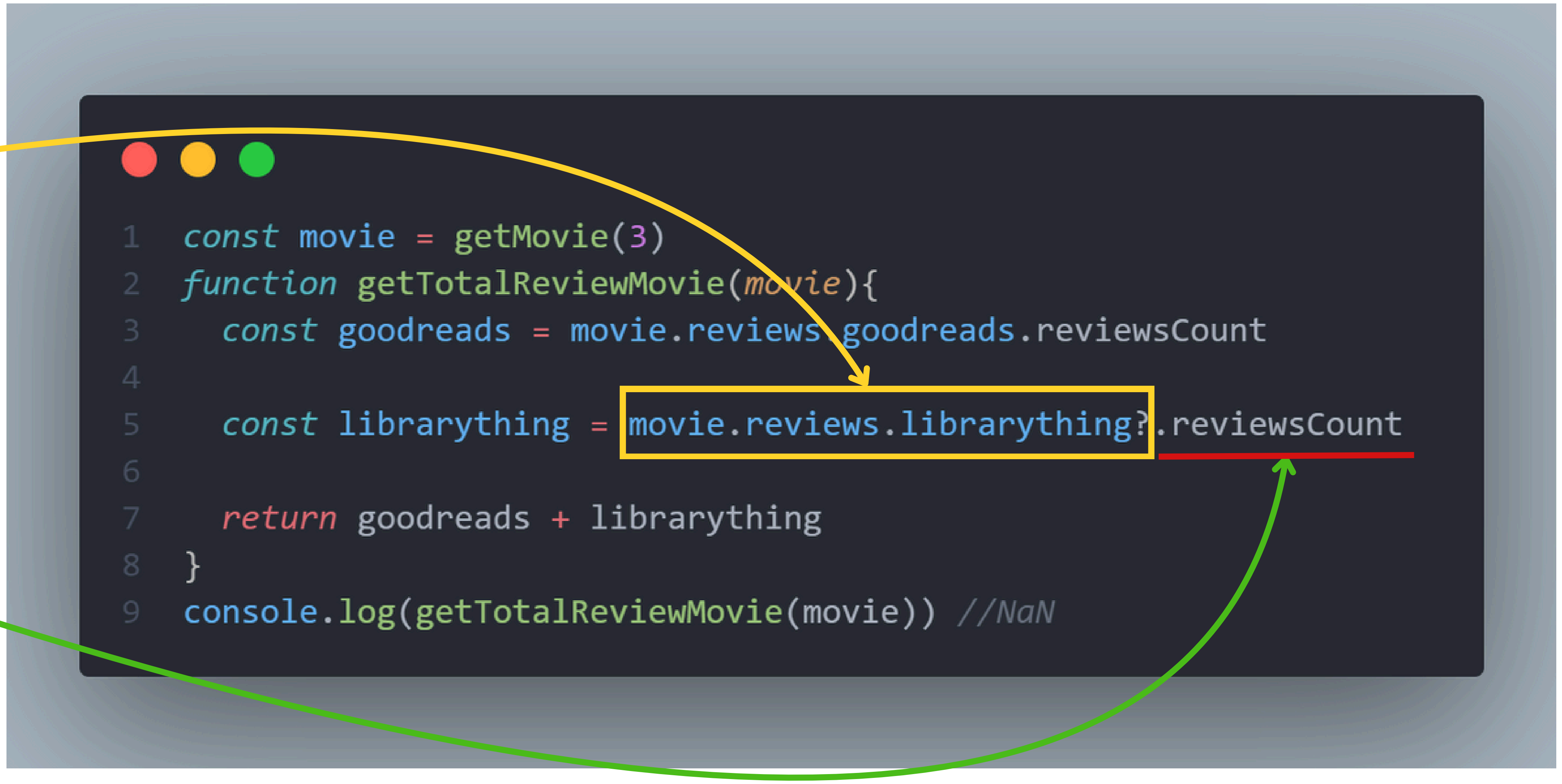
OPTIONAL CHAINING

OPTIONAL CHAINING

JavaScript has a solution for this, which is called optional chaining.

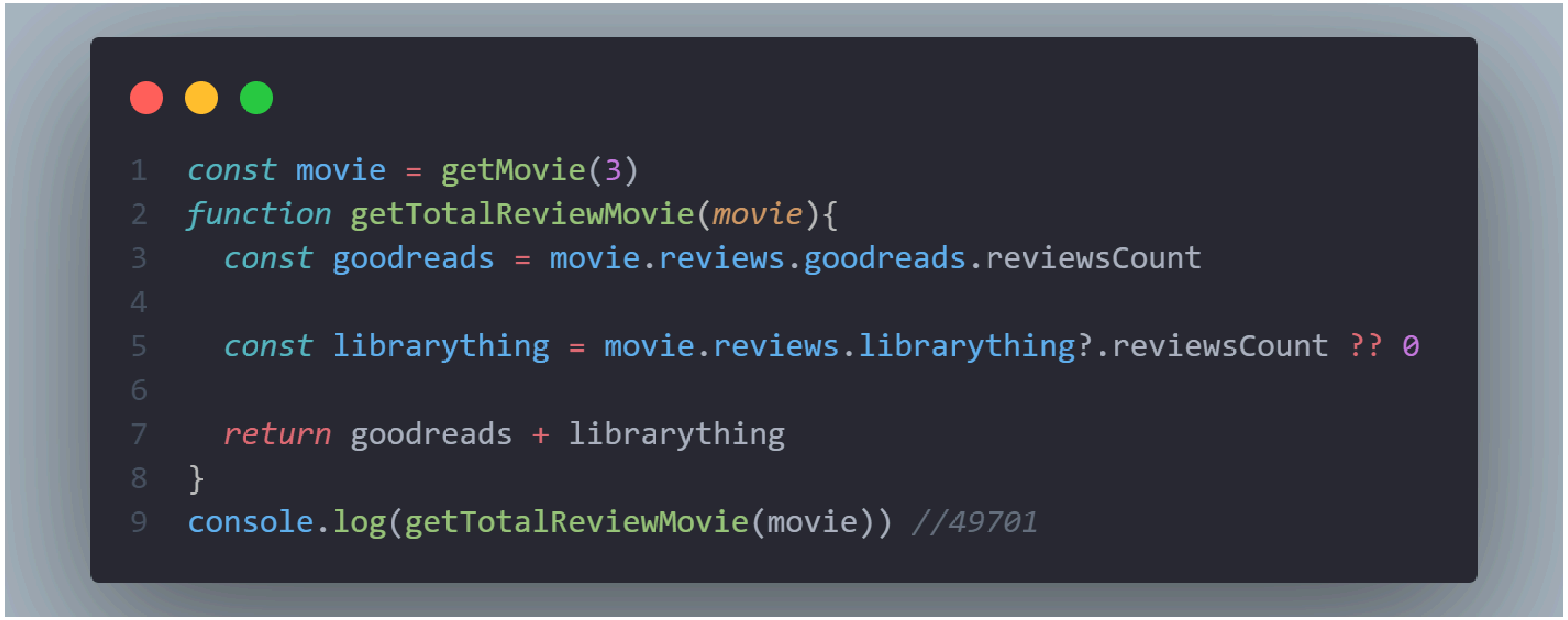
So what happens now is that whenever this here is undefined.

Then JavaScript will no longer even try to read reviewsCount out of this.



```
const movie = getMovie(3)
function getTotalReviewMovie(movie){
  const goodreads = movie.reviews.goodreads.reviewsCount

  const librarything = movie.reviews.librarything?.reviewsCount

  return goodreads + librarything
}
console.log(getTotalReviewMovie(movie)) //NaN
```

OPTIONAL CHAINING

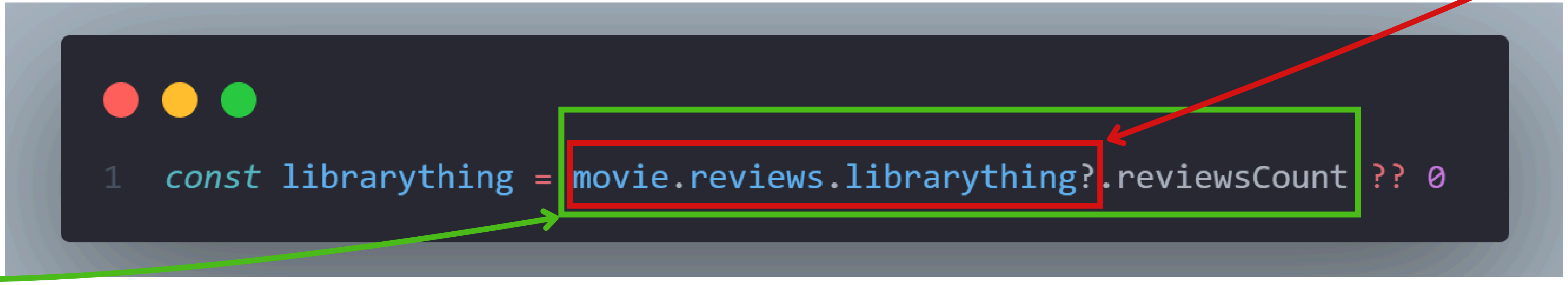


```javascript
const movie = getMovie(3)
function getTotalReviewMovie(movie){
  const goodreads = movie.reviews.goodreads.reviewsCount

  const librarything = movie.reviews.librarything?.reviewsCount ?? 0

  return goodreads + librarything
}
console.log(getTotalReviewMovie(movie)) //49701
```

OPTIONAL CHAINING

SUMMARY

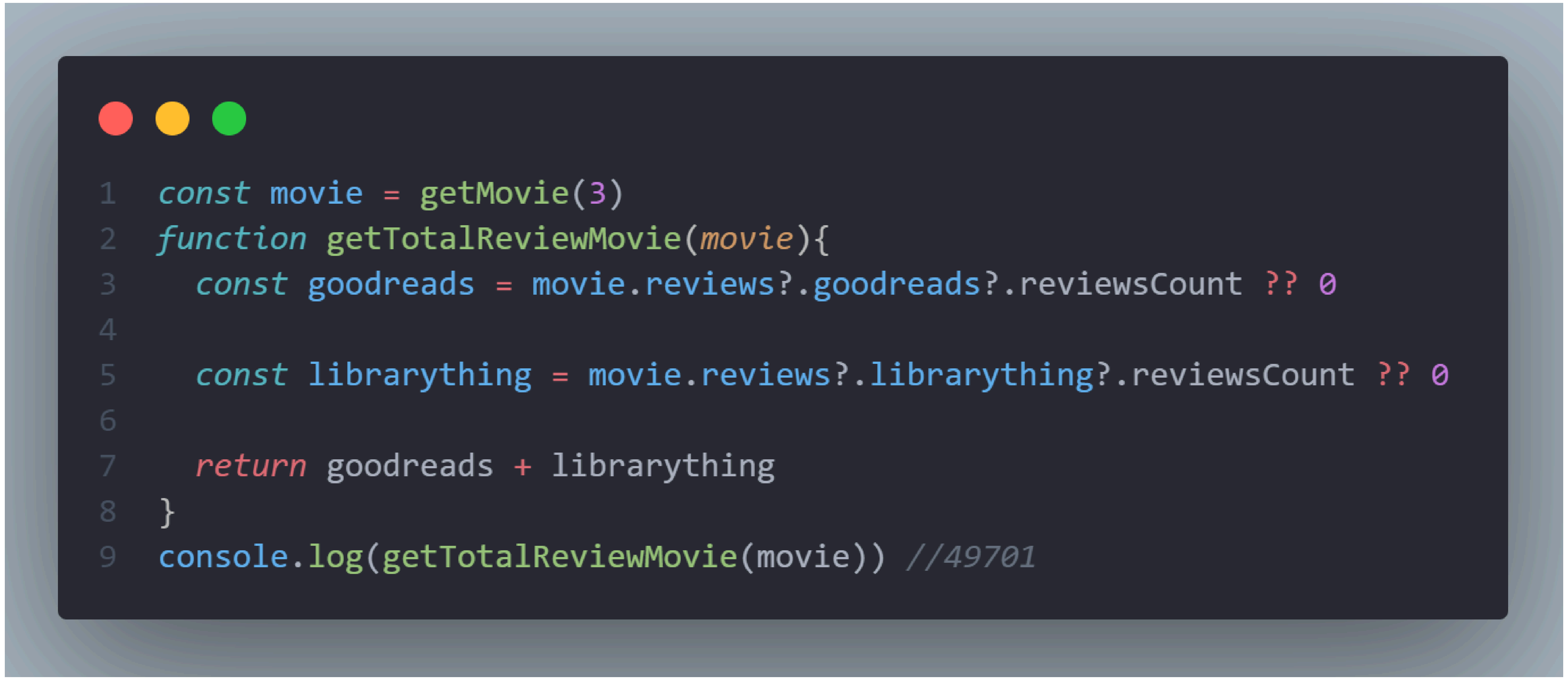
👉 what's happening here is that because of this optional chaining, JavaScript will no longer try to read **reviewsCount** out of this undefined thing here.
So before we had **undefined.reviewsCount**, which, of course, did not exist and gave us an error.
But with optional chaining, as soon as this here is undefined, the entire thing, so all of this, will automatically become undefined.
And so then we use this ?? operator to then simply give us zero instead of that undefined.



```
const librarything = movie.reviews.librarything?.reviewsCount ?? 0
```

OPTIONAL CHAINING

👉 So this is helpful whenever we are **not sure** that all the values that we expect, **exist in an object.**



```javascript
const movie = getMovie(3)
function getTotalReviewMovie(movie){
  const goodreads = movie.reviews?.goodreads?.reviewsCount ?? 0

  const librarything = movie.reviews?.librarything?.reviewsCount ?? 0

  return goodreads + librarything
}
console.log(getTotalReviewMovie(movie)) //49701
```

SEE YOU SOON...