Regular Expression: Non-Greedy Quantifiers

Summary: in this tutorial, you'll learn how to use non-greedy quantifiers to match their preceding elements as few as possible.

Introduction to JavaScript regex non-greedy (or lazy) quantifiers

In regular expressions, quantifiers allow you to match their preceding elements with specified numbers of times.

By default, quantifiers use the greedy mode for matching. In the greedy mode, quantifiers try to match as many as possible and return the largest matches. When quantifiers use the greedy mode, they are called greedy quantifiers. In the quantifier tutorial, you learned how to work with greedy quantifiers such as `*`, `+`, and `?`.

Besides the greedy mode, quantifiers also work in the non-greedy mode or lazy mode. In the lazy mode, the quantifiers match their preceding elements as few as possible and return the smallest matches. When quantifiers use the lazy mode, they're often referred to as non-greedy quantifiers or lazy quantifiers.

To transform a greedy quantifier into a non-greedy quantifier, you add an extra question mark to it. The following table lists the greedy quantifiers and their corresponding lazy quantifiers:

Greedy quantifier	Lazy quantifier	Meaning
`*`	`*?`	Match its preceding element zero or more times.
`+`	`+?`	Match its preceding element one or more times.
`?`	`??`	Match its preceding element zero or one time.

Greedy quantifier	Lazy quantifier	Meaning
{ *n* }	{ *n* }?	Match its preceding element exactly *n* times.
{ *n* ,}	{ *n* ,}?	Match its preceding element at least *n* times.
{ *n* , *m* }	{ *n* , *m* }?	Match its preceding element from *n* to *m* times.

JavaScript non-greedy quantifiers example

The following program uses the non-greedy quantifier (`+?`) to match the text within the quotes (`""`) of a button element:

```
const s = '<button type="submit" class="btn">Send</button>'
const pattern = /".+?"/g;

const result = s.match(pattern)
console.log(result);
```

Output:

```
['"submit"', '"btn"']
```

Summary

- Lazy quantifiers match their preceding elements as few as possible to return the smallest possible matches.
- Use a question mark (?) to transform a greedy quantifier into a lazy quantifier.