Regex Alternation

Summary: in this tutorial, you'll learn about JavaScript regex alternation, which is the "OR" operator in regular expressions.

Introduction to the regex alternation

Regex uses the pipe operator (`|`) to represent an alternation, which is like the logical OR operator in regular expressions. The alternation allows you to match either A or B:

```
A | B
```

The following example uses the alternation to match either the `JavaScript` or `JS` in the string `'JavaScript and JS'` :

```
const s = 'JavaScript and JS';
const pattern = /JavaScript|JS/g;
const match = s.match(pattern);

console.log(match);
```

Output:

```
[ 'JavaScript', 'JS' ]
```

Regex alternation examples

The following example illustrates the practical applications of the regex alternation.

1) Using regex alternation to match time string in the hh:mm format

The following regular expression that combines the `\d` character set with the quantifiers `{}` to match a time string in the format `hh:mm` :

```
/\d{2}:\d{2}/
```

In this regular expression:

- `\d{2}` matches two digits.
- `:` matches the colon character
- `\d{2}` matches two digits

But the rule `\d{2}` also matches an invalid hour or minute for example `99` . To make it match more precisely, you can use an alternation.

Since the valid hours are from `01` to `23` , you can use the following pattern to match the hour part:

```
[01]\d|2[0-3]
```

In this pattern:

- The rule `[01]` matches a single digit 0 or 1 and the rule `\d` matches a single digit from 0 to 9. Therefore, the rule `[01]\d` matches 00, 01 to 19
- The literal number `2` matches the digit 2 and the rule `[0-3]` matches a single digit from 0 to 3 including 0, 1, 2, 3. Therefore, the rule `2[0-3]` matches two digits 20, 21, 22, and 23.

Hence, the rule `[01]\d|2[0-3]` matches two digits from 00 to 23

Similarly, you can use the following rule to match a valid minute that ranges from 00 to 59:

```
[0-5]\d
```

The following regular expression combines those rules to match a time string in the `hh:mm` format:

```
/[01]\d|2[0-3]:[0-5]\d/g
```

However, this regular expression will not work as expected. For example:

```
const time = '05:30 31:62 23:45 26:99';
const pattern = /[01]\d|2[0-3]:[0-5]\d/g;
const match = time.match(pattern);

console.log(match);
```

Output:

```
[ '05', '23:45' ]
```

In this example, the regex engine treats the pattern `[01]\d|2[0-3]:[0-5]\d` as two parts separated by the alternation:

```
[01]\d
OR
2[0-3]):([0-5]\d)
```

To fix it, you use parentheses to wrap the alternation. It indicates that only the wrapped part is alternated, not the entire pattern:

```
([01]\d|2[0-3]):[0-5]\d
```

Now, the script works as expected:

```
const time = '05:30 31:62 23:45 26:99';
const pattern = /([01]\d|2[0-3]):[0-5]\d/g;
const match = time.match(pattern);

console.log(match);
```

Output:

```
[ '05:30', '23:45' ]
```

Summary

- The alternation `A | B` matches either `A` or `B`.
- The alternation is like an OR operator in regular expressions.
- Use parentheses `()` to wrap the parts that you want to apply the alternation.