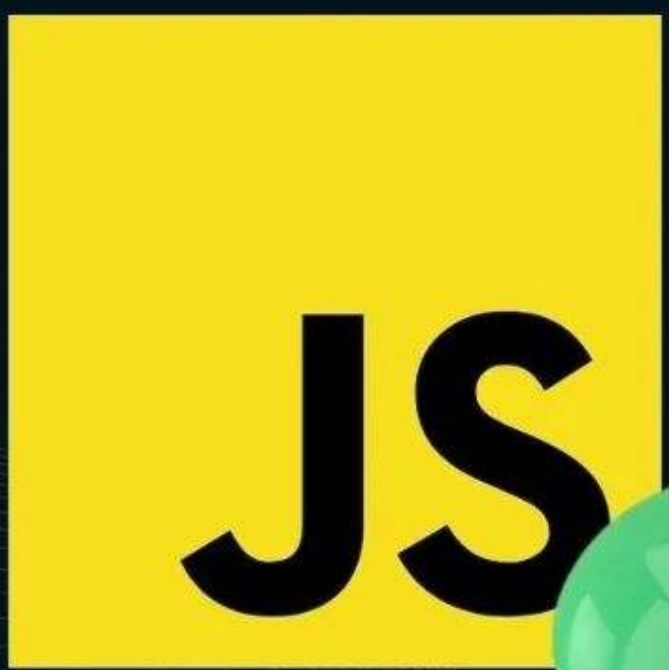
@pyplane_code

Simple regular expressions



introduction

Regular expressions enable finding patterns in a string. The two most common use cases are related to: text validation and searching through a string i.e. to find a number

Simple example - let's find hello world in a string:

```
const text = "People often write hello world while programming"
const regex = /hello world/

const result = text.match(regex)
console.log(result)


(1) ['hello world', index: 19, input: 'People often write
hello world while programming', groups: undefined]
```

find many

Let's take our previous example to the next level and find many "hello world" in a text. We need to modify the string itself and add a global option to the regex instructing it to create an array of all occurrences that will be found matching the given pattern

using match

```
const text = "People often write hello world while hello world programming"
const regex = /hello world/g

const result = text.match(regex)
console.log(result)

(2) ['hello world', 'hello world']
```

using matchAll

```
const text = "People often write hello world while hello world programming"
const regex = /hello world/g

const results = [...text.matchAll(regex)]
results.forEach(res=> console.log(res))

(1) ['hello world', index: 19, input: 'People often write hello
world while hello world programming', groups: undefined]
(1) ['hello world', index: 37, input: 'People often write hello
world while hello world programming', groups: undefined]
```

case insensitive

Add a "i" option to the modifier to make the pattern case insensitive. Without this the capital letter HELLO WORLD wouldn't be found

```
const text = "People often write hello world while HELLO WORLD programming"
const regex = /hello world/gi

const results = [...text.matchAll(regex)]
results.forEach(res=> console.log(res))
```

```
(1) ['hello world', index: 19, input: 'People often write hello
world while HELLO WORLD programming', groups: undefined]
(1) ['HELLO WORLD', index: 37, input: 'People often write hello
world while HELLO WORLD programming', groups: undefined]
```

find capital letters

In this example we are going to search for all the capital letters in the text

```
const text = "People often write hello world while HELLO WORLD programming"
const regex = /[A-Z]/g

const result = text.match(regex)
console.log(result)

(11) ['P', 'H', 'E', 'L', 'L', 'O', 'W', 'O', 'R', 'L', 'D']
```

If we would like to ignore the "P" letter:

```
const text = "People often write hello world while HELLO WORLD programming"
const regex = /[A-OQ-Z]/g
```

or

```
const text = "People often write hello world while HELLO WORLD programming"
const regex = /(?![P])[A-Z]/g

const result = text.match(regex)
console.log(result)

(10) ['H', 'E', 'L', 'L', 'O', 'W', 'O', 'R', 'L', 'D']
```

find digits

Let's now find digits in a string

```
const text = "Some 2000 text hello world x190 xyz"
const regex = /\d/g

const result = text.match(regex)
console.log(result)
```

```
(7) ['2', '0', '0', '0', '1', '9', '0']
```

Add + to the patter to get full numbers

```
const text = "Some 2000 text hello world x190 xyz"
const regex = /\d+/g

const result = text.match(regex)
console.log(result)
```

```
(2) ['2000', '190']
```

replace

As the next step let's see how to find a number in a text and immediately replace it with a question mark

```
const text = "My favourite number is 7"
const regex = /\d+/g

const blank = text.replace(regex, "?")
console.log(blank)
```

```
My favourite number is ?
```

search and exec

The search method will give us the position of the first match

```
const text = "Hello world world 1000 xyz"
const regex = /world/g

const result = text.search(regex)
console.log(result)
```

```
6
```

exec is similar to match but with the purpose to be used in a loop. It returns an array of captures or null

```
const text = "Hello world world 1000 xyz"
const regex = /world/g
let arr;

while ((arr = regex.exec(text)) !== null) {
  console.log(`Found ${arr[0]}. Next starts at ${regex.lastIndex}.`);
}
```

```
Found world. Next starts at 11.
Found world. Next starts at 17.
```

validate email

At the end let's take a look at a liitle bit more complicated example, however very practical - to validate email

```
const email = "test-email@pyplane.com"
const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/g

const result = email.match(regex)
console.log(result)
```

```
(1) ['test-email@pyplane.com']
```

Multiple "@"

```
const email = "test-email@pyplane@.com"
const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/g

const result = email.match(regex)
console.log(result)
```

```
null
```

Did you find it Useful?

Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost