02

[JavaScript] Regex com JS

Transcrição

No mundo JavaScript, temos duas formas de construir uma expressão regular. A primeira é usada através da classe `RegExp`, como já vimos no nosso script:

```
var exp = new RegExp("(\\d\\d)(\\w)");
```

A segunda forma usa uma expressão literal. Nesse caso, devemos colocar a expressão entre `/pattern aqui/`, por exemplo:

```
var exp = /(\d\d)(\w)/;
```

A segunda forma deve ser utilizada quando o *pattern* é constante, e possui um melhor desempenho, pois o navegador já compila a regex na hora de carregar o script.

Flags importantes

Em ambos os casos podemos combinar a regex com algumas flags, que indicam como aplicar o *pattern*. Por exemplo, no nosso script sempre usamos a flag `g` para fazer um *match global*, no alvo inteiro:

```
var exp = /(\d\d)(\w)/g;
```

Caso quisermos usar o objeto `RegExp`, é possível aplicar a flag no segundo parâmetro do construtor:

```
var exp = new RegExp('(\d\d)(\w)', 'g');
```

Temos mais duas flags para utilizar: `i` e `m`. A flag `i` significa *ignorecase*, ou seja, independente de letra maiúscula ou minúscula. A flag `m` significa *multiline*, para aplicar a regex linha por linha. Nesse caso, as âncoras `^` e `$` selecionam o início e o fim de uma linha, e não da string inteira.

Principais métodos do mundo JavaScript

Uma vez o objeto regex criado, tanto faz com a classe ou de maneira literal, podemos usufruir dos métodos para encontrar o padrão no alvo.

O primeiro método é o `exec`, que executa a regex e devolve um array com as informações sobre o *match*. Por exemplo, vamos definir uma regex com um grupo e chamar o método `exec`:

```
var regex = /(\d\d)(\w)/g; //2 dígitos e 1 word char, dois grupos
var resultado = regex.exec('11a22b33c ');
```

O resultado possui as informações sobre primeiro *match:*

```
console.log(resultado[0]); //devolve o match inteiro: 11a
console.log(resultado[1]); //devolve o primeiro grupo: 11
console.log(resultado[2]); //devolve o segundo grupo a
console.log(resultado.index); //devolve a posição onde o match começo no alvo: 0
console.log(regex.lastIndex); //devolve a última posição do match: 3
```

Para pegar o próximo *match,* devemos chamar novamente o método `exec` :

```
var resultado = regex.exec('11a22b33c ');
console.log(resultado[0]); //devolve o match inteiro: 22b
console.log(resultado[1]); //devolve o primeiro grupo: 22
console.log(resultado[1]); //devolve o primeiro grupo: b
console.log(resultado.index); //devolve a posição onde o match começo no alvo: 3
console.log(regex.lastIndex); //devolve a última posição do match: 6
```

Se não há mais nenhum *match,* o método `exec` devolve `null` .

Caso só quisermos saber se há algum *match,* existe o método `test` , que devolve um booleano:

```
console.log(regex.test('11a22b')); //true
```