

Introdução

Caso Simples

```
const regex = /(a|b)(c)*d/

const palavras = ['ad', 'accccd', 'a', 'c', 'd', 'cccd', '']

for(w of palavras){
  console.log(`Palavra[${w}]: ${regex.test(w) ? 'aceita' : 'rejeita'}`)
}
```

Com 'd' opcional e (a|b) no inicio sendo 1 ou mais vezes

```
const regex = /(a|b)+(c)*(d?)/

//Informar porque adaccdaccccccd foi aceita (uso do ^$)
const palavras = ['ad', 'accccd', 'a', 'aaaaaaaaaccccccc', 'c', 'd',
'cccd', 'adaccdaccccccd', '']

for(w of palavras){
  console.log(`Palavra[${w}]: ${regex.test(w) ? 'aceita' : 'rejeita'}`)
}
```

Palavra vazia

```
const regex = /^(a|b)+(c)*(d?))*$/

const palavras = ['ad', 'accccd', 'a', 'aaaaaaaaaccccccc', 'c', 'd',
'cccd', '', 'bdaaaaccccd']

for(w of palavras){
  console.log(`Palavra[${w}]: ${regex.test(w) ? 'aceita' : 'rejeita'}`)
}
```

Teste de tipos de palavras

```
const regex = {
  BRANCO: /^\\s$/,
  BRANCOS: /^\\s*$/,
  DIGITO: /^[0-9]$/,
```

```

DIGITOS: /^[0-9]+$/,
LETRA: /^[A-Za-z]$/,
LETRAS: /^[A-Za-z]+$/,
IDENT: /^[A-Za-z]([A-Za-z]|[0-9])*$/,
REAL: /^[^-\?[0-9]+\.[0-9]+$/,
INTEIRO: /^[^-\?[0-9]+$/,
};

const confere = (padrao, sentenca) => {
  console.log(
    `W: '${sentenca}' ..... ${
      padrao.test(sentenca) ? "ACEITA" : "rejeitada."
    }`
  );
};

confere(regex.DIGITOS, "000511200021"); // Deve ser aceito
confere(regex.DIGITOS, "000511200021ADAF"); // Deve ser rejeitado

confere(regex.LETRAS, "ASDFEAFdafsafdsf"); // Deve ser aceito
confere(regex.LETRAS, "ASDFEAFdafsafdsf4565"); // Deve ser rejeitado

confere(regex.IDENT, "Altura1"); // Deve ser aceito
confere(regex.IDENT, "1Altura"); // Deve ser rejeitado

confere(regex.REAL, "-123.908777"); // Deve ser aceito
confere(regex.REAL, "0.17"); // Deve ser aceito
confere(regex.REAL, "0."); // Deve ser rejeitado

confere(regex.INTEIRO, "10"); // Deve ser aceito
confere(regex.INTEIRO, "-10"); // Deve ser aceito
confere(regex.INTEIRO, "-1a0"); // Deve ser rejeitado

```

Groups

Uso para uma única expressão

```

const regex = /^(?<PARAMETRO_1>[a-zA-Z0-9]+)\s*?,\s*?(?<PARAMETRO_2>[a-zA-Z0-9]+)$/;

const str = "mar1cos    ,    pedro22";
let i = 0

const match = str.match(regex);

if (match) {
  const { PARAMETRO_1, PARAMETRO_2 } = match.groups;

  console.log(
    `Parametro1: ${PARAMETRO_1}\nParametro2: ${PARAMETRO_2}`
  );
}

```

```
    );  
  } else {  
    console.log("Nenhuma correspondência encontrada.");  
  }
```

Uso para várias (flag g)

```
const regex = /(?(?<PARAMETRO_1>[a-zA-Z0-9]+)\s*,\s*(?<PARAMETRO_2>[a-zA-Z0-9]+)/g;  
  
const str = "mar1cos, pedro22 john3, do4n maria5, clara6";  
let i = 0  
  
const matches = str.matchAll(regex);  
  
for (const { groups } of matches) {  
  const { PARAMETRO_1, PARAMETRO_2 } = groups;  
  
  console.log(  
    `Correspondência ${++i}:  
    Parametro1: ${PARAMETRO_1}  
    Parametro2: ${PARAMETRO_2}  
  `);  
}
```