Introdução

Caso Simples

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

const palavras = ['ad', 'acccccd', '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 adaccdacccccccd foi aceita (uso do ^$)
const palavras = ['ad', 'acccccd', 'a', 'aaaaaaaaaccccccc', 'c', 'd',
'cccd', 'adaccdacccccccd', '']

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

Palavra vazia

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

const palavras = ['ad', 'acccccd', '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 = {
  BRANCOS: /^\s*$/,
  DIGITOS: /^[0-9]+$/,
  LETRAS: /^[A-Za-z]+$/,
```

```
  FRASE: /^([A-Za-z,!]+(\s?))*$/,
  IDENT: /^[A-Za-z]([A-Za-z]|[0-9])*$/,
  REAL: /^\-?[0-9]+\.[0-9]+$/,
  INTEIRO: /^\-?[0-9]+$/,
  CPF: /^\d{3}\.\d{3}\.\d{3}\-\d{2}$/,
};

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

confere(regex.BRANCOS, " \n\t"); // Deve ser aceito
confere(regex.BRANCOS, " "); // Deve ser aceito
confere(regex.BRANCOS, " 000"); // Deve ser rejeitado

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.FRASE, "Ola mundo, que bom te ver!"); // Deve ser aceito
confere(regex.FRASE, "ASDFEAFdafsafdsf4565"); // Deve ser rejeitado
confere(regex.FRASE, "Ola  mundo"); // 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

confere(regex.CPF, "123.456.789-10"); // Deve ser aceito
confere(regex.CPF, "019.876.543-21"); // Deve ser aceito
confere(regex.CPF, "12345678910"); // Deve ser aceito
confere(regex.CPF, "2312312"); // Deve ser rejeitado
confere(regex.CPF, "fsdafsaf"); // 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}
    `);
}
```