06

Faça o que eu fiz na aula

Para validar um CPF que foi digitado com todos os números repetidos, vamos começar criando um arquivo chamado `validarCPF.js` dentro da pasta `/services`.

Dentro, vamos criar uma função chamada `ehUmCPFComNumerosRepetidos`, que irá receber como parâmetro o cpf com apenas números.

```
const ehUmCFPComNumerosRepetidos = cpf => {
}
```

Agora, precisamos de um `array` com todos os CPF inválidos de números repetidos.

```
  const CPFsINVALIDOS = [
    11111111111,
    22222222222,
    33333333333,
    44444444444,
    55555555555,
    66666666666,
    77777777777,
    88888888888,
    99999999999
  ];
```

Com isso, já podemos verificar se o cpf passado como parâmetro está dentro deste `array`. Caso esteja, isso significa que o cpf digitado pelo usuário é um CPF inválido e vamos retornar o valor `true`, e caso não esteja dentro do `array`, vamos retornar `false`.

```
if (CPFsINVALIDOS.includes(cpf)) {
    return true;
}

return false;
```

A função inteira ficou assim

```
const ehUmCFPComNumerosRepetidos = cpf => {
  const CPFsINVALIDOS = [
    11111111111,
    22222222222,
    33333333333,
    44444444444,
    55555555555,
    66666666666,
    77777777777,
    88888888888,
    99999999999
```

```
  ];

  if (CPFsINVALIDOS.includes(cpf)) {
    return true;
  }

  return false;
};
```

Agora, precisamos ter uma maneira de chamar essa função. Para isso, vamos criar uma outra função que será nossa função principal. Vamos chamá-la de `validarCPF`.

```
export const validarCPF = input => {

}
```

Dentro dela, vamos criar uma variável que irá receber o apenas o números que foram digitados pelo usuário. Apenas os números, porque o usuário pode digitar caracteres especiais como hífens, barras e pontos, por exemplo, porém para nossa validação isso não é interessante. Sendo assim, vamos utilizar do método `replace`, do próprio Javascript, passando um regex para ele. Dessa forma, iremos substituir todos os caracteres especiais digitados por uma string vazia, ficando apenas com os números.

```
  const cpfNumeros = input.value.replace(/\D/g, "");
```

Agora, vamos chamar nossa função `ehUmCPFComNumerosRepetidos` passando o valor da variável `cpfNumeros`. Vamos fazer essa chamada dentro de uma condicional, analisando o retorno da função. Caso ela retorne `true`, iremos atribuir um erro ao `input` de CPF; caso retorne um `false`, iremos remover o erro do `input` e continuar com o cadastro.

```
  if (ehUmCFPComNumerosRepetidos(cpfNumeros)) {
    input.setCustomValidity("Este não é um CPF válido");
    return;
  }

  input.setCustomValidity("");
```

O código final fica desta forma:

```
const ehUmCFPComNumerosRepetidos = cpf => {
  const CPFsINVALIDOS = [
    11111111111,
    22222222222,
    33333333333,
    44444444444,
    55555555555,
    66666666666,
    77777777777,
    88888888888,
    99999999999
  ];
```

```
    if (CPFsINVALIDOS.includes(cpf)) {
      return true;
    }

    return false;
  };

  export const validarCPF = input => {
    const cpfNumeros = input.value
      .replace(/\D/g, "");

    if (ehUmCFPComNumerosRepetidos(cpfNumeros)) {
      input.setCustomValidity("Este não é um CPF válido");
      return;
    }

    input.setCustomValidity("");
  }
```