Javascript

Condicionais e Repetições

O comando if

- Javascript possui o comando if cuja sintaxe se encontra nos grupos de comandos abaixo:

```
const idade = 20;
let classificacao = "Criança";
if (idade >= 18) {
    classificacao = "Adulto";
}
```

```
const idade = 20;
let classificacao;
if (idade < 13) {
    classificacao = "Criança";
} else if (idade < 18) {
    classificacao =
"Adolescente";
} else if (idade < 65) {
    classificacao = "Adulto";
} else {
    classificacao = "Idoso";
}
```

O operador ?

- Quando o valor de uma expressão depende de uma condição, é possível utilizar o operador ?

```
const idade = 20;
let classificacao = idade < 18 ? "Criança" : "Adulto";
```

- Se a condição que vem antes do operador ? resultar em verdadeiro, o valor que vem após o operador ? será retornado. Caso contrário o valor que vem após o : será retornado.

O operador ??

- O operador ?? retorna o operando à esquerda se o valor for diferente de null e de undefined. Caso contrário retorna o valor à direita do operador.

```
const logradouro = "Rua X";
const numero = null;
const endereco = logradouro + ", " + (numero ?? "Sem Número");
```

```
const logradouro = "Rua X";
const numero = 123;
const endereco = logradouro + ", " + (numero ?? "Sem Número");
```

O comando switch

- O operador switch também pode ser utilizado em Javascript. O comando case pode receber expressões e não somente constantes.

```
const trimestre = 3;
let semestre;
switch(trimestre) {
    case 1: case 1 + 1:
        semestre = 1;
        break;
    case 3: case 4:
        semestre = 2;
        break;
    default:
        semestre = 0;
}
```

O comando while

- O comando while executa o seu bloco de código enquanto a sua condição for verdadeira.

```
let numero = 0;


while (numero < 10) {
    console.log(numero); //Imprime o número no console
    numero++;
}
```

O comando do...while

- O comando do...while executa o seu bloco de código enquanto a sua condição for verdadeira, porém o teste da condição só ocorre após a primeira execução de seu bloco de código.

```
let numero = 0;


do {
    console.log(numero); //Imprime o número no console
    numero++;
} while (numero < 10);
```

O comando for

- O comando for executa o seu bloco de código enquanto a sua condição for verdadeira, porém este comando permite que a inicialização, o teste e a atualização sejam feitas em uma mesma linha.

```
for (let numero = 0; numero < 10; numero++) {
    console.log(numero); //Imprime o número no console
}
```

O saindo de laços

- É possível sair de qualquer laço utilizando para isso o comando break. Veja abaixo o código que calcula se um número é primo:

```
const numero = 29;
let i;
for (i = 2; i < numero; i++) {
    if (numero % i == 0) {
        break;
    }
}
console.log(numero, (i != numero ? "não" : "") + " é primo!");
```

O saltando execuções de bloco em laços

- É possível saltar do bloco para o teste de qualquer laço utilizando para isso o comando continue. Veja abaixo o código que imprime os números pares entre 1 e 10:

```
for (let i = 1; i <= 10; i++) {
    if (i % 2 != 0) {
        continue;
    }
    console.log(i);
}
```

Nomeando laços

- Note que no o código abaixo o comando break sai do laço mais interno, ou seja, do laço cujo bloco o comando break está inserido.

```
for (let i = 0; i < 5; i++) {
    for (let j = 0; j < 5; j++) {
        if (j == 2) {
            break;
        }
        console.log(i, j);
    }
}
```

Nomeando laços

- É possível nomear um comando de repetição de forma que seja possível indicar de qual laço se deseja sair. Veja o exemplo abaixo:

```
primeiroLaco: for (let i = 0; i < 5; i++) {
    for (let j = 0; j < 5; j++) {
        if (j == 2) {
            break primeiroLaco;
        }
        console.log(i, j);
    }
}
```

Fim de material

Dúvidas?