Unidade III: Estruturas de repetição soulcode.com

**1 Estruturas de repetição**

**1.1 While-loop**

O while é um comando usado para fazer um laço (loop), isto é, repetir um trecho de código algumas vezes. A ideia é que esse trecho de código seja repetido enquanto uma determinada condição permanecer verdadeira.

Listagem 1: Exemplo de while

1 let idade = 15;

2

3 while (idade < 18) {

4 console.log(idade);

5 idade = idade + 1; // ou idade++

6 }

O trecho dentro do bloco do while será executado até o momento em que a condição idade < 18 passe a ser falsa. E isso ocorrerá exatamente no instante em que idade == 18, o que não o fará imprimir 18.

Listagem 2: Exemplo de while

1 let i = 0;

2

3 while (i < 10) {

4 console.log(i);

5 i = i + 1; // ou i++

6 }

**1.2 For-loop**

Outro comando de loop extremamente utilizado é o for. A ideia é a mesma do while: fazer um trecho de código ser repetido, enquanto uma condição continuar verdadeira.

Mas, além disso, o for isola também um espaço para inicialização de va riáveis e o modificador dessas variáveis. Isso faz com que as variáveis relaci onadas ao loop fiquem mais legíveis:

Listagem 3: Sintaxe do for

1 for (inicializacao; condicao; incremento) {

2 // codigo

3 }

Um exemplo é:

Listagem 4: Sintaxe do for

1 for (let i = 0; i < 10; i++) {

2 console.log("ola!");

3 }

4

5 // Repare que esse for poderia ser trocado por:

1

Unidade III: Estruturas de repetição soulcode.com

6 let i = 0;

7

8 while (i < 10) {

9 console.log("ola!");

10 i++;

11 }

Porém, o código do for indica claramente que a variável i serve, em especial, para controlara quantidade de laços executados. Quando usar o for? Quando usar o while? Depende do gosto e da ocasião.

**1.3 Controlando loops**

Apesar de termos condições booleanas nos nossos laços, em algum momento, podemos decidir parar o loop por algum motivo especial sem que o resto do laço seja executado.

Listagem 5: Exemplo de break

1 for (let i = x; i < y; i++) {

2 if (i % 19 == 0) {

3 console.log("Achei um numero divisivel por 19 entre x e y"); 4 break;

5 }

6 }

O código acima percorrerá os números de x a y e irá parar quando encontrar um número divisível por 19, uma vez que foi utilizada a palavra-chave break. Da mesma maneira, é possível obrigar o loop a executar o próximo laço. Para isso, usamos a palavra-chave continue.

Listagem 6: Exemplo de continue

1 for (let i = 0; i < 100; i++) {

2 if (i > 50 && i < 60) {

3 continue;

4 }

5 console.log(i);

6 }

O código acima não imprimirá alguns números. (Quais exatamente?)

**1.4 Escopo de variáveis**

O escopo é a vida de uma variável em Javascript, representa os limites de uma variável, ou seja, onde ela poderá ser acessada (onde ela existir). Agora que aprendemos mais sobre controle de fluxos (if/else, switch-case, while e for), podemos entender o impacto do escopo das variáveis. Para declarar uma variável temos três formas diferentes: **let**, **const** e **var**. As duas primeiras formas tem escopo delimitado, porém, declarar variáveis com o **var** nos trás a ideia de hoisting abordada nas unidades anteriores. Ob serve o seguinte exemplo:

2

Unidade III: Estruturas de repetição soulcode.com

Listagem 7: Efeito de hoisting

1 if(true) {

2 var nome = "John";

3 }

4

5 console.log(nome);

6 // A variavel 'nome' existe mesmo fora do if

7 // Teste o codigo acima com let/const

Quaisquer escopos, seja de loops ou condicionais, caso seja utilizado **var**, o escopo da variável será "elevado"para o escopo mais externo. Nesse caso do exemplo acima, o escopo é o principal do arquivo, se estivesse dentro de uma função este if, o escopo da variável seria elevado para a própria função (veremos adiante).

3