**CONTROLE DE FLUXO DE REPETIÇÃO**

**ESTRUTURA FOR**

A estrutura for é, possivelmente, a estrutura mais utilizada para as soluções de looping de repetição. Segue a sintaxe: Iniciar uma variável de controle, realizar uma lógica de decisão e alterar o valor da variável de controle decrementando-a ou incrementando-a.

A variável de controle sempre inicia seu valor no início do looping e tem seu valor modificado no início do próximo looping. Perceba o comportamento abaixo:

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

console.log(i)

}

O que ocorrerá como saída é a impressão do número 0 até 9, isso porque a iteração do *i++* se ocorre logo no início do primeiro looping, faria com que a primeira impressão de *i* no console já fosse com o valor incrementado, se assim fosse, o que seria impresso seria o número 1 até 10, ocorrendo em uma falha, pois a variável *i*, pelo o que foi estabelecido na lógica de decisão, não pode ser igual ou superior a 10.

A lógica de decisão em um looping se dá através de uma comparação da variável de controle com um valor que sugere um "todo". Esse "todo" pode ser expresso definido como um valor inteiro em uma variável previamente declarada ou escrita diretamente na estrutura do for. Uma outra forma muito comum de representarmos o "todo" seria através de uma estrutura de dados como um array (que ainda será aprendido durante o curso dessa disciplina), mas perceba como seria:

var cestaDeFrutas= ["banana","maça","uva","pêra","jaca", "laranja"]

for (passo=0; passo<cestaDeFrutas.length; passo++){

console.log(cestaDeFrutas[passo])

}

**ESTRUTURA WHILE E DO..WHILE**

A principal diferença entra essas estruturas e a estrutura for, está na possibilidade de se dispensar a variável de controle. Na estrutura *for* a variável de controle existe para controlar a quantidade de looping que será realizada. Nessas novas estruturas o controle está em avaliar se o “todo” já está no estado em que se espera. Como em um quarto bagunçado, iremos arrumar o quarto bagunçado enquanto houver bagunça. Veja em um outro exemplo:

var capacidadeDeArmazenamentoDoBau = 50;

while (capacidadeDeArmazenamentoDoBau > 0) {

capacidadeDeArmazenamentoDoBau--

// Executa a quantidade de vezes necessárias para esvaziar o baú

console.log('Retire uma bolinha');

}

A estrutura do...while tem a única diferença para a anterior no momento em que se executa a ação do looping. Essa estrutura executa a ação e depois testa o estado do “todo”. Perceba:

do{

capacidadeDeArmazenamentoDoBau --

// Executa a quantidade de vezes necessárias para esvaziar o baú

console.log('Retire uma bolinha');

}while (capacidadeDeArmazenamentoDoBau > 0)