4.0 Caixa de Diálogo

Um recurso interessante de JavaScript é a possibilidade de criar caixas de diálogo simples, que podem ser muito informativas aos usuários que a visualizam.

Essas caixas de diálogo podem ser de alerta, de confirmação ou de prompt de entrada. Todas elas são chamadas de forma simples e intuitiva por uma função.

4.1 Alert

As caixas de diálogo de alerta são simples e informativas. Elas, geralmente, são utilizadas em validação de formulários ou bloqueio de ações.

Sua função é mostrar apenas uma mensagem com um botão de confirmação para que esta seja fechada.

Para chamar esta caixa de diálogo usamos a função **alert()**. Esta função recebe como parâmetro uma string que será a mensagem a ser exibida. Vejamos o código abaixo:

```
<script type="text/javascript">
    alert ("Esta é uma caixa de diálogo ALERT do javascript!")
</script>
```

Em caixas de diálogo há a possibilidade de controlar o fluxo de texto usando `\n` para a quebra de linhas.

```
<script type="text/javascript">
    alert ("javascript \n Caixa de dialogo \n Alert")
</script>
```

4.2 Prompt

A caixa de diálogo de prompt nos possibilita requerer uma entrada ao usuário apesar de não ser tão útil, pois esse recurso pode facilmente ser substituído por um campo de texto feito em HTML.

Para chamarmos esta caixa de diálogo, usamos a função `prompt()` que recebe uma string como parâmetro. Esse parâmetro será a mensagem a ser exibida dentro da caixa de diálogo.

A caixa de diálogo de prompt possui três elementos: um campo input para texto, um botão OK e outro CANCELAR.

A função `prompt ()` sempre irá retornar um valor, ou seja, podemos gravar o resultado da função em uma variável ou algo assim. Se clicarmos no botão OK, o valor a ser retornado será o que estiver escrito no campo de texto, mesmo se ele estiver vazio. Se clicarmos em CANCELAR, o valor retornado será **null**.

Abaixo criamos um exemplo no qual exige que o usuário digite o nome dele. Para isso, colocamos o prompt dentro de uma estrutura de repetição *while* que tem a seguinte condição: se o resultado for null (ou seja, se o usuário clicar em cancelar), ou então, se o resultado for vazio (ou seja, se o usuário não digitar nada e clicar no OK), neste caso, deve-se executar a repetição.

Dessa forma nos asseguramos que o usuário sempre irá digitar alguma coisa dentro da caixa de diálogo.

```
<script type="text/javascript">
    var nome;
    do {
        nome = prompt ("Qual é o seu nome?");
    }while (nome == null || nome == "");
    alert ("Seu nome é: "+nome);
</script>
```

4.3 Confirm

A caixa de diálogo de confirmação é chamada pela função `confirm()` e tem apenas dois botões: um OK e outro CANCELAR. Assim como a função prompt(), a função confirm() também retorna um valor que pode ser true (verdadeiro) ou false (falso).

Como confirm() retorna um valor booleano, isso o torna ideal para ser usado com uma estrutura seletiva *if*. Por exemplo, podemos usar a caixa de diálogo de confirmação antes de redirecionarmos uma página para executar uma rotina para apagar algum registro do banco de dados.

No exemplo abaixo, não iremos tão profundamente quanto o cenário acima, pois envolve mais do que simples JavaScript. Aqui, apenas iremos demonstrar o resultado do clique em algum dos dois botões.

```
<script type="text/javascript">
    decisao = confirm("Clique em um botão!");
    if (decisao) {
        alert ("Você clicou no botão OK, \n"+
                "porque foi retornado o valor: "+decisao);
    } else{
        alert ("Você clicou no botão CANCELAR, \n"+
                "porque foi retornado o valor: "+decisao);
    };
</script>
```

Exercícios Propostos

Usando a instrução alert, faça um algoritmo que exibe para o usuário a mensagem: “Bom dia usuário.”.

Faça um algoritmo que solicite ao usuário que digite seu nome, após o usuário dar entrada da informação exiba a seguinte mensagem para o mesmo: “Bom dia nome_usuario.”. Onde há nome_usuario será

colocado o nome que o usuário digitou.

Faça um algoritmo que solicite o ano que o usuário nasceu e depois o ano atual, seu código irá pegar essas informações e calcular a idade do usuário e exibi-la na tela usando alert().

Faça um algoritmo que receba dois números inteiros, digitados pelo usuário, e exiba em um único alert() o resultado da soma, subtração, multiplicação e divisão entre os dois números digitados.

Faça um algoritmo que peça sua amado(a) em casamento ou namoro, utilize a instrução confirm(). Caso ele(a) aceite exiba a mensagem "Eu aceito", caso contrário exiba a mensagem "Desculpa, não posso aceitar".

Faça um algoritmo de validação de login. O usuário irá digitar o login e depois a senha. Se o login digitado é igual a "root" e a senha "qwe123", então exiba a mensagem "Acesso permitido", senão exiba "Login/Senha inválidos";

Escreva um algoritmo que calcula quantos meses um funcionário trabalhou para uma empresa. Solicite o dia, mês e ano que um funcionário foi contratado e depois o dia, mês e ano que ele foi demitido. Calcule quantos meses esse funcionário trabalhou nessa empresa.

Escreva um algoritmo que receba a idade do usuário e se a idade for maior o u igual a 18 exiba a mensagem "Sou um adulto", senão exiba a mensagem "Sou uma Criança/Adolescente".

Escreva um algoritmo que solicite ao usuário uma lista de compras para se fazer no supermercado. Essa lista terá 10 produtos e após todos serem digitados exiba a lista em um só alert() utilizando essa formatação:

1 – arroz

2 – feijão

3 – macarrão

4 – farofa

Escreva um algoritmo que receba 3 valores inteiros que representam a tamanho dos lados de um triângulo. Usando o operador ternário exiba a mensagem "Sou equilátero" se todos os lados forem iguais, "Sou isósceles" se dois lados forem iguais e "Sou escaleno ", se todos os lados forem diferentes.

5.0 Estruturas de controle e repetição

Objetivos

Mostra estruturas de controle e sua aplicação prática em JavaScript; Definir qual a principal finalidade dessas estruturas; Mostrar exemplos em sua sintaxe;.

As estruturas que veremos a seguir são comuns para as linguagens de programação imperativas, bastando descrever a sintaxe de cada uma delas resumindo o funcionamento. Independente do JavaScript, boa parte das outras linguagens de programação tem estruturas bem semelhantes, mudando apenas algumas sintaxes.

5.1 Blocos de controle

Um bloco consiste de vários comandos agrupados com o objetivo de relacioná-los com determinado comando ou função. Em comandos como if, for, while, switch e em declarações de funções blocos podem ser utilizados para permitir que um comando faça parte do contexto desejado. Blocos em JavaScript são delimitados pelos caracteres "{" e "}". A utilização dos delimitadores de bloco em uma parte qualquer do código não relacionada com os comandos citados ou funções não produzirá efeito algum, e será tratada normalmente pelo interpretador. Outro detalhe importante: usar as estruturas de controle sem blocos delimitadores faz com que somente o próximo comando venha ter ligação com a estrutura. Observe os exemplos:

```
<script type="text/javascript">
    var a = 0;
    if (a>2)
        alert("comando 1");
        alert("comando 2");
</script>
```

Observe que temos um comando IF, onde é passado a ele uma expressão booleana que retorna verdadeiro ou falso.

O resultado da expressão é FALSE (falso), pois 0 não é maior que 2, fazendo com que o IF não execute o **echo** com "comando1". Somente o segundo **echo** é executado, pois não pertence ao IF declarado.

Mas se quisermos que mais de um comando pertença a estrutura de controle, será usado blocos de comandos ({comando;}), onde através deles podemos delimitar e organizar os códigos.

```
<script type="text/javascript">
    var a = 0;
    if (a>2) {
        alert("comando 1");
        alert("comando 2");
    }
</script>
```

No código ao lado, temos um bloco onde inserimos dois comandos. Observe que eles não serão executados, pois a expressão booleana passada para o IF é falsa.

5.2 IF e ELSE

Essa estrutura condicional está entre as mais usadas na programação. Sua finalidade é induzir um desvio condicional, ou seja, um desvio na execução natural do programa. Caso a condição dada pela expressão seja satisfeita, então serão executadas a instruções do bloco de comando. Caso a condição não seja satisfeita, o bloco de comando será simplesmente ignorado. Em lógica de programação é o que usamos como “SE (expressão) ENTÃO {comando:}”.

–

```
if (expressão)
    comando;
if
(expressão){
    comando1;
    comando2
```

Sintaxe:

```
<script type="text/javascript">
    var idade = 16;
    if (idade >18) {
        var texto = "Maior idade";
    };
    alert(texto);
</script>
```

Exemplo:

Caso a condição não seja satisfatória (FALSE), podemos atribuir outro comando pertencente ao IF chamado ELSE, como se fosse a estrutura SENÃO em lógica de programação.

Sintaxe:

```
if
(expressão)
    comando;
eles

    comando

; if

(expressão)

{
    comand
    o1;
```

Exemplo:

```
<script type="text/javascript">
    var idade = 16;
    if (idade>18) {
        var texto = "Maior de idade";
    }else{
        var texto = "Menor de idade";
    }
    alert(texto);
</script>
```

Nesse exemplo temos uma expressão booleana onde retorna falso, com isso o IF não executa, passando a execução para o eles, que por sua vez executa e atribuí o valor “menor idade” a variável texto.

Em determinadas situações é necessário fazer mais de um teste, e executar condicionalmente diversos comandos ou blocos de comandos. Isso é o que podemos chamar de “If's encadeados”, onde usamos a estrutura **ELSE IF** .Para facilitar o entendimento de uma estrutura do tipo:

```
if (expressao1)
    comando1;

    else
if (expressao2)

    comando2;
else

    if (expressao3)
        comando3;
    else comando4;
```

Sintaxe:

```
<script type="text/javascript">
    var idade = 16;
    if (idade>18) {
        var texto = "Maior de idade";
    }else if (idade < 12){
        var texto = "Criança";
    }else{
        var texto = "Adolescente";
    }
    alert(texto);
</script>
```

Exemplo:

Exercício rápido:

1°) Faça uma script em JavaScript que possua 4 notas de um aluno (cada uma em uma variável). Depois calcule e imprima a média aritmética das notas e a mensagem de aprovado para média superior ou igual a 7.0 ou a mensagem de reprovado para média inferior a 7.0.

2°) Faça um script em JavaScript que receba a idade de um nadador (representada por uma variável chamada “idade”) e imprima a sua categoria seguindo as regras:

Categoria	Idade
Infantil A	5 - 7 anos

Infantil B	8 - 10 anos
Juvenil A	11- 13 anos
Juvenil B	14- 17 anos

5.3 Atribuição condicional (ternário)

Como já vimos exemplos de atribuição condicionais (ternários), podemos defini-los usando a sintaxe: (Expressão booleana) ? (executa caso verdadeiro) : (executa caso falso);

Isso se aplica quando queremos uma estrutura resumida, onde podemos ter um resultado mais direto, como por exemplo, atribuir um valor a uma variável dependendo de uma expressão. Observe o exemplo abaixo onde envolvemos uma variável do tipo string, porém o valor atribuído a essa variável deverá ser de acordo com o valor da idade:

```
<script type="text/javascript">
    var idade = 16;
    var texto = idade > 18 ? "Maior de idade" :"Menor de idade";
    alert(texto);
</script>
```

É uma estrutura parecida com IF e ELSE, onde dependendo da expressão booleana podemos executar um bloco ou não.

Exercício rápido:

1°) Faça uma script em JavaScript que receba um número representado por uma variável. Verifique se este número é par ou ímpar e imprima a mensagem.
2°) Crie outro script baseando em um seguro de vida com as seguintes regras:

Idade:	**Grupo de Risco :**
18 a 24	- Baixo
25 a 40	- Médio
41 a70	- Alto

5.4 SWITCH

Observe que quando temos muitos “if's encadeados” estamos criando uma estrutura que não é considerada uma boa prática de programação. Para resolver esse problema temos uma estrutura onde sua funcionalidade é semelhante ao **ELSE IF.** O comando SWITCH é uma estrutura que simula uma bateria de teste sobre uma variável. Frequentemente é **necessário** comparar a mesma variável com valores diferentes e executar uma ação especifica em cada um desses valores.


```
switch(expressão){

    case "valor
      1":
      comandos;

    case "valor
      1":
      comandos;
    case "valor
      1":
      comandos;
    case "valor
```

Sintaxe:

```
<script type="text/javascript">
    var numero = 2;
    switch(numero){
        case 1:
            alert("opção 1");
        case 2:
            alert("opção 2");
        case 3:
            alert("opção 3");
        case 4:
            alert("opção 4");
        case 5:
            alert("opção 5");
    }
</script>
```

Exemplo:

Resultado: opção 2: opção 3:opção 4:opção 5:

Nesse exemplo temos o número = 2, onde o switch compara com os case's o valor recebido, o bloco que é executado é do segundo case, porém, os demais também são executados para que tenhamos um resultado satisfatório temos que usar em cada case um comando chamado **break**. No qual tem a função de para o bloco de execução.

5.5 SWITCH com BREAK

Break é uma instrução (comando) passada quando queremos parar o fluxo da execução de um programa. Em JavaScript, ele tem a mesma função que é “abortar” o bloco de código correspondente.

Observe o mesmo exemplo com o uso de break:

Temos agora como resultado “opção 2:”. O comando break fez com que os demais case's abaixo do 'case 2' não sejam executados.

```
<script type="text/javascript">
    var numero = 2;
    switch(numero){
        case 1:
            alert("opção 1");
            break;
        case 2:
            alert("opção 2");
            break;
        case 3:
            alert("opção 3");
            break;
        case 4:
            alert("opção 4");
            break;
        case 5:
            alert("opção 5");
            break;
    }
</script>
```

Obs.: Além de números podemos também comparar outros tipos como string, pontos flutuantes e inteiros, veja um exemplo abaixo:

```
<script type="text/javascript">
    var numero = "opc 2";
    switch(numero){
        case "opc 1":
            alert("opção 1");
            break;
        case "opc 2":
            alert("opção 2");
            break;
    }
</script>
```

Mas o que acontece se não tivermos um valor que seja satisfatório aos casos existentes no switch? A resposta é bem simples, nenhum dos blocos seriam executados, porém temos um comando onde determinamos uma opção padrão caso nenhuma das outras venha ter resultado que satisfaça a expressão passada para o switch chamada default (padrão).

```
<script type="text/javascript">
    var numero = "opc 3";
    switch(numero){
        case "opc 1":
            alert("opção 1");
            break;
        case "opc 2":
            alert("opção 2");
            break;
        default:
            alert("opção inválida");
    }
</script>
```

Veja um exemplo:

Resultado: opção inválida

A instrução passada não condiz com nenhum dos casos existentes. Por esse motivo o bloco pertencente ao comando default será executado.

O comando default pode ser inserido em qualquer lugar dentro do switch, porém caso isso aconteça, o uso do comando break deve ser adicionado para evitar que os case's abaixo

A partir de agora trabalharemos as estruturas de repetição. Elas muito utilizadas nas linguagens de programação.

1°) Faça um script em JavaScript usando switch, onde receba uma variável e mostre as seguintes opções: 1 - módulo.
2 - somar.
3 - subtrair.
4 - multiplicar.

5.6 WHILE

O WHILE é uma estrutura de controle similar ao IF, onde possui uma condição para executar um bloco de comandos. A diferença primordial é que o WHILE estabelece um laço de repetição, ou seja, o bloco de comandos será executado repetitivamente enquanto a condição passada for verdadeira. Esse comando pode ser interpretado como “ENQUANTO (expressão) FAÇA { comandos...}”.

Sintaxe:

```
while
    (expressão){
    comandos;
```

Quando estamos usando um laço de repetição, podemos determinar quantas vezes ele deve ou não se repetir. Para trabalharmos com essa contagem de quantas vezes o laço deve se repetir, usaremos incremento ou decremento de uma variável conforme vimos no capitulo de operadores em JavaScript. Observe o exemplo abaixo:

```
<script type="text/javascript">
    var a = 1;
    while(a < 10){
        alert(a);
        a++;  //incrementa a variável
    }
</script>
```

Sequência do resultado em caixa de diálogo: 1 2 3 4 5 6 7 8 9

Nesse exemplo criamos um laço de repetição que tem como condição a < 10, a cada laço é executado um incremento na variável a, fazendo com que o seu valor aumente até a condição não ser mais satisfatória.

Dicas: Tenha cuidado quando estiver trabalhando com loop's (laço de repetição), pois caso a expressão passada esteja errada, pode ocasionar em um loop infinito fazendo com que o bloco de código se repita infinitamente. Isso pode ocasionar um travamento do navegador ou até mesmo do próprio servidor WEB.

Vamos ver agora um exemplo em que o laço se repete de forma automática, onde quem determina o loop são propriedades do JavaScript e não um número determinado pelo programador.

A propriedade length é chamada logo após uma variável de texto e retorna a quantidade de caracteres incluindo também os espaços em branco. Ele poderia ser aplicado diretamente no alert, mas no exemplo, ele determina a quantidade de loop's.

```
<script type="text/javascript">
    var a = 1;
    var texto = "Projeto e-Jovem";
    //length conta o tamanho da string
    while(a < texto.length){
        a++;  //incrementa a variável
    }
    alert("O texto possui "+a+" caracteres");
</script>
```

Resultado: a frase possui 15 caracteres

Exercício rápido:

1°) Faça um script que conte de 1 até 100.
2°) Faça um script que imprima na tela números de 3 em 3 iniciando com 0 até 90, ex: 0,3,6,9...

5.7 DO...WHILE

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a expressão é testada ao final do bloco de comandos. O laço do...while possui apenas uma sintaxe que é a seguinte:

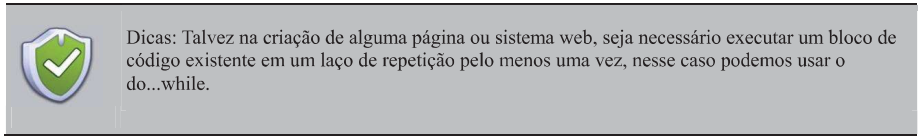
–

```
do {
    comando
    ;
. . .
    comando;
} while
```

Sintaxe:

```
<script type="text/javascript">
    var a = 1;
    do{
        alert(a);
        a++;
    }while(a<10);
</script>
```

Exemplo:



Dicas: Talvez na criação de alguma página ou sistema web, seja necessário executar um bloco de código existente em um laço de repetição pelo menos uma vez, nesse caso podemos usar o do...while.

Sequência do resultado em caixa de diálogo: 1 2 3 4 5 6 7 8 9

Exercício rápido:

1°) Faça um script que conte de -1 até -100 usando "do while".

2°) Faça um script que imprima na tela somente números pares de 2 até 20 com do while.

5.8 FOR

Outra estrutura semelhante ao while é o for, onde tem a finalidade de estabelecer um laço de repetição em um contador. Sua estrutura é controlada por um bloco de três comandos que estabelecem uma contagem, ou seja, o bloco de comandos será executado determinado número de vezes.

Sintaxe:

```
for( inicialização; condição;
incremento ){ comandos;
}
```

Parâmetros	Descrição
Inicialização	Parte do for que é executado somente uma vez, usado para inicializar uma variável.
Condição	Parte do for onde é declarada uma expressão booleana.
Incremento	Parte do for que é executado a cada interação do laço.

Lembrando que o loop do for é executado enquanto a condição retornar expressão booleana verdadeira. Outro detalhe importante é que podermos executar o incremento a cada laço, onde possibilitamos adicionar uma variável ou mais. Mostraremos agora um exemplo fazendo um comparativo entre a estrutura de repetição do while e também a do for de forma prática.

Exemplo:

```
<script type="text/javascript">
// COM WHILE
    var a = 1;
    while(a<10){
        alert(a);
        a++;
    }
</script>
```

WHILE

```
<script type="text/javascript">
// COM FOR
    for(a=1;a<10;a++){
        alert(a);
    }
</script>
```

FOR

*Ambos exemplos geram a mesma sequência em caixa de diálogo com o resultado:*1 2 3 4 5 6 7 8 9

O **for** não precisa ter necessariamente todas as expressões na sua estrutura, com isso podemos criar um

exemplo de **for** onde suas expressões são declaradas externamente.

```
<script type="text/javascript">
    a = 1;
    for(;a<10;){
        alert(a);
        a++;
    }
</script>
```

Observe nesse exemplo uma proximidade muito grande do comando while. Apesar de ser funcional, não é uma boa prática de programação utilizar desta forma.

A estrutura **"while"** e **"do while"** normalmente é usado quando o programador não sabe quantos loops o sistema irá realizar, já o **"for"** informamos explicitamente qual será o ponto de partida, o seu limite, e valor de incremento ou decremento.

Exercício rápido:

1°) Faça um script que receba duas variáveis "a" e "b", logo após imprima os números de intervalos entre eles com o uso de "for".ex: a=5 ,b = 11, imprime : 5,6,7,8,9,10,11.

5.9 FOREACH

O **foreach** é um laço de repetição para interação em array's ou matrizes, o qual estudaremos com mais detalhes no próximo capítulo. Trata-se de um **for** mais simplificado que compõe um vetor ou matriz em cada um de seus elementos por meio de sua cláusula **IN.**

```
vetor = new Array();
foreach(incide in vetor){
  comandos;
}
```

Sintaxe:

```
<script type="text/javascript">
    var texto = new Array("HTML","CSS","javascript");

    for (var indice in texto) {
        alert(texto[indice]);
    };
</script>
```

Exemplo:

Resultado: *HTML CSS JAVASRIPT*

Veremos adiante que um array é uma variável composta por vários elementos. No caso do exemplo anterior, esses elementos são nomes de cursos. A finalidade do foreach é justamente a cada laço, pegar cada índice desses valores e atribuir a uma variável, até que tenha percorrido todo array e assim, finalizar o laço. Podemos saber em qual posição o elemento se encontra no array, para isso basta imprimir o índice.

Observe o exemplo:

```
<script type="text/javascript">
    var texto = new Array("HTML","CSS","javascript");

    for (var indice in texto) {
        alert(indice+" - "+texto[indice]);
    };
</script>
```

Resultado:

0-HTML
1-CSS
2-JAVASCRIPT

Nesse exemplo observamos que cada elemento do array possui um índice (chave), imprimindo na tela o número da posição e o valor guardado.

5.10 BREAK.

Outro comando importante é o break, usado para abortar (parar) qualquer execução de comandos como SWITCH, WHILE, FOR, FOREACH, ou qualquer outra estrutura de controle. Ao encontrar um break dentro de um desses laços, o interpretador JavaScript interrompe imediatamente a execução do laço, seguindo normalmente o fluxo do script.

Sintaxe:

```
while....
   for....
      break <quantidades de
      níveis>;
```

Vamos ver um exemplo com o uso de **break** dentro de um laço de repetição (no caso o **for**), onde criamos

```
<script type="text/javascript">
    var a = 1;
    for ( ; ; ) { //Loop Infinito
        document.write("<p>"+a+"</p>");
        if (a===10) {
            break;   //Para o laço
        };
        a++;  // Incremento
    };
</script>
```

um laço infinito, porém colocamos um if com a condição de parar o laço através do **break**. Observe:

Podemos notar nesse exemplo a criação de um laço(loop) infinito, que ocorre quando tiramos a condição

do **for**, ou atribuímos "for(; true ;)", porém a condição fica na responsabilidade do if, quando o valor de a e igual a 10, faz com que o if execute o **break**, fazendo com que o laço pare de funcionar.

5.11 CONTINUE

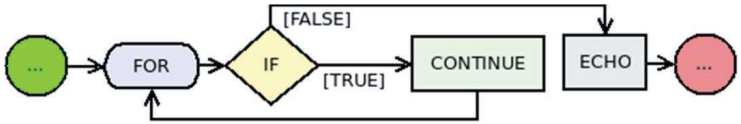
A instrução **continue**, quando executada em um bloco de comandos for/while, ignora as instruções restantes até o fechamento em "}". Dessa forma, o programa segue para a próxima verificação da condição de entrada do laço de repetição, funciona de maneira semelhante ao break, com a diferença que o fluxo ao invés de sair do laço volta para o início dele. Veja um exemplo:

```
<script type="text/javascript">
    for ( var a = 0; a<20 ;a++) {
        if( a % 2){
            continue;
        }
        document.write(+a+" ");
        };
</script>
```

***Resultado**:* 0,2,4,6,8,10,12,14,16,18

Podemos observar a seguinte lógica no exemplo acima:

Criamos um laço que tem 20 interações de repetição. Logo após temos um if, onde, quando o resto da divisão por 2 for igual a 0 (número par), o valor booleano será "false". Quando não for igual a 0, significa que a variável a é um número ímpar (ex: 5%2 = 1), então temos um valor booleano "true". Isso significa que o if executa somente quando os números forem ímpares. Adicionamos um continue, que ao executar o if, faz com que volte novamente para o início do for impedindo de alcançar o echo em seguida. Com isso, em vez de mostramos os números impares, imprimimos somente os números pares incluindo o 0. Resumimos que o código só passa adiante quando o if não executa o continue. Fluxograma:



Exercícios Propostos

Qual a principal finalidade de uma estrutura de controle?

Qual a principal finalidade de uma estrutura de repetição?

Crie um código com a uma condição ternária onde receba um valor booleano e de acordo com o valor passado na expressão, deve imprimir "sim" ou "não".

Com o comando IF e ELSE crie um código que determine se uma expressão é verdadeira ou

falsa.

Qual a finalidade da estrutura de controle SWITCH e cite um exemplo onde comparamos uma opção com 4 casos diferente?

Crie um contador de 1 até 20 usando a estrutura de repetição WHILE.

Crie um contador de 1 até 100 usando DO WHILE. EP06.8: Crie um contador de 100 até 1 usando FOR.

Qual a fonalidade de um FOREACH?

Crie um código onde podemos para a execução de um laço infinito com o uso de BREAK.

Como podemos determinar o uso de CONTINUE e qual a sua aplicação prática em JavaScript.

Crie um código com as seguintes características:

Deverá receber um valor inicial e outro final (crie duas variáveis para esse fim).

Como o comado FOR crie um laço onde a contagem e determinada pelo valor inicial e final?

Dentro do for deverá conter um IF e ELSE responsável por compara os valores passado a ele e imprimir os pares e impares. Exemplo:

IF(valor%2==0)

echo valor. "é um número par"; ELSE

echo valor. "é um número ímpar";

Exemplo prático: foi passado o número inicial 8 e o final 15, então o script JavaScpit deverá imprimir o intervalo entre esse número ou seja 8,9,10,11,12,13,14,15, mostrando quais deles são pares e quais são ímpares.