

# 1 Conceitos Iniciais

## 1.1 Definição

PHP é uma linguagem de script que pode ser embutida no HTML (linguagens de script são linguagens que podem ser embutidas em outros programas ou em outras linguagens); A sintaxe do PHP é baseada, em grande parte, nas linguagens C, Java e Perl - incluindo algumas características específicas do próprio PHP; O objetivo da linguagem PHP é possibilitar que os desenvolvedores web codifiquem páginas dinâmicas e de forma rápida; A sigla PHP significa: Hypertext Preprocessor ou Pré-processador de hipertexto; A infra-estrutura da Internet é baseada no modelo cliente x servidor, onde clientes estão conectados a servidores que possuem e fornecem cópias de documentos; O PHP é uma tecnologia "server-side" (do lado do servidor) , onde um servidor faz sua interpretação (do seu código) e retorna como resultado dados que serão exibidos pelo navegador (browser); Portanto, todos os processos, rotinas e funções são processadas no lado servidor, ou seja, o usuário recebe apenas o resultado desse processamento no seu navegador; Para que o PHP funcione corretamente é necessário o servidor Apache, responsável por interpretar o código PHP e dar suporte a Sistemas Gerenciadores de Banco de dados (ex: MySQL, Oracle);

## 1.2 Estrutura Básica

```
1 <?php
2     echo "Hello";
3     print "World!";
4 ?>
```

1. A sequência de caracteres **<?php** indica que um novo trecho de código em linguagem PHP está sendo iniciado. Em outras palavras, tal sequência é utilizada para especificar onde inicia-se o conjunto de instruções que deve ser interpretado pelo servidor Apache.
2. Trecho de código em linguagem PHP que será interpretado. As rotinas **echo** e **print** são utilizadas para apresentar dados no navegador;
3. A sequência de caracteres **?>** é utilizada para especificar onde terminar o trecho de código em linguagem PHP.

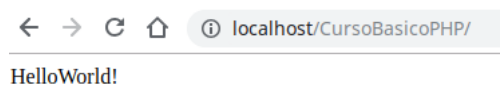


Figura 1: Resultado da Execução

## 1.3 Tipos de Dados

### 1.3.1 Variáveis/Sintaxe

Na linguagem PHP, assim como outras linguagens interpretadas (ex.: Python), não há a necessidade de declarar as variáveis (como acontece nas linguagens C e Java); Sendo assim, a linguagem PHP é denominada fracamente tipada, já que ao não declararmos uma variável não definimos o seu tipo; Por essa característica (não declaração das variáveis), a linguagem PHP faz uso da chamada "tipagem dinâmica"; Na tipagem dinâmica a escolha do tipo da variável ocorre de forma dinâmica no momento que o código está sendo interpretado/executado;

### 1.3.2 Variáveis/Sintaxe (codificação)

#### 1º exemplo

```

1  <?php
2
3      $curso = "Técnico em Informática";
4      $disciplina = "Aplicações para Web II";
5      $turma = 2023;
6
7      echo "[CURSO]: $curso <br>";
8      echo "[DISCIPLINA]: $disciplina <br>";
9      echo "[TURMA]: $turma";
10 ?>

```

1. O nome das variáveis sempre iniciam pelo caractere reservado \$.
2. Variáveis não precisam ser declaradas, sendo criadas no momento que são utilizadas, elas podem receber valores de qualquer tipo (no exemplo, tipos "string" e "inteiro").
3. Obs.: repare que o comando "echo" permite apresentar textos estáticos juntamente com o conteúdo de variáveis e "tags" HTML.

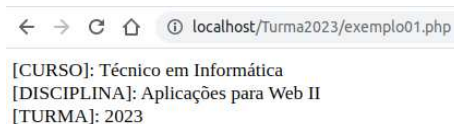


Figura 2: Resultado da Execução

Para a nomeação de variáveis, as dicas a seguir são necessárias:

- Não inicie o nome de uma variável com números;
- Não utilize espaços em brancos;
- Não utilize caracteres especiais, somente underline;
- Crie variáveis com nomes que ajudarão a identificar melhor a mesma;
- Evite utilizar letras maiúsculas.

### 1.3.3 Constantes no PHP

O valor de uma constante jamais poderá ser alterado enquanto estiver sendo executada e para defini-la utilizamos a função define() ou const:

```

1  <?php
2      define("PHP", "Linguagem Open - Source");
3
4      const HTML = "Linguagem de marcação";
5
6      echo PHP; // Linguagem Open - Source
7
8      echo HTML; // Linguagem de marcação
9  ?>

```

### 1.3.4 Como comentar o código no PHP

Para comentarmos o nosso código PHP usamos duas barras ou # para comentários de uma linha, e para comentários de múltiplas linhas usamos /\* \*/, o mesmo usado no CSS. Observe alguns exemplos abaixo:

```
1 <?php
2     echo "Oi, Eu serei visto na sua tela";
3     // Eu não! Sou apenas um comentário.
4
5     echo "Oi, Eu também serei visto por você";
6     # Já eu não serei!
7
8     echo "E eu aqui novamente na sua tela, rs";
9     /* Eu não aparecerei na sua tela novamente
10    pois sou um comentário */
11 ?>
```

### 1.3.5 Operadores Aritméticos no PHP

Os operadores matemáticos disponíveis em PHP são:

- Adição: +
- Subtração: -
- Multiplicação: \*
- Divisão: /
- Módulo: %

### 1.3.6 Operadores de Atribuição no PHP

```
1 <?php
2     $a = 1; // A variável $a é igual a 1
3     $a += 2; // Somamos 2 ao valor da $a;
4     $a -= 2; // Subtraímos 2 ao valor da variável $a;
5     $a *= 2; // Multiplicamos o valor da variável $a por 2;
6     $a /= 2; // Dividimos o valor da variável $a por 2.
7
8     $a = 1;
9     echo ++$a; // Incrementamos 1 e retornamos o valor
10    echo $a++; // Retornamos o valor e incrementamos 1
11    echo --$a; // Decrementamos 1 e retornamos o valor
12    echo $a--; // Retornamos o valor e decrementamos 1
13 ?>
```

### 1.3.7 Operadores Relacionais

- Igual: ==
- Idêntico: ===
- Diferente: != ou <>
- Menor que: <
- Maior que: >
- Menor ou igual: <=

- Maior ou igual: `>=`

É importante lembrar que `==` não checa o tipo da variável, apenas seu valor. Já o `===` checa tanto o valor da variável quanto o seu tipo.

```
1 <?php
2     $a = 2;
3     $b = '2';
4 ?>
```

### 1.3.8 Operadores Lógicos

- `$a and $b`: ambos verdadeiros;
- `$a or $b`: ambos ou apenas um seja verdadeiro;
- `$a xor $b`: apenas A ou B forem verdadeiros, mas não os dois;
- `!$a`: verdadeiro se A for falso;
- `$a && $b`: ambos verdadeiros;
- `$a || $b`: ambos ou apenas um seja verdadeiro;

### 1.3.9 Estrutura de Decisão if/else

```
1 <?php
2     $a = 2;
3     $b = '2';
4
5     if($a == $b)
6         echo "igual verdadeiro <br>";
7     else
8         echo "igual falso <br>";
9
10    if($a === $b)
11        echo "idêntico verdadeiro <br>";
12    else
13        echo "idêntico falso <br>";
14 ?>
```

### 1.3.10 Estruturas de Decisão (elseif/switch)

```
1 <?php
2 $nome = 'Eddie';
3
4 if($nome == 'Richard Kreisson') {
5     echo 'E ae Richard Kreisson!';
6 } elseif ($nome == 'Epaminondas Quintana') {
7     echo 'E ae Epaminondas Quintana!';
8 } elseif ($nome == 'Eddie') {
9     echo 'Up the Irons!!!';
10 } else {
11     echo "E ae $nome!";
12 }
13
14 switch($nome) {
15     case 'Richard Kreisson':
16         echo 'E ai Richard Kreisson!';
17         break;
```

```
18
19     case 'Epaminondas Quintana':
20         echo 'E ai Epaminondas Quintana!';
21         break;
22
23     case 'Eddie':
24         echo 'Up the Irons!!!';
25         break;
26
27     default:
28         echo 'Qual é o seu nome?';
29         break;
30 }
31 ?>
```

### 1.3.11 Operador Ternário no PHP

No PHP existe uma forma mais curta de criar condições através do Operador Ternário:

```
1 <?php
2     $number1 = 1;
3     $number2 = 2;
4
5     $ternario = ($number2 > $number1) ?
6     'Número 2 é maior que número 1' : 'Número 2 não é maior que número 1';
7
8     echo $ternario; // Número 2 é maior que número 1
9 ?>
```

### 1.3.12 Laços de Repetição

```
1 <?php
2     $num = 0;
3
4     while($num < 10 ) {
5         echo $num++;
6     }
7
8     $cont = 2000;
9
10    do{
11        $dobro = $cont + $cont;
12        echo "O dobro de $cont é $dobro";
13        $cont++;
14    } while ($cont <= 1999);
15
16    for($a = 1; $a <= 10; $a++){
17        $cubo = $a * $a * $a;
18        echo "O cubo de $a é $cubo<br />";
19    }
20 ?>
```

## Foreach

O Foreach faz o mesmo que as demais estruturas já apresentadas, porém, com ela podemos trabalhar com arrays de forma muito eficiente:

```
1 <?php
2 $info = array('Aqui no CEFET ', 'você se torna um ', 'Técnico em
   Informática');
3
4 foreach($info as $valor){
5     echo "$valor";
6 }
7 ?>
```

### 1.3.13 Quebrando loops

```
1 <?php
2
3 for($a = 1; $a <= 10; $a++){
4     if($a == 3) {
5         continue;
6     }
7
8     $cubo = $a * $a * $a;
9     echo "O cubo de $a é $cubo<br />";
10 }
11
12 for($a = 1; $a <= 10; $a++){
13     if($a == 3) {
14         break;
15     }
16
17     $cubo = $a * $a * $a;
18     echo "O cubo de $a é $cubo<br />";
19 }
20
21 ?>
```

### 1.3.14 Tipos de dados - arrays

Os arrays, em PHP, são mapas ordenados de chaves e valores, ou seja, é possível atribuir a um elemento do array uma chave e um valor; Tal característica permite que os arrays, em PHP, representem listas, tabelas hash, pilhas, filas, coleções, etc.; Os arrays, em PHP, também permitem que o elemento de um array "x" seja um outro array "y"; Essa característica possibilita que estruturas (arrays) multidimensionais sejam facilmente criadas ? uma árvore, por exemplo, pode ser facilmente criada em PHP pelo uso de arrays;

Existem duas forma de definir um array em PHP:

1. de maneira explícita através do construtor array().

- Ex.: Array([chave] => valor, ...);

2. de maneira implícita ou direta.

- Ex.: \$array\_exemplo[chave] = valor;

### Exemplo 01

```

1 <?php
2 // Array: Definição Explícita (sem chave)
3 $var = array(1, 2, 3, 4);
4
5 echo "[for]: ";
6 for($a=0; $a<count($var); $a++){
7     echo "$var[$a] ";
8 }
9
10 echo "<br>[foreach]: ";
11 foreach($var as $dado){
12     echo "$dado ";
13 }
14 ?>

```

1. Quando utilizamos o construtor array() durante a definição de um novo array dizemos que essa definição é explícita. Perceba que não são utilizadas chaves nesta construção, mas apenas valores, quando isso acontece são atribuídos índices numéricos crescentes, partindo do "0", para os dados inseridos no array.

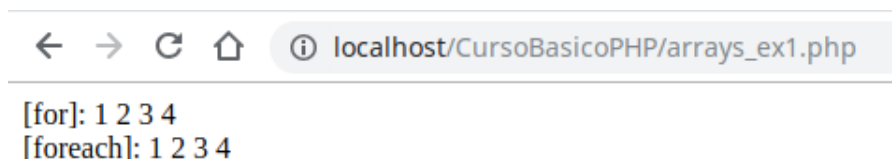


Figura 3: Resultado da Execução

### Exemplo 02

```

1 <?php
2 // Array: Definição Explícita (com chave)
3 $var = array( "Maria" => 28,
4               "João"   => 40,
5               "José"   => 59,
6               "Silvia" => 30
7             );
8
9 foreach($var as $chave => $valor){
10     echo "$chave: $valor<br> ";
11 }
12 echo "<br>";
13 print_r($var);
14 ?>

```

1. Neste exemplo de definição explícita foram utilizadas chaves durante a construção do array. Sendo assim, os valores inteiros (28, 40, 59, 30) estão vinculados as suas respectivas chaves (Maria, João, José, Silvia).
2. Observe que função "print\_r" permite apresentar, de maneira compreensível, todo o array de uma só vez.

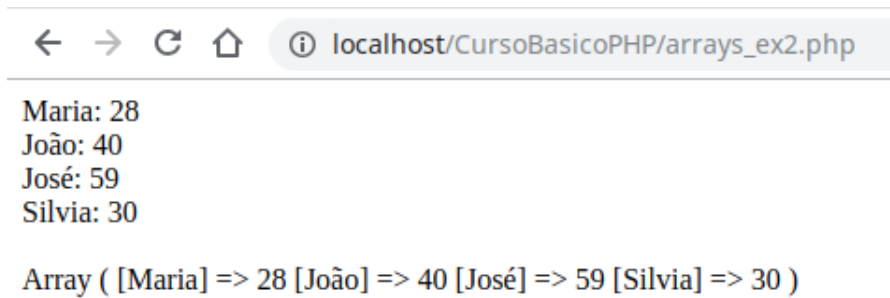


Figura 4: Resultado da Execução

### Exemplo 03

```

1 <?php
2 // Array: Definição Explícita (com chave)
3 $var = array( "000.000.000-00" => "Maria",
4               "001.001.001-11" => "João"
5             );
6
7 foreach($var as $chave => $valor){
8     echo "$chave: $valor<br> ";
9 }
10
11 $var = array( 28 => "Maria",
12              40 => "João"
13            );
14
15 echo "<br>";
16
17 foreach($var as $chave => $valor){
18     echo "$chave: $valor<br> ";
19 }
20 ?>

```

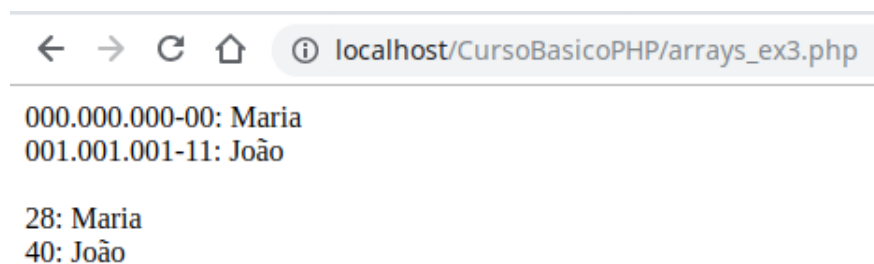


Figura 5: Resultado da Execução



### Exemplo 04

```

1 <?php
2 // Array: Definição Direta (sem chave)
3 $var[0] = "Aplicações";
4 $var[1] = "Web II";
5 $var[2] = "Introdução";
6
7 echo "[for]: ";
8 for($a=0;$a<count($var); $a++){
9     echo "$var[$a] ";
10 }
11 echo "<br>[foreach]: ";
12 foreach($var as $dado){
13     echo "$dado ";
14 }
15 ?>

```

1. Quando não utilizamos o construtor array(), durante a criação de um novo array, dizemos que essa definição é direta. Nela a sintaxe torna-se muito próxima da utilizada quando codificamos uma aplicação na linguagem C.

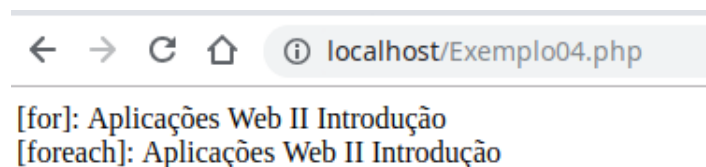


Figura 6: Resultado da Execução

#### 1.3.15 Arrays Multidimensionais - (codificação)

```

1 <?php
2 // Array Multidimensional: Definição Explícita
3 $arr = array( "Maria" => array("endereço" => "Rua Leopoldina 1500",
4                                "bairro"    => "Centro",
5                                ),
6             "João"   => array("endereço" => "Avn dos Imigrantes 1000",
7                                "bairro"   => "Vargem",
8                                )
9             );
10 foreach($arr as $chave => $aux){
11     echo strtoupper($chave).": <br>";
12     foreach($aux as $chave => $valor){
13         echo "    - $valor<br>";
14     }
15     echo "<br>";
16 }
17 ?>

```

1. Os valores vinculados às chaves (Maria, João) são outros arrays contendo novas chaves (endereço, bairro) e novos valores vinculados a elas.

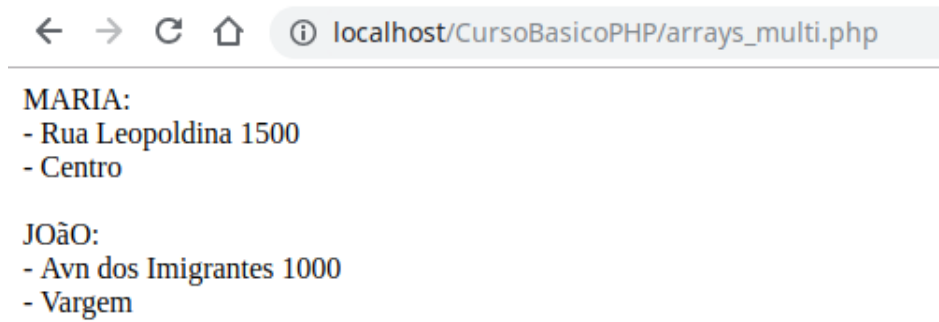


Figura 7: Resultado da Execução