TEMA 07 - Tipos de Funções

As funções são rotinas criadas para efetuar diversos serviços, estes serviços são tarefas que tem uma finalidade dentro do código fonte que estamos escrevendo, basta utilizar a função adequada para que não haja a necessidade de repetir um mesmo código já escrito por diversas vezes, chamando essa função em qualquer lugar do seu código.

7.1 Introdução sobre funções

Temos **dois tipos de funções dentro do PHP**, as **nativas** e as **internas**, as nativas são funções desenvolvidas pelos engenheiros de software que organizam a linguagem do PHP, já as funções consideradas internas são as que vamos criar no decorrer do nosso projeto, software e sistema web.
Vamos agora criar nossas próprias funções, dentro do PHP, para isso temos que respeitar algumas regras e boas práticas da linguagem seguindo sua sintaxe.
As funções são blocos de código que podem ser reutilizados em diferentes partes do programa, evitando repetições e facilitando a manutenção. As funções também permitem modularizar o código, separando-o em partes menores e mais claras. Para criar uma função no PHP, usamos a palavra-chave function, seguida do nome da função e dos parâmetros entre parênteses. Os parâmetros são variáveis que recebem os valores que serão usados pela função. Dentro das chaves, colocamos o corpo da função, que é o conjunto de instruções que serão executadas quando a função for chamada. Para retornar um valor da função, usamos a palavra-chave return. Veja um exemplo de uma função que calcula a soma de dois números:

```
<?php
// Definição da função
function soma($num1, $num2) {
   $resultado = $num1 + $num2;
   return $resultado;
}

// Chamada da função e exibição do resultado
$x = 10;
$y = 5;
$saida = soma($x, $y);
echo "A soma de $x e $y é: $saida";
?>
```

Para usar uma função, **basta invocá-la pelo seu nome, passando os argumentos entre parênteses**. Os argumentos são os valores reais que serão atribuídos aos parâmetros da função.

As funções são muito **úteis para organizar e simplificar o código**, além de permitir a reutilização de lógicas comuns. Existem muitas funções nativas do PHP que realizam diversas tarefas, como manipular strings, arrays, datas, arquivos, etc.

7.2 Funções Ambientes nativas e criação

Sintaxe de uma função

```
function nomedafuncao(arg ...) {
   código;
   return
}
```

O nome da função sempre temos que seguir as boas práticas da programação igual quando declaramos uma variável.
Código – Onde colocamos o trecho da tarefa do código.
Return – Caso a função tenha um retorno no argumento é usado de forma opcional.

Segue um exemplo de uma função simples sem retorno:

```
<?php
// Definição da função
function cumprimentar() {
   echo "Olá, mundo!";
}

// Chamada da função
cumprimentar();
?>
```

Outro exemplo com retorno:

```
<?php
// Definição da função
function gerarNumeroAleatorio() {
   $numero = rand(1, 100); // Gera um número aleatório entre 1 e 100
   return $numero; // Retorna o número gerado
}

// Chamada da função e exibição do resultado
$numeroAleatorio = gerarNumeroAleatorio();
echo "Número aleatório gerado: $numeroAleatorio";
?>
```

Exemplo com argumentos e retorno

Um exemplo com argumentos que é comumente usado na programação é uma função para calcular a média de uma lista de números.

```
<?php
// Definição da função
function calcularMedia($numeros) {
   $soma = 0;
   foreach ($numeros as $numero) {
      $soma += $numero;
   }
   $media = $soma / count($numeros);
   return $media;
}

// Lista de números
$listaNumeros = array(10, 20, 30, 40, 50);

// Chamada da função e exibição do resultado
```

```
$media = calcularMedia($listaNumeros);
echo "A média dos números " . implode(", ", $listaNumeros) . " é: $media";
?>
```

A função **count()** em PHP é usada para contar o número de elementos em uma variável, que pode ser um array. O **implode** é uma função em PHP que é usada para juntar elementos de um array em uma única string, utilizando um delimitador especificado.

7.3 Funções de texto, data, codificadas e numéricas

Função de texto – Retorna a um carácter específico.

```
<?php
        echo chr(65);
?>
```

Exemplo simples usando a função ord() para obter o código ASCII de um caractere específico:

```
<?php
// Obtendo o código ASCII do caractere 'A'
$codigo_ASCII = ord('A');
echo "O código ASCII de 'A' é: $codigo_ASCII";
?>
```

Neste exemplo, a função ord('A') retorna o código ASCII do caractere 'A', que é 65. Em seguida, esse valor é impresso na tela.

No script PHP (abaixo) é criada uma tabela HTML que lista os valores ASCII e seus caracteres correspondentes de 0 a 255. Cada linha da tabela mostra o valor ASCII (de 0 a 255) na primeira coluna e o caractere correspondente obtido usando a função chr() na segunda coluna.

```
<?php
echo "<table border='1'>";
echo "<tr><th>Valor</th><th>Caractere</th></tr>";

for ($i = 0; $i <= 255; $i++) {
   echo "<tr>";
   echo "<td>$i</td>";
   echo "<td>" . chr($i) . "</td>";
   echo "</tr>";
}

echo "</table>";
?>
```

Função de data – Permite a formatação da data e da hora de um local.

```
<?php
// Configura o fuso horário para o fuso horário local
date_default_timezone_set('America/Sao_Paulo');
```

```
// Exibe a data e hora atual formatada
$dataFormatada = date("d/m/Y H:i:s");
echo "Data e hora atual: $dataFormatada";
?>
```

A função date() do PHP exibirá a data e hora de acordo com o fuso horário configurado no servidor.

Se você precisa exibir a data e hora de acordo com o fuso horário local do seu sistema, você pode configurar o fuso horário utilizando a função date_default_timezone_set() antes de chamar a função date().

Funções codificadas – retornando uma matriz de string, frações da string original.

```
<?php
// Exemplo usando implode
$array = array("apple", "banana", "orange");
echo "Array antes do implode: ";
var_dump($array); // Visualiza o array antes do implode
echo "<br>";

$string = implode(", ", $array);
echo "String resultante após o implode: $string"."<br>"; // Visualiza a string resultante

// Exemplo usando explode
$string = "apple, banana, orange";
echo "String antes do explode: $string <br>";
$array = explode(", ", $string);
echo "Array resultante após o explode: ";
var_dump($array); // Visualiza o array resultante
?>
```

Resultado:
Array antes do implode: array(3) { [0]=> string(5) "apple" [1]=> string(6) "banana" [2]=> string(6) "orange" }
String resultante após o implode: apple, banana, orange
String antes do explode: apple, banana, orange
Array resultante após o explode: array(3) { [0]=> string(5) "apple" [1]=> string(6) "banana" [2]=> string(6) "orange" }

Funções numéricas – Formatar uma string

```
<?php
	$numero = 12345.67;
	$separador_decimal = ',';
	$separador_milhar = '.';
	$casas_decimais = 2;
	$prefixo = 'R$ ';
	$sufixo = ' (BRL)';
	$numero_formatado = number_format($numero, $casas_decimais, $separador_decimal, $separador_milhar);
	$numero_formatado = $prefixo . $numero_formatado . $sufixo;
```

```
        echo $numero_formatado; // Saída: R$ 12.345,67 (BRL)

        //Exemplo de função
        function formatar($numero) {
                $retorno = number_format($numero, 2, ',', '.');
                return $retorno;
        }

        echo '<br>' . formatar($numero);
?>
```

RESUMO:

Neste capítulo, exploramos os tipos de funções no contexto do desenvolvimento back-end em PHP. As funções desempenham um papel crucial ao permitir a criação de blocos de código reutilizáveis que executam tarefas específicas, melhorando a organização e a modularidade do código.

Destacamos dois tipos principais de funções: as nativas, que são desenvolvidas pela equipe de engenheiros de software para organizar a linguagem PHP, e as internas, que podemos criar em nossos próprios projetos. Para criar funções personalizadas, seguimos a sintaxe padrão do PHP, usando a palavra-chave "function", especificando o nome da função e os parâmetros entre parênteses. O corpo da função contém as instruções a serem executadas, e podemos usar a palavra-chave "return" para retornar valores, se necessário.

Demonstramos como criar uma função simples, como calcular a soma de dois números, e como invocá-la passando argumentos. Essas funções são valiosas para evitar a repetição de código e melhorar a legibilidade.Além disso, mencionamos que o PHP oferece uma vasta biblioteca de funções nativas para realizar diversas tarefas, desde manipular strings e arrays até trabalhar com datas e números.

Para aprofundar ainda mais o conhecimento sobre criação de funções em PHP, sugerimos consultar materiais complementares, como o curso de PHP 7 mencionado, que pode fornecer informações adicionais e práticas sobre o assunto. A compreensão de funções é fundamental para um desenvolvimento back-end eficaz e organizado.

ATIVIDADES:

1. Qual é a sintaxe básica para criar uma função em PHP? Liste os principais elementos que compõem a declaração de uma função.
2. Como você chama uma função em PHP? Dê um exemplo simples de como você invocaria uma função personalizada.
3. Qual é a diferença entre uma função que retorna um valor e uma função que não retorna nada em PHP? Dê exemplos de ambas.
4. O que uma função retorna quando não usa o comando return?
5. Mencione pelo menos três tipos de funções que podem ser úteis ao trabalhar com strings em PHP. Dê um exemplo de cada tipo.
6. O que são funções?
7. O que são funções nativas?
8. O que são funções ambiente?
9. Explique a importância das funções no desenvolvimento back-end em PHP. Como elas contribuem para a organização do código e a reutilização de lógica?
10. Liste pelo menos cinco funções nativas do PHP e descreva brevemente o que cada uma delas faz.

RESPOSTAS:

1. A sintaxe básica para criar uma função em PHP é:

```php
function nomeDaFuncao($parametro1, $parametro2, ...) {
   // Corpo da função
   // Código a ser executado
   return $resultado; // opcional
}
```

Os principais elementos que compõem a declaração de uma função são:
- A palavra-chave `function`.
- O nome da função.
- Parênteses contendo os parâmetros da função (opcionais).
- Chaves que delimitam o corpo da função.
- Opcionalmente, a palavra-chave `return` seguida do valor a ser retornado pela função.

2. Para chamar uma função em PHP, basta utilizar o nome da função seguido pelos parênteses contendo os argumentos, se houver. Aqui está um exemplo simples de como invocar uma função personalizada:

```php
function minhaFuncao($parametro) {
   echo "O parâmetro recebido é: $parametro";
}

minhaFuncao("Olá, mundo!");
```

3. A diferença entre uma função que retorna um valor e uma função que não retorna nada em PHP é que a primeira retorna um valor específico, enquanto a segunda não retorna nenhum valor. Aqui estão exemplos de ambas:

Função que retorna um valor:
```php
function soma($a, $b) {
   return $a + $b;
}

$resultado = soma(3, 5);
echo $resultado; // Saída: 8
```

Função que não retorna nada:
```php
function exibirMensagem($mensagem) {
   echo $mensagem;
}

exibirMensagem("Olá, mundo!");
```

4. Quando uma função em PHP não usa o comando `return`, ela simplesmente não retorna nenhum valor. Nesse caso, ao chamar a função, ela executa suas instruções internas, mas não há nenhum valor específico sendo retornado. O retorno será `null`.

5. Três tipos de funções úteis ao trabalhar com strings em PHP são:

- `strlen()`: Retorna o comprimento de uma string.
```php
$string = "Olá, mundo!";
$tamanho = strlen($string);
echo $tamanho; // Saída: 11
```

- `substr()`: Retorna uma parte de uma string.
```php
$string = "Hello, world!";
$sub_string = substr($string, 0, 5);
echo $sub_string; // Saída: Hello
```

- `str_replace()`: Substitui todas as ocorrências de uma substring por outra em uma string.
```php
$string = "Hello, world!";
$nova_string = str_replace("Hello", "Olá", $string);
echo $nova_string; // Saída: Olá, world!
```

6. Funções são blocos de código que podem ser reutilizados em diferentes partes do programa, evitando repetições e facilitando a manutenção. Elas permitem modularizar o código, separando-o em partes menores e mais claras, o que torna o código mais organizado e legível.

7. Funções nativas são aquelas que já são fornecidas pela linguagem PHP. Elas são desenvolvidas pela equipe de engenheiros de software para organizar a linguagem PHP e fornecer funcionalidades básicas para os desenvolvedores.

8. Funções ambiente são funções que não fazem parte da linguagem PHP em si, mas estão disponíveis no ambiente de execução, como funções definidas pelo usuário ou funções de bibliotecas externas.

9. As funções são importantes no desenvolvimento back-end em PHP porque permitem a organização do código, a reutilização de lógica e a simplificação de tarefas complexas. Elas contribuem para tornar o código mais legível, modular e fácil de manter.

10. Cinco funções nativas do PHP são:

- `strlen()`: Retorna o comprimento de uma string.
- `implode()`: Junta elementos de um array em uma string usando um delimitador.
- `explode()`: Divide uma string em um array de substrings com base em um delimitador.
- `date()`: Formata a data e hora atual de acordo com o formato especificado.
- `number_format()`: Formata um número com os milhares agrupados e a quantidade de casas

decimais especificada.