INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Charqueadas

# Desenvolvimento Back-end I

## Funções

Prof. Sérgio Yoshimitsu Fujii
sergiofujii@ifsul.edu.br

# Funções

## O que são?

- Uma função é um trecho de código encapsulado com um objetivo específico que pode (ou não) receber parâmetros, utilizar qualquer parte do PHP (incluindo outras funções) para manipular os dados e retornar um valor de qualquer tipo;

- O código dentro de uma função é encapsulado do código externo (e vice-versa), ou seja, a menos que você queira, nada que estiver entre as chaves { ... } (ações da função) é acessível ao código fora delas.

# Funções

## Criando uma função

- Para declarar uma função em PHP, utilizamos o operador "**function**" seguido do nome, dos argumentos (parâmetros) que a função poderá receber, e, por fim, das ações vêm entre chaves { ... }.

```php
<?php
    // Função com dois parâmetros
    function nome_da_funcao( $par1, $par2 ) {
        // Cria uma variável local $total
        $total = $par1 + $par2;

        // Retorna a soma dos parâmetros 1 e 2
        return $total;
    }
?>
```

# Funções

## Criando uma função

- A palavra “**return**” (retorno) serve para retornar o valor desejado depois de executar as ações da função. O interpretador do PHP para de ler o restante da função assim que encontrar a palavra “return”.

```php
<?php
    // Função com dois parâmetros
    function nome_da_funcao( $par1, $par2 ) {
        // Cria uma variável local $total
        $total = $par1 + $par2;

        // Retorna a soma dos parâmetros 1 e 2
        return $total;
    }
?>
```

# Funções

## Exemplo

- Função que calcula o quadrado de qualquer valor enviado.

```php
<?php
    // Função para calcular o quadrado
    function calcula_quadrado( $numero ) {
        // Garante que o parâmetro é do tipo numérico
        $numero = (int)$numero;

        // Retorna o número ao quadrado
        return $numero * $numero;
    }

    // Executa a função e exibe 15625 na tela
    echo calcula_quadrado(125);
?>
```

# Funções

## Chamada de função

- Uma função não executa nenhuma ação se ela não for chamada. Para chamar uma função, utilizamos apenas seu nome e os parâmetros (entre parênteses) separados por vírgula.

```php
// Executa a função e exibe 15625 na tela
echo calcula_quadrado(125);
```

# Funções

## Função sem retorno

- É possível criar funções sem parâmetro, ou sem retorno.

```php
<?php
    // Função sem parâmetros e sem retorno
    function exibe_frase() {
        echo 'Eu sou uma função que sem retorno ;)';
    }

    // Chama a função
    exibe_frase();
?>
```

# Funções

## Escopo de variáveis

- As variáveis criadas dentro das função têm escopo **local**. Isso significa que o código externo à função não sabe da existência de qualquer variável dentro da mesma.

```php
<?php
    // Uma variável fora da função
    $var_externa = 'Valor';

    function vai_dar_erro() {
        // Variável local
        $var_local = 'Valor';

        // Tenta acessar a variável externa
        echo $var_externa;
    }

    /* Chama a função e obtém um erro:
       Undefined variable: var_externa */
    vai_dar_erro();

    /* Tenta acessar a var_local
       Erro: Undefined variable: var_local */
    echo $var_local;
?>
```

# Funções

## Escopo de variáveis

- Para definir que uma variável terá escopo **global**, ou seja, acessível em qualquer parte do código, devemos utilizar a palavra chave "**global**".

```php
<?php
    // Uma variável fora da função
    $var_externa = 'Valor externo';

    function vai_dar_certo() {
        // A variável acessível dentro e fora da função
        global $var_global;

        // A variável externa estará acessível na função
        global $var_externa;

        // Define o valor da variável global
        $var_global = 'Valor local';

        // Exibe a variável externa
        echo $var_externa;
    }

    // Chama a função e exibe a variável externa
    vai_dar_certo();

    // Acessa a variável criada dentro da função
    echo $var_global;
?>
```

# Funções

## Variáveis Estáticas

- Variáveis estáticas são variáveis que mantêm o valor da última execução da função;

- Se você executar a função duas vezes somando o valor da variável estática (apenas como exemplo), ele sempre irá aumentar de onde parou na última execução.

# Funções

## Variáveis Estáticas

```php
<?php
    function andar( $distancia ) {
        // Define que a variável é estática
        static $soma;

        // Soma a distancia
        $soma += $distancia;

        // Retorna a distancia
        return $soma;
    }

    // O valor da soma será sempre aumentado
    echo andar(1) . "<br>"; // 1
    echo andar(10) . "<br>"; // 11
    echo andar(21) . "<br>"; // 32
    echo andar(220) . "<br>"; // 252
    echo andar(1) . "<br>"; // 253
?>
```

# Funções

## Parâmetros

- Podemos (ou não) passar parâmetros para a função. Os parâmetros são simplesmente valores que podemos manipular nas ações da função para retornar um valor qualquer.

```php
<?php
    // Uma função que soma os parâmetros
    function soma_parametros( $num1, $num2, $num3 ) {
        // Retorna a soma
        return $num1 + $num2 + $num3;
    }

    // Chama a função
    echo soma_parametros(10, 50, 101); // 161
?>
```

# Funções

## Parâmetros - Valor Padrão

- Você também pode definir valores padrão para os parâmetros, assim, se o desenvolvedor esquecer de enviar os parâmetros, o valor definido é assumido.

```php
<?php
    // Uma função que soma os parâmetros
    function soma_parametros ( $num1 = 10, $num2, $num3 = 1 ) {
        // Retorna a soma
        return $num1 + $num2 + $num3;
    }

    // Chama a função
    echo soma_parametros(false, 50); // 51
?>
```

# Funções

## Parâmetros - Valor Padrão

- Se a função for chamada com valor **falso** no local onde apareceria o parâmetro 1, o valor **0** será assumido. Como o parâmetro 2 não tem valor definido, ele será “50”. O parâmetro 3 tem valor definido, portanto, a função assumirá o valor "1" para ele.

```php
<?php
    // Uma função que soma os parâmetros
    function soma_parametros ( $num1 = 10, $num2, $num3 = 1 ) {
        // Retorna a soma
        return $num1 + $num2 + $num3;
    }

    // Chama a função
    echo soma_parametros(false, 50); // 0 + 50 + 1 = 51
?>
```

# Funções

## Parâmetros - Passagem por Referência

- Até agora, estamos passando valores para a função de maneira comum, ou seja, alterações feitas no valor da variável, não afetaram seu valor fora da função;

- Se você precisa alterar o valor da variável dentro da função e obter esse mesmo valor fora dela, é possível passar o parâmetro por referência;

- Para isso, basta adicionar um **&** antes do parâmetro.

# Funções

## Parâmetros - Passagem por Referência

```php
<?php
    // Cria uma função que soma o valor um com o valor
    function soma( &$valor1, $valor2 ) {
        // Faz a soma
        $valor1 += $valor2;
    }

    // Cria uma variável chamada $qlq_coisa
    $qlq_coisa = 50;

    // Executa a função
    soma($qlq_coisa, 10);

    // Mostra 60 na tela
    echo $qlq_coisa;
?>
```

# Funções

## Parâmetros - Passagem por Referência

- Perceba que, pelo simples fato de ter um **&** antes do primeiro parâmetro, qualquer valor que for enviado para a função no primeiro parâmetro, **será passado por referência** e as **alterações feitas em seu valor dentro da função, afetam seu valor fora dela**.

```php
<?php
    // Cria uma função que soma o valor um com o valor
    function soma( &$valor1, $valor2 ) {
        // Faz a soma
        $valor1 += $valor2;
    }
```

# Funções

## Parâmetros dinâmicos

- Caso seja necessário criar funções que tenham vários parâmetros criados dinamicamente, basta criar uma função sem parâmetros e enviar vários argumentos (valores) quando for chamar a mesma.

- O PHP possui algumas funções para extrair valores enviados para a função, como ***func_get_args***, que retorna um *array* contendo os argumentos enviados na chamada da função.

# Funções

## Parâmetros dinâmicos

```php
<?php
    // Cria uma função que soma o valor um com o valor
    function exibe_parametros () {
        // Array com os parâmetros da função
        $paramentros = func_get_args();

        // Laço para exibir os parâmetros
        foreach ( $paramentros as $posicao => $valor ) {
            // Exibe todos os parâmetros
            echo 'Parâmetro ' . $posicao . ': ' . $valor . '<br>';
        }
    }

    // Chama a função com quantos parâmetros quiser
    exibe_parametros('Valor 1', 'Valor 2', 'Valor 3');
?>
```

# Funções

## Parâmetros dinâmicos

- No exemplo a função não tem parâmetros:

```
2       // Cria uma função que soma o valor um com o valor
3       function exibe_parametros () {
```

- Porém, na chamada, vários parâmetros são passados:

```
14      // Chama a função com quantos parâmetros quiser
15      exibe_parametros('Valor 1', 'Valor 2', 'Valor 3');
```

- Com isso, pode-se obter esses valores dinamicamente dentro da função.

```
4           // Array com os parâmetros da função
5           $paramentros = func_get_args();
```

# Funções

## Recursividade

- O PHP permite recursividade, ou seja, **uma função que chama ela mesma** dentro de suas ações.

```php
<?php
    //Função recursiva que calcular o fatorial de um número
    function fatorial($numero){
        if($numero <= 1){
            return $numero;
        }else{
            return $numero * fatorial($numero - 1);
        }
    }

    //4*3*2 = 24
    echo fatorial(4);
?>
```

# Funções

## Recursividade

```php
<?php
    // Conta de 1 até um limite
    function conta_ate( $limite ) {
        // Especifica que o valor será estático
        static $valor = 0;

        // Verifica se o valor é menor ou igual ao limite
        if ( $valor < $limite ) {
            // Soma mais um no valor
            $valor++;

            // Mostra o valor
            echo $valor . '<br>';

            // Chama a função novamente e começa tudo de novo
            conta_ate( $limite );
        }
    }

    // Chama a função apenas uma vez
    // A recursividade fará o resto das chamadas
    conta_ate(8);
?>
```

# Funções

## Funções com especificação de tipo

- Exemplo de função com especificação do tipo do parâmetro e do tipo de retorno:

```php
<?php
    // Função com especificação de tipo de parâmetro e retorno
    function somaCinco( int $par1 ) : int {
        return $par1 + 5;
    }

    // Chama a função
    echo somaCinco(8);
?>
```

# Funções

## Funções com especificação de tipo

- Erro caso o tipo de parâmetro seja diferente do especificado:

```php
<?php
    // Função com especificação de tipo de parâmetro e retorno
    function somaCinco( int $par1 ) : int {
        return $par1 + 5;
    }

    // Chama a função
    echo somaCinco("A");
?>
```

**Fatal error**: Uncaught TypeError: Argument 1 passed to somaCinco() must be of the type int, string given, called in C:\xampp\htdocs\funcoes.php on line 8 and defined in C:\xampp\htdocs\funcoes.php:3 Stack trace: #0 C:\xampp\htdocs\funcoes.php(8): somaCinco('A') #1 {main} thrown in **C:\xampp\htdocs\funcoes.php** on line **3**