Curso de PHP

FATEC - Jundiaí

PHP – Aula 5

Funções

Uma função é um bloco de código reutilizável que é executado devido a um evento ou pela chamada de outra função. Deve-se usar a declaração function para criar uma função.

```
function nome_da_função([arg1, arg2, arg3]) {
        Comandos;
... ;
        [return <valor de retorno>];
}
```

A função deve ter um nome único, não podendo iniciar com um número e nem ter o caractere "." em sua formação. A referência à função é feita por meio deste nome. O PHP, ao encontrar a referência, interromperá a execução linear do programa, executará a função referenciada, para só então retornar ao programa a partir daquele ponto.

PHP – Aula 5

Funções

As funções devem ser definidas antes de serem referenciadas, podendo ser utilizadas de duas formas:

- Ter o seu resultado atribuído a alguma variável por meio do operador de atribuição “=”, fazendo com que a variável à esquerda do operador receba o resultado da função referenciada à direita

- Simplesmente executar alguma tarefa sem nenhum retorno. Para isto basta apenas referenciar o nome da função.

As funções são geralmente criadas para uma tarefa específica e também por vezes repetitiva.

PHP – Aula 5

Funções

Os parâmetros usados pela função são declarados entre parênteses. Os comandos a serem executados pela função devem estar entre chaves. A declaração return retorna um valor quando a função é chamada. Esta declaração não é necessária se a função não retorna nenhum valor.

```
<?php
  Function escreveTexto()
  {
    echo "Criando uma função simples! <br>";
  }

  Function escreveAno()
  {
    $num = 2004;
    echo "Estamos no ano de: $num";
  }
  escreveTexto();
  escreveAno();
?>
```

PHP – Aula 5

Funções

Qualquer código PHP válido pode estar contido no interior de uma função. Como a checagem de tipos em PHP é dinâmica, o tipo de retorno não deve ser declarado, sendo necessário que fique atento para que a função retorne o tipo desejado!

```
<?php
  function soma($valor1, $valor2)
  {
     $resultado = $valor1 + $valor2;
     return ($resultado);
  }
  $x = soma(7, 8);
  echo "O valor da soma é: $x";
?>
```

Obs.: em muitos casos o devemos estar atento aos tipos dos valores passados como parâmetros, pois se não for passado o tipo esperado não é emitido nenhum alerta pelo interpretador PHP, já que este não testa os tipos.

PHP – Aula 5

Função

Não é possível que uma função retorne mais de um valor, mas é permitido fazer com que uma função retorne um valor composto, como listas ou arrays.

```
<?php
   function animais()
   {
       return array("Boi", "Cavalo","Macaco","Hiena");
   }
   $animal = animais();
   for ($x=0;$x<sizeof($animal);$x++)
      Echo "animal[$x] = $animal[$x]<br>";
?>
```

Resultado apresentado no terminal seria:

```
Animal[0] = Boi
Animal[1] = Cavalo
Animal[2] = Macaco
Animal[3] = Hiena
```

PHP – Aula 5

Função

Uma das grandes vantagens das funções é sua capacidade de receber parâmetros para sua execução. Os parâmetros podem ser passados por referência ou valor. A condição padrão do PHP é por valor. A variável passada não tem seu conteúdo alterado durante a execução da função.

```
<?php
  function soma($oper) {
    for ($x=0;$x<sizeof($oper);$x++) {
        $soma+=oper[$x];
        $oper[$x] = 0;
    }
    return $soma;
  }
  $oper = array(10,15,2,25,37);
  echo soma($oper) . "<br>";
  echo $oper[2];
?>
```

Resultado apresentado no terminal seria: 89 e 2

PHP – Aula 5

Função

Nas situações em que é necessário que a variável passada como parâmetro seja alterada dentro da função, deve-se acrescentar o símbolo & antes do nome da variável. Isto indica ao PHP que estamos passando o parâmetro por referência.

```
<?php
  function soma(&$oper) {
    for ($x=0;$x<sizeof($oper);$x++) {
        $soma+=oper[$x];
        $oper[$x] = 0;
    }
    return $soma;
  }
  $oper = array(10,15,2,25,37);
  echo soma($oper) . "<br>";
  echo $oper[2];
?>
```

Resultado apresentado no terminal seria: 89 e 0

PHP – Aula 5

Função

Agora podemos ter uma situação em que temos que passar o parâmetro como referência e como valor.

```
<?php
  function junta($texto) {
     $texto .= " usando referência e valor como parâmetro";
  }
  $str = "Este é um teste";
  junta($str);
  echo $str . "<br>";
  junta(&$str);
  echo $str;
?>
```

Resultado apresentado no terminal seria:

Este é um teste
Este é um teste usando referência e valor como parâmetro

PHP – Aula 5

Função

Argumentos pré-definidos

Em PHP é possível ter valores default para argumentos de funções, ou seja, valores que serão assumidos em caso de nada ser passado no lugar do argumento. Quando algum parâmetro é declarado desta maneira, a passagem do mesmo na chamada da função torna-se opcional.

```
<?php
  function teste($aula = "iniciando") {
          echo $aula;
  }
  teste();
  teste("função e classe");
?>
```

Resultado apresentado no terminal seria:

iniciando
função e classe

PHP – Aula 5

Função

Argumentos pré-definidos

Devemos lembrar que quando a função tem mais de um parâmetro, o que tem valor default deve ser declarado por último.

```
<?php
  function teste($figura = “circulo”, $cor) {
    echo "a figura é um ". $figura. " de cor " $cor;
  }
  teste(azul); // vai funcionar da maneira esperada - erro no interpretador

  function teste2($cor, $figura = “circulo”) {
    echo "a figura é um ". $figura. " de cor " $cor;
  }
  teste2(azul);
?>
```

Obs.: O valor padrão deve ser uma constante, pois o PHP irá recusar qualquer outra forma de padrão - uma variável, por exemplo.

PHP – Aula 5

Função

Funções podem conter outras funções dentro de sua estrutura, ou fazer referência a outra funções. Quando construímos funções dentro de funções (funções aninhadas), elas somente poderão ser referenciadas dentro da função principal.

```
<?php
  function a() {
    function b() {
      echo “Esta é a função b() dentro da função a()!”;
    }
    b();
  }
  a();
  b();
?>
```

Obs.: Se não usássemos a referência da função a() antes da função b(), o PHP provocaria um erro, pois a função b() está declarada dentro da função a().

PHP – Aula 5

Função

Ao incluir funções aninhadas, devemos ter cuidado com os seus nomes, pois podem causar erros durante a execução do programa.

```
<?php
  function a() {
    function b() {
      echo “Esta é a função b() dentro da função a()!”;
    }
    b();
  }
  function b() {
    echo “Função principal b()”;
  }
  a();
?>
```

Obs.: O PHP provocará um erro, pelo fato de existirem duas funções com o mesmo nome b(), não importando se é uma função principal ou aninhada.

PHP – Aula 5

Função

O modificador static

Uma variável estática é visível num escopo local, mas ela é inicializada apenas uma vez e seu valor não é perdido quando a execução do script deixa esse escopo. Veja o seguinte exemplo:

```
<?php
  function Teste() {
    $a = 0;
    echo $a;
    $a++;
  }
  Teste();
  Teste();
  Teste();
?>
```

O último comando da função é inútil, pois assim que for encerrada a execução da função a variável $a perde seu valor.

PHP – Aula 5

Função

Vamos usar o mesmo exemplo anterior, colocando o modificador static.

```
<?php
  function Teste() {
    static $a = 0;
    echo $a . “<br>”;
    $a++;
  }
  Teste();
  Teste();
  Teste();
?>
```

Neste exemplo, a cada chamada da função a variável $a terá seu valor impresso e será incrementada.

Resultado no terminal será:
```
0
1
2
```


PHP – Aula 5

Funções

Buscando um número arbitrário de argumentos passados para um função.

```
<?php
  function nova() {
    $args = func_get_args();
    for ($i=0;$i<sizeof($args);$i++) {
       echo $args[$i] . “<br>”;
    }
  }
  nova(“ola”);
  nova(“aluno”, “como”, “você”);
  nova(“foi”,“hoje”,“na”,“faculdade”);
?>
```

PHP – Aula 5

Funções

Verificando o tipo de uma variável

Por causa da tipagem dinâmica utilizada pelo PHP, nem sempre é possível saber qual o tipo de uma variável em determinado instantes. Podemos contar com a ajuda de algumas funções que verificam isso. A verificação pode ser feita de duas maneiras:

1) Função que retorna o tipo da variável

Esta função é a gettype. Sua sintaxe é a seguinte:

string gettype(mixed var);

A palavra "mixed" indica que a variável var pode ser de diversos tipos.
A função gettype pode retornar as seguintes strings: "boolean", "integer", "double", "string", "array", "object", "NULL" e "unknown type".

PHP – Aula 5

Funções

2) Funções que testam o tipo da variável

São as funções is_int, is_integer, is_numeric, is_real, is_long, is_float, is_string, is_array, is_object, is_bool e is_null. Todas têm o mesmo formato, seguindo modelo da sintaxe a seguir:

bool is_integer(mixed var);

Todas essas funções retornam true se a variável for daquele tipo, e false em caso contrário.

```
<?php
  $x = 10;
  if (is_integer($x)) {
    echo “É um inteiro<br>”;
  } else {
    echo “Não é um inteiro<br>”;
  }
?>
```

PHP – Aula 5

Funções

Utilizando as instruções include e require para reutilização de função.

São instruções que auxiliam na programação para reaproveitar as funções dentro de vários programas. Assim podemos criar nossas próprias bibliotecas de funções.

Include e Require

Sintaxe:

include '<nome_do_arquivo>';
require '<nome_do_arquivo>';

Os dois tem quase a mesma função - um inclui o conteúdo do arquivo especificado (seja um script PHP ou um arquivo HTML normal) e o outro requer que o arquivo especificado seja incluído. Se esse arquivo por algum motivo não puder ser incluído, um erro aparecerá na página.

PHP – Aula 5

Include

Permite a inclusão de outros arquivos php dentro do script que está sendo executado. Pode-se criar uma função que imprime a data atual e pode-se reusá-lo sem precisar reescrever o código cada vez que for necessário. No exemplo a seguir, pode-se chamar o primeiro script de cabecalho.php e o próximo script o inclui através do comando include.

```
<?php
   $meses = array(1 => "Janeiro",2 => "Fevereiro",3 => "Março",4 => "Abril",
       5 => "Maio",6 => "Junho",7 => "Julho",8 => "Agosto",9 => "Setembro",
      10 => "Outubro",11 => "Novembro",12 => "Dezembro");
   $hoje = getdate();
   $dia = $hoje["mday"];
   $mes = $hoje["mon"];
   $nomeMes = $meses[$mes];
   $ano = $hoje["year"];
   echo "Olá. Hoje é dia $dia de $nomeMes de $ano.“
?>
```

Chamaremos este arquivo de inicio.php

PHP – Aula 5

Include

```
<?php
    include(“inicio.php");
    echo “<br> com a inclusão acima o cabeçalho mostra a data de hoje”;
?>
```

A função **Include** é semelhante à função **require**, com a diferença que o código do arquivo incluído é processado em tempo de execução, permitindo que sejam usados “includes” dentro de estruturas de controle como **for** e **while**.

PHP – Aula 5

Include

```
<?php
  $arqs = array(‘a1.inc’,’a2.inc’,’a3.inc’) ;
  for ($i=0;$i<count($arqs); $i++) {
       include($arqs[$i]);
  }

  If ($x == $y) {
     include($arquivo1);
  }
  else {
       include($arquivo2);
  }
?>
```

Obs.: Note que quando se utiliza a função include dentro de estruturas é necessário a utilização das chaves

PHP – Aula 5

Require

A função **require** põe o conteúdo de um outro arquivo no arquivo php atual, antes de ser executado. Quando o interpretador do PHP ler este arquivo, ele encontrará todo o conteúdo dos “require´s” adicionado no arquivo corrente.

```
Require(“nomedoarquivo”);
```

Criando o hábito de usar essa função, o programador pode vir a encontrar um erro de arquivo já declarado. Para evitar isso é recomendável que sempre que a função **require** for utilizada ela seja substituída pela função **require_once**.

```
Require_once(“nome_do_arquivo“);
```


PHP – Aula 5

Termino da aula