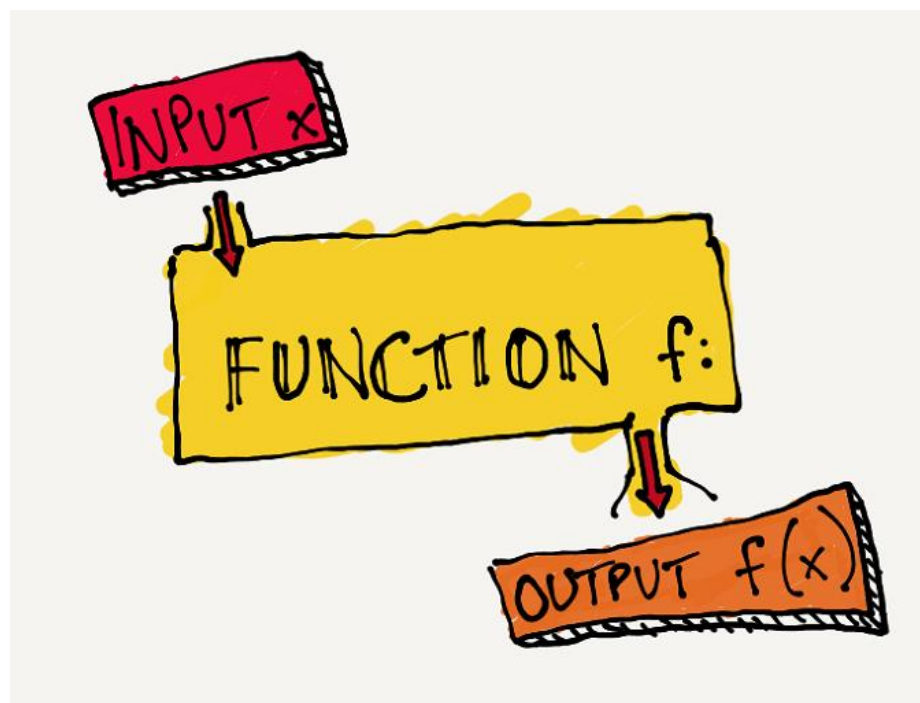


Funções e Tratamento de erros em PHP



Conteúdo da aula

Nesta aula veremos as **funções definidas por usuários**.

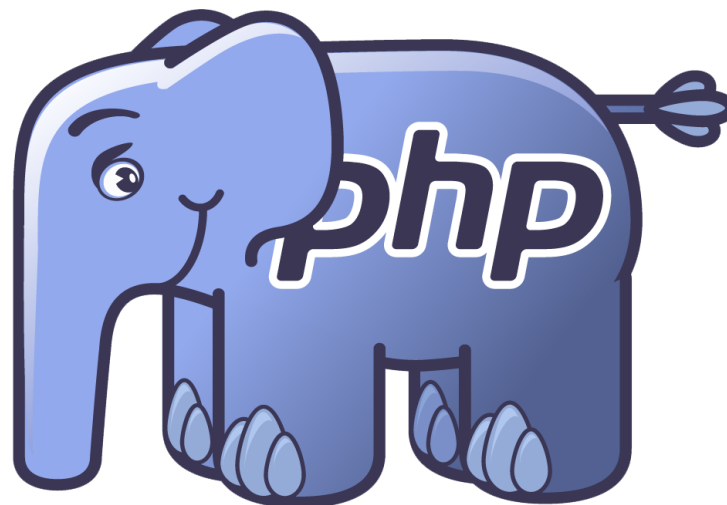




INSTITUTO
FEDERAL
São Paulo

Conteúdo da aula

E também estudaremos um pouco sobre o **tratamento de**
erros em PHP.



Funções (definidas por usuários)



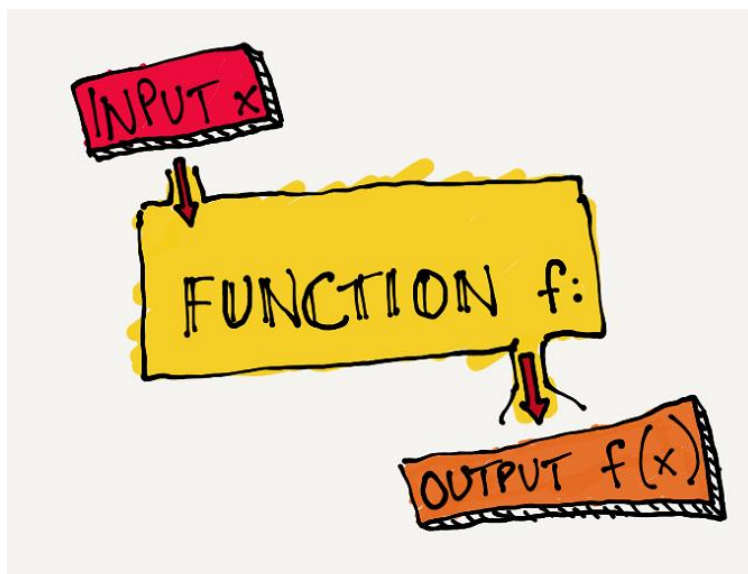
Funções

No PHP (e em praticamente todas as linguagens de programação), **funções** são blocos de código que **realizam uma tarefa específica**. Elas permitem que um trecho de código seja escrito uma única vez e **reutilizado** sempre que necessário.



Funções

Uma função pode receber valores de entrada (parâmetros), processar informações e retornar um resultado.





Funções

Funções são úteis porque permitem:

- **Evitar repetição de código** – em vez de reescrever um trecho várias vezes, basta chamá-lo por meio de uma função.



Funções

- Facilitar a leitura e organização do código – funções segmentam o programa em partes menores e mais fáceis de entender.
- Facilitar a manutenção e atualização – se precisar mudar a lógica, basta modificar a função em um único lugar.



Funções

Exemplo de código PHP sem função:

```
<?php
echo '<div class="alert alert-success" role="alert">
    Operação realizada com sucesso!
</div>';

echo '<div class="alert alert-warning" role="alert">
    Atenção! Verifique os dados informados.
</div>';

echo '<div class="alert alert-danger" role="alert">
    Erro! Algo deu errado.
</div>';
?>
```



Funções

Se precisarmos mudar a estrutura do <div>, teremos que editar cada um separadamente, aumentando as chances de erro e dificultando a manutenção.



Funções

Exemplo de código PHP com função:

```
<body>
<?php
function exibirAlerta($tipo, $mensagem)
{
    echo '<div class="alert alert-' . $tipo . '" role="alert">' . $mensagem . '</div>';
}

// Chamadas da função para exibir os alertas
exibirAlerta("success", "Operação realizada com sucesso!");
exibirAlerta("warning", "Atenção! Verifique os dados informados.");
exibirAlerta("danger", "Erro! Algo deu errado.");
?>
```



Funções

- Código mais limpo e fácil de entender.
- Se for necessário mudar a estrutura do alerta, só precisamos alterar a função.
- Podemos exibir quantos alertas quisermos apenas chamando `exibirAlerta()`.



Funções

O uso de funções permite dividir o código em módulos menores e independentes facilita a **manutenção**. Se houver um erro em uma parte, ele pode ser corrigido sem afetar o restante do programa.



Funções

Exemplo: Um sistema de e-commerce pode ter funções separadas para **calcular impostos**, **descontos** e **preços finais**, tornando o código mais organizado:



Funções

Funções também deixam o código mais **autoexplicativo**,
facilitando o entendimento por outros desenvolvedores.



Funções

Por fim, a **facilidade de manutenção** proporcionada pelo uso de funções, em minha opinião, é a maior **vantagem**, pois se for necessário fazer alterações na lógica, basta modificar a função uma vez, em vez de alterar várias partes do código.



Funções

Exemplo: Se quisermos mudar a fórmula do cálculo de imposto, basta alterar a função, e todas as chamadas dela serão automaticamente atualizadas.



Funções

Criar uma função:

Example

```
function myMessage() {  
    echo "Hello world!";  
}
```



Funções

Chamar a função:

Example

```
function myMessage() {  
    echo "Hello world!";  
}
```

```
myMessage();
```



Funções

Função com argumentos:

Example

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}  
  
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");
```



Example

```
function familyName($fname) {  
    echo "$fname Refsnes.<br>";  
}
```

```
familyName("Jani");  
familyName("Hege");  
familyName("Stale");  
familyName("Kai Jim");  
familyName("Borge");
```



Funções

Função com argumentos com valor padrão:

Example

```
function setHeight($minheight = 50) {  
    echo "The height is : $minheight <br>";  
}  
  
setHeight(350);  
setHeight(); // will use the default value of 50  
setHeight(135);  
setHeight(80);
```



Funções

Example

```
function setHeight($minheight = 50) {  
    echo "The height is : $minheight <br>";  
}  
  
setHeight(350);  
setHeight(); // will use the default value of 50  
setHeight(135);  
setHeight(80);
```



Funções

Função com retorno de valores:

Example

```
function sum($x, $y) {  
    $z = $x + $y;  
    return $z;  
}  
  
echo "5 + 10 = " . sum(5, 10) . "<br>";  
echo "7 + 13 = " . sum(7, 13) . "<br>";  
echo "2 + 4 = " . sum(2, 4);
```




Example

```
function sum($x, $y) {  
    $z = $x + $y;  
    return $z;  
}  
  
echo "5 + 10 = " . sum(5, 10) . "<br>";  
echo "7 + 13 = " . sum(7, 13) . "<br>";  
echo "2 + 4 = " . sum(2, 4);
```



Funções

Referências:

https://www.w3schools.com/php/php_functions.asp

https://www.php.net/manual/pt_BR/functions.user-defined.php

Tratamento de erros



Tratamento de erros

O tratamento de erros consiste em **preparar seu sistema/aplicação/código** para rodar em qualquer situação e, se não for possível, avisar porquê não pôde ser executado.



Tratamento de erros

Ao implementar estruturas para o tratamento de erros você se **prepara para eventuais falhas** em determinadas situações e é capaz de contorná-las ou, se não for possível, **finalizar a execução de forma segura** informando adequadamente **o usuário do que está acontecendo.**



Tratamento de erros

O tratamento de erros em seu código permite a apresentação de **mensagens personalizadas e amigáveis** para informar aos os usuários, evitando a sensação de **quebra do sistema**.



Tratamento de erros

Adicionalmente é possível informar aos administradores o que exatamente está acontecendo e como proceder para resolver o problema. Isso pode ser feito por mensagens no programa ou por registros (*logs*) de erros da aplicação.



Tratamento de erros

Uma das formas mais simples e rudimentares de tratar erros é usando a função die() ou a função exit().

Em ambos os casos a execução do programa é abortada.



Tratamento de erros

exit — Mostra uma mensagem e termina o script atual

```
<?php

$filename = '/caminho/para/arquivo';
$file = fopen ($filename, 'r')
    or exit("Não pude abrir o arquivo ($filename)");

?>
```



Tratamento de erros

die — Equivalente a `exit()`

[https://www.php.net/manual/pt BR/function.die.php](https://www.php.net/manual/pt_BR/function.die.php)



Tratamento de erros

Utilizar as funções *exit* ou *die* é uma alternativa minimamente boa, pois é possível impedir o prosseguimento do script caso alguma ação essencial não tenha sido feita.



Tratamento de erros

Por exemplo, é possível **abortar a execução de uma página que armazena imagens em um diretório no servidor se este diretório não existir ou não for gravável (não ter permissões).**



Tratamento de erros

Outro exemplo comum consiste em **abortar a execução** de **uma ou mais páginas** que necessitem de dados do **banco de dados** quando a **conexão com o banco** **não tenha sido estabelecida**.



Example (MySQLi Procedural)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = mysqli_connect($servername, $username, $password);

// Check connection
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
echo "Connected successfully";
?>
```



Example (MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// Create connection
$conn = new mysqli($servername, $username, $password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
echo "Connected successfully";
?>
```



**INSTITUTO
FEDERAL**
São Paulo

Tratamento de erros

Exemplo,

https://www.w3schools.com/php/php_mysql_connect.asp



Tratamento de erros

No entanto, **essa é uma forma ruim de lidar com erros**, pois ela aborta completamente a execução do programa, o que em muitos casos não é o comportamento desejado, visto que nem todos os tipos de erros são fatais para a execução de uma aplicação.



Tratamento de erros

Por isso existe uma forma bem mais adequada e elegante de tratamento e manipulação de erros em PHP, as **Exceções do PHP**.

Exceções em PHP

(PHP Exception Handling)



Exception Handling

As exceções são usadas para alterar o fluxo normal de um script se ocorrer um erro especificado.

https://www.w3schools.com/php/php_exception.asp

https://www.php.net/manual/pt_BR/class.exception.php



Exception Handling

A partir do **PHP 5** surgiu uma forma de lidar com erros orientada a objetos as **exceções** em PHP (em inglês *PHP Exception handling*).



Exception Handling

Elas consistem em alterar o fluxo normal da execução do código se ocorrer uma condição de erro (*exception*) especificada.

Essa condição é chamada de exceção.



Exception Handling

O W3Schools possui uma explicação textual sobre o funcionamento das exceções em PHP, que pode ser consultada nesse link:

https://www.w3schools.com/php/php_exception.asp



Exception Handling

Vamos ver alguns códigos e um exemplo na prática.



Exception Handling


O primeiro passo é **criar uma exceção**, isso é feito usando a palavra reservada ***throw*** seguida de um objeto derivado da classe ***Exception***, que é nativa da linguagem.



Exception Handling

Observe:

```
function dividir($x, $y)
{
    if ($y == 0) {
        throw new Exception('Não é possível realizar uma divisão por zero');
    }
    $resultado = $x / $y;
    return $resultado;
}
```





Exception Handling

Em seguida deve-se fazer a chamada da função dentro de um bloco ***PHP Try Catch***, que é responsável por pegar essa exceção lançada pelo ***throw***.



Exception Handling

Veja:

```
try {  
    echo dividir(10,0)."<br/>";  
} catch (Exception $e) {  
    echo 'Exceção capturada: ', $e->getMessage(), "\n";  
}
```



Exception Handling

Código completo do exemplo:

```
<?php
function dividir($x, $y)
{
    if ($y == 0) {
        throw new Exception('Não é possível realizar uma divisão por zero');
    }
    $resultado = $x / $y;
    return $resultado;
}

try {
    echo dividir(10,0)."<br/>";
} catch (Exception $e) {
    echo 'Exceção capturada: ', $e->getMessage(), "\n";
}

?>
```



```
<?php
function dividir($x, $y)
{
    if ($y == 0) {
        throw new Exception('Não é possível realizar uma divisão por zero');
    }
    $resultado = $x / $y;
    return $resultado;
}

try {
    echo dividir(10,0)."<br/>";
} catch (Exception $e) {
    echo 'Exceção capturada: ', $e->getMessage(), "\n";
}

?>
```



Exception Handling

O bloco de tratamento de exceção **PHP Try Catch** pode ser comparada a uma estrutura de decisão “*if ... else*” porém com diferencial que utiliza-se o primeiro quando o desenvolvedor não tem como garantir que aquele código será executado com sucesso.



Exception Handling

try - Uma função que usa uma exceção deve estar em um bloco "**try**". Se a exceção não disparar, o código continuará normalmente. No entanto, se a exceção for disparada, uma exceção é “lançada”.



Exception Handling

catch - Um bloco “catch” recupera uma exceção e cria um objeto contendo as informações da exceção.



Exception Handling

É possível definir mais de uma exceção em um código.

Considerando o exemplo anterior seria possível restringir algum valor de divisor caso o programador quisesse.



Exception Handling

Veja:

```
function dividir($x, $y)
{
    if ($y == 0) {
        throw new Exception('Não é possível realizar uma divisão por zero');
    }
    if ($y < 1 ) {
        throw new Exception('Não é permitido realizar uma divisão por um valor menor que 1');
    }
    $resultado = $x / $y;
    return $resultado;
}
```



```
<?php
function dividir($x, $y)
{
    if ($y == 0) {
        throw new Exception('Não é possível realizar uma divisão por zero');
    }
    if ($y < 1 ) {
        throw new Exception('Não é permitido realizar uma divisão por um valor menor que 1');
    }
    $resultado = $x / $y;
    return $resultado;
}

try {
    echo dividir(10,0.5)."<br/>";
} catch (Exception $e) {
    echo 'Exceção capturada: ', $e->getMessage(), "\n";
}

?>
```





Exception Handling

Resultado:





Exception Handling

Existem algumas dezenas de exceções nativas na linguagem PHP, como as aritméticas (divisão por zero), de funções (chamada a uma função inexistente ou chamada a uma função/método sem passar os devidos parâmetros ou argumentos).



Exception Handling

Uma lista com essas exceções pode ser vista neste endereço:

https://www.php.net/manual/pt_BR/spl.exceptions.php



Exception Handling

Índice

- [BadFunctionCallException](#)
- [BadMethodCallException](#)
- [DomainException](#)
- [InvalidArgumentException](#)
- [LengthException](#)
- [LogicException](#)
- [OutOfBoundsException](#)
- [OutOfRangeException](#)
- [OverflowException](#)
- [RangeException](#)
- [RuntimeException](#)
- [UnderflowException](#)
- [UnexpectedValueException](#)



**INSTITUTO
FEDERAL**
São Paulo

Exception Handling

Leitura complementar e referência:

<https://www.homehost.com.br/blog/tutoriais/php/php-try-catch/>



Exception Handling

Leitura complementar:

<https://rberaldo.com.br/tratamento-erros-php/>

Operador de controle de erro em PHP



Operador de controle de erros

O PHP suporta um **operador de controle de erro: o sinal 'arroba' (@)**. Quando ele precede uma expressão em PHP, qualquer mensagem de erro que possa ser gerada por aquela expressão será **ignorada**.



Operador de controle de erros

Se o recurso ***track_errors*** estiver habilitado, qualquer mensagem de erro gerada pela expressão será gravada na variável ***\$php_errormsg***. Esta variável será sobrescrita em cada erro, assim verifique-a constantemente se você quiser usá-la.

Operador de controle de erros

Exemplo, observe o seguinte código.

```
<?php  
  
$my_file = file('non_existent_file') or  
    die("Failed opening file: error was '" . error_get_last()['message'] . "'");  
  
?>  
</body>
```



Operador de controle de erros

Ele produzirá a seguinte saída:






Operador de controle de erros

No entanto é possível ocultar os erros usando o operador de controle de erros “@”:

```
<?php
    $my_file = @file('non_existent_file') or
        die("Failed opening file: error was '" . error_get_last()['message'] . "'");
?>
</body>
```



Operador de controle de erros

Observe agora o resultado:



Operador de controle de erros

Atenção!

Esta não é uma boa prática!

O intuito de apresentar este recurso é apenas para conhecimento, **seu uso não é recomendado!!**



**INSTITUTO
FEDERAL**
São Paulo

Operador de controle de erros

Inclusive não há muito material sobre o assunto, mesmo na internet.

Operador de controle de erros

Leitura recomendada e referência:

<https://www.php.net/manual/en/language.operators.errorcontrol.php>

<https://rberaldo.com.br/tratamento-erros-php/>

Dúvidas?

Perguntas?

Sugestões?