Aula 02 - PHP

Na última aula, aprendemos a fazer uma lista com php, usando uma conexão com um banco mysql e exibindo os dados em uma tabela. Esse exemplo mostrou vários conceitos importantes do PHP, como o uso de variáveis, a concatenação de variáveis e o conceito de GET. Para continuar nosso estudo do PHP, vamos ver mais alguns conceitos básicos que nos ajudarão a construir um site completo em PHP depois.

Integrando PHP com HTML

A integração do PHP com HTML é uma técnica essencial para desenvolvedores web. PHP é uma linguagem de script do lado do servidor que pode gerar HTML dinamicamente, permitindo a criação de conteúdo interativo e personalizado para os usuários.

Técnicas de Mistura

Inserção Direta

Você pode escrever código PHP diretamente dentro de um documento HTML usando as tags `<?php ... ?>`. O servidor processará o PHP e enviará o resultado como HTML para o navegador.

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de PHP com HTML</title>
</head>
<body>
    <h1><?php echo "Olá, mundo!"; ?></h1>
</body>
</html>
```

Echo de HTML

O `echo` pode ser usado para imprimir grandes blocos de HTML:

```
<?php
echo '<div>';
echo '<p>Este é um parágrafo gerado pelo PHP.</p>';
echo '</div>';
?>
```

Short Echo Tags

Se o `short_open_tag` estiver habilitado no seu servidor PHP, você pode usar a sintaxe `<?= ... ?>` para imprimir diretamente o resultado de uma expressão PHP:

```
<p>O ano atual é <?= date('Y'); ?></p>
```

HTML em Strings PHP

Você pode armazenar HTML em strings PHP e imprimi-las quando necessário:

```
<?php
$html = '<footer>Rodapé da página</footer>';
echo $html;
```

```
?>
```

Controle de Fluxo com HTML

Estruturas de controle de fluxo, como `if`, `for`, `foreach`, `while`, podem ser usadas para controlar a exibição de HTML:

```
<ul>
<?php for ($i = 0; $i < 5; $i++): ?>
    <li>Item <?= $i + 1 ?></li>
<?php endfor; ?>
</ul>
```

Boas Práticas

- **Separação de Código**: Mantenha a lógica PHP e o layout HTML separados tanto quanto possível para facilitar a manutenção.
- **Templates**: Considere usar um sistema de templates, como Twig ou Blade, para uma separação ainda mais clara entre lógica e apresentação.
- **Segurança**: Sempre escape saídas para evitar ataques XSS (Cross-Site Scripting).

Conclusão

Misturar PHP com HTML permite que os desenvolvedores criem páginas web dinâmicas e interativas. Compreender e aplicar essas técnicas é fundamental para o desenvolvimento web moderno.

Variáveis em PHP

PHP suporta vários tipos de variáveis, mas os principais são:

- **Inteiros**: São números sem decimais. Podem ser positivos ou negativos.

```
$inteiro = 42;
```

- **Ponto flutuante (floats)**: São números com decimais.

```
$float = 10.5;
```

- **Strings**: Sequências de caracteres, usadas para texto.

```
$string = "Olá, mundo!";
```

- **Booleanos**: Representam verdadeiro ou falso.

```
$verdadeiro = true;
$falso = false;
```

- **Arrays**: Coleções de valores, que podem ser de diferentes tipos.

```
$array = array(1, 2, "três", 4.5);
```

- **Objetos**: Instâncias de classes, que podem conter dados e métodos para manipular esses dados.

```
class Carro {
    public $cor;
```

```
    public function __construct($cor) {
        $this->cor = $cor;
    }
}

$objeto = new Carro("azul");
```

- **NULL**: Representa uma variável sem valor.

```
$nulo = NULL;
```

- **Recursos**: São variáveis especiais que mantêm referências a recursos externos, como arquivos.

```
$recurso = fopen("arquivo.txt", "r");
```

Cada tipo de variável é importante dependendo do contexto em que é usado. Por exemplo, inteiros são frequentemente usados em contagens, enquanto strings são essenciais para manipulação de texto. Arrays e objetos são estruturas de dados mais complexas que permitem organizar e manipular conjuntos de dados de maneira mais eficiente.

Concatenação de Variáveis em PHP

Em PHP, a concatenação de strings é realizada utilizando o operador ponto (.). Este operador permite unir duas ou mais strings em uma única string. A concatenação pode ser usada para construir mensagens dinâmicas, gerar código HTML de forma programática, ou simplesmente para juntar informações em uma única variável.

Exemplos Práticos de Concatenação

Concatenação Simples

Para concatenar duas strings, você pode fazer o seguinte:

```
$primeiraParte = "Olá, ";
$segundaParte = "mundo!";
$saudacao = $primeiraParte . $segundaParte;

echo $saudacao; // Saída: Olá, mundo!
```

Concatenação com Variáveis

Você também pode concatenar strings com variáveis:

```
$nome = "Maria";
$saudacao = "Bem-vinda, " . $nome . "!";

echo $saudacao; // Saída: Bem-vinda, Maria!
```

Concatenação na Prática

A concatenação é especialmente útil quando você precisa criar uma string baseada em várias condições ou fontes de dados:

```
$produto = "livro";
$preco = 20;

$mensagem = "O preço do " . $produto . " é R$" . $preco . ",00.";
```

```
echo $mensagem; // Saída: O preço do livro é R$20,00.
```

Concatenação com Operador de Atribuição

PHP também oferece um operador de concatenação combinado com atribuição (`.=`), que adiciona a string à variável existente:

```
$mensagem = "Você tem ";
$mensagem .= "uma nova mensagem.";

echo $mensagem; // Saída: Você tem uma nova mensagem.
```

Boas Práticas

- **Legibilidade**: Prefira quebrar linhas longas de concatenação para melhorar a legibilidade do código.
- **Espaçamento**: Inclua espaços em branco ao redor do operador de concatenação para distinguir claramente as diferentes partes da string.
- **Aspas Duplas**: Use aspas duplas se precisar avaliar variáveis dentro da string, mas lembre-se de que isso pode ser menos eficiente do que a concatenação direta.

Conclusão

A concatenação de strings é uma técnica poderosa e flexível em PHP, permitindo aos desenvolvedores manipular e combinar texto de maneira eficaz. Compreender e utilizar corretamente a concatenação pode levar a um código mais dinâmico e responsivo.

Entendendo `echo` e `print` em PHP

PHP oferece duas construções básicas para imprimir texto e variáveis na saída padrão: `echo` e `print`. Ambas são usadas frequentemente para exibir dados na tela, mas possuem algumas diferenças sutis.

`echo`

`echo` é uma construção da linguagem PHP que pode ser usada para exibir uma ou mais strings. É ligeiramente mais rápido do que `print` porque não retorna nenhum valor.

Exemplos de Uso do `echo`:

```
echo "Olá, mundo!";
```

Você também pode passar múltiplos argumentos para `echo` separados por vírgulas:

```
echo "Olá, ", "mundo", "!";
```

`print`

`print` é outra construção da linguagem PHP que realiza uma tarefa similar ao `echo`, mas só pode aceitar um único argumento e sempre retorna 1, o que significa que pode ser usado em expressões.

Exemplo de Uso do `print`:

```
print "Olá, mundo!";
```

Diferenças Chave

- **Retorno**: `print` retorna um valor (1), enquanto `echo` não retorna nada.
- **Argumentos**: `echo` pode aceitar múltiplos argumentos, `print` aceita apenas um.
- **Velocidade**: `echo` é marginalmente mais rápido do que `print`.

Quando Usar Cada Um?

A escolha entre `echo` e `print` geralmente se resume à preferência pessoal, já que a diferença de desempenho é mínima. No entanto, se você precisar verificar o sucesso da operação de impressão, `print` pode ser útil porque retorna 1. Por outro lado, se você quiser passar múltiplos argumentos sem concatenar, `echo` é a escolha ideal.

Conclusão

Tanto `echo` quanto `print` são ferramentas essenciais no arsenal de qualquer desenvolvedor PHP. Eles são simples, mas poderosos, e permitem uma comunicação eficaz entre o script PHP e o usuário final.

Aspas Simples vs. Aspas Duplas em PHP

No PHP, tanto as aspas simples (`'`) quanto as duplas (`"`) são usadas para definir strings, mas elas têm comportamentos ligeiramente diferentes que podem afetar como você escreve seu código.

Aspas Simples (`'`)

As aspas simples são usadas para definir uma string literal. Qualquer coisa entre aspas simples é tratada exatamente como está, sem interpretação de variáveis ou caracteres de escape (exceto `\\` e `\'`).

Exemplos:

```
$nome = 'Maria';
echo 'Olá, $nome'; // Saída: Olá, $nome
```

Note que a variável `$nome` não foi interpretada, e o PHP imprimiu o nome da variável como parte da string.

Aspas Duplas (`"`)

As aspas duplas interpretam variáveis e caracteres especiais dentro da string. Isso significa que o PHP substituirá variáveis pelo seu valor e interpretará sequências de escape como `\n` (nova linha) ou `\t` (tabulação).

Exemplos:

```
$nome = "Maria";
echo "Olá, $nome"; // Saída: Olá, Maria
```

Aqui, a variável `$nome` foi interpretada, e o PHP imprimiu o valor da variável.

Quando Usar Cada Uma?

Use Aspas Simples Quando:

- A string é literal e não requer variáveis.
- Você quer garantir que o conteúdo seja tratado exatamente como está escrito.

- Performance é uma preocupação (muito marginal), pois as aspas simples são ligeiramente mais rápidas.

Use Aspas Duplas Quando:

- Sua string inclui variáveis que você quer que sejam interpretadas.
- Você está usando caracteres de escape como `\n`, `\r`, `\t`, etc.
- Você quer incorporar expressões complexas dentro da string usando a sintaxe `${...}`.

Conclusão

Escolher entre aspas simples e duplas depende do contexto do seu código. Se você precisa de uma string simples e direta, aspas simples são a melhor escolha. Para strings mais complexas com variáveis e caracteres de escape, aspas duplas são mais apropriadas.

Operações Matemáticas Básicas em PHP

PHP é uma linguagem de programação rica em recursos para operações matemáticas. Desde operações simples até funções matemáticas complexas, PHP oferece uma ampla gama de funcionalidades. Vamos explorar as operações matemáticas básicas e como elas são realizadas em PHP.

Operações Aritméticas

Adição

A adição é realizada usando o operador `+`.

```
$soma = 3 + 4; // Resultado: 7
```

Subtração

A subtração usa o operador `-`.

```
$subtracao = 10 - 4; // Resultado: 6
```

Multiplicação

A multiplicação é feita com o operador `*`.

```
$multiplicacao = 5 * 2; // Resultado: 10
```

Divisão

A divisão utiliza o operador `/`.

```
$divisao = 15 / 3; // Resultado: 5
```

Módulo

O módulo (resto da divisão) é obtido com o operador `%`.

```
$modulo = 7 % 2; // Resultado: 1
```

Exponenciação

A exponenciação usa o operador `**`.

```
$exponenciacao = 2 ** 3; // Resultado: 8
```

Funções Matemáticas

PHP também oferece várias funções matemáticas integradas:

`abs()`

Retorna o valor absoluto de um número.

```
echo abs(-4.2); // Resultado: 4.2
```

`ceil()`

Arredonda números para cima.

```
echo ceil(4.3); // Resultado: 5
```

`floor()`

Arredonda números para baixo.

```
echo floor(4.9); // Resultado: 4
```

`round()`

Arredonda números para o inteiro mais próximo.

```
echo round(4.5); // Resultado: 5
```

`rand()`

Gera um número aleatório.

```
echo rand(1, 10); // Resultado: um número entre 1 e 10
```

Operações com Números Complexos

Para operações com números complexos, PHP oferece a extensão `ext/complex` (não nativa).

Conclusão

As operações matemáticas em PHP são simples e seguem a lógica matemática padrão. Combinadas com as funções matemáticas integradas, elas tornam o PHP uma ferramenta poderosa para cálculos e manipulações numéricas.

Caracteres de Escape em PHP

Caracteres de escape são uma parte fundamental de muitas linguagens de programação, incluindo PHP. Eles são usados para representar caracteres especiais dentro de strings, permitindo que você inclua caracteres que normalmente não seriam interpretados literalmente ou que não podem ser digitados diretamente.

O que são Caracteres de Escape?

Em PHP, um caractere de escape é precedido por uma barra invertida (`\`). Esta barra invertida indica que o caractere seguinte tem um significado especial ou deve ser interpretado de maneira diferente do usual.

Caracteres de Escape Comuns

Aqui estão alguns dos caracteres de escape mais comuns em PHP:

- `\n`: Nova linha
- `\r`: Retorno de carro
- `\t`: Tabulação horizontal
- `\\`: Barra invertida
- `\$`: Sinal de dólar
- `\"`: Aspas duplas
- `\'`: Aspas simples

Exemplos de Uso

Nova Linha e Tabulação

```
echo "Linha um.\nLinha dois.\n\tEste é um texto com tabulação.";
```

Aspas Duplas e Simples

Usando aspas duplas:

```
echo "Ele disse: \"Olá, mundo!\"";
```

Usando aspas simples:

```
echo 'Ele disse: \'Olá, mundo!\'';
```

Barra Invertida e Sinal de Dólar

```
echo "Caminho do arquivo: C:\\Pasta\\Subpasta\\arquivo.txt";
echo "Variável de custo: \$preco";
```

Quando Usar Caracteres de Escape

Você deve usar caracteres de escape sempre que precisar incluir caracteres especiais em uma string que de outra forma teriam um significado diferente para o interpretador PHP. Por exemplo, para incluir uma nova linha, aspas ou uma barra invertida dentro de uma string.

Conclusão

Entender e utilizar corretamente os caracteres de escape é essencial para o desenvolvimento em PHP, pois permite um controle mais preciso sobre o conteúdo das strings e a forma como são exibidas.

Casting de Tipos em PHP

O casting de tipos é uma operação importante em PHP que permite converter uma variável de um tipo para outro. Isso é útil quando você precisa garantir que uma variável seja tratada como um tipo específico para operações subsequentes.

Tipos de Dados em PHP

Antes de mergulhar no casting, é importante entender os tipos de dados em PHP:

- Inteiros (`int`)
- Ponto flutuante (`float`)
- Strings (`string`)
- Booleanos (`bool`)
- Arrays (`array`)
- Objetos (`object`)
- NULL

Como Realizar o Casting

O casting em PHP é feito colocando o tipo desejado entre parênteses antes da variável que você deseja converter.

Exemplos de Casting

De String para Inteiro

```
$variavelString = "10";
$variavelInteira = (int)$variavelString; // $variavelInteira é agora um inteiro 10
```

De Float para Inteiro

```
$variavelFloat = 3.14;
$variavelInteira = (int)$variavelFloat; // $variavelInteira é agora um inteiro 3
```

De Inteiro para String

```
$variavelInteira = 20;
$variavelString = (string)$variavelInteira; // $variavelString é agora "20"
```

De String para Booleano

```
$variavelString = "true";
$variavelBooleana = (bool)$variavelString; // $variavelBooleana é agora true
```

Casting Implícito vs. Explícito

PHP realiza casting implícito quando uma operação entre tipos diferentes é necessária. Por exemplo, ao adicionar um inteiro a uma string numérica, PHP converte a string para um inteiro automaticamente. No entanto, o casting explícito é feito pelo programador e é útil para garantir que o tipo de dado seja o esperado.

Boas Práticas

- Use casting explícito para evitar comportamentos inesperados em operações matemáticas ou lógicas.
- Prefira funções de conversão de tipo, como `intval()`, `floatval()`, e `strval()`, quando precisar de mais clareza no código.
- Lembre-se de que o casting pode levar à perda de dados, como quando um float é convertido para um inteiro.

Conclusão

O casting de tipos é uma ferramenta poderosa em PHP que oferece controle sobre como os dados são manipulados e apresentados. Usar casting de forma adequada pode ajudar a evitar erros e garantir que seu código funcione como esperado.

Exercícios Práticos de PHP

Exercício 1: Tipos de Variáveis

Crie um script PHP que declare variáveis para cada um dos tipos de dados em PHP e imprima o tipo de cada variável usando a função `gettype()`.

Exemplo de Saída:

```
O tipo da variável $inteiro é: integer
O tipo da variável $float é: double
...
```

Exercício 2: Concatenação de Strings

Escreva um script PHP que concatene várias strings para formar um endereço completo (rua, cidade, estado e CEP). Imprima o endereço completo em uma única string.

Exercício 3: Uso de `echo` e `print`

Crie um script PHP que use tanto `echo` quanto `print` para exibir uma mensagem na tela. Explique com comentários no código a diferença entre os dois.

Exercício 4: Misturando PHP com HTML

Desenvolva uma página HTML simples com formulários e use PHP para processar os dados do formulário. Mostre como o PHP pode ser inserido no meio do HTML para dinamizar o conteúdo da página.

Exercício 5: Aspas Simples vs. Aspas Duplas

Escreva um script PHP que demonstre a diferença entre usar aspas simples e aspas duplas. Inclua exemplos com variáveis e caracteres de escape.

Exercício 6: Operações Matemáticas

Crie um script PHP que realize e imprima o resultado de várias operações matemáticas, como adição, subtração, multiplicação, divisão e módulo. Inclua também o uso de algumas funções matemáticas como `abs()`, `ceil()`, `floor()`, e `round()`.

Exercício 7: Caracteres de Escape

Escreva um script PHP que utilize caracteres de escape para incluir aspas simples, aspas duplas, e uma nova linha em uma string. Imprima o resultado na tela.

Exercício 8: Casting de Tipos

Desenvolva um script PHP que demonstre o casting explícito de tipos. Crie variáveis de diferentes tipos e converta-as para outros tipos, imprimindo o valor e o tipo antes e depois do casting.

Exercícios Práticos de PHP

Exercício 1: Tipos de Variáveis

Crie um script PHP que declare variáveis para cada um dos tipos de dados em PHP e imprima o tipo de cada variável usando a função `gettype()`.

Resposta:

Script PHP que declara variáveis para cada um dos principais tipos de dados em PHP e imprime o tipo de cada variável usando a função `gettype()`:

```
<?php
// Inteiro
$inteiro = 42;
echo "O tipo da variável \$inteiro é: " . gettype($inteiro) . "\n";

// Ponto flutuante
$float = 10.5;
echo "O tipo da variável \$float é: " . gettype($float) . "\n";

// String
$string = "Olá, mundo!";
echo "O tipo da variável \$string é: " . gettype($string) . "\n";

// Booleano
$booleano = true;
echo "O tipo da variável \$booleano é: " . gettype($booleano) . "\n";

// Array
$array = array(1, 2, "três", 4.5);
echo "O tipo da variável \$array é: " . gettype($array) . "\n";

// Objeto
class Carro {
    public $cor;
    public function __construct($cor) {
        $this->cor = $cor;
    }
}
$objeto = new Carro("azul");
echo "O tipo da variável \$objeto é: " . gettype($objeto) . "\n";

// NULL
$nulo = NULL;
echo "O tipo da variável \$nulo é: " . gettype($nulo) . "\n";

// Recurso
$recurso = fopen("arquivo.txt", "r");
echo "O tipo da variável \$recurso é: " . gettype($recurso) . "\n";
fclose($recurso);
?>
```

Este script irá produzir a seguinte saída:

```
O tipo da variável $inteiro é: integer
O tipo da variável $float é: double
O tipo da variável $string é: string
O tipo da variável $booleano é: boolean
O tipo da variável $array é: array
O tipo da variável $objeto é: object
O tipo da variável $nulo é: NULL
```

```
O tipo da variável $recurso é: resource
```

Lembre-se de que o arquivo “arquivo.txt” deve existir no mesmo diretório do script para que o exemplo de recurso funcione corretamente. Caso contrário, você precisará criar o arquivo ou modificar o caminho conforme necessário.

Exercício 2: Concatenação de Strings

Escreva um script PHP que concatene várias strings para formar um endereço completo (rua, cidade, estado e CEP). Imprima o endereço completo em uma única string.

Resposta:

Script PHP que concatena várias strings para formar um endereço completo e imprime o resultado:

```
<?php
// Definindo as variáveis de endereço
$rua = "Av. Paulista";
$numero = "1000";
$cidade = "São Paulo";
$estado = "SP";
$cep = "01310-000";

// Concatenando as strings para formar o endereço completo
$enderecoCompleto = $rua . ", " . $numero . " - " . $cidade . ", " . $estado . " - CEP:
" . $cep;

// Imprimindo o endereço completo
echo $enderecoCompleto;
?>
```

Quando executado, este script PHP irá imprimir a seguinte string:

```
Av. Paulista, 1000 - São Paulo, SP - CEP: 01310-000
```

Você pode substituir os valores das variáveis com informações de endereço reais conforme necessário.

Exercício 3: Uso de `echo` e `print`

Crie um script PHP que use tanto `echo` quanto `print` para exibir uma mensagem na tela. Explique com comentários no código a diferença entre os dois.

Resposta:

Script PHP que utiliza tanto `echo` quanto `print` para exibir mensagens na tela, com comentários explicando a diferença entre os dois:

```
<?php
// Usando echo para imprimir uma mensagem.
// Echo pode aceitar múltiplos argumentos e é ligeiramente mais rápido que print.
echo "Olá, mundo com echo!";

// Usando print para imprimir uma mensagem.
// Print pode aceitar apenas um argumento e sempre retorna 1, o que significa que pode
ser usado em expressões.
print "Olá, mundo com print!";
?>
```

Quando executado, este script irá imprimir as mensagens “Olá, mundo com echo!” seguido por “Olá, mundo com print!” na tela. A principal diferença entre `echo` e `print` é que `echo` pode aceitar múltiplos argumentos e não retorna nenhum valor, enquanto `print` aceita apenas um argumento e retorna 1, o que o torna utilizável em expressões como parte de uma operação maior. Por exemplo, você pode usar `print` em uma instrução `if` porque ele retorna um valor, enquanto `echo` não pode ser usado dessa maneira. Além disso, `echo` é ligeiramente mais rápido do que `print`, mas na prática, essa diferença de desempenho é geralmente insignificante.

Exercício 4: Misturando PHP com HTML

Desenvolva uma página HTML simples com formulários e use PHP para processar os dados do formulário. Mostre como o PHP pode ser inserido no meio do HTML para dinamizar o conteúdo da página.

Resposta:

Exemplo simples de como você pode criar uma página HTML com um formulário e usar PHP para processar os dados do formulário. Este exemplo mostra como o PHP pode ser inserido no meio do HTML para tornar o conteúdo dinâmico:

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
    <title>Formulário Simples com PHP</title>
</head>
<body>

<!-- O formulário HTML onde os usuários podem inserir dados -->
<form action="processa_formulario.php" method="post">
    Nome: <input type="text" name="nome"><br>
    Email: <input type="text" name="email"><br>
    <input type="submit" value="Enviar">
</form>

<!-- Aqui começa o código PHP -->
<?php
// Verifica se os dados do formulário foram enviados
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // Atribui os dados do formulário a variáveis PHP
    $nome = $_POST['nome'];
    $email = $_POST['email'];

    // Imprime os dados processados
    echo "<h2>Dados Recebidos:</h2>";
    echo "<p>Nome: " . $nome . "</p>";
    echo "<p>Email: " . $email . "</p>";
}
?>

</body>
</html>
```

Neste exemplo, quando o usuário preenche o formulário e clica em “Enviar”, os dados são enviados para o mesmo arquivo PHP (`processa_formulario.php`). O PHP então verifica se os dados foram enviados via método POST. Se foram, ele processa os dados e os exibe na página.

Para que este exemplo funcione corretamente, você deve salvar o código em um arquivo chamado `processa_formulario.php` no servidor que suporta PHP. Quando o formulário for enviado, os dados inseridos pelo usuário serão exibidos na mesma página abaixo do formulário.

Lembre-se de que, em um ambiente de produção, você deve sempre validar e limpar os dados do formulário para evitar vulnerabilidades de segurança, como ataques de injeção de SQL ou scripts entre sites (XSS). Este exemplo é apenas para fins educacionais e não inclui medidas de segurança.

Exercício 5: Aspas Simples vs. Aspas Duplas

Escreva um script PHP que demonstre a diferença entre usar aspas simples e aspas duplas. Inclua exemplos com variáveis e caracteres de escape.

Resposta:

Script PHP que ilustra a diferença entre o uso de aspas simples e aspas duplas, incluindo exemplos com variáveis e caracteres de escape:

```
<?php
// Variável para demonstração
$variavel = 'mundo';

// Exemplo com aspas simples
// As variáveis e caracteres de escape não são interpretados
echo 'Olá, $variavel\n';

// Exemplo com aspas duplas
// As variáveis e caracteres de escape são interpretados
echo "Olá, $variavel\n";

// Caracteres de escape com aspas simples
// Apenas a barra invertida e as próprias aspas simples são escapadas
echo 'Linha com barra invertida: \\ e aspas simples: \'';

// Caracteres de escape com aspas duplas
// Vários caracteres de escape são interpretados, como nova linha (\n) e tab (\t)
echo "Linha com nova linha: \n e tabulação: \t";
?>
```

Explicação:

- Com aspas simples, o PHP trata o conteúdo literalmente. A variável `$variavel` e a sequência `\n` não são interpretadas e são exibidas como texto puro.
- Com aspas duplas, o PHP interpreta a variável `$variavel` e substitui pelo seu valor, e a sequência `\n` é interpretada como uma nova linha.
- Caracteres de escape específicos como `\\` e `\'` são necessários dentro de aspas simples para exibir uma barra invertida e aspas simples, respectivamente.
- Dentro de aspas duplas, caracteres de escape como `\n` (nova linha) e `\t` (tabulação) são interpretados e executam suas funções correspondentes na saída.

Este script ajudará a entender quando usar cada tipo de aspas no PHP e como eles afetam a interpretação de strings.

Exercício 6: Operações Matemáticas

Crie um script PHP que realize e imprima o resultado de várias operações matemáticas, como adição, subtração, multiplicação, divisão e módulo. Inclua também o uso de algumas funções matemáticas como `abs()`, `ceil()`, `floor()`, e `round()`.

Resposta:

Script PHP que realiza várias operações matemáticas e imprime os resultados, incluindo o uso de funções matemáticas como `abs()`, `ceil()`, `floor()`, e `round()`:

```
<?php
// Definindo algumas variáveis para as operações
$num1 = 25;
$num2 = 7;

// Adição
$soma = $num1 + $num2;
echo "A soma de $num1 e $num2 é: $soma\n";

// Subtração
$subtracao = $num1 - $num2;
echo "A subtração de $num1 por $num2 é: $subtracao\n";

// Multiplicação
$multiplicacao = $num1 * $num2;
echo "A multiplicação de $num1 por $num2 é: $multiplicacao\n";

// Divisão
$divisao = $num1 / $num2;
echo "A divisão de $num1 por $num2 é: $divisao\n";

// Módulo
$modulo = $num1 % $num2;
echo "O módulo de $num1 por $num2 é: $modulo\n";

// Funções matemáticas
$numeroNegativo = -15;
$numeroDecimal = 2.8;

// Valor absoluto
echo "O valor absoluto de $numeroNegativo é: " . abs($numeroNegativo) . "\n";

// Arredondamento para cima
echo "O arredondamento de $numeroDecimal para cima é: " . ceil($numeroDecimal) . "\n";

// Arredondamento para baixo
echo "O arredondamento de $numeroDecimal para baixo é: " . floor($numeroDecimal) . "\n";

// Arredondamento para o inteiro mais próximo
echo "O arredondamento de $numeroDecimal para o inteiro mais próximo é: " .
round($numeroDecimal) . "\n";
?>
```

Este script PHP define duas variáveis, `$num1` e `$num2`, e realiza operações de adição, subtração, multiplicação, divisão e módulo com elas. Em seguida, ele usa as funções `abs()`, `ceil()`, `floor()`, e `round()` para demonstrar o cálculo do valor absoluto, arredondamento para cima, arredondamento para baixo e arredondamento para o inteiro mais próximo, respectivamente. Os resultados de cada operação são impressos na tela.

Exercício 7: Caracteres de Escape

Escreva um script PHP que utilize caracteres de escape para incluir aspas simples, aspas duplas, e uma nova linha em uma string. Imprima o resultado na tela.

Resposta:

Script PHP que utiliza caracteres de escape para incluir aspas simples, aspas duplas e uma nova linha em uma string. O resultado é então impresso na tela:

```
<?php
// String com aspas simples, aspas duplas e uma nova linha
echo "Ele disse: \"Olá, \$variavel!\"\nEla respondeu: 'Tudo bem?'";

// Resultado na tela:
// Ele disse: "Olá, $variavel!"
// Ela respondeu: 'Tudo bem?'
?>
```

Neste exemplo, a barra invertida (`\`) é usada como caractere de escape para permitir a inclusão de aspas duplas e simples na string sem encerrar a string prematuramente. Além disso, `\n` é usado para criar uma nova linha na saída. Quando você executa este script PHP, ele imprimirá as frases com as aspas corretamente exibidas e em linhas separadas.

Exercício 8: Casting de Tipos

Desenvolva um script PHP que demonstre o casting explícito de tipos. Crie variáveis de diferentes tipos e converta-as para outros tipos, imprimindo o valor e o tipo antes e depois do casting.

Resposta:

Script PHP que demonstra o casting explícito de tipos. O script cria variáveis de diferentes tipos e as converte para outros tipos, imprimindo o valor e o tipo antes e depois do casting:

```
<?php
// Variável inteira
$inteiro = 10;
echo "Valor original: $inteiro, Tipo: " . gettype($inteiro) . "\n";
$inteiroParaFloat = (float)$inteiro;
echo "Valor após casting: $inteiroParaFloat, Tipo: " . gettype($inteiroParaFloat) .
"\n\n";

// Variável float
$float = 3.14;
echo "Valor original: $float, Tipo: " . gettype($float) . "\n";
$floatParaInteiro = (int)$float;
echo "Valor após casting: $floatParaInteiro, Tipo: " . gettype($floatParaInteiro) .
"\n\n";

// Variável string
$string = "4.5";
echo "Valor original: $string, Tipo: " . gettype($string) . "\n";
$stringParaInteiro = (int)$string;
echo "Valor após casting: $stringParaInteiro, Tipo: " . gettype($stringParaInteiro) .
"\n";
$stringParaFloat = (float)$string;
echo "Valor após casting: $stringParaFloat, Tipo: " . gettype($stringParaFloat) . "\n\n";

// Variável booleana
$booleano = true;
```

```
echo "Valor original: $booleano, Tipo: " . gettype($booleano) . "\n";
$booleanoParaString = (string)$booleano;
echo "Valor após casting: $booleanoParaString, Tipo: " . gettype($booleanoParaString) .
"\n";
?>
```

Este script irá imprimir o valor e o tipo de cada variável antes e depois do casting, demonstrando como o valor pode mudar quando é convertido de um tipo para outro. Por exemplo, ao converter um float para um inteiro, a parte decimal é descartada. Ao converter uma string numérica para um inteiro ou float, o PHP tentará interpretar o número dentro da string. E ao converter um booleano para uma string, `true` se torna “1” e `false` se torna “” (uma string vazia).