

A presentation slide titled "Tipos de dados". The top bar is blue with the Softblue logo. The main content area is white and contains a bulleted list of PHP data types with corresponding code examples in blue boxes: "String: textos e caracteres" with "\$var = 'Curso de PHP';", "Número: números inteiros e decimais" with "\$var = 50.03;", "Data: datas e horas em diversos formatos" with "\$var = date('Y-m-d');", "Booleano: verdadeiro ou falso" with "\$var = TRUE;", and "Objeto: qualquer tipo de objeto" with "\$var = array();".

Variáveis

- Variáveis utilizam o símbolo \$
 - Para atribuir valores em variáveis utilize o símbolo de igual (=)

```
$meuNome = "Andre";
$ano = 1982;

echo $meuNome." nasceu em ".$ano;
// Resultado: Andre nasceu em 1982
```
- Verificando se uma variável foi setada
 - Função isset(var) retorna valor booleano indicando se a variável está setada

```
echo isset($empresa);
// Resultado: Vazio (falso)

$empresa = "Softblue";
echo isset($empresa);
// Resultado: 1 (verdadeiro)
```

Constantes

- Definindo constantes
 - A função define(*nome*, *valor*) define uma constante no PHP
 - Em constantes não utiliza-se o símbolo \$

```
define("PI", 3.14);
define("NOME_EMPRESA", "Softblue");
```
- Acessando constantes


```
echo PI * 2;
// Resultado: 6.28

echo "Conheça a ".NOME_EMPRESA;
// Resultado: Conheça a Softblue
```

Operadores e comandos

- Operadores matemáticos
- Operadores condicionais
- Operadores lógicos
- Comandos de controle
- Comandos de repetição
- Tabelas disponíveis para download

Operadores matemáticos



- = Atribui o valor da sua direita à variável a sua esquerda
`$x = 3;` // Resultado: \$x armazena o valor 3
- . Concatena dois valores de formato texto (string) ou numéricos
`$x = "Andre"."Milani";` // Resultado: \$x armazena o valor "Andre Milani"
- + Soma dois valores numéricos
`$x = 3 + 5;` // Resultado: \$x armazena o valor 8
- Subtrai dois valores numéricos
`$x = 3 - 5;` // Resultado: \$x armazena o valor -2
- * Multiplica dois valores numéricos
`$x = 2 * 6;` // Resultado: \$x armazena o valor 12
- / Divide dois valores numéricos
`$x = 18 / 3;` // Resultado: \$x armazena o valor 6
- % Obtém o resto da divisão entre dois valores numéricos
`$x = 19 % 3;` // Resultado: \$x armazena o valor 1

Operadores matemáticos



- ++ Incrementa em 1 o valor da variável acoplada
`$x = 3;`
`$x++;` // Resultado: \$x neste momento armazena o valor 4
- Decrementa em 1 o valor da variável acoplada
`$x = 3;`
`$x--;` // Resultado: \$x neste momento armazena o valor 2
- É possível utilizar estes operadores em expressões e outros comandos
`$x = 3;`
`$y = 2 + $x++;` // Resultado: \$y = 5 e \$x = 4
- Diferença entre ++\$x e \$x++
`$x = 3;`
`$y = 2 + ++$x;` // Resultado: \$y = 6 e \$x = 4

Operadores matemáticos



Assumir \$x = 5 para o início de cada exemplo apresentado neste slide

- . = Concatena à variável à sua esquerda o valor apresentado em sua direita
`$x .= 3;` // Resultado: \$x armazena o valor 53
- += Soma à variável à sua esquerda o valor apresentado em sua direita
`$x += 3;` // Resultado: \$x armazena o valor 8
- = Subtrai da variável à sua esquerda o valor apresentado em sua direita
`$x -= 3;` // Resultado: \$x armazena o valor 2
- * = Multiplica a variável à sua esquerda pelo valor apresentado em sua direita
`$x *= 3;` // Resultado: \$x armazena o valor 15
- / = Divide à variável à sua esquerda o valor apresentado em sua direita
`$x /= 3;` // Resultado: \$x armazena o valor 1.6666666666667
- % = Atribui à variável da esquerda o resto de sua divisão pelo valor apresentado em sua direita
`$x %= 3;` // Resultado: \$x armazena o valor 2

Arredondamento de números



- `round(valor[, precisão])`

– Permite arredondar um número de diversas formas diferentes

```
echo round(5.5, 0). "<br>"; // 6
echo round(5.5, 1). "<br>"; // 5.5
echo round(5.5, 2). "<br>"; // 5.5
echo round(5.5, 3). "<br>"; // 5.5

echo round(5.55, 0). "<br>"; // 6
echo round(5.55, 1). "<br>"; // 5.6
echo round(5.55, 2). "<br>"; // 5.55
echo round(5.55, 3). "<br>"; // 5.55
```

Operadores condicionais



- Utilizados para avaliar uma condição

Operador	Significado
<code>==</code>	Compara se dois valores tem o mesmo valor, mesmo em tipos distintos
<code>===</code>	Compara se dois valores são idênticos
<code>!=</code>	Compara se dois valores são diferentes
<code>!==</code>	Compara se dois valores são diferentes inclusive no tipo de dado
<code><></code>	Equivalente à <code>!=</code>
<code><</code>	Compara se o valor da esquerda é menor que o da direita
<code>></code>	Compara se o valor da esquerda é maior que o da direita
<code><=</code>	Compara se o valor da esquerda é menor ou igual que o da direita
<code>>=</code>	Compara se o valor da esquerda é maior ou igual que o da direita
<code><=></code>	Comparador especial: 0 se igual, -1 se menor, 1 se maior

Operadores lógicos



- Utilizados para montar uma condição

Operador	Significado
<code>!</code>	Nega a condição informada (inverte seu resultado lógico)
<code>&</code>	Retorna verdadeiro se ambas as condições da esquerda e direita forem satisfeitas
<code> </code>	Retorna verdadeiro se pelo menos uma das condições (da direita ou da esquerda) for satisfeita
<code>&&</code>	O mesmo que <code>&</code> , porém é otimizado: Se a condição do lado esquerdo for falsa, não verifica o lado direito da expressão. Resulta a operação toda em FALSA neste caso.
<code> </code>	O mesmo que <code> </code> , porém é otimizado: Se a condição do lado esquerdo for verdadeira, não verifica o lado direito da expressão. Resulta a operação toda em VERDADEIRA neste caso.

Comandos de controle e repetição



- `if(condição) {} [else if(condição) {}] [else {}]`
 - Permite executar um bloco de código se determinada condição for verdadeira

```
if(3 > 5) {
    echo('Não entra no primeiro if. ');
}

if(1 < 10) {
    echo('Entra no segundo if. ');
}
else {
    echo('Não entra no else do segundo if. ');
}
```

```
$i = 5;
if($i == 3)
{
    echo('O valor de i é 3. ');
}
else if($i == 4)
{
    echo('O valor de i é 4. ');
}
else {
    echo('O valor de i não é 3 nem 4. '); }
}
```

Comandos de controle e repetição



- Short if (if reduzido)
 - Forma reduzida de escrever o comando if
 - Deve ser utilizado somente em casos onde o bloco de código for curtíssimo
- `condição ? bloco_true : bloco_false;`

```
$x = 5;

if($x % 2 == 0) {
    $y = "Par";
}
else {
    $y = "Ímpar";
}

echo $y; // Ímpar
```

```
$x = 5;
$y = $x % 2 == 0 ? $y = "Par" : $y = "Ímpar";
echo $y; // Ímpar
```

Comandos de controle e repetição



- `switch(valor) { cases }`
 - Permite executar um bloco de código específico dependendo do valor avaliado

```
$i = 1;
switch($i)
{
    case 0:
        echo ('O valor de i é 0');
        break;
    case 1:
        echo ('O valor de i é 1');
        break;
    case 2:
        echo ('O valor de i é 2');
        break;
    default:
        echo ('Nenhum');
        break;
}
```

Comandos de controle e repetição

• **for**(inicialização; condição; incremento)

- Permite criar um laço de repetição, executando para cada valor o mesmo bloco de código

```
for($i = 0; $i < 10; $i++)  
{  
    echo($i.' ');  
}  
  
// Resultado: 0 1 2 3 4 5 6 7 8 9
```

Comandos de controle e repetição

• **foreach**(array as parte)

- Permite navegar por todos os registros de um array executando o mesmo bloco de código para cada registro do array

```
$meuArray = array('a', 'b', 'c', 'd');  
  
foreach($meuArray as $valor)  
{  
    echo($valor.' ');  
}  
  
// Resultado: a b c d
```

Comandos de controle e repetição

• **while**(condição)

- Permite executar várias vezes um bloco de código enquanto sua condição for verdadeira

```
$i = 0;  
  
while($i < 5)  
{  
    echo($i.' ');  
    $i++;  
}  
  
// Resultado: 0 1 2 3 4
```

Comandos de controle e repetição



• do { } while (condição);

- Similar ao while, com a diferença de que o do while tem o bloco de código executado obrigatoriamente uma vez antes de avaliar a condição

```
$i = 0;
do
{
    echo($i . ' ');
    $i++;
}
while($i < 5);
// Resultado: 0 1 2 3 4
```

Comandos de controle e repetição



• break

- Permite interromper suspender o comando de repetição e ir para o próximo comando

```
for($i = 0; $i < 10; $i++)
{
    if($i == 4)
    {
        break;
    }
    echo($i . ' ');
}
// Resultado: 0 1 2 3
```

Comandos de controle e repetição



• continue

- Permite instruir ao comando de repetição que interrompa esta iteração e avance para a próxima iteração da repetição

```
for($i = 0; $i < 10; $i++)
{
    if($i == 4)
    {
        continue;
    }
    echo($i . ' ');
}
// Resultado: 0 1 2 3 5 6 7 8 9
```

Comandos de controle e repetição



- goto

- Permite direcionar o fluxo do PHP para um bloco de código específico
- Este recurso só está disponível a partir da versão 5.3 do PHP

```
echo('Código iniciando...');
goto saindo;

echo('Código executando...');

saindo:
echo('Código encerrando...');
```

Comandos de controle e repetição



- exit (antigo die)

- Permite encerrar a execução do arquivo PHP

```
$arquivo = '/caminho/inexistente';
$file = fopen($arquivo, 'r') or exit('Problemas ao abrir o arquivo');


// Resultado:
// Warning: fopen(/caminho/inexistente) [function.fopen]: failed to open
// stream: No such file or directory in /home/httpd/htdocs/teste.php on
// line 4
// Problemas ao abrir o arquivo
```

Supervariáveis



Nome	Descrição
\$GLOBALS	Variáveis de escopo global
\$_SERVER	Variáveis do servidor
\$_GET	Variáveis enviadas por HTTP pelo método GET
\$_POST	Variáveis enviadas por HTTP pelo método POST
\$_FILES	Variáveis de upload de arquivos
\$_REQUEST	Variáveis de URL
\$_SESSION	Variáveis de sessão
\$_ENV	Variáveis de ambiente
\$_COOKIE	Variáveis de cookies
\$php_errormsg	Armazena a última mensagem de erro ocorrida
\$argc	Armazena o número de argumentos passados para o script PHP
\$argv	Array de argumentos passados para o script PHP

Supervariável \$_SERVER



SERVER_ADDR

```
echo $_SERVER["SERVER_ADDR"]; // Resultado: 127.0.0.1
```

SERVER_NAME

```
echo $_SERVER["SERVER_NAME"]; // Resultado: localhost
```

HTTP_ACCEPT_ENCODING

```
echo $_SERVER["HTTP_ACCEPT_ENCODING"]; // Resultado: gzip, deflate, sdch
```


HTTP_USER_AGENT

```
echo $_SERVER["HTTP_USER_AGENT"]; // Resultado: Mozilla/5.0
// (Windows; U; Windows NT 5.1; en-US) AppleWebKit/534.3
// (KHTML, like Gecko) Chrome/6.0.472.63 Safari/534.3
```

REMOTE_ADDR

```
echo $_SERVER["REMOTE_ADDR"]; // Resultado: 127.0.0.1
```

Criando funções



- Permite centralizar blocos de código

Sintaxe

```
function nomeDaFunção([parâmetros]): tipo de retorno { }
```


Exemplo

```
function calcularDobro(int $num): int
{
    $dobro = $num * 2;
    return $dobro;
}

$i = 5;
$iDobro = calcularDobro($i);
echo "O dobro de ".$i." é ".$iDobro;

// Resultado: O dobro de 5 é 10
```

Includes e requires



- Comandos que permitem incluir ou importar outros arquivos PHP e executá-los
- Organização de código
- Centralização de código
- Facilidade de manutenção
- Facilidade de reutilização entre diferentes projetos
- Otimização de recursos do servidor

Includes e requires

• Como funciona

```

MeuArquivo1.php
include("MinhasFuncoes.php")
$i = 5;
$idobro = calcularDobro($i);
echo "O dobro de ".$i." é ".$idobro;

MeuArquivo2.php
include("MinhasFuncoes.php");
$i = 5;
$idobro = calcularDobro($i);
echo "O dobro de ".$i." é ".$idobro;

MinhasFuncoes.php
function calcularDobro($num) {
    $dobro = $num * 2;
    return $dobro;
}
  
```

Includes e requires

• include

- Inclui um arquivo, continuando a execução mesmo que o arquivo não seja localizado

• require

- Inclui um arquivo, cancelando a execução caso o arquivo não seja localizado

• include_once

- Inclui um arquivo somente se não tiver sido incluído ainda, continuando a execução mesmo que o arquivo não seja localizado

• require_once

- Inclui um arquivo somente se não tiver sido incluído ainda, cancelando a execução caso o arquivo não seja localizado

Redirecionamento de fluxo

• header

- Permite definir parâmetros e comandos do cabeçalho


• Redirecionando a requisição

- Location

```
header("Location: http://www.google.com.br");
```

IMPORTANTE
Só é possível manipular o cabeçalho por meio do comando header se nenhuma informação tiver sido impressa na tela, pelo arquivo PHP, antes do comando ser utilizado.

Redirecionamento de fluxo



- Exemplo de mal uso do header



```
include("outroArquivo");
echo "Olá usuário(a)!";
header("Location: http://www.google.com.br");
```

IMPORTANTE
Ao realizar includes ou comandos de impressão na tela antes do comando header, erros do tipo *Cannot modify header information ou Headers already sent* podem ocorrer.

- Exemplo de bom uso do header


```
header("Location: http://www.google.com.br");
include("outroArquivo");
echo "Olá usuário(a)!";
```



Redirecionamento de fluxo



- Solução alternativa (ob_start e ob_flush)
 - ob_start
 - ob_flush
- Exemplo


```
ob_start();
include("outroArquivo");
echo "Olá usuário(a)!";
header("Location: http://www.google.com.br");
ob_flush();
```

Aulas práticas e manuais on-line

Assista agora as aulas práticas, que apresentam o uso dos comandos abordados nesta aula teórica.

[Clique aqui](#) para visualizar as aulas práticas disponíveis
