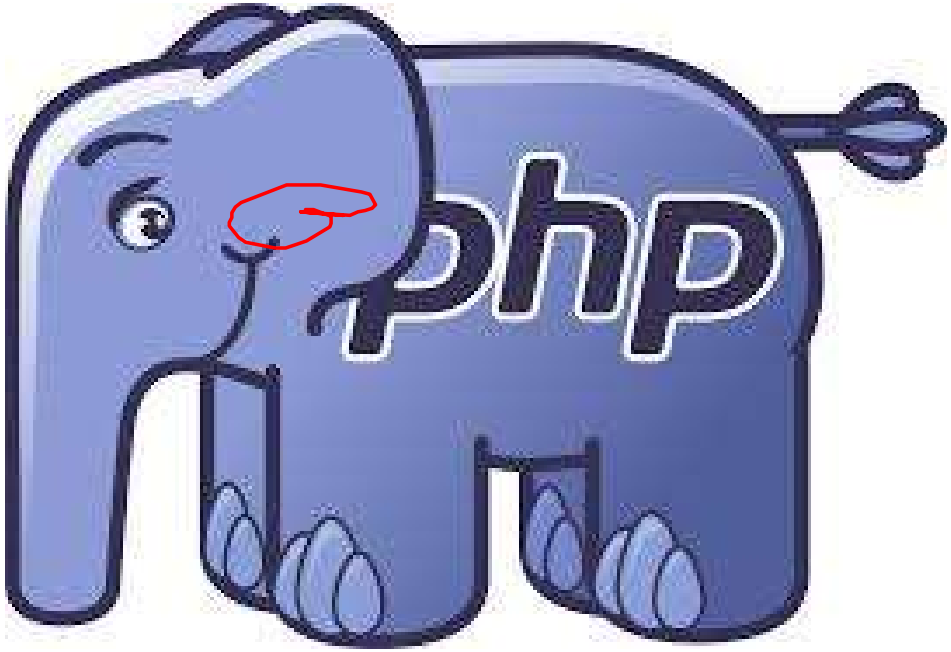
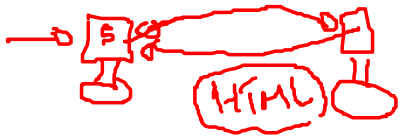


AULA 1

CONCEITOS BÁSICOS E CARACTERÍSTICAS

O PHP é uma das linguagens mais utilizadas na Web. Hoje mais de 10 milhões de sites no mundo inteiro utilizam PHP. A principal diferença em relação às outras linguagens é a capacidade que o PHP tem de interagir com o mundo Web, transformando totalmente os websites que possuem páginas estáticas.

PÁGINAS DINÂMICAS



CONCEITOS BÁSICOS E CARACTERÍSTICAS

PAG. HTML

PAG. PHP

- É Gratuito com código aberto

- Embutido no HTML

Outra característica do PHP é que ele é embutido no HTML. Veremos mais adiante as facilidades que isso pode nos trazer. Uma página que contém programação PHP normalmente possui extensão *.php* (isso depende da configuração do seu servidor Web). Sempre que o servidor Web receber solicitações de páginas que possuem essa extensão, ele saberá que essa página possui linhas de programação. Porém, você verá que o HTML e o PHP estão misturados, pois começa a escrever em PHP, de repente escreve um trecho em HTML, depois volta para o PHP, e assim por diante.

```
<html>
<body>
<?php
    // Legal, estou escrevendo meu primeiro programa em PHP
    echo "<h2 align='center'>Parabéns para mim!</h2>";
?>
</body>
</html>
```

- Pode ser trabalhado com Orientação a Objetos

ARQ.. HTML

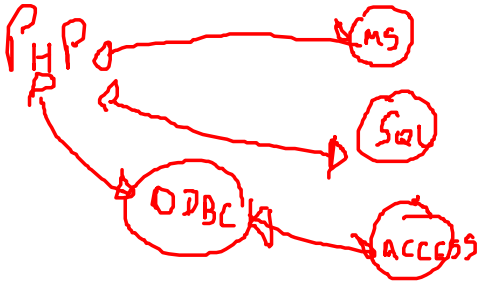
ARQ.PHP

- Portabilidade

Podemos executar o PHP no Linux, Unix ou Windows NT. Vamos falar mais sobre a utilização do PHP no Linux, embora haja poucas diferenças em relação aos demais sistemas operacionais.

Note que o browser não recebe nenhuma linha em PHP, somente recebe código HTML puro, pois, como já vimos, o PHP roda no servidor. Toda a programação PHP é processada no servidor, que retorna somente o resultado final para seu browser.

Dica: quando as páginas possuem extensão .html, o servidor Web as tratará como HTML puro, e não reconhecerá códigos PHP. Se a página possuir extensão *.php*, o servidor Web ativará o processador de hipertexto do PHP para verificar linha a linha em busca de códigos de programação, por isso o processo fica um pouco mais lento. A dica é a seguinte: só coloque extensão *.php* nas páginas que realmente possuem códigos PHP, senão você estará gastando um tempo desnecessário, procurando a cada linha códigos que não existem na página.



Gratuito e Código Aberto

Embutido no HTML

Suporte a diversos Bancos de Dados

MySQL, SQLite, PostgreSQL, Sybase, Oracle, SQLServer e outros por ODBC.

→ APLICAÇÃO DA CONEXÃO COM SGBD NÃO NATIVO/ACEITO PELO PHP

Portabilidade

Linux, Windows, MacOS

Primeiro Programa

OBS: CONSTANTE NÃO USA $

```
<html>
  <body>
    <?php
      // Meu primeiro programa
      $mensagem = "<h1>Olá, Mundo!</h1>";
      echo $mensagem;
    ?>
  </body>
</html>
```

PARA VARIÁVEL

```
<html>
  <body>
    <h1>Olá, Mundo!</h1>
  </body>
</html>
```

Segundo Programa

```
<html>
  <body>
    <?php
      $hoje = date ("d/m/Y", time());
    ?>
    <p>Hoje é dia <?php echo $hoje ?>!!!</p>
  </body>
</html>
```

```
<html>
  <body>
    <p>Hoje é dia 07/03/2015!!!</p>
  </body>
</html>
```

```php
<html>
<body>
<?php
    $time = "Grêmio";
    $ano = 1983;
    $frase1 = "O $time é o melhor clube de futebol do mundo!";
    $frase2 = "O $time foi campeão do mundo em $ano";
    echo "<h3>$frase1</h3>";
    echo "<h3>$frase2</h3>";
?>
</body>
</html>
```

DINAMICAMENTE TIPADO

CASE SENSITIVE

NÃO PRECISA DECLARAR VARIÁVEL!

GRÊMIO

1993

DUPLAS

IMPORTANTE

<?php ?> super tag PHP (padrão)

<? ?> short tags (alterar php.ini)

<% %> ASP tags (alterar php.ini)

Constantes & Variáveis

Variáveis

- Começa com $
- Próximo deve ser letra
- Pode conter letras ou números
- Não pode conter espaço
- Não pode conter acentos
- Não pode conter símbolos (exceto underline)
- Não pode ser palavra reservada

* PHP é case-sensitive

```
$nome = "Gustavo";
echo "Eu sou $NOME";
```

Constantes

```
define("curso","EXPLICADORES");
echo "Eu estudo no" . curso;
```

CONCATENAÇÃO

EXPLICADORES

```
const NOME = 'Alex';
```

OUTRA FORMA DE CRIAR CONSTANTES;

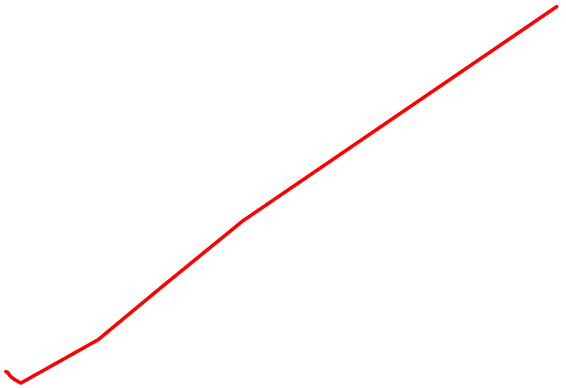
POO

ORIENTADO A OBJETOS

Constantes & Variáveis

```php
<?php
    $x = "tri";
    echo "Eu sou ${x}color";
?>
```

```php
<?php
    $x = "tri";
    echo "Eu sou " . $x . "color";
?>
```



Constantes & Variáveis

Além de você poder definir suas próprias constantes, o PHP já possui diversas constantes próprias predefinidas. Vamos conhecer algumas na tabela a seguir:

Constante	Descrição
TRUE	Valor verdadeiro (utilizado para comparação).
FALSE	Valor falso.
__FILE__	Contém o nome do script que está sendo executado.
__LINE__	Contém o número da linha do script que está sendo executado.
PHP_VERSION	Contém a versão corrente do PHP.
PHP_OS	Nome do sistema operacional no qual o PHP está rodando.
E_ERROR	Exibe um erro ocorrido em um script. A execução é interrompida.
E_WARNING	Exibe uma mensagem de aviso do PHP. A execução não pára.
E_PARSE	Exibe um erro de sintaxe. A execução é interrompida.
E_NOTICE	Mostra que ocorreu algo, mas não necessariamente um erro. A execução não pára.

$A = "ALEX";
ECHO "MEU NOME É $A"; MEU NOME É ALEX
ECHO " MEU NOME É \$A"; MEU NOME É $A

$NOME = "ALEX";

Caractere

Aspas Duplas - Interpolação de variáveis

```
echo "meu nome é $nome";
```

MEU NOME É ALEX

Aspas Simples - Sem interpolação de variáveis

```
echo 'meu nome é $nome';
echo 'meu nome é ' . $nome;
```

MEU NOME É $NOME

CONCATENAR

Aspas Invertidas - Executa comandos de ambiente

```
echo `ls -l *.html`;
```

3 40

Alfanuméricas (strings)

São cadeias de caracteres que, conforme vimos anteriormente, podem ser delimitadas por aspas simples (') ou aspas duplas ("). Quando utilizamos aspas duplas, podemos incluir caracteres de controle no meio da string. Exemplos:

```
$nome = 'Claudiomar';
$profissao = "Pedreiro";
$texto = "Boa tarde!\nSeja bem-vindo ao meu site!"
```

LITERAL

ECHO 'MEU NOME É $A';

MEU NOME É $A

ECHO " MEU NOME É : $A;

MEU NOME É ALEX

PULAR DE LINHA

Tipos Primitivos

Numéricas

Inteiro

```
$valor = 10;
```

Reais

```
$pi = 3.1415;
```

Hexadecimal

```
$valor = 0x0b;
```

Tipos das variáveis

No PHP existem variáveis dos tipos numéricas, alfanuméricas (strings), arrays e objetos. Vamos ver cada um desses tipos:

Numéricas

As variáveis numéricas podem possuir valores inteiros ou reais (ponto flutuante). Uma variável é definida como numérica no momento em que atribuímos um valor numérico a ela. Veja alguns exemplos:

```
$numero = 10;
$x = 5;
$numero_hexa = 0x0b;    // valor em hexadecimal
$y = 15.002;
$a = 200.3;
```

Tipos Primitivos

Lógicas

```
$aprovado = True;
$casado = False;
```

* False é considerado como VAZIO
* True é considerado como 1

ESTRUTURA DE DADOS!

Arrays

```
$aluno [0] = 8.0;
$aluno [1] = 9.0;
$aluno [2] = 2.5;
```

Arrays

As variáveis comuns (também chamadas de variáveis escalares) podem armazenar apenas um valor por vez. Um array (vetor) pode armazenar vários valores ao mesmo tempo, pois se trata de uma estrutura de armazenamento que, assim como as variáveis, possui um identificador, mas além disso há um índice associado (que pode ser um número ou um texto), e cada índice indica uma posição de memória em que fica armazenado um elemento do array. O índice deve aparecer entre colchetes ([]) logo após o identificador do array.

```
$aluno [] = 8.7;    // Acrescenta ao próximo espaço livre

$notas = array range (1, 10);

$notas = array (8.5, 7.0, 4.5, 8.0);

$cadastro ["nome"] = "José";
$cadastro ["idade"] = 23;
$cadastro ["peso"] = 93.5;   // Chave associativa

$pessoa = array ("nome" => "Paulo", "idade" =>30);
```

Objetos

Veremos mais adiante o que são classes. Dentro das classes temos funções definidas. Criamos uma variável para instanciar uma classe, e essa variável é chamada de objeto. Um objeto pode acessar funções definidas dentro de uma classe. Se você alguma vez já estudou programação orientada a objetos, esse conceito deve lhe ser familiar. Veja um pequeno exemplo:

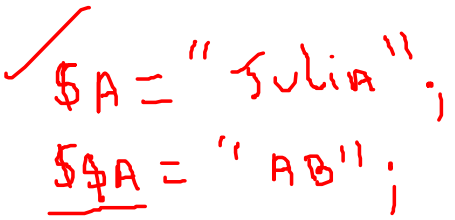
exemplo4_13.php

```
<?php
    class Teste
    {
        function Saudacao()  {
        echo "Oi pessoal!";
        }
    }
    $objeto = new Teste;  // $objeto se torna uma instância da classe Teste
    $objeto -> Saudacao();
?>
```

Variáveis durante execução

```
$resultado = "aprovado";
$tipo = "aluno";
$$tipo = $resultado;

echo "$aluno";
```



$A	$JULIA
JULIA	AB

OPERADORES

Por meio dos operadores nós informamos ao PHP o que deve ser executado. Exemplos: atribuir um valor a uma variável, realizar operações aritméticas (soma, subtração etc.), realizar comparação de valores, para testar se um é maior ou menor que o outro, etc. Vamos ver os seguintes tipos de operadores:

- Operadores aritméticos.
- Operadores binários.
- Operadores de comparação.
- Operadores de atribuição.
- Operadores lógicos.
- Operador ternário.

Aritméticos

$a = 3
$b = 2

+ adição
```
echo $a + $b;  //5
```

- subtração
```
echo $a - $b;  //1
```

***** multiplicação
```
echo $a * $b;  //6
```

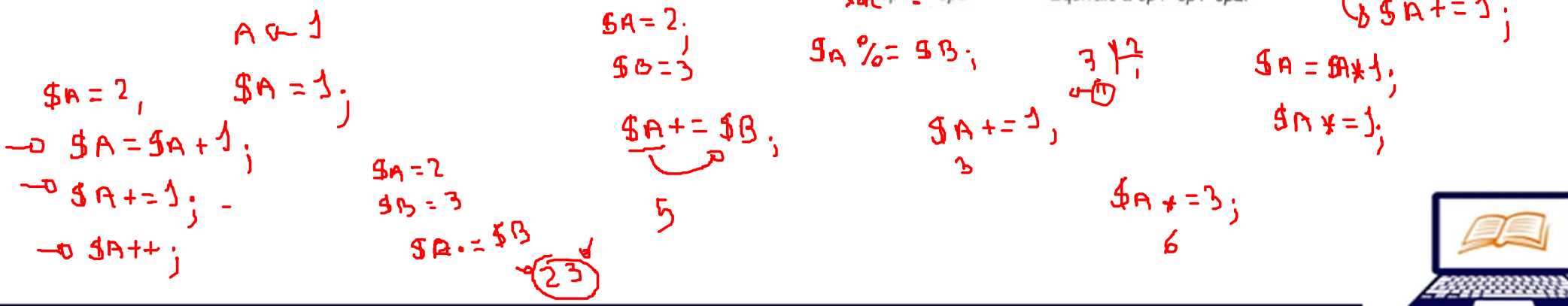
/ divisão
```
echo $a / $b;  //1.5
```

% módulo
```
echo $a % $b;  //1
```

Operadores de atribuição

Atribuição é o termo usado para representar a colocação de um valor em uma variável. A variável que receberá a atribuição encontra-se sempre do lado esquerdo do operador, e esta recebe o valor gerado pela expressão ou operador que está a direita. Além disso temos diversas variações dos comandos de atribuição, que podemos utilizar para facilitar a programação. São operadores que, assim como os operadores de incremento (++) e decremento (--), servem para deixar o código mais simples e mais fácil de ser programado. Veja a seguir a tabela dos comandos de atribuição:

Operador	Descrição
op1 = *op2*	*op1* recebe o valor de *op2*.
op1 += *op2*	Equivale a *op1=op1+op2*.
op1 -= *op2*	Equivale a *op1=op1-op2*.
op1 *= *op2*	Equivale a *op1=op1*op2*.
op1 /= *op2*	Equivale a *op1=op1/op2*.
op1 .= *op2*	Concatenação: equivale a *op1=op1.op2*.
op1 %= *op2*	Equivale a *op1=op1%op2*.
op1 <<= *op2*	Equivale a *op1=op1<<op2*.
op1 >>= *op2*	Equivale a *op1=op1>>op2*.
op1 &= *op2*	Equivale a *op1=op1&op2*.
op1 |= *op2*	Equivale a *op1=op1|op2*.
op1 ^= *op2*	Equivale a *op1=op1^op2*.



$A = 2; INT

$B = "2"; STRING

Comparação (Relacionais)

$A == $B

TRUE

$A === $B

FALSE

$A = = $B

TRUE

$A = = = $B

FALSE

== igual

=== idêntico (igual e do mesmo tipo)

> < >= <= maior, menor, maior e igual, menor e igual

!= <> diferente

!=

Operadores lógicos

São aqueles que retornam o valor verdadeiro ou falso. Veja a tabela a seguir:

Operador	Descrição
!op1	Verdadeiro se *op1* for falso.
op1 AND op2	Verdadeiro se *op1* E *op2* forem verdadeiros.
op1 OR op2	Verdadeiro se *op1* OU *op2* forem verdadeiros.
op1 XOR op2	Verdadeiro se só *op1* ou só *op2* for verdadeiro.
op1 && op2	Verdadeiro se *op1* E *op2* forem verdadeiros.
op1 || op2	Verdadeiro se *op1* OU *op2* forem verdadeiros.

Operador ternário

É uma forma abreviada de usar o comando condicional *if*, que veremos mais adiante. Uma condição é avaliada, e, se ela for verdadeira, atribui-se um valor à variável, e se a condição for falsa atribui-se um outro valor. A sintaxe é a seguinte:

```
cond ? exp1 : exp2
```

Vamos ver um exemplo de uso desse operador, embora seja mais recomendado utilizar o comando condicional *if*, por ser mais simples de usar. Observe:

```
$nota = ($frequencia >= 0.75) ? ($nota+2) : ($nota-2);
```

UNÁRIOS

Operador	Descrição
++oper	Pré-incremento. Primeiro incrementa o valor do operando e depois realiza a operação.
--oper	Pré-decremento. Primeiro decrementa o valor do operando e depois realiza a operação.
oper++	Pós-incremento. Primeiro realiza a operação e depois incrementa o operando.
oper--	Pós-decremento. Primeiro realiza a operação e depois decrementa o operando.

Os operadores mostrados na tabela também são conhecidos como operadores unários, pois necessitam apenas de um operando, ao contrário da adição, subtração e outras operações que necessitam de pelo menos dois operandos.

```php
<?php
    $a = 1;
    $b = 3;
    $c = 5;
    $res1 = ++$b - $a;
    $res2= $c-- + $a;
    $res3= --$a + $c++;
    echo "a = $a<br> b = $b<br> c = $c<br><br>";
    echo "res1 = $res1<br> res2 = $res2<br> res3 = $res3<br>";
?>
```

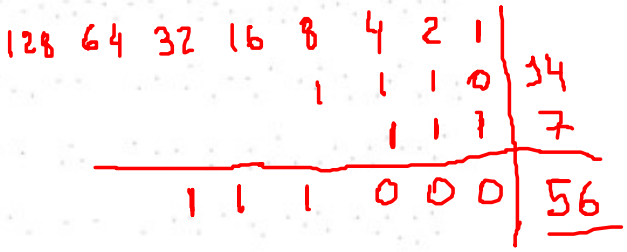
$A = 0
$B = 4
$C = 5
$RES1 = 3
$RES2 = 6
$RES3 = 4

Operadores binários

Esses operadores atuam em um nível de abstração bem mais baixo: trabalham diretamente com bits. Podem ser utilizados para fazer comparações binárias (bit a bit), inverter os bits de um operando, deslocar bits para direita (cada deslocamento para a direita equivale a uma divisão por 2) ou esquerda (cada deslocamento para a esquerda equivale a multiplicar o número por 2). Em alguns casos é interessante usar os operadores binários. Veja a tabela a seguir para conhecer esses operadores:

Operador	Descrição
~op1	Inverte os bits de op1.
op1 & *op2*	Operação E (AND) bit a bit.
op1 | op2	Operação OU (OR) bit a bit.
op1 ^ *op2*	Operação OU exclusivo (XOR).
op1 >> *n*	Desloca op1 n bits à direita.
op1 << *n*	Desloca op1 n bits à esquerda.

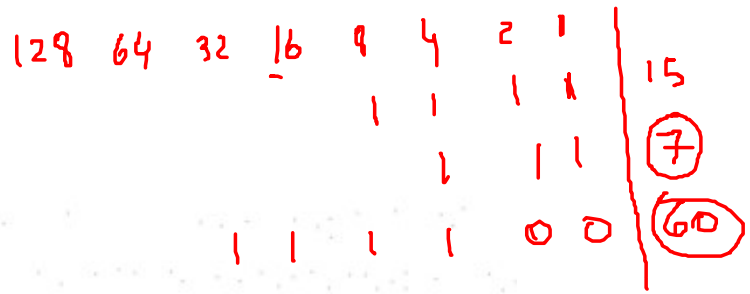
```
<html>
<body>
<?php
    $num = 14;
    $deslocado = $num >> 1;   // desloca 1 bit para direita
    echo $deslocado;
?>
</body>
</html>
```

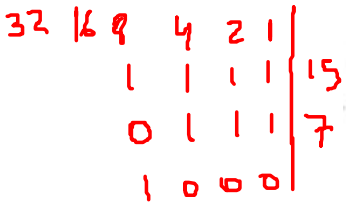


$num << 2

7

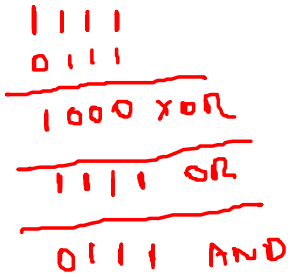
No exemplo apresentado estamos pegando o valor 14 (que equivale a 1110 na base binária) e, deslocando um bit a direita, o que equivale a dividi-lo por 2. O resultado escrito na tela será a divisão de 14 por 2, que dá 7 (equivalente na base binária a 0111).






```php
<html>
<body>
<?php
    $num = 15;
    $resultado1 = $num >> 1;  // desloca 1 bit para direita
    $resultado2 = $num << 2;  // desloca 2 bits para esquerda
    echo $resultado1;
    echo "<br>";
    echo $resultado2;
?>
</body>
</html>
```

7
60




```
$A = 15;
$B = 7;
$C = $A ^ $B;  // XOR
$D = $A | $B;  // OR
$E = $A & $B;  // AND
Echo $C, $D, $E;
8 15 7
```

PRECEDÊNCIA

```php
<?php
    $num = 5;
    $resultado = 8 + 3 * 2 + ++$num;
    echo "$num<br>";
    echo $resultado;
?>
```

20

```php
<?php
    $num = 7;
    $resultado = 8 * $num % 2;
    echo $resultado;
?>
```

0

58 – Em relação à linguagem de programação PHP, assinale a alternativa correta.

a) A linguagem PHP não permite a criação de sites dinâmicos.
b) O código PHP não pode ficar embutido no código HTML.
c) O código PHP é executado no servidor, sendo enviado para o cliente apenas HTML puro.
d) O código-fonte PHP pode ser visualizado pelo cliente, bastando, para isso, acionar a opção "Exibir código-fonte do browser."

60 – Dado o script em php abaixo:

```
<?php
$y = "EEAR";
$$y = "FAB";
print $EEAR;
?>
```

O resultado que será impresso é:

a) FAB
b) EEAR
c) EEARFAB
d) FABEEAR

59 – Na linguagem de programação PHP, a sintaxe correta para a composição do nome de uma variável **CPF** é:

a) $CPF
b) #CPF
c) @CPF
d) *CPF

56 – O PHP tem um recurso conhecido como variáveis variáveis. A utilização deste recurso é feita através do

a) $
b) $$
c) #
d) //

69 – Qual tag abaixo **não** delimita um código PHP?

a)<? php
comandos
?>

b) <?
comandos
?>

c) <%
comandos
%>

d) </
comandos
/>