PHP

(Hypertext Preprocessor)

Introduction

- PHP stands for Hypertext Preprocessor
- Open Source scripting language
- Syntax: C, Java, Perl mix
- HTML can be embedded in code
- Goal
  - To allow web developers to write quickly dynamically generated web pages

Why PHP?

- Advantages
  - Easy
  - No compiler
  - No special software
  - No special editors
- Disadvantages
  - debugging
- Can be tested in university labs
- I know it! ☺

Example

```
<html>
      <head>
         <title>Example</title>
      </head>
      <body>

         <?php
            echo "Hi, I'm a PHP script!";
         ?>

      </body>
</html>
```

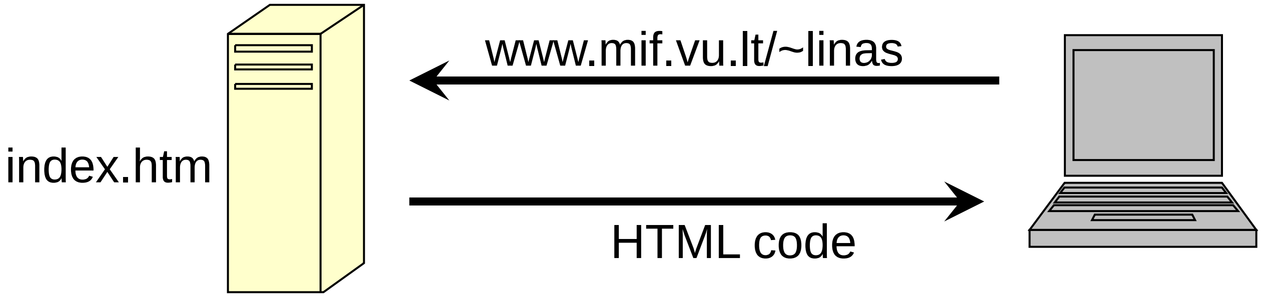
Example (2)

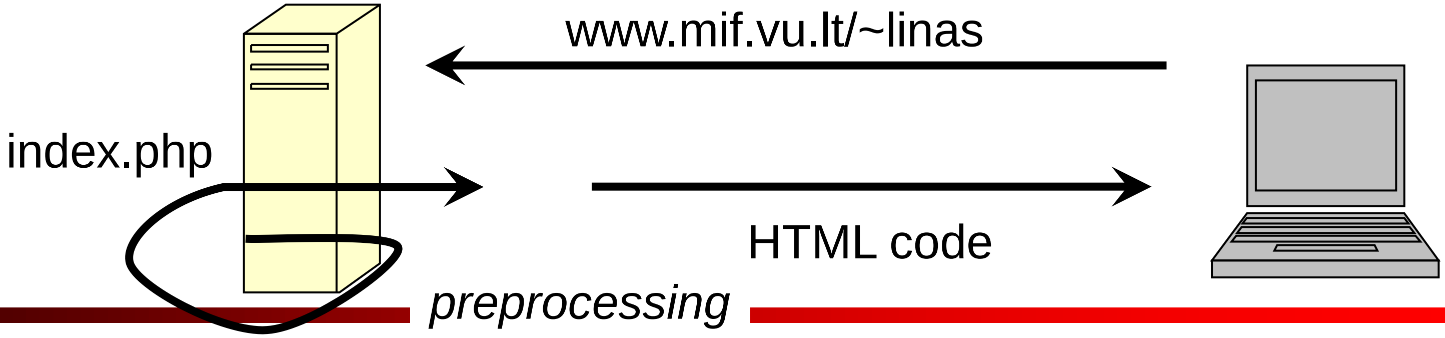
```
<html>
       <head>
        <title>PHP Test</title>
       </head>
       <body>
       <?php echo '<p>Hello World</p>'; ?>
      </body>
</html>
```

```
<html>
  <head>
  <title>PHP Test</title>
  </head>
  <body>
  <p>Hello World</p>
  </body>
</html>
```

What can PHP do?

- Server-side scripting
  - You need a <u>web server</u>, a PHP <u>parser</u> and a <u>web browser</u>





Comparing with C++, JAVA, …

- Similarities
  - Use any text editor
- Differences
  - Easy to learn and program
  - No compiler, so …
  - Code is interpreted, not executed in low level machine language
  - Slower
  - Not suitable for data processing
  - Hard to debug

PHP Purpose

- PHP is for creating HTML pages
  - No calculations
  - No speed
  - No long runs (for web client response)
  - No optimization
- Just easy HTML creation !
- ☺ That’s enough

Something else?

- Command line scripting
- Writing client-side GUI applications, desktop applications – PHP–GTK (extension to PHP)

Flexibility

- PHP can be used on:
  - Linux, many Unix variants, Microsoft Windows, Mac OS X, RISC OS
- PHP has support for most of the web servers:
  - Apache, Microsoft Internet Information Server, Personal Web Server, …
- PHP supports a wide range of databases:
  - MySQL, Oracle, DB2, PostgreSQL, Sybace, … and ODBC
- PHP supports protocols:
  - LDAP, IMAP, SNMP, NNTP, POP3, HTTP, …

Flexibility (2)

- Support for XML:
  - Parsing XML documents
  - Supports SAX and DOM standards
  - XSLT can be used to transform XML documents
  - SimpleXML
- Many other:
  - Compression, coding, hashing utilities
  - Image, .pdf making and transforming
  - Object programming

Syntax

- Tags denoting PHP code blocks:
  - <?php … ?>
  - <script language="php">. . .</script>
  - <? … ?> or <?= …?>
  - <% … %> or <%= … %>

Syntax: Advanced escaping

```
<?php
  if ($expression) {
     ?>
      <strong>This is true.</strong>
     <?php
  } else {
     ?>
      <strong>This is false.</strong>
     <?php
  }
?>
```

Comments

```
<?php
      echo "This is a test"; // This is a one-line c++ style comment
      /* This is a multi line comment
         yet another line of comment */
      echo "This is yet another test";
      echo "One Final Test"; # This is shell-style style comment
?>
```

Types

- Scalar:
  - boolean
  - integer
  - float
  - string
- Compound:
  - array
  - object
- Special:
  - resource
  - NULL

Example

```
<?php
    $bool = TRUE;          // a boolean – TRUE or FALSE
    $str  = "foo";  // a string
    $int  = 12;      // an integer

    echo gettype($bool);    // prints out "boolean"
    echo gettype($str);      // prints out "string"

    // If this is an integer, increment it by four
    if (is_int($int)) {
      $int += 4;
    }

    // If $bool is a string, print it out
    // (does not print out anything)
    if (is_string($bool)) {
      echo "String: $bool";
    }
?>
```

Example

```
<?php
  $bool = TRUE;
// a boolean – TRUE or FALSE
  $str  = "foo"; // a string
  $int  = 12;    // an integer

  echo gettype($bool);   // prints out "boolean"
  echo gettype($str);    // prints out "string"

  // If this is an integer, increment it by four
  if (is_int($int)) {
     $int += 4;
  }

  // If $bool is a string, print it out
  // (does not print out anything)
  if (is_string($bool)) {
     echo "String: $bool";
```

Example - float

```
<?php    // float
    $a = 1.234;
    $b = 1.2e3;
    $c = 7E-10;
?>
```

Array

```
array(
    [key =>] value ,        // key may be an integer or string
     ...                    // value may be any value
)
```

- Array index starts from 0;

```
<?php
    $myarray = array();
    $myarray[] = 14          //   $myarray[0] = 14;
?>
```

Array example

```
$a = array(
   0 => 101,
   1=> “labas”
);

$a = array(
   0 => 101,
   6=> “labas”
);
```

Array example (2)

```
$a = array(
    6 => 101,
    16 => “labas”,
    “geras” => “lala”
);
```

Pavyzdys

- $_SESSION[0] = “true”;
- $_SESSION[1] = “linas”;

- $_SESSION[“logedin”] = “true”;
- $_SESSION[“username”] = “linas”;

Array example (3)

```
$a = array(
    6 => 101,
    16 => “labas”,
    “geras” => “lala”,
    4657, 13, “labas dienas”
);
```

Array

```
<?php

$myarray = array(“name" => “bob", 12 => true);

echo $myarray[“name"];      // bob
echo $myarray[12];          // 1 ???
$myarray[] = 124;           // $myarray[13] = 124;
?>
```

Arrays

```
<?php
    // This array is the same as ...
    array(5 => 43, 32, 56, "b" => 12);
    // ...this array
    array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

```
<?php
    $arr = array(
            "somearray" => array(6 => 5, 13 => 9, "a" => 42, 77)
    );
    echo $arr["somearray"][6];          // 5
    echo $arr["somearray"][13];         // 9
    echo $arr["somearray"]["a"];        // 42
    echo $arr["somearray"][14];         // 77
?>
```

Array – examples

```
$arr[“user”][“login”] = true;
$arr[“user”][“name”] = “Jonas”;
$arr[“user”][“type”] = “admin”;

$arr[„user“][„favorites“][„links“][] = 23;

 echo $arr[„user“][„favorites“][„links“][0];
```

Arrays - klausimai

- Kas bus jeigu:
- var_dump($arr[„user“][„favorites“][„links“][0]);
- var_dump($arr[„user“][„favorites“][„links“]);
- var_dump($arr[„user“][„favorites“]);
- var_dump($arr);

Array examples 2

- DB yra lentelė vartotojo nustatymų saugojimui

Id	Setting	Value
21	color	red

- Įrašų reikšmės yra nežinomos
- Įrašus norime saugoti masyve

- $arr[„settings“][„{$setting}“] = $value

Array examples 3

- $arr[„settings_“.$setting.“_a“] = $value

Example: String (single quoted)

```
<?php
   echo 'this is a simple string';

   echo 'You can also have embedded newlines in
   strings this way as it is
   okay to do';

   // Outputs: Arnold once said: "I'll be back"
   echo 'Arnold once said: “I\'ll be back“ ';
?>
```

Example: String (single quoted)

```php
<?php
  // Outputs: You deleted C:\*.*?
  echo 'You deleted C:\*.*?';

  // Outputs: This will not expand: \n a newline
  echo 'This will not expand: \n a newline';

  // Outputs: Variables do not $expand $either
  echo 'Variables do not $expand $either';
?>
```

Example: String (double quoted)

```
<?php
    echo “this is a simple string”;

    echo “You can also have embedded newlines in
    strings this way as it is
    okay to do”;

    // Outputs: Arnold once said: "I'll be back"
    echo “Arnold once said: \“I'll be back\” “;
    ?>
```

Example: String (double quoted)

```
<?php
   // Outputs: You deleted C:\*.*?
   echo “You deleted C:\\*.*?”;

   // Outputs: This will not expand:
 a newline
   echo “This will expand: \n a newline”;

   // Outputs: Variables do
   echo “Variables do $expand $either”;
?>
```

Example – String (double quoted)

```
$fruits = array('strawberry' => 'red', 'banana' =>
   'yellow');

 // Works
echo "A banana is {$fruits['banana']}.";


   // Works but note that this works differently
outside string-quotes
echo "A banana is $fruits[banana].";
```

Example – String (double quoted)

```
// Works but PHP looks for a constant named
   banana first
   // as described below.
   echo "A banana is        {$fruits[banana]}.";

   // Won't work, use braces.  This results in a
   parse error.
   echo "A banana is         $fruits['banana'].";

   // Works
   echo "A banana is " . $fruits['banana'] . ".";
```

Pavyzdys

```
for($i=0; $i<10; $i++)
    echo "A banana is " . $fruits[‘fruit_‘.$i] . ".";

//
$fruits[‘fruit_0’]
$fruits[‘fruit_1’]
…
```

String – examples

```
echo “Aš sakau:\”PHP yra gerai\” ”;

echo ‘Aš sakau: “PHP yra gerai” ’;

echo “ Slashas: \\ ”;
```

Example – String (heredoc)

```
<?php
   $str = <<<EOD
   Example of “my” string
   spanning multiple lines
   using ‘heredoc’ syntax.
   EOD;

    echo $str;
?>
```

Interesting

```
<?php
    $foo = 1 + "10.5";               // $foo is float (11.5)
    $foo = 1 + "-1.3e3";            // $foo is float (-1299)
    $foo = 1 + "bob-1.3e3";           // $foo is integer (1)
    $foo = 1 + "bob3";                // $foo is integer (1)
?>
```

Interesting

```
<?php
   $foo = 1 + "10 Small Pigs";      // $foo is integer
   (11)
   $foo = 4 + "10.2 Little Piggies"; // $foo is float
   (14.2)
   $foo = "10.0 pigs " + 1;          // $foo is float (11)
   $foo = "10.0 pigs " + 1.0;        // $foo is float (11)
?>
```

Pavyzdys

```
$a = “35”;

$b = (int)$a;

If ($a < 40) {                    // ??? Turbūt veiks gerai

}
```

Control structures

- if () , elseif (), else, *endif*
- while(), do … while ()
- for (), foreach()
- switch () case …:

Klausimas

foreach($arr as [$key =>] $element)

Klausimas

```
foreach($arr as $key => $element) {
    $arr[$key] => 3;
}
```

If structure

```
if  ( <expresion> )
    <statement>
elseif ( <expresion> )
    <statement>
else
    <statement>
```

```
if ( <expresion> ):
    <statement>
elseif ( <expresion> ):
    <statement>
else:
    <statement>
endif;
```

```
echo $a > 5 ? "big" : "small";
```

If structure (1)

```
if  ( <expresion> )
    <statement>
elseif ( <expresion> )
    <statement>
else
    <statement>
```

If structure (1) – pvz.

```
if  ( $a > 100 )
    echo $a;
elseif ( $a < 0)
    echo “<0”;
elseif ($a == 0)
    echo “0”
else
    echo “?”;
```

If structure (2)

```
if ( <expresion> ):
    <statement>
elseif ( <expresion> ):
    <statement>
else:
    <statement>
endif;
```

If structure (2)

```
if ($a > 5) {
   echo "big";
} else {
   echo "small";
}
```

=

```
echo $a > 5 ? "big" : "small";
```

Casting type

```
<?php
  $foo = 10;                    // $foo is an integer
  $str = "$foo";                // $str is a string
  $fst = (string) $foo; // $fst is also a string

  // This prints out: "they are the same"
  if ($fst = = = $str) {
     echo "they are the same";
  }
?>
```

Casting type

```
<?php

    // Šitas bus atspausdintas
    if (10 = = "10") {
       echo "they are the same";
    }

    // O šitas ne
    if (10 = = = "10") {
       echo "they are the same";
    }
?>
```

Dumping variables

```
<?php
  $a = 1;
  var_dump($a);
  // output: int(1)


  print_r($a);                 // use <pre> also
?>
```

Variables – assigning by reference

```
<?php
    $foo = 'Bob';                    // Assign the value 'Bob' to $foo
    $bar = &$foo;                    // Reference $foo via $bar.
    $bar = "My name is $bar";   // Alter $bar...
    echo $bar;
    echo $foo;                       // $foo is altered too.
?>
```

The result:

```
My name is Bob
My name is Bob
```

Variable scope

```
<?php
   $a = ‘My name is Bob’;       /* global scope */

   function Test() {
     echo $a;                   /* local scope variable */
   }


   Test();
?>
```

Result: nothing

Variable scope (2)

```
<?php
    $a = ‘My name is Bob’;              /* global scope */

    function Test1() {
      global $a;
       echo $a;                         /* global scope variable */
    }

     function Test2() {
      echo $GLOBALS[‘a’];               /* global scope variable */
    }

    Test1();                  // Result: My name is Bob
    Test2();                  // Result: My name is Bob
?>
```

```
$a = 3;

funcion sum($b) {
   $b += 2;
}

sum(&$a);                    // $a == 5
```

Variables - static

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

Result: every time the Test() function will print the value of $a and increment it

- Svarbu: statinis kintamasis išsaugo reikšmę vieno PHP script’o paleidimo metu

Predefined variables

- $GLOBALS
- $_SERVER
- $_POST, $_GET
- $_SESSION
- …

Predefined variables - $GLOBALS

```
<?php
    $a = 1;
    $b = 2;

    function Sum()
    {
      // global $a, $b;
       $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
    }

    Sum();
    echo $b;
?>
```

Predefined variables - $_SERVER

- $_SERVER['HTTP_USER_AGENT‘]
- $_SERVER['REMOTE_ADDR']
- …

$_POST, $_GET

- Some times works:

```
http://www.mif.vu.lt/~linas/example.php?id=10

echo $id;                    // not recommended at all !
echo $_GET[id];              // if no reserved word “id” is founded, PHP 4.x
                             // PHP 5.x produces warning
```

- Always works:

```
http://www.mif.vu.lt/~linas/example.php?id=10

echo $_GET[‘id’];
```

$_POST

```
<form action=“myfile2.php" method="post">

    <input type=“text” name=”vardas” />
    …
</form>
-------------------------------------


echo $_POST[‘vardas’]
```

$_POST (2)

```
<form action=“myfile2.php?id=3" method="post">

   <input type=“text” name=”vardas” />
   ...
</form>
-----------------------------------
echo $_GET[‘id’]
echo $_POST[‘vardas’]
```

$_POST (3)

```
<form action=“myfile2.php" method="post">

    <input type=“text” name=”vardas” />
    <input type=“hidden” name=“id” value=“3” />
</form>
----------------------------------------
echo $_POST[‘id’]
echo $_POST[‘vardas’]
```

Variables from outside

```
<?php    // file myfile1.php
<form action="myfile2.php" method="post">
    Name: <input type="text" name="username" />
    <br />
    <input type="submit" name="submit" value="Submit me!" />
</form>
?>

<?php    // file myfile2.php
    echo $_POST['username'];
?>
```

Result: the text in the “Name” field will be output in the sceen.

$_SESSION

- Why?
  - Keeps track of variables while user browses
- How?
  - Assigns unique ID to browser window
- Properties
  - Is alive for some time period (usually 30 min.)
- Tricks
  - can be passed (to other window, …)

$_SESSION

- Should be started first!

```
<?                                    // no output before!
session_start();                      // and 90% of mistakes solved already ☺
```

- Is an ordinary array:

```
$_SESSION['loged_in'] = true;
```

- Values can be
  - Registered
  - Unregistered

$_SESSION

- Klausimai iš studentų:
  - Kur saugoti Session ID? Ar duomenų bazėje?
    - Session ID saugoti nereikia, reikiamą sesiją PHP interpretatorius gauna iš karto, pagal tai koks browseris į ji kreipiasi – tai Web serverio lygyje atliekama ☺

Tricks with variables

- Passing/not passing a variable

```
http://www.mif.vu.lt/~linas/example.php?id=10
```

```
if (!empty($_GET[‘id’]) && $_GET[‘id’] == 10)
  echo $_GET[‘id’];              // PHP 5.x produces warning, if there is no
                                          // (!empty($_GET[‘id’])
```

Variable variables

```
<?php
   $a = "hello";
   $$a = "world“;
    echo "$a  ${$a}";          // echo "$a $hello";
?>
```

Result: hello world

Example

HTML form:

```
<form action=“myfile2.php" method="post">

   <input type=“text” name=”vardas1” />
   <input type=“text” name=”vardas2” />
   …
</form>
----------------------------------------
echo $_POST[‘vardas1’]
echo $_POST[‘vardas2’]
```

Example 2

```
-----------------------------------------
for ($i=0; $i < 100; $i++)
   echo $_POST[“vardas”.$i];


-----------------------------------------
for ($i=0; $i < 100; $i++)
   if (!empty($_POST[“vardas”.$i])) {

}
```

Example 3

```
<form action=“myfile2.php" method="post">

    <input type=“text” name=”funkcija” />
    …
</form>
--------------------------------------
$_POST[‘funkcija’]();        // php_info();
```

Example 4

```
$lentele = array(
    “users” => array(
        “pavarde” => rasykJuodai,
        “gimData”   => spausdinkData
    )
);
```

id	**pavarde**	**gimData**
1	Jonaitis	34898739182
3	Petraitė	32547436543

Example 5

HTML lentelė

Nr.	Pavardė	Gimimo data
1	Jonaitis	1977.03.13
2	Petraitė	1985.06.11

Example 6

```
function rasykJuodai($text) {
    return “<span style=\”text-weight:bold;\”> {$text}
    </span>”;
}

function pieskLentele($lentelesVardas) {
    …
    foreach ($lentele[$lentelesVardas] as $key=> $el)
        echo $el($row[$key]);
    …
}
```

Example 7

```
$lentele = array(
    “users” => array(
        “pavarde” => “rasykJuodai”,
        “gimData”      => “spausdinkData”
    )
);

function pieskLentele($lentelesVardas) {
    …
    foreach ($lentele[$lentelesVardas] as $key=> $el)
        echo $el($row[$key]);
    …
}
```

Example – Parsing Var… Var…

```
<?php

    $var_name = "a";
    $a = "text";

    echo “$var_name = {${$var_name}}”;
    echo ‘$var_name = ‘ .”{$$var_name}”; !!!!!!
    //  a = text

    echo “$var_name: = $$var_name<br>”;
    //  a = $a
?>
```

Constants

-
  ```
  <?php
  define("CONSTANT", "Hello world.");
  echo CONSTANT;          // outputs "Hello world."
  ?>
  ```

- Constants do not have a dollar sign ($) before them;
- Constants may only be defined using the **define()** function, not by simple assignment;
- Constants may be defined and accessed anywhere without regard to variable scoping rules;
- Constants may not be redefined or undefined once they have been set; and
- Constants may only evaluate to scalar values (**boolean**, **integer**, **float** and **string**) .

Functions

```
<?php
   function myfunction ($number)
   {
     return $number++;
   }


   echo “number” . myfunction(5);
?>
```

Result: number 6

Functions - declaration

```
<?php
    function foo ([$arg_1, [$arg_2, […]]])
    {
        echo "Example function.\n";
        return <$returnval>;
    }
?>
```

Functions

```
<?php
    function myfunction ($number=3)
    {
       return $number++;
    }


    echo “number” . myfunction(5);  // Result:  number 6

    echo “number” . myfunction();  // Result:  number 4


?>
```

Blogai - gerai

```
Function manoFunkcija($a=3, $b, $c) {
…
}
manoFunkcija(3,44);

Function manoFunkcija($a, $b, $c=3) {
…
}
manoFunkcija(3,44);
manoFunkcija(3,44,"a");
```

Connection to Database

```
$connection = mysql_connect(<host>, <username>,<password>);

mysql_select_db(<database name>);
```

Sending query to Database

$result = mysql_query("SELECT id FROM
  Persons WHERE Name=‘James’“,$connection)

$line = mysql_fetch_assoc($result)

echo $line[‘Name’];   // James

Example

```
$result = mysql_query("SELECT id FROM
   Persons WHERE Name=‘James’“,$connection)


while($line = mysql_fetch_assoc($result)) {

        echo $line[‘id’];

}
```

Example 2

id	Name	Surname	email
1	Linas	Xfile	
2	Jonh	Smith	

```
while( $line = mysql_fetch_assoc($result) ) {

        echo $line[‘id’];

}
```

The power of PHP

- Example of variable in variable construction
- Calling functions with variable names

variable in variable construction

```
$_POST[“email_{$nr}”]

$masyvas[“email_”.$_POST[“input1_{$a}”]]
```

Includes

- include()
  - include('myfile.php');
- Differences with C++, and other
  - Includes file in the place of the line
  - Variables take the scope of the place where

Includes

- include()
  - include(‘myfile.php’);
- include_once()
  - produces warning

- require()
- require_once()
  - produces a fatal error

Cookie

- When cookie is loaded?
  - Cookies are part of the HTTP header, so setcookie() must be called before any output is sent to the browser.[1]
  - When cookie values are recieved from computer?

Function empty()

- The following things are considered to be empty:
  - *""* (an empty string)
  - *0* (0 as an integer)
  - *0.0* (0 as a float)
  - *"0"* (0 as a string)
  - **NULL**
  - **FALSE**
  - *array()* (an empty array)
  - *var $var;* (a variable declared, but without a value)

http://php.net/manual/en/function.empty.php

Not talked about…

- Avoiding “Refresh”
- Callback functions
- Socket
- Autoload for classes

- …? what else?

Literature

[1] http://www.php.net/manual/en/