Pemrograman Berbasis Web

Pertemuan 8

Nori Wilantika
Lya Hulliyyatus Suadaa
Yeni Rimawati

Politeknik Statistika STIS
Prodi DIV Komputasi Statistik

Materi s.d UAS

08
Server Side Scripting (1)
Dasar server side scripting dengan PHP pada aplikasi web

09
Server Side Scripting (2)
Form handling dan database connection dengan PHP pada aplikasi web

10
Session&Cookies
Penggunaan session dan cookies pada halaman web

11
AJAX, JSON, API
AJAX, JSON, PHP pada halaman web

12
Web Framework (1)
Dasar penggunaan web framework

13
Web Framework (2)
CRUD dengan Web Framework

14
Project Akhir Web
Presentasi dan Demo

08

SERVER-SIDE SCRIPTING
<?PHP ?>

PHP

PHP -> akronim dari "PHP: Hypertext Preprocessor"
PHP adalah server scripting language dan powerful tool untuk membuat halaman web dinamis dan interaktif.

Kode PHP dieksekusi di server, dan hasil HTML dikirim kembali ke browser. Langkah pemrosesannya yaitu:

- Web server menerima client request
- Web server mengenali bahwa client request adalah suatu dokumen HTML yang terdiri dari kode PHP
- Server mengeksekusi kode PHP, substitusi output ke dalam dokumen HTML, lalu halaman yang dihasilkan dikirim ke client

Aplikasi dan website yg ditulis dg PHP: Drupal, Moodle, Wordpress, Facebook, Wikipedia

PHP SYNTAX

- Script PHP dapat ditempatkan di mana saja dalam dokumen.
- Script PHP dimulai dengan **<?php** dan diakhiri dengan **?>**.

```
<?php
// PHP code goes here
?>
```

- Komentar pada script PHP:

```
// This is a single-line comment

# This is also a single-line comment

/*
This is a multiple-lines comment block
that spans over multiple lines
*/
```

- Statement PHP diakhiri dengan titik koma.
- PHP **TIDAK case sensitive**. Namun untuk **nama variabel**, case sensitive.
- Ekstensi file default untuk file PHP adalah " **.php**".

OUTPUT

Dua cara dasar untuk menampilkan output: **echo** dan **print**.

```
echo(arg1)
echo arg1, arg2, ...
```

- Menampilkan semua argumen sebagai output
- Tidak boleh menggunakan tanda kurung jika ada lebih dari satu argumen
- Lebih efisien dari print (sehingga lebih sering digunakan)

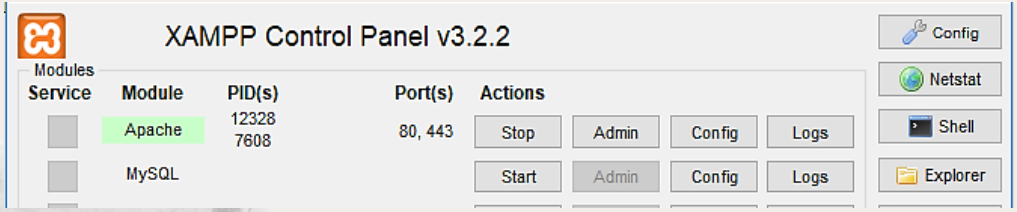
```
print(arg)
```

- Menampilkan argumen sebagai output
- Hanya satu argument yang diperbolehkan
- Mengembalikan nilai 1 (return value)
- Tanda kurung dapat diabaikan

Untuk mulai menggunakan PHP, kita memerlukan web host yang mendukung PHP. Atau kita dapat menginstall perangkat lunak yang menyediakan fungsi HTTP Server seperti XAMPP atau WAMP.

```
<html lang="en-GB">
<head><title>Hello World</title></head>
<body>
<h1>Our first PHP script</h1>
<?php
  print ("<p><b>Hello World!</b></p>\n");
?>
</body></html>
```

Ekstensi file default untuk file PHP adalah " .php“, contoh: hello_world.php
Simpan file pada direktori yang dapat diakses oleh web server, contoh “C:\xampp\htdocs”
Untuk melihat hasilnya pada web browser, jalankan Apache HTTP Server pada xampp dan akses URL http://localhost/hello_world.php.



```php
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "This ", "string ", "was ", "made ", "with multiple parameters.";
?>
```

```php
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

TYPES AND VARIABLES

TYPES

- PHP memiliki delapan datatypes
- Empat primitive types:
  - boolean
  - integer
  - float
  - string
- Dua compound types: array, object
- Dua special types: resource, NULL

INTEGERS AND FLOAT

- PHP membedakan antara integer numbers dan floating-point numbers
- PHP mendukung banyak pre-defined mathematical functions

`abs(number)`	absolute value
`ceil(number)`	round fractions up
`floor(number)`	round fractions down
`round(number [,prec,mode])`	round fractions
`log(number [,base])`	logarithm
`rand(min,max)`	generate an integer random number
`sqrt(number)`	square root

- PHP menyediakan pre-defined number constants:

`M_PI`	3.14159265358979323846
`NAN`	'not a number'
`INF`	'infinity'

NAN AND INF

- NAN dan INF digunakan sebagai return value untuk beberapa aplikasi mathematical functions yang tidak mengembalikan number

- log(0) returns -INF (negative 'infinity')
- sqrt(-1) returns NAN ('not a number')

- 1/0 returns INF dan menghasilkan PHP warning
- 0/0 returns NAN dan menghasilkan PHP warning

STRING

- PHP mendukung heredocs untuk menspesifikasikan multi-line strings. Operator yang digunakan untuk string concatenation yaitu '.'

```
'hello'          ↪ hello
'don\'t'         ↪ don't
'"hello"'        ↪ "hello"
'backslash\\'    ↪ backslash\
'glass\\table'   ↪ glass\table
'glass\table'    ↪ glass\table
```

```
"hello"          ↪ hello
"don't"          ↪ don't
"\"hello\""      ↪ "hello"
"backslash\\"    ↪ backslash\
"glass\\table"   ↪ glass\table
"glass\table"    ↪ glass    able
```

```
print<<<EOF
<body>Some text
<img alt="Picture␣of␣Crowne␣Plaza" src="pic.png">
</body>
</html>
EOF;
```

```
<?php
$str = <<<LABEL_SAYA
Contoh penulisan heredoc pada PHP.
Menulis string tanpa escape (back slash) pada PHP.
Contoh kata Jum'at dengan satu tanda kutip atau
Jum"at dengan dua tanda kutip
LABEL_SAYA;

echo $str;
```

VARIABLES

- Semua **variable names** pada PHP dimulai dengan $ diikuti dengan **PHP identifier**
- **PHP identifier** terdiri dari letters, digits, dan underscores, tetapi tidak dapat dimulai dengan digit
- **PHP identifiers** adalah case sensitive

- Pada PHP, variabel **tidak perlu dideklarasikan** sebelum digunakan
- Variabel **tidak perlu diinisialisasi** sebelum digunakan, walaupun inisialisasi direkomendasikan

Type	Default	Type	Default
bool	FALSE	string	empty string
int/float	0	array	empty array

- Jika tidak ada nilainya, maka default value yaitu **NULL**

ASSIGNMENTS

- Seperti Java, JavaScript and Python, PHP menggunakan equality sign = untuk assignments

```
$student_id = 200846369;
```

- PHP juga mendukung standard binary assignment operators:

Binary assignment	Equivalent assignment
$a += $b	$a = $a + $b
$a -= $b	$a = $a - $b
$a *= $b	$a = $a * $b
$a /= $b	$a = $a / $b
$a %= $b	$a = $a % $b
$a **= $b	$a = $a ** $b
$a .= $b	$a = $a . $b

ASSIGNMENTS

- Konstanta mirip seperti variabel, kecuali setelah ditentukan, konstanta tidak dapat diubah atau tidak ditentukan.
- Nama konstanta yang valid dimulai dengan letter atau underscore (tidak ada tanda $ sebelum nama konstanta).
- Tidak seperti variabel, konstanta secara otomatis bersifat global untuk keseluruhan script.

```
define(name, value, case-insensitive)
```

```
define("PI",3.14159);
define("SPEED_OF_LIGHT",299792458,true);
// PHP 7
define("ANIMALS",["bird","cat","dog"]);

$circumfence = PI * $diameter;
$distance    = speed_of_light * $time;
$myPet       = ANIMALS[1];
```

TYPE CASTING & COMPARISONS

TYPES CHECKING

gettype(expr)	returns the type of expr as string
is_type(expr)	checks whether expr is of type type
var_dump(expr)	displays structured information about expr that includes its type and value

```
<?php print "Type␣of␣23:␣␣␣".gettype(23)."\n";
      print "Type␣of␣23.0:␣".gettype(23.0)."\n";
      print "Type␣of␣\"23\":␣".gettype("23")."\n";

      if (is_int(23)) { echo "23␣is␣an␣integer\n"; }
         else { echo "23␣is␣not␣an␣integer\n"; }
?>
```

```
Type of 23:   integer
Type of 23.0: double
Type of "23": string
23 is an integer
```

TYPES JUGGLING

- PHP secara otomatis mengkonversi nilai ke tipe yang tepat sesuai dengan **pemilik operator** yang digunakan. Contoh:

```
2 . "␣worlds"          ⇝  "2␣worlds"
"2" * 3                ⇝  6
"1.23e2" + 0           ⇝  123
```

Pada **operator aritmatika**:

- jika kedua operan adalah floats, maka hasil operasi adalah berupa floats. Jika tidak, maka operan akan diinterpretasikan sebagai integer, dan hasil operasi adalah berupa integer.

- Operan Numeric string dan Leading-numeric string akan dikonversi menjadi bilangan terdekat.

- Operan Non Numeric String menghasilkan **TypeError.**

JENIS-JENIS STRING

- Berlaku sejak PHP 8, string dibedakan menjadi Numeric String, Leading-Numeric string, dan Non Numeric String

- **Numeric string** adalah string yang hanya berisi angka, **dapat diawali dengan spasi**, contoh:
  `"10.5"` `"42"` `" 42"` `"-1.3e3"` `"0E1"` `"2E1"`

- **Leading-numeric string** adalah string yang dimulai dengan numeric string tetapi diikuti oleh karakter non-angka **termasuk spasi**, contoh:
  `"10hello5"` `"123abc"` `"42 foo"` `"105.a“`
  `"10.0 pigs "` `"10 Small Cats"` `"10.2 Little Cats"`

- **Non Numeric String** adalah string yang bukan merupakan numeric string maupun leading-numeric string, contoh:
  `"bob-1.3e3"` `"bob3“` `"0D1"`

CONTOH TYPE JUGGLING

Contoh type juggling pada operator aritmatika dengan operan string:

```
$foo = 1 + "10.5";                // (float) 11.5
$foo = 1 + "-1.3e3";              // (float) -1299
$foo = 1 + "bob-1.3e3";           // TypeError, PHP 7: 1
$foo = 1 + "bob3";                // TypeError, PHP 7: 1
$foo = 1 + "10 Small Pigs";       // (int) 11 and E_WARNING
$foo = 4 + "10.2 Little Piggies"; // (float) 14.2 and E_WARNING
$foo = "10.0 pigs " + 1;          // (float) 11 and E_WARNING
$foo = "10.0 pigs " + 1.0;        // (float) 11 and E_WARNING
$foo = "hello" * 3;               // TypeError, PHP 7: 0
$foo = "10hello5" + 5;            // (int) 15 and E_WARNING
```

TYPE CASTING

PHP mendukung explicit type casting menggunakan **(type)**

(int) "12"	:	12
(int) "1.23e2"		123
(int) ("1.23e2" * 1)		123
(int) (float) "1.23e2"		123
(int) "1.23e2h5"		123
(int) "10.5"		10
(int) "10hello5"		10
(int) "foo"		**0** (bukan TypeError, berbeda dengan type juggling)
(float) "1.23e2"		123
(float) "1.23e2h5"		123
(float) "10hello5"		10
(float) "foo"		**0** (bukan TypeError, berbeda dengan type juggling)
(bool) "0"		**FALSE** (Pada javascript, hasilnya TRUE)
(bool) "foo"		TRUE
(array) "0"		{ [0]=> string(1) "0" }

TYPE CASTING

- Dapat menerapkan identity function dari tipe target untuk memaksa type conversion.

```
"12" * 1   ⇝ 12        !!1   ⇝ TRUE     !!"1"       ⇝ TRUE
"12" * 1.0 ⇝ 12.0      !!0   ⇝ FALSE    !!"0"       ⇝ FALSE
"12.1"* 1  ⇝ 12.1      !!1.0 ⇝ TRUE     !!""        ⇝ FALSE
12 . ""    ⇝ "12"      !!0.0 ⇝ FALSE    FALSE . ""  ⇝ ""
12.1 . ""  ⇝ "12.1"                     FALSE * 1   ⇝ 0
```

- Tidak bisa konversi arrays menjadi string atau number

TYPE CONVERSION TO BOOLEAN

- Ketika mengkonversi ke boolean, nilai nilai berikut menjadi **FALSE**:
  - the boolean FALSE
  - the integer 0
  - the float 0.0
  - **the string “0”**
  - the empty string “” (tidak ada spasi di antara petik)
  - **an array with zero elements**
  - the special type NULL

- Nilai lainnya bernilai **TRUE** (including any resource)

- Ketika mengkonversi boolean menjadi string,
  - **TRUE menjadi "1"**
  - **FALSE menjadi ""**

TYPES JUGGLING: OPERATOR COMPARISONS

- Pada operator non-strict comparisons (==) juga terjadi type juggling

Tipe Operand 1	Tipe Operand 2	Hasil
bool atau null	Apapun	Convert both sides to bool, false < true
numeric string, Int, atau float	numeric string, Int, atau float	Convert to numbers
Leading-numeric string atau non-numeric string	**int atau float**	**Convert to string**

- Non-strict comparisons between numbers and non-numeric strings work by casting the **number to string** and **comparing the strings**.

CONTOH TYPE JUGGLING: COMPARISONS

```
"123" == 123          ⇝  TRUE
"123" != 123          ⇝  FALSE
"1.23e2" == 123       ⇝  TRUE
"1.23e2" == "12.3e1"  ⇝  TRUE
"10hello5" == 10      ⇝  FALSE
5 == TRUE             ⇝  TRUE
```

```
"123" === 123          ⇝  FALSE
"123" !== 123          ⇝  TRUE
1.23e2 === 123         ⇝  FALSE
"1.23e2" === "12.3e1"  ⇝  FALSE
"10hello5" === 10"     ⇝  FALSE
5 === TRUE             ⇝  FALSE
```

```
'35.5' > 35           ⇝  TRUE
'ABD' > 'ABC'         ⇝  TRUE
'1.23e2' > '12.3e1'   ⇝  FALSE
"F1" < "G0"           ⇝  TRUE
TRUE > FALSE          ⇝  TRUE
5 > TRUE              ⇝  FALSE
```

```
'35.5' >= 35          ⇝  TRUE
'ABD' >= 'ABC'        ⇝  TRUE
'1.23e2' >= '12.3e1'  ⇝  TRUE
"F1" <= "G0"          ⇝  TRUE
TRUE >= FALSE         ⇝  TRUE
5 >= TRUE             ⇝  TRUE
```

CONTOH TYPE JUGGLING: COMPARISONS

Contoh type juggling pada operator comparisons :

`0 == "0"`	`true`
`0 == "0.0"`	`true`
`0 == "foo"`	`false`
`0 == ""`	`false`
`42 == " 42"`	`true`
`42 == "42foo"`	`false`

Best Practice

- Gunakan operator strict comparions (===) dalam operasi perbandingan.
- Untuk membandingkan dua buah string, gunakan **`strcmp()`**

SPACESHIP OPERATOR

- Digunakan untuk three-way comparisons (yang mengkonversi numeric strings ke numbers). Mengembalikan nilai 0, -1, atau 1

strcmp(*expr1*, *expr2*)	String comparison	Returns < 0 if *expr1* is less than *expr2*, > 0 if *expr1* is greater than *expr2*, 0 if *expr1* is equal to *expr2*
expr1 <=> *expr2* (PHP 7 only)	Three-way comparison	Returns -1 if *expr1* < *expr2*, +1 if *expr1* > *expr2*, 0 if *expr1* == *expr2*

```
strcmp('ABD','ABC')              ⇝  1
strcmp('aaa',"aaa")              ⇝  0
strcmp('1.23e2','12.3e1')        ⇝  -1
'ABD' <=> 'ABC'                  ⇝  1
'aaa' <=> "aaa"                  ⇝  0
'1.23e2' <=> '12.3e1'            ⇝  0
'35.5' <=> 35                    ⇝  1
TRUE <=> FALSE                   ⇝  1
5 <=> TRUE                       ⇝  0
```

```
strcmp("F1","G0")                ⇝  -1
strcmp('aaa',"AAA")              ⇝  1

"F1" <=> "G0"                    ⇝  -1
'aaa' <=> "AAA"                  ⇝  1

'10hello5' <=> 10                ⇝  0
0.0 <=> FALSE                    ⇝  0
'FALSE' <=> TRUE                 ⇝  0
```

COMPARISONS: NAN & INF

- **NAN** dan **INF** dapat dibadingkan dengan menggunakan operator equality dan comparison.

```
NAN == NAN ⇝ FALSE
INF == INF ⇝ TRUE
NAN < NAN  ⇝ FALSE
NAN < INF  ⇝ FALSE
NAN < 1    ⇝ FALSE
```

```
NAN === NAN ⇝ FALSE
INF === INF ⇝ TRUE
INF < INF   ⇝ FALSE
INF < NAN   ⇝ FALSE
1 < NAN     ⇝ FALSE
```

```
NAN == 1 ⇝ FALSE
INF == 1 ⇝ FALSE
1 < INF  ⇝ TRUE
INF < 1  ⇝ FALSE
```

- PHP menyediakan tiga fungsi untuk menguji suatu nilai adalah **NAN**, **INF** or **–INF**
  - `is_nan(value)` returns TRUE if value is NAN
  - `is_infinite(value)` returns TRUE if value is INF or -INF
  - `is_finite(value)` returns TRUE if value is neither NAN nor INF/-INF

- Untuk konversi ke nilai boolean, NAN dan INF dikonversi menjadi TRUE
- Untuk konversi ke string, NAN dikonversi ke 'NAN' dan INF dikonversi ke 'INF'

ARRAY

ASSOCIATIVE ARRAY

- Pada PHP, fungsi array ( ) digunakan untuk membuat array

```
array(key => value, ... )
[key => value, ...]
```

- Pada PHP, ada tida tipe dari array:
  - Indexed arrays - Arrays dengan numeric index
  - Associative arrays - Arrays dengan named keys
  - Multidimensional arrays - Arrays terdiri dari satu atau lebih arrays

```
$arr3 = array("Peter", "Paul", "Mary");
```

```
$arr1 = [1 => "Peter", 3 => 2009, "a" => 101];
$arr2 = array(200846369 => array("name" => "Jan␣Olsen",
                                 "COMP518" => 69,
                                 "COMP519" => 52));
```

- Individual array element dapat diakses melalui key

ARRAY ASSIGNMENTS

- Pada JavaScript, arrays adalah objects dan konsekuensinya array assignments dilakukan dengan reference

```
$array2 = $array1;
```

- Pada PHP, menggunakan copy-on-write untuk array assignments
- Jika ingin dua variabel array merujuk ke array yang sama, maka perlu secara eksplisit menggunakan reference

```
$array2 = &$array1;
```

ARRAY LOOP

- Indexed arrays – menggunakan for loop.
- Associative arrays - menggunakan foreach loop.

```
foreach (array as $value)
    statement

foreach (array as $key => $value)
    statement
```

- Multidimensional arrays – menggunakan for loop di dalam for loop lainnya untuk mendapatkan elemen.

Example 1:

```
foreach (array("Peter", "Paul", "Mary") as $key => $value)
   print "The␣array␣maps␣$key␣to␣$value\n";
```

```
The array maps 0 to Peter
The array maps 1 to Paul
The array maps 2 to Mary
```

Example 2:

```
$arr5[2] = "Mary";
$arr5[0] = "Peter";
$arr5[1] = "Paul";
// 0 => 'Peter',  1 => 'Paul',  2 => 'Mary'
foreach ($arr5 as $key => $value)
   print "The␣array␣maps␣$key␣to␣$value\n";
```

```
The array maps 2 to Mary
The array maps 0 to Peter
The array maps 1 to Paul
```

ARRAY FUNCTION

- Size dari array didapatkan dengan fungsi **count**.

```
count(array, mode)
```

```
print count($arr1);      // prints 3
print count($arr2);      // prints 1
print count($arr2,1);    // prints 4
```

- Mengakses undefind key akan menghasilkan PHP notice dan returns NULL

```
array_key_exists("a",$arr1)   # returns TRUE
array_key_exists("c",$arr1)   # returns FALSE
```

- Semua keys dari array dapat diambil menggunakan **array_keys($array1)**
- Semua values dari array dapat diambil menggunakan **array_values($array1)**
- Key-value pair dapat dihapus dari array menggunakan fungsi **unset**:

```
$arr1 = array(1 => "Peter", 3 => 2009, "a" => 101);
unset($arr1[3]);    // Removes the pair 3 => 2009
unset($arr1);       // Removes the whole array
```

ARRAY FUNCTION

- **array_push($array, value1, value2,...)**
  appends one or more elements at the end of the end of an array variable; returns the number of elements in the resulting array
- **array_pop($array)**
  extracts the last element from an array and returns it
- **array_shift($array)**
  shift extracts the first element of an array and returns it
- **array_unshift($array, value1, value2,...)**
  inserts one or more elements at the start of an array variable; returns the number of elements in the resulting array.
- Sorting array: **sort( ), rsort( ), asort( ), ksort( ), arsort( ), krsort( )**

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

Contoh

```php
<?php
$a=array("red","green");
array_push($a,"blue","yellow");
print_r($a);
print("<br />");
array_pop($a);
print_r($a);
print("<br />");
echo array_shift($a);
print("<br />");
print_r ($a);
?>
```

```
Array ( [0] => red [1] => green [2] => blue [3] => yellow )
Array ( [0] => red [1] => green [2] => blue )
red
Array ( [0] => green [1] => blue )
```

SPECIAL TYPES

NULL

- NULL adalah suatu special type dan suatu value

- Suatu variable akan bertipe NULL dan bernilai NULL pada tiga situasi:
  - Variabel belum di-assign suatu value (not equal to NULL)
  - Variabel telah kosong dengan melakukan assign value NULL
  - Dilakukan unset terhadap variable menggunakan operasi **unset**

- Untuk mengecek variable apakah bernilai NULL:
  - **isset($variable)** TRUE if $variable exists and does not have value NULL
  - **is_null(expr)** is identical to NULL

Contoh

```php
<?php
$a = 0;
// True because $a is set
if (isset($a)) {
  echo "Variable 'a' is set.<br>";
}

$b = null;
// False because $b is NULL
if (isset($b)) {
  echo "Variable 'b' is set.";
}
?>
```

Variable 'a' is set.

```php
<?php
$a = 0;
echo "a is " . is_null($a) . "<br>";

$b = null;
echo "b is " . is_null($b) . "<br>";

$c = "null";
echo "c is " . is_null($c) . "<br>";

$d = NULL;
echo "d is " . is_null($d) . "<br>";
?>
```

a is
b is 1
c is
d is 1

RESOURCES

- Tipe data resources bukanlah tipe data yang sebenarnya. Ini adalah penyimpanan referensi ke fungsi dan sumber daya eksternal di luar PHP.

- Digunakan pada fungsi filesystem untuk mengakkses dan memanipulasi filesystem.
  - **fopen(filename, mode)** returns a file pointer resource for filename access using mode on success, or FALSE on error
  - **fclose(resource)** closes the resource
  - **fgets(resource [, length])** returns a line read from resource
  - **fread(resource,length)** returns length characters read from resource
  - **fwrite(resource, string [, length])** writes a string to a resource

Contoh

```
<!DOCTYPE html>
<html>
<body>

<?php
$file = fopen("test.txt","r");
//Output lines until EOF is reached
while(! feof($file)) {
  $line = fgets($file);
  echo $line. "<br>";
}

fclose($file);
?>

</body>
</html>
```

Hello, this is a test file.
There are three lines here.
This is the last line.

CONDITIONS

IF-ELSE

```
if (condition)
  statement
elseif (condition)
  statement
else
  statement
```

*elseif instead of elif or else if

TERNARY OPERATOR

```
condition ? if_true_expr : if_false_expr
```

SWITCH-CASE

```
switch (expr) {
  case expr1:
      statements
      break;
  case expr2:
      statements
      break;
  default:
      statements
      break;
}
```

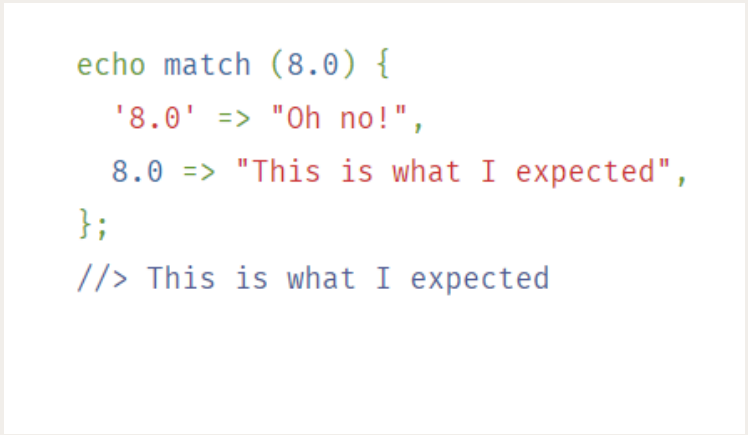
- The switch statement uses loosely comparisons (==)
- The match expression uses strict comparison (===) instead.

```php
<?php
$t = date("H");

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>
```

```php
<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>
```



Match expression baru ada sejak PHP 8

LOOPS

```
while (condition)
    statement
```

```
do
    statement
while (condition);
```

```
for (initialisation; test; increment)
     statement
```

```php
<?php
$x = 0;

while($x <= 100) {
  echo "The number is: $x <br>";
  $x+=10;
}
?>
```

```php
<?php
$x = 6;

do {
  echo "The number is: $x <br>";
  $x++;
} while ($x <= 5);
?>
```

```php
<?php
for ($x = 0; $x <= 100; $x+=10) {
  echo "The number is: $x <br>";
}
?>
```

EXCEPTIONS

- Code may be surrounded in a **try** block, to help catch potential exceptions
- Each **try** block or "**throw**" must have at least one corresponding **catch** block
- A simple rule: If you throw something, you have to catch it.
- a finally clause can be added

```php
<?php
$number=0;
//trigger exception in a "try" block
try {
    if($number>1) {
        throw new Exception("Value must be 1 or below");
    }
  //If the exception is thrown, this text will not be shown
    echo 'If you see this, the number is 1 or below';
}

//catch exception
catch(Exception $e) {
  echo 'Message: ' .$e->getMessage();
}
?>
```

ERROR HANDLING

- PHP membedakan antara exceptions dan errors
- Beberapa cara error handling (https://www.w3schools.com/Php/php_error.asp)
  - Simple "die( )" statements

```php
<?php
if(file_exists("mytestfile.txt")) {
  $file = fopen("mytestfile.txt", "r");
} else {
  die("Error: The file does not exist.");
}
?>
```

  - Creating a Custom Error Handler
  - Error Logging

FUNCTION

DECLARATION

- Functions are defined as follows in PHP:

```
function identifier($param1,&$param2, ...) {
  statements
}
```

- Functions can be placed anywhere in a PHP script Function names are case-insensitive
- The function name must be followed by parentheses
- PHP supports **anonymous functions**

```
$div = function($x,$y) { return $x / $y; };
```

- A function has **zero**, one, or more parameters
- Parameters can be given a default value using `$param = const_expr`
- When using default values, any defaults must be on the right side of any parameters without defaults
- A function can contain more than one return statement

FUNCTION WITH VARIABLE ARGUMENTS

- Saat mendeklarasikan fungsi, jumlah parameter boleh tidak ditentukan.

```php
function sum() { // no minimum number of arguments
  if (func_num_args() < 1) return 0;
  $sum = 0;
  foreach (func_get_args() as $value) { $sum += $value; }
  return $sum;
}
```

- `func_num_args()` returns the number of arguments passed to a function
- `func_get_arg(arg_num)` returns the specified argument, or FALSE on error
- `func_get_args()` returns an array with copies of the arguments passed to a function

FUNCTION WITH VARIABLE ARGUMENTS

- We can use the `...` token in an argument list to denote that the function accepts a variable number of arguments
- The arguments will be passed into the given variable as an array

```
function sum(...$numbers) {
  if (count($numbers) < 1) return 0;
  $sum = 0;
  foreach ($numbers as $value) { $sum += $value; }
  return $sum;
}
echo "1:␣",sum(0,1,2,3), "\n";

echo "2:␣",sum(0,TRUE,"2",3e0), "\n";
```

```
1: 6
2: 6
```

CALLING A FUNCTION

- A function is called by using **the function name** followed by a **list of arguments in parentheses**

```php
<?php
function sum($num1, $num2) {
  return $num1 + $num2;
}

echo sum(5, 4);      //9
echo sum(5, 4, 3);   //9
echo sum(5);         //Fatal error
?>
```

- The list of arguments can be shorter as well as longer as the list of parameters
- If it is shorter, then **default values** must have been specified for the parameters without corresponding arguments
- If no default values are available for ‘missing’ arguments, then we get a **PHP fatal error**

FUNCTION AND VARIABLES SCOPE

PHP has three different variable scopes:

- **Local variables**: A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function
- **Global variables**: A variable declared **outside** a function has a GLOBAL SCOPE and can only be accessed **outside** a function
- **Static variables**: Local to the function, variable will still have the information it contained from the last time the function was called. Normally, when a function is completed/executed, all of its variables are deleted.
- There is **no** hoisting of variable ‘declarations’

```php
$x = "Hello";

function f1() {
  echo "1: $x\n";
}

function f2() {
  echo "2: $x\n";
  $x = "Bye";
  echo "3:",$x,"\n";
}

f1();
f2();
echo "4: $x\n";
```

```
1: PHP Notice:  Undefined variable: x
2: PHP Notice:  Undefined variable: x
3: Bye
4: Hello
```

```php
$x = "Hello";

function f1() {
  global $x;
  echo "1: $x\n";
}
function f2() {
  $x = "Bye";
  echo "2: $x\n";
  global $x;
  echo "3: $x\n";
}
f1();
f2();
echo "4: $x\n";
```

```
1: Hello
2: Bye
3: Hello
4: Hello
```

```
$x = "Hello";


function f3($x) {
  $x .= '!';
  echo "1:␣$x\n";
}
f3('Bye');
echo "2:␣$x\n";
f3($x)
echo "3:␣$x\n";
```

```
1: Bye!
2: Hello
1: Hello!
3: Hello
```

```php
<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>
```

SUPERGLOBALS VARIABLES

- Merupakan variabel bawaan yang selalu tersedia di semua scope

- Beberapa variabel standar dalam PHP adalah "superglobals", yang berarti variabel tersebut selalu dapat diakses dari fungsi, kelas, atau file apa pun tanpa harus melakukan sesuatu yang khusus.

`$GLOBALS` `$_REQUEST` `$_GET` `$_ENV` `$_SESSION`
`$_SERVER` `$_POST` `$_FILES` `$_COOKIE`

- `$GLOBALS` adalah variabel PHP super global yang digunakan untuk mengakses variabel global dari mana saja di skrip PHP (juga dari dalam fungsi).

- PHP menyimpan semua variabel global dalam array yang disebut `$GLOBALS[index]`, dimana index adalah nama variabel.

```php
<?php
$x = 75;
$y = 25;

function addition() {
  $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

- PHP allows the definition of nested functions
- The inner function does not have access to local variables of the outer function
- The inner function can be called from outside the outer function

```
function outer($param1, &$param2, ...) {
  function inner($param3, &$param4, ...) { ... }
}
```

- It is possible to include HTML markup in the body of a function definition, The HTML markup can in turn contain PHP scripts.

```
<?php
function print_form($fn, $ln) {
?>
<form action="process_form.php" method=POST">
<label>First Name: <input type="text" name="f" value="<?php echo $fn ?>">
</label><br>
<label>Last Name<b>*</b>:<input type="text" name="l" value="<?php echo $ln ?>">
</label><br>
<input type="submit" name="submit" value="Submit"> <input type=reset></form>
<?php
}
print_form("Peter","Pan");
?>
```

```
<form action="process_form.php" method=POST">
<label>First Name: <input type="text" name="f" value="Peter"></label><br>
<label>Last Name<b>*</b>:<input type="text" name="l" value="Pan"></label><br>
<input type="submit" name="submit" value="Submit"> <input type=reset></form>
```

FUNCTION AS ARGUMENTS

- PHP allows function names and anonymous functions to be passed as arguments to other functions

```
function apply($f,$x,$y) {
  return $f($x,$y);
}
function mult($x,$y) {
  return $x * $y;
}
$div = function($x,$y) { return $x / $y; };

echo "2 * 3 = ",apply('mult',2,3),"\n";
```

```
2 * 3 = 6
```

```
echo "6 / 2 = ",apply($div,  6,2),"\n";
```

```
6 / 2 = 3
```

```
echo "2 + 3 = ",apply(function($x,$y) { return $x + $y; },
                      2,3),"\n";
```

```
2 + 3 = 5
```

TYPE DECLARATION

- By default, type juggling is still applied in function
- In PHP 7, type declarations were added. This gives us an option to specify the expected data type **for function parameters and for the return value of a function**
- You can specify a different return type, than the parameter types, but make sure the return is the correct type. it will throw a "Fatal Error" on a type mismatch.

```
function addNumbers(float $a, float $b) : int {
  return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
```

INCLUDING AND REQUIRING FILES

- It is often convenient to build up function stored in separated files (library), that are then reused in PHP scripts.
- PHP provides the commands include, include_once, require, and require_once to incorporate the content of a file into a PHP script
- If a library file requested by include and include_once cannot be found, PHP generates a warning but continues the execution of the requesting script
- If a library file requested by require and require_once cannot be found, PHP generates a error and stops execution of the requesting script
- Include_once or require_once commands for the same library file results in the file being incorporated only once

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

REFERENSI

Materi disadur dari:

1. COMP519 Web Programming (2020-21). Diakses pada Januari 2021, dari https://cgi.csc.liv.ac.uk/~ullrich/COMP519/
2. W3Schools Online Web Tutorials. Diakses pada Januari 2021, dari https://www.w3schools.com/

Thanks!

Do you have any questions?

wilantika@stis.ac.id
lya@stis.ac.id
yeni.rima@stis.ac.id

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik