Web Programming with The Laravel Framework

#3 - PHP Fast Track

Steven Lolong

steven.lolong@uni-tuebingen.de

Programming Languages Research Group
Tuebingen University

26 April 2022

Eberhard Karls
Universität
Tübingen

PHP

Start php file with **<?php** and close with **?>**

```php
<?php
/**
 * @author: Steven L.
 * @date: 26 April 2022
 */

//  Show "Hello World!" on web page
$notify = "Hello World!";
echo $notify;

?>
```

Listing: PHP structure – 3-helloWorld.php

Variable

- **$** is the key for variable's name.

```
6  $vStr = "This is string";
7  $vInt = 323;
8  $vBoolean = true; //false
9  $vDouble = 9.9;
10 $vNull = null;
11 $vArray = Array(1,3,5,7);
```

Listing: Variable – 3-var-and-type.php

Type Juggling

- PHP is **dynamic type checking**, it means checking the type when running.
- There is no need to annotate type explicitly.

```
12 echo '$vStr = ' . $vStr . " : " . gettype($vStr) . "<br/>";
13 echo '$vInt = ' . $vInt . " : " . gettype($vInt) . "<br/>"
       ;
14 echo '$vBoolean = ' . $vBoolean . " : " . gettype($vBoolean
       ) . "<br/>";
15 echo '$vDouble = ' . $vDouble . " : " . gettype($vDouble)
       . "<br/>";
16 echo '$vNull = ' . $vNull . " : " . gettype($vNull) . "<br
       />";
17 echo '$vArray = ' . "<br/>";
18 print_r($vArray);
19 echo "<br/> : " . gettype($vArray) . "<br/>";
```

Listing: Variable – 3-var-and-type.php

Type Casting

- (*new_type*) *variable* or *value*
- *settype*(*variable_name*, *new_type*)

```
21 echo 'cast $vInt to string, new type: ' . gettype((string)
       $vInt) . "<br/>";
22 echo 'cast $vBoolean to integer, new type: ' . gettype((int
       )$vBoolean) . "<br/>";
23 echo "cast $vStr to integer, old value: $vStr <br/>";
24 settype($vStr, "integer");
25 echo "New type: " . gettype($vStr) . ", value $vStr <br/>";
```

Listing: Type casting – 3-var-and-type.php

Operator

- Arithmetic: $+, -, *, /, \%, **$
- Assignment: $=, + =, - =, * =, / =, \% =$
- Comparison: $==, ===, ! = (<>), ! ==, <, >, <=, >=, <=>$
- Increment / decrement: $++\$x, \$x++, --\$x, \$x--$
- Logical: *and*, *or*, *xor*, &&, ||, !
- String: ., . =
- Array: $+, ==, ===, ! =, <>, ! ==$
- Conditional: ? :, ??

Operator equivalent

- **$a += $b** is equal to **$a = $a + $b**. This is also applied to $-=, *=, /=, \%=$.
- **++$a** is equal to **$a = $a + 1**. This is also applied to $--$.

Control Statement

- **if** statement, executes some code if on condition is true.
- **if...else** statement, executes some code if a condition is true and another code if that condition is false.
- **if...elseif...else** statement, executes different codes for more than two conditions.
- **switch** statement, selects one from many blocks of code to be executed.

if and if-else

```
if Condition then
    "code to be executed if Condition is true";
end
```

Algorithm 1: if statatement

```
if Condition1 then
    "code to be executed if Condition1 is true";
else
    "code to be executed if Condition1 is false";
end
```

Algorithm 2: if-else statatement

Eberhard Karls Universität Tübingen

if Statement

```
2 $weather = "rain";
3 $temperatur = 10;
4 // if statement
5 if($weather == "rain"){
6   echo "use umbrella <br/>";
7 }
```

Listing: if – 3-control-statement.php

if-else Statement

```
10 if($weather == "rain"){
11   echo "bring umbrella <br/>";
12 }else{
13   echo "leave umbrella at home <br/>";
14 }
```

Listing: if-else – 3-control-statement.php

if-elseif-else Statement

```
17 if($temperatur < 2){
18   echo "wear 4 layers <br/>";
19 }elseif(($temperatur >= 2) && ($temperatur < 10)){
20   echo "wear 3 layers <br/>";
21 }elseif(($temperatur >= 10) && ($temperatur < 18)){
22   echo "wear 2 layers <br/>";
23 }elseif(($temperatur >= 18) && ($temperatur < 25)){
24   echo "wear 1 layers <br/>";
25 }else{
26   echo "wear t-shirt <br/>";
27 }
```

Listing: if-elseif-else – 3-control-statement.php

Switch-Case Statement

```
switch Variable do
    case expression-1 do
        "case-1";
        break;
    end
    case expression-2 do
        "case-2";
        break;
    end
    otherwise do
        "Other";
    end
end
```

Algorithm 3: switch-case-default statatement

Eberhard Karls Universität Tübingen

switch-case statement

```
30 switch($temperatur){
31   case ($temperatur >= 2) && ($temperatur < 10):
32     echo "wear 3 layers <br/>";
33     break;
34   case ($temperatur >= 10) && ($temperatur < 18):
35       echo "wear 2 layers <br/>";
36     break;
37   case ($temperatur >= 18) && ($temperatur < 25):
38       echo "wear 1 layers <br/>";
39     break;
40   default:
41     echo "wear t-shirt <br/>";
42 }
```

Listing: switch-case – 3-control-statement.php

Looping

- **while** pre-test looping, loops through a block of code as long as the specified condition is true.
- **do-while** post-test looping, loops through a block of code once, and then repeats the loop as long as the specified condition is true.
- **for**, loops through a block of code a specified number of times.

While Statement

Pre-test
while Condition do
  Statement;
end

Algorithm 4: while statatement

```
2 $a = 1;
3 // while
4 while($a < 5){
5   echo "a = $a <br/>";
6   $a++;
7 }
```

Listing: while – 3-looping.php

Do-While Statement

Post-test

do

    Statement;

while Condition;

Algorithm 5: while statatement

```
10 do{
11   echo "a = $a <br/>";
12   $a++;
13 }while($a < 5);
```

Listing: do-while – 3-looping.php

```
16 $b = 2;
17 do{
18   echo "b = $b<br/>";
19 }while($b < 2);
```

Listing: post test – 3-looping.php

For Statement

Condition: start value ; condition ; post statement

for Condition do
  Statement;
end

Algorithm 6: for statatement

```
22 for($i = 1; $i < 10; $i++){
23   echo "i = $i <br/>";
24 }
```

Listing: for – 3-looping.php

Nested Looping

```
26 // nested looping
27 for($i = 1; $i < 5; $i++){
28   for($j = $i; $j < 5; $j++){
29     echo "*";
30   }
31   echo "<br/>";
32 }
```

Listing: nested looping – 3-looping.php

What is the output?

Iteration

- **foreach**, loops through a block of code for each element in an array.

foreach *array* **as** *value* **do**
    Statement;
end

Algorithm 7: foreach statatement

```
35 $colors = array("Red", "Green", "Blue");
36 foreach($colors as $value){
37   echo "$value <br/>";
38 }
```

Listing: iteration – 3-looping.php

Iteration

```
41 $colorsHex = array("red" => "#ff0000", "green" => "#00ff00",
       "blue" => "#0000ff");
42 foreach($colorsHex as $indx => $val){
43   echo "Index = $indx, Value = $val <br/>";
44 }
```

Listing: extracting the index – 3-looping.php

What is the output?

Function

- A function is a block of statements that can be used repeatedly in a program.
- A function will not execute immediately when a page loads.
- A function will be executed by a call to the function (apply over value).

Function

syntax
function **funcName** (*parameters*){
statement-1;
⋮
statement-n;
return *value*;
}

Function

```php
// function abstraction
function sayHello(){
  echo "Hello <br/>";
}
// function application (call)
sayHello();
```

Listing: function – 3-functions.php

Function

```
9  function funEmpty(){
10   echo "Hello <br/>";
11 }
```

Listing: No parameter and no return value – 3-functions.php

```
13 function funRet(){
14   return "Hello <br/>";
15 }
```

Listing: Parameter but no return value – 3-functions.php

```
17 function funParam($param1, $param2){
18   echo "param1 = $param1, param2 = $param2 <br/>";
19 }
```

Listing: No parameter but return value – 3-functions.php

```
21 function funAdd($a, $b){
22   return ($a + $b);
23 }
```

Listing: Parameter and return value – 3-functions.php

UNIVERSITÄT TÜBINGEN

Array

An **array** is a special variable, which can hold more than one value at time.

Normal Variabl vs Array Variable

```
2 $bike1 = "Hercules";
3 $bike2 = "Cube";
4 $bike3 = "Rose";
5 // Or
6 $bikes = array("Hercules", "Cuba", "Rose");
7 echo count($bikes) . "<br/>";
```

Listing: Array – 3-array.php

Indexed Arrays

```
10 $bikers = array();
11 $bikers[0] = "Biker_A";
12 $bikers[1] = "Biker_B";
13 $bikers[2] = "Biker_C";
```

Listing: Indexed array – 3-array.php

Associative Arrays

```
16 $bikersAndBikes = array("Biker_A" => "Hercules", "Biker_B"
       => "Cube", "Biker_C" => "Rose");
```

Listing: Associative array – 3-array.php

echo $bikersAndBikes["Biker_B"];

Multidimensional Arrays

```php
$cars = array(
  array("Mercedes", 22, 18),
  array("BMW", 15, 13),
  array("Audi", 4, 12),
);
```

Listing: Multidimensional array – 3-array.php

How to print all data in Multidimensional Array?