Unit 2: Control Structures and Loop

Presented By:
Bipin Maharjan

Table of Contents

Control Structures

Control structures are core features of the PHP language that allow your script to respond differently to different inputs or situations. This could allow your script to give different responses based on user input, file contents, or some other data.

Control structure define the flow of program execution.

There are two main types of control structures in PHP:

- Conditional statements => Decision Making
- Loops => Repetition

Conditional Statements

Conditional statements in PHP are used to perform different actions based on different conditions. They control the flow of execution in a script.

Conditional statements allow you to branch the path of execution in a script based on whether a single or multiple conditions evaluate to true or false.

Types:

- if statement
- if-else statement
- if-elseif-else statement
- Switch statement
- Ternary operator

if Statement

- The **if** statement is used to execute a block only if a given condition evaluates to true.
- If the condition is true, the code inside the block runs; otherwise, it is skipped.

Syntax:

```
if (condition) {
    // code to execute if condition is true
}
```

if Statement: Example

```
<?php
$name = “Akira”;
$age = 20;

if ($age >= 18) {
	echo “$name is eligible to vote.”;
}
?>
```

if-else Statement

- The **if-else** statement executes the if block of code if the condition is true, otherwise, if the condition is false, the else block of code is executed.

Syntax:

```
if (condition) {
        // code if condition is true
} else {
        // code if condition is false
}
```

if-else Statement: Example

```
<?php
$name = “Akira”;
$score = 95;

if ($score >= 50) {
	echo “$name has passed the test.”;
} else {
	echo “$name has failed the test.”;
}
?>
```

if-elseif-else Statement

- The **if-elseif-else** statement checks multiple conditions sequentially and executes the first matching block.
- Helps handle multiple possible outcomes instead of just two.
- If the first condition is false, it moves to the next elseif; if none match, the else block executes.

Syntax:

```
if (condition1) {
        // code if condition1 is true
} elseif (condition2) {
        // code if condition2 is false
} else {
        // code if all conditions are false
}
```

if-elseif-else Statement: Example

```
<?php
$name = “Akira”;
$score = 95;

if ($score >= 90) {
	echo “$name got an A grade.”;
} elseif ($score >=75) {
	echo “$name got a B grade.”;
} else {
	echo “$name got a C grade.”;
}
?>
```

switch Statement

- The **switch** statement checks a variable multiple values and executes the corresponding block.
- If a match is found, the corresponding block executes; **break** prevents fall-through to the next case.
- It is similar to **if-elseif-else** statement.

Syntax:

```
switch (variable) {
        case value1:
                //code to execute if variable == value1
                break;
        case value2:
                //code to execute if variable == value2
                break;
                …
        default:
                //code if no case matches
}
```

Working of Switch Statement

- First we have a single expression n (most often a variable), that is evaluated once.
- The value of the expression is then compared with the values for each case in the structure.
- If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically.
- The default statement is used if no match is found.

switch Statement: Example

```
<?php
$name = “Akira”;
$day = “Monday”;

switch ($day) {
	case “Monday”:
		echo “$name has a test today.”;
		break;
	case “Wednesday”:
		echo “$name is planning to sleep.”;
		break;
	default:
		echo “$name has a regular day.”;
}
?>
```

Ternary Operator

- The ternary operator is a shorthand for **if-else,** returning one of two values based on a condition.
- Also known as conditional operator.
- The first statement is executed if the condition is true, else the second statement is executed.

Syntax:

variable = (condition) ? value_if_true : value_if_false;

Ternary Operator: Example

```
<?php
$name = “Akira”;
$score = 95;

$result = ($score >=50) ? “$name passed the exam!” : $name failed the exam!”;

echo $result;
?>
```

Looping Statements

- Often when we write code, you want the same block of code to run over and over again a certain number of times. So, instead of adding several almost equal code-lines in a script, we can use loops.
- Looping statements are used to execute a block of code repeatedly based on a condition.
- Loops are useful for performing repetitive tasks, such as iterating over arrays, processing user input, generating output, etc.
- Types:
  - **while** Loop
  - **do-while** Loop
  - **for** Loop
  - **foreach** Loop

while Loop

- The **while** loop executes code as long as the given condition remains **true**.
- Used when the number of iterations is **not predetermined.**
- It is an entry controlled loop, i.e. the condition is checked before executing the block.

Syntax:

```
while (condition) {
    //code to execute
}
```

while Loop: Example

```
<?php
$counter = 1;

while ($counter <= 4) {
	echo “Akira is studying for exam $counter. <br>”;
	$counter++;
}
?>
```

do-while Loop

- Similar to **while** loop, but block executes at least one, even if the condition is false.
- It is an exit controlled loop, i.e. the condition is checked after the block of code is executed.

Syntax:

```
do {
    //code to execute
} while (condition)
```

do-while Loop: Example

```
<?php
$counter = 1;

do {
	echo “Akira is studying for exam $counter. <br>”;
	$counter++;
} while ($counter <= 3)
?>
```

for Loop

- The **for** loop runs a block of code a specific number of times.
- It consists of three parts:
  - Initialization: Sets a starting value.
  - Condition: Loop continues while this is true.
  - Increment/Decrement: Updates the loop variable.

Syntax:

```
for (initialization; condition; increment/decrement) {
    //code to execute
}
```

for Loop: Example

```
<?php
for ($i = 1; $i <=5; $i++) {
        echo “Task no. $i. <br>”;
}
?>
```

foreach Loop

- The **foreach** loop is used to iterate over arrays.
- It automatically assigns each array element to a variable.
- Used to access each individual elements of an array.

Syntax:

```
foreach ($array as $value) {
    //code to execute
}
```

foreach Loop: Example

```
<?php
$names = [“Akira”, “Ayaka”, “Ayumi”];

foreach ($names as $name) {
	echo “Student name is $name. <br>”;
}
?>
```

Jumping Statements

- Jumping statements are used to control the flow of execution by skipping or stopping iterations inside loops and conditional structures.
- Types:
  - break Statement
  - continue Statement

break Statement

- Immediately exists a loop or **switch** statement.
- Used when a specific condition is met, and further iterations are unnecessary.

break Statement: Example

```
<?php
for ($i = 1; $i <= 5; $i++){
	if ($i == 3) {
		echo “Stopping at $i<br>”;
		break;
	}
	echo “Step $i completed.<br>”;
}
?>
```

continue Statement

- Skips the current iteration without exiting the loop.
- The loop continues with the next iteration.

continue Statement: Example

```
<?php
for ($i = 1; $i <= 5; $i++){
	if ($i == 3) {
		echo “Skipping step $i … <br>”;
		continue;
	}
	echo “Step $i completed.<br>”;
}
?>
```

Any Questions?