Control Structures

Disclaimer: вы смотрите просто запись лекции, это **НЕ** специально подготовленный видеокурс!

Intro

Most control structures in PHP are similar to any other C-style language, still there are some special cases…

Basics: if … elseif … else

These structures are the most common:

```php
<?php

// The simples case:
if ($someVariable > 10) {
    // Do something
}

// Extended case:
if ($someVariable > 10) {
    // Do something
} else {
    // Do something else
}
```

```php
<?php

// PHP-specific
if ($someVariable > 10) {
    // Do something
} elseif ($someVariable < 1) {
    // Do other something
} else {
    // Do something else
}
```

Basics: if … elseif … else

These structures may be compound and nested:

```php
<?php

if (($someVariable > 15) && ($someOtherVariable < 5)) {
    if ($someCondition) {
        // Do something
    }
}
```

Basics: switch

In case you need a lot of alternatives to consider, you may use switch…

```php
<?php

$someVariable = 10.7;

// In PHP switch supports integers, doubles, strings, booleans, etc...
switch ($someVariable) {
    case 10:
        echo 'Ten';
        break; // This keyword is mandatory here.
    case 10.7:
        echo 'Ten point seven';
        break;
    case 'ABC':
        echo 'Some string';
        break; // This keyword is mandatory here.

    // This section is optional:
    default:
        echo 'Something unknown';
}
```

Basics: match

Since PHP 8 another approach is available:

```php
<?php

$inputValue = '2';

$resultValue = match ($inputValue) {
    0 => "hello",
    '1', '2', '3' => "world",
};

echo $resultValue; // 'world'
```

Loops: while

In PHP **while** loops behave just like their C counterparts.

```php
<?php

$someVariable = 3;
while ($someVariable > 0) {
    echo $someVariable-- . "\n"; // 3 2 1
}
```

Loops: do … while

Do … while loop body will be executed at least one time always.

```php
<?php

$someVariable = 10;
do {
    echo $someVariable-- . "\n"; // 10
} while ($someVariable > 50);
```

Loops: for

In PHP **for** loops behave just like their C counterparts.

```php
<?php

for ($someVariable = 0; $someVariable < 3; $someVariable++) {
    echo $someVariable . "\n"; // 0 1 2
}
```

Loops: foreach

Many other languages support this functionality via **for** loops.

```php
<?php

$someArray = ['name' => 'John', 'surname' => 'Smith'];

foreach ($someArray as $key => $value) {
    echo $key . ' = ' . $value . "\n";
}
// name = John
// surname = Smith

foreach ($someArray as $value) {
    echo $value . "\n";
}
// John
// Smith
```

Loops: how to skip an iteration or to exit the loop

There are two useful keywords for loops management: continue and break.

```php
<?php

for ($i = 0; $i < 999; $i++) {
    if ($i < 500) {
        continue; // Skips the rest of the body, initiates new iteration.
    }
    echo $i . "\n";
    break; // Forces the loop to end.
}
```

Methods and functions: return

To return a value from a method or a function we may use **return** keyword.

```php
<?php

function sqr(int|float $x): float
{
    return $x * $x;
}

echo sqr(2); // 4

class Math
{
    public static function sqr(int|float $x): float
    {
        return $x * $x;
    }
}

echo Math::sqr(2); // 4
```

PHP-specific: declare

The **declare** construct is used to set execution directives for a block of code.

```php
<?php

declare(strict_types=1);
declare(encoding='ISO-8859-1');

// Code here
```

Control Structures

Disclaimer: вы смотрите просто запись лекции, это **НЕ** специально подготовленный видеокурс!