

Webmediaproductie 5

PHP - Basisconcepten

Kristof Michiels

Lesinhoud: waar gaan we heen?

Dit zal je kunnen na afloop van deze les

- Het plaatsen van PHP binnen webpagina's
- Data sturen als output naar de browser
- Comments toevoegen in je code
- Variabelen en hun gebruik
- PHP foutmeldingen
- Een HTML5 "template" creëren

Dit zal je kunnen na afloop van deze les

- Objecten en hun gebruik
- Strings samenvoegen
- URL variabelen oproepen met \$_GET
- Een class definitie "declareren"
- Dynamische CSS embedden

Plaatsen van PHP binnen webpagina's

PHP scripts insluiten

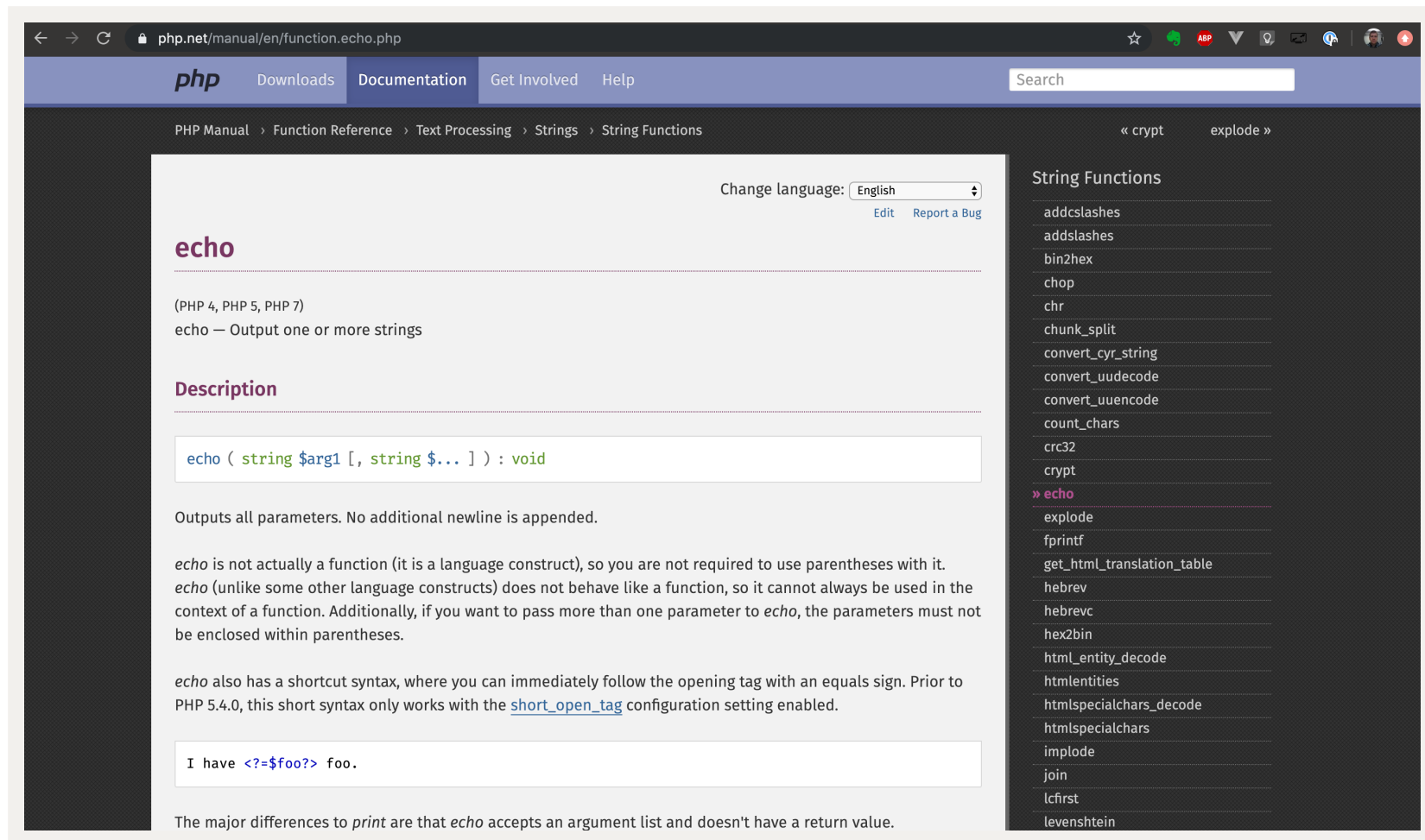
- Voor de server een opgevraagd bestand naar de browser stuurt, wordt eerst het PHP script uitgevoerd
- Hoe weet de server waar te zoeken naar PHP?
- Om te beginnen zoekt de server enkel in bestanden met de .php-extensie
- Een .php-bestand kan uiteraard ook andere zaken bevatten (zoals html)
- Om efficiënt te kunnen zoeken maken we gebruik van PHP scheidingstekens: `<?php` en `?>`
- Alles buiten die scheidingstekens wordt behandeld als html
- Je mag zoveel PHP blokken toevoegen als je wil

PHP scripts insluiten

```
<p>Dit is wat statische tekst</p>  
<?php echo "<p>Deze tekst werd aangemaakt door PHP!</p>"; ?>  
<p>Deze tekst niet.</p>
```

```
<?php echo "<p>Dit is wat tekst.</p>"; ?>  
<p>Deze tekst is statisch,  
<?php echo " maar deze tekst dan weer niet"; ?>  
</p>  
<?php echo "<p>"; ?>  
Deze tekst wordt begrensd door p-elementen aangemaakt door php.  
<?php echo "</p>"; ?>
```

Echo: output sturen naar de browser



The screenshot shows the PHP Manual page for the `echo` function. The page is titled "echo" and is part of the "String Functions" section. It includes a "Description" section that explains that `echo` is a language construct, not a function, and provides a code example: `echo (string $arg1 [, string $...]) : void`. The example shows the output: "I have <?=\$foo?> foo." The page also includes a "Change language" dropdown menu and a "Search" bar. On the right side, there is a list of "String Functions" including `addslashes`, `bin2hex`, `chr`, `chunk_split`, `convert_cyr_string`, `convert_uuencode`, `count_chars`, `crc32`, `crypt`, `explode`, `fprintf`, `get_html_translation_table`, `hebrew`, `hebrevc`, `hex2bin`, `html_entity_decode`, `htmlentities`, `htmlspecialchars_decode`, `htmlspecialchars`, `implode`, `join`, `lcfirst`, and `levenshtein`.

php.net/manual/en/function.echo.php

php Downloads Documentation Get Involved Help

PHP Manual > Function Reference > Text Processing > Strings > String Functions

Change language: English Edit Report a Bug

echo

(PHP 4, PHP 5, PHP 7)

echo — Output one or more strings

Description

```
echo ( string $arg1 [, string $... ] ) : void
```

Outputs all parameters. No additional newline is appended.

`echo` is not actually a function (it is a language construct), so you are not required to use parentheses with it. `echo` (unlike some other language constructs) does not behave like a function, so it cannot always be used in the context of a function. Additionally, if you want to pass more than one parameter to `echo`, the parameters must not be enclosed within parentheses.

`echo` also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign. Prior to PHP 5.4.0, this short syntax only works with the [short_open_tag](#) configuration setting enabled.

```
I have <?=$foo?> foo.
```

The major differences to `print` are that `echo` accepts an argument list and doesn't have a return value.

String Functions

- `addslashes`
- `addslashes`
- `bin2hex`
- `chop`
- `chr`
- `chunk_split`
- `convert_cyr_string`
- `convert_uuencode`
- `convert_uuencode`
- `count_chars`
- `crc32`
- `crypt`
- » echo**
- `explode`
- `fprintf`
- `get_html_translation_table`
- `hebrew`
- `hebrevc`
- `hex2bin`
- `html_entity_decode`
- `htmlentities`
- `htmlspecialchars_decode`
- `htmlspecialchars`
- `implode`
- `join`
- `lcfirst`
- `levenshtein`

Variabelen

Variabelen

- Een variabele is een woord dat verwijst naar een waarde die in het systeemgeheugen is opgeslagen
- Je code kan een reeks acties uitvoeren afhankelijk van de inhoud van een variabele
- De waarde van de variabele veranderen resulteert dan in een andere output
- In één enkele lijn kan je in PHP een variabele declareren en er een waarde aan toekennen

```
<?php
    $mijnNaam = "Pedro";
    $vriendNaam = "Ronny";
    echo "<p>Mijn naam is $mijnNaam en mijn vriend heet $vriendNaam.</p>";
?>
```

Naamgeving van je variabelen

- In PHP beginnen alle variabelen verplicht met een \$
- Er zijn andere beperkingen, maar als je enkel alfabetische karakters gebruikt is er nooit een probleem
- Gebruik geen spaties of speciale karakters zoals !"#€%&/.
- Je mag nummers gebruiken, maar niet als eerste karakter. \$1a kan niet, \$a1 wel
- bij PHP kan je camelCasing gebruiken, net als bij JS

PHP foutmeldingen

PHP foutmeldingen

- Fouten schrijven in je PHP-code is onvermijdelijk
- Feedback krijgen over die fouten is belangrijk. PHP kan foutmeldingen voorzien
- Je kan de fout dan verbeteren en er uit leren
- Niet alle fouten worden getoond door PHP, tenzij je er om vraagt
- Onthou: plaats beide regels op volgende slide bovenaan elke php-pagina
- Pas op: foutmeldingen zijn soms cryptisch. Vaak moeten google of bvb stackoverflow.com raad brengen

PHP foutmeldingen

- <https://www.php.net/manual/en/function.error-reporting.php>
- <https://www.php.net/manual/en/function.ini-set.php>

```
<?php
    // deze lijnen dragen PHP op om alle foutmeldingen weer te geven in browser
    error_reporting( E_ALL );
    ini_set( "display_errors", 1 );
?>
```

Een HTML-template creëren

Een HTML-pagina maken in PHP

- PHP is een fantastische taal om op spaarzame wijze dynamische HTML-pagina's te maken
- Hier maak je in het geheugen een valide HTML5 pagina die je vervolgens echo-t naar de browser
- Probeer zelf de code uit en bekijk zeker ook de source code in je browser

```
<?php
    error_reporting( E_ALL );
    ini_set( "display_errors", 1 );
    $titel = "Mijn portfolio - startpagina";
    $content = "<h1>Wereld, maak kennis met mijn portfolio!</h1>";
    $pagina = "<!DOCTYPE html><html><head><title>$titel</title>
    <meta http-equiv='Content-Type' content='text/html; charset=utf-8' /></head>
    <body>$content</body></html>";
    echo $pagina;
?>
```


Omzetten naar een HTML-template voor alle pagina's

- We zetten onze denkoefening verder
- Maak een pagina aan en noem ze pagina.php. We stoppen het bestand in een folder met de naam templates
- Het return statement is zeer handig: vergelijkbaar met een echo die je maar éénmaal uitvoert per pagina
- Elke waarde die volgt op return wordt teruggegeven. Daarna stopt het script met uitvoeren, ook al zou er nog extra code onder de return statement staan

```
<?php
    return "<!DOCTYPE html><html><head><title>$title</title>
    <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
    </head><body>$content</body></html>";
?>
```

Omzetten naar een HTML-template voor alle pagina's

- Om een ander bestand in te voegen in een ander bestand gebruik je het statement include once
- We spreiden hiermee code over verschillende bestanden. We noemen dit bestand index.php.
- De template zit nu in een afzonderlijk bestand en kan herbruikt worden in verschillende contexten

```
<?php
    error_reporting( E_ALL );
    ini_set( "display_errors", 1 );
    $titel = "Mijn portfolio - startpagina";
    $content = "<h1>Hallo wereld!</h1>";
    //gebruik het relatieve pad naar het bestand dat je wil gebruiken
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
?>
```

PHP Comments

Comments in je code

- Gebruik // voor een enkele regel commentaar
- of /* en */ voor commentaar gespreid over meerdere regels
- Net zoals bij css dus :-)

```
<?php
    //dit is een enkele lijn-commentaar
    /* Dit is commentaar gespreid over verschillende regels
    we noemen dat een block-commentaar */
?>
```

Objecten en classes

Naamgevingsconflicten vermijden

- Ga even mee in het onderstaande gedachtenexperiment
- Je kan op termijn PHP projecten maken met honderden regels code.
- Je zal veel variabelen aanmaken die allen uniek moeten zijn qua naamgeving
- In volgend voorbeeld wordt tegen deze regel gezondigd

```
<?php
    $titel = "Welkom op mijn blog";
    /* honderden regels code later */
    $titel = "Webontwikkelaar";
    /* tweemaal dezelfde variabele aanmaken, dat is een probleem
    Gelukkig zijn er oplossingen :-) */
?>
```

PHP Objecten

- Je kan problemen te vermijden door een context te creëren in de vorm van een object
- Je creëert een standaard PHP object door gebruik van stdClass
- Een PHP object is net zoals een variabele waarin je waarden kan opslaan
- Een variabele kan één waarde bevatten. In een object kan je zoveel waarden onderbrengen als je nodig hebt

```
<?php
    $paginaData = new stdClass();
    $paginaData->titel = "Welkom op mijn blog";
    /* honderden regels code later */
    $jobData = new stdClass();
    $jobData->titel = "Web ontwikkelaar";
?>
```

PHP Objecten

- Objecten kunnen dus gebruikt worden als namespaces voor eigenschappen. Zo worden conflicten vermeden en wordt een duidelijke context geboden
- Om waarden van een object eigenschap te krijgen moet je 2 zaken aangeven: het object en de eigenschap
- De object operator ziet eruit als een pijltje. Dat geeft aan dat je een bepaalde eigenschap wenst binnen een specifiek object

```
$objectNaam = new StdClass();  
$objectNaam->eigenschapNaam = "waarde";  
$objectNaam->andereEigenschapNaam = "andere waarde";  
echo $objectNaam->eigenschapNaam;
```


Aanpassen van onze index.php en pagina.php

```
<?php
    error_reporting( E_ALL );
    ini_set( "display_errors", 1 );
    $paginaData = new stdClass();
    $paginaData->titel = "Nieuwe object-georiënteerde test titel";
    $paginaData->content = "<h1>Hallo, object!</h1>";
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
?>
```

```
<?php
    return "<!DOCTYPE html><html><head><title>$paginaData->titel</title><
    meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
    </head><body>$paginaData->content</body></html>";
?>
```

De HTML-template gebruiken voor elke pagina

De HTML-template gebruiken voor elke pagina

We creëren twee pagina's in een nieuwe subfolder views: views/films.php en views/muziek.php

```
<?php
    return
        "<h1>Mijn favoriete films</h1>
        <p>Hier lees je alles over mijn passie voor films</p>";
?>
```

```
<?php
    return
        "<h1>Mijn favoriete muziek</h1>
        <p>Hier komt mijn CD top 10</p>";
?>
```

De HTML-template gebruiken voor elke pagina

Vervolgens maken we views/navigatie.php aan, om dynamische site navigatie toe te voegen

```
<?php
    return "
        <nav>
            <a href='index.php?pagina=films'>Films</a>
            <a href='index.php?pagina=muziek'>Muziek</a>
        </nav>";
?>
```

De HTML-template gebruiken voor elke pagina

De navigatie voegen we als volgt toe aan index.php (maar we zijn nog niet helemaal klaar):

```
<?php
    error_reporting( E_ALL );
    ini_set( "display_errors", 1 );
    $paginaData = new stdClass();
    $paginaData->titel = "Mijn persoonlijke site";
    $paginaData->content = include_once "views/navigatie.php";
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
?>
```

De HTML-template gebruiken voor elke pagina

- We moeten er nog voor zorgen dat de bezoeker kan navigeren tussen "muziek" en "films"
- Informatie doorgeven van pagina naar pagina is belangrijk bij dynamische pagina's
- We kunnen informatie doorgeven via URL- variabelen
- Als we op de links in de nav klikken dan wordt in de adresbalk een URL variabele meegegeven naar de volgende pagina. Die kan daar gebruikt worden

```
index.php?pagina=films  
index.php?pagina=muziek
```

De HTML-template gebruiken voor elke pagina

- Om de informatie uit de URL-variabelen te krijgen gebruik je de \$_GET superglobal array
- We maken de volgende aanpassing aan onze index.php:

```
<?php
    $paginaData = new stdClass();
    $paginaData->titel = "Mijn persoonlijke site";
    $paginaData->content = include_once "views/navigatie.php";
    $navigatieIsClicked = isset($_GET['pagina']);
    if ($navigatieIsClicked ) {
        $bestandTeLaden = $_GET['pagina'];
        $paginaData->content .= "<p>Hier moet $bestandTeLaden.php komen</p>";
    }
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
?>
```

De HTML-template gebruiken voor elke pagina

- De `isset()` functie gaat na of de variabele bestaat. Deze zal `true` teruggeven als de variabele effectief bestaat

```
$navigatieIsClicked = isset($_GET['pagina']);
```

- Met `$_GET` krijg je toegang tot alle URL variabelen

```
$_GET['variabele_naam'];
```

- Heb je daarnet de `.=` in de code gespot? Wordt de incrementele concatenatieoperator genoemd

```
$paginaData->content .= "<p>Hier moet $bestandTeLaden.php komen</p>";
```


De HTML-template gebruiken voor elke pagina

We wijzigen onze index.php: op deze manier laden we de juiste view op basis van de aanwezige URL-variabele

```
<?php
    $paginaData = new stdClass();
    $paginaData->titel = "Mijn persoonlijke site";
    $paginaData->content = include_once "views/navigatie.php";
    $navigatieIsClicked = isset($_GET['pagina']);
    if ($navigatieIsClicked ) {
        $bestandTeLaden = $_GET['pagina'];
        $paginaData->content .= include_once "views/$bestandTeLaden.php";
    }
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
?>
```

De HTML-template gebruiken voor elke pagina

We kunnen index.php een standaardpagina laten tonen indien er geen URL-variabele wordt meegegeven

```
//deelcode voor index.php
if ($navigatieIsClicked ) {
    $bestandTeLaden = $_GET['pagina'];
} else {
    $bestandTeLaden = "muziek";
}
$paginaData->content .=include_once "views/$bestandTeLaden.php";
```

CSS toevoegen aan onze structuur



CSS toevoegen aan onze structuur

We doen dit met een nieuwe paginadata eigenschap in onze pagina.php

```
<?php
    return "<!DOCTYPE html><html> <head><title>$paginaData->titel</title>
    <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
    $paginaData->css</head><body>$paginaData->content</body></html>";
?>
```

CSS toevoegen aan onze structuur

... en in index.php vullen we deze eigenschap in

```
<?php
    // ...
    $paginaData = new stdClass();
    $paginaData->titel = "Mijn persoonlijke site";
    $paginaData->content = include_once "views/navigatie.php";
    $paginaData->css = "<link href='css/styling.css' rel='stylesheet' />";
    // ...
?>
```

Van stdClass naar custom class

Werken met custom classes

- We gaan een custom class aanmaken voor pagina_data
- Maak een nieuwe folder aan die je classes noemt
- Maak in deze folder een nieuwe file aan die je Pagina_Data.class.php noemt
- Conventie = begint met een hoofdletter, woorden scheiden met een underscore, laten eindigen op .class.php

Pagina_data class aanmaken

We voegen eerst nog snel een extra eigenschap toe aan pagina.php ;-)

```
<?php
    return "<!DOCTYPE html><html> <head><title>$paginaData->titel</title>
    <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
    $paginaData->css $paginaData->embeddedStyle</head><body>
    $paginaData->content</body></html>";
?>
```


Pagina_data class aanmaken

En vervolgens maken we Pagina_Data.class.php aan:

```
<?php
class Pagina_Data {
    public $titel = "";
    public $content = "";
    public $css = "";
    public $embeddedStyle = "";
}
```

Een object instantiëren vanuit een custom class

Doe je als volgt in je index.php

```
//...  
include_once "classes/Pagina_Data.class.php";  
$paginaData = new Pagina_Data();  
//...
```

De volledige index.php wordt dan...

```
<?php
    include_once "classes/Pagina_Data.class.php";
    $paginaData = new Pagina_Data();
    $paginaData->titel = "Mijn persoonlijke site";
    $paginaData->content = include_once "views/navigatie.php";
    $paginaData->css = "<link href='css/styling.css' rel='stylesheet'>";
    $navigatieIsClicked = isset($_GET['pagina']);
    if ($navigatieIsClicked ) {
        $bestandTeLaden = $_GET['pagina'];
    } else {
        $bestandTeLaden = "muziek";
    }
    $paginaData->content .=include_once "views/$bestandTeLaden.php";
    $paginaData->embeddedStyle = " <style> nav a[href *= '?pagina=$fileToLoad']{
        background-color:white;} </style>";
    $pagina = include_once "templates/pagina.php";
    echo $pagina;
```

Tot besluit

- In deze presentatie: belangrijke basisconcepten van PHP gezien
- Deze kennis zal je toelaten data aan te maken, op te slaan, te manipuleren en uit te sturen naar de browser
- We breiden in de volgende presentatie onze PHP-kennis verder uit
- Het is van het grootste belang dat je de opgedane kennis inoefent
- Breid de site uit die we in deze presentatie zijn begonnen
- Meng je ondertussen opgedane skills op vlak van HTML, CSS en JavaScript met PHP scripting. Experimenteer!
Het is de sleutel tot een goede kennis van PHP
- Opdracht: maak een eenvoudige persoonlijke website met minstens 5 pagina's waarin je de concepten die we vandaag zagen gebruikt

WMP5 - PHP Basisconcepten