3.1 van functie naar object (oop)

In deze les gaan we leren wat een global variabele is en we gaan een functie maken waarbij we gebruik gaan maken van global variabelen. In sommige gevallen kan dat functioneel zijn. Zeker bij grotere programma's heeft het gebruik van global variabele nadelen en leidt snel tot fouten. Hoe we dit probleem oplossen is met object oriented programmeren en we gaan onze eerste stukje OOP maken.

Functies en globale variabelen

In het volgende filmpje wordt een functie gemaakt en wordt een voorbeeld gegeven van het gebruik van *globale vari*abele in een functie. Een *globale* variabele heeft een globale scope en kan *overal* in de code worden gebruikt. Dit in tegenstelling tot een variabele in een functie; die heeft een *lokale* scope en kan alleen in de functie worden gebruikt.

https://www.youtube.com/embed/9CCGc4xZy_0

In de video wordt de onderstaande besproken. We gaan deze code aanpassen.

```
<?php

global $som;
global $aantal;

$som=0;
$aantal=0;

function berekenGemiddelde($cijfer){
  global $som, $aantal;
  $som=$som+$cijfer;
  $aantal++;
  $gemiddelde=$som/$aantal;
  return($gemiddelde);
}

echo "Gemiddelde: ".berekenGemiddelde(6)."<br>";
echo "Gemiddelde: ".berekenGemiddelde(7)."<br>";
```

```
echo "Gemiddelde: ".berekenGemiddelde(8)."<br>";


?>
```

Opgave 1

De functie *berekenGemmidelde($cijfer)*, moet worden gesplits in twee nieuwe functies.

cijferToevoegen($cijfer), met deze functie voegen we een cijfer toe.

gemiddeldeBerekenen(), met deze functie wordt het gemiddelde berekend

Met deze twee nieuwe functies moet het gemiddelde cijfer worden berekend van alle cijfers uit het volgende array.

```
$cijfersPHP=[5,5,6,5,7,6,7,5,8,7,8,6];
```

Gebruik een for-loop en voeg elke cijfer uit het array toe aan de lijst van gemiddelden met de functie *cijferToevoegen($cijfer).*

De functie *cijferToevoegen($cijfer)* wordt dus voor elk cijfer uit het array cijferPHP een keer aangeroepen.

Als alle cijfers zijn toegevoegd, druk je het gemiddelde af met de functie *gemiddeldeBerekenen()*.

Zet de nieuwe code in de file opdracht1.php en lever deze in.

OOP

Om de basis van OOP uit te leggen ga ik eerst aan de hand van een voorbeeld met een hond een paar nieuwe begrippen uitleggen. Als we die begrippen dan kennen ga ik in het tweede filmpje uitleggen hoe we het voorbeeld van het gemiddelde kunnen omzetten naar een object waarbij we de functies en de variabelen combineren in één object.

Begrippen

Class, Object, Properties en Methods worden nog een keer duidelijk gemaakt in:

> https://slides.com/jamescandan/oop-php-for-beginners-1-2/fullscreen#/5

https://www.youtube.com/embed/tZJcsFp9ttU

Dus een class is een sjabloon van een object en met het commando new <class_naam> maken we een nieuw object van de sjabloon. In een class heten de variabelen proeprties en de fucnties heten methods.

Code

Nu gaan we ons gemiddelde in een class zetten.

https://www.youtube.com/embed/z7mB1jRRy_Y

De code die we in het filmpje uitleggen:

```
<?php

class MijnGemiddelde {
  private $som;
  private $aantal;

  public function __construct() {
    $this->reset();
  }

  public function addNumber($number) {
    $this->som=$this->som+$number;
    $this->aantal++;
  }

  public function reset() {
    $this->som=0;
    $this->aantal=0;
  }

  public function gemiddelde() {
    if ($this->aantal) {
      return($this->som/$this->aantal);
    } else {
      return(0);
    }
  }
```

```
}

$gemiddeldePHP = new MijnGemiddelde;
$gemiddeldeJava = new MijnGemiddelde;

$gemiddeldePHP->addNumber(8);
$gemiddeldePHP->addNumber(6);
echo "Gemiddelde PHP: ".$gemiddeldePHP->gemiddelde();

echo "<hr>";

$gemiddeldeJava->addNumber(5);
$gemiddeldeJava->addNumber(7);
echo "Gemiddelde Java: ".$gemiddeldeJava->gemiddelde();

echo "<hr>";

$gemiddeldeJava->reset();
echo "Gemiddelde Java: ".$gemiddeldeJava->gemiddelde();

?>
```

In de volgende opdrachten wordt het voorbeeld van hierboven aangepast en uitgebreid.

Maak alle opdrachten en zet de aangepaste code in opdracht2.php.

Opdracht 2

Maak een nieuw object $*gemiddeldeLinux*.

Voeg de cijfers 8 en 7 toe aan het object $*gemiddeldeLinux*.

Druk het gemiddelde van Linux af op dezelfde manier zoals in het voorbeeld de andere gemiddelde zijn afgedrukt.

Zet de aangepaste code in opdracht2.php.

Opdracht 3

Pas de method gemiddelde aan, zodat het gemiddelde cijfer op één decimaal afgerond wordt gereturned.

zet de aangepaste code in opdracht2.php.

Opdracht 4

Voeg aan de class *MijnGemiddelde* een method toe waarmee je cijfers kunt toevoegen die 2x meetellen.

Dus stel je hebt een 5 en een 8 gehaald, maar de 8 telt twee maal mee. Dan is het gemiddelde 5+8+8=21 en 21/3=7

Maak nu een method en noem die *gewogenCijfer($cijfer, $gewicht).* Je roept de method dan aan op de volgende manier.

```
$gemiddeldePHP = new MijnGemiddelde;
$gemiddeldePHP = $gemiddeldePHP->gewogenCijfer(5,1); // voeg het cijfer 5 toe, deze telt 1x mee
$gemiddeldePHP = $gemiddeldePHP->gewogenCijfer(8,2); // voeg het cijfer 8 toe, deze telt 2x mee

echo "Gemiddelde PHP: ".$gemiddeldePHP->gemiddelde(); // het resultaat zou 7 moeten zijn
```

Pas de class *MijnGemiddelde* aan en maak een de nieuwe method *gewogenCijfer($cijfer, $gewicht)*

Zet de aangepaste code in opdracht2.php.

Lever de file opdracht2.php (waarin de uitwerking van opdracht 2, 3 en 4 zit) in.

--

Revision #15
Created 2020-02-26 20:16:42 UTC by Max
Updated 2020-04-29 14:52:21 UTC by Max