Back End-utveckling i PHP

50 YHP

```
this.week() {

 return {
   tuesday: "php",
   wednesday: "php",
   thursday: "php",
   friday: "php"
  }
 }
```

FORTSÄTTNING PHP

Fredag 31 Januari 2018

what we know so far...

- PHP är en preparser för html, och andra, filer
- **Input** -> PHP Parsning -> **Output** -> Annan parser
- PHP är sketasvårt att installera på Windows
- PHP liknar Javascript men är på många ställen enklare.

what we know so far...

- Öppnings- och stängningstaggarna **<?php ?>**
- "värden" och $variabler //även kommentarer
- operationer **+ - = .= <>**
- Några enkla statements (for, echo, switch, while)

Installera PHP

WINDOWS:

- Nerladdning: http://windows.php.net/download/
- Guide: http://kizu514.com/blog/install-php7-and-composer-on-windows-10/

MAC:

- curl -s http://php-osx.liip.ch/install.sh | bash -s 7.1
- echo 'export PATH=/usr/local/php5/bin:$PATH' >>~/.bash_profile

PHPs utvecklingsserver

- PHP installationen innehåller en server för utveckling
- Startas med > *php -S hostname:port*
- e.g. **php -S localhost:8080**

Funktioner

- Förutom dom inbyggda operationerna kan man i PHP skriva egna funktioner

- En funktion skapas med keywordet **function**

```
function namnetPåMinFunktion(argument...) {}
```

- Koden skriven i funktionen körs först när man *anropar* eller *kallar* på funktionen

Funktioner

- Koden som skrivs innanför funktionens {} blir ett eget **local scope**
- Värden kan skickas in i en funktion som argument
- Ett värde kan skickas ut ur en funktion med keywordet **return**

Funktioner

Exempel på en funktion:

```
function fullName($firstName, $familyName) {
   $fullName = $firstName." ".$familyName;

   return $fullName;
} // dekleration

fullName("lisa", "simpson"); // anrop
```

Funktioner

- $variabler är som lådor med värden i.
- När man deklarerar en funktion så skriver man argument (*$firstName*, *$lastName*) som finns som variabler i funktionens local scope
- Vad händer när man kallar på en funktion med en variabel som argumentets värde??
  fullName($myFirstName)

Funktioner

- $variabler är som lådor med värden i.
- När man deklarerar en funktion så skriver man argument (*$firstName*, *$lastName*) som finns som variabler i funktionens local scope
- Vad händer när man kallar på en funktion med en variabel som argumentets värde??
  fullName($myFirstName)

Värdet i variabeln flyttas från en låda till en annan. $firstName = $myFirstName

Övning: Funktioner

- Börja med filen vi skrev igår.
- Skriv en ny funktion (myCoolLoop) och flytta in Loopen vi skrev igår till funktionens scope.
- Istället för att alltid skriva ut 10 gånger, ändra så att myCoolLoop tar ett argument med hur många gånger loopen ska köras. Kör loopen på detta sätt tjugo gånger.

Funktioner i funktioner

- En av fördelarna med funktioner är att dom kan anropas på vid flera olika tillfällen.
- Detta gör dem återanvändningsbara
- Vissa problem kan lösas genom att kalla en funktion från samma funktion, detta kallas rekursion 💪
- Exempel på vanliga rekursiva funktioner är sökningar i mappar, sortering samt vissa matematiska problem.

Demo: Rekursion

- För ett heltal större än noll är fakulteten lika med produkten av alla heltal från 1 upp till och med talet självt.
- För talet 0 är fakulteten 1
- För talet 5 är fakulteten 5 * 4 * 3 * 2 * 1

Scope

Allt mellan taggarna <php? ?> kallas **global scope**.
Detta scope återställs varje gång filen körs

Varje funktion har ett **local scope.** Detta scope återställs för varje gång funktionen körs

Inga variabler existerar mellan dessa scope automatiskt d.v.s. globala variabler finns inte inuti era funktioner!

Övning: Global Scope

- Skapa en ny **php fil** med två styckten php styckten (<?php ?>)
- I det första stycket skapar ni en variabel och ger den värdet "foobar"
- Finns denna variabel i det andra stycket? Har den värdet foobar eller något annat?
- **Skrik inte ut svaret!** 🤫

Lösning: Global Scope

- Yes :) Varje fil är samma global scope
- *Men mellan olika filer är scopet olika*

Arrays

I PHP skapas listor / arrayer med en funktion som heter **array**

```
$myArray = array("first", "second", "third");
```

värden läses ut som i JS

```
$myArray[0] //"first"
```

I PHP får man ut hur lång en array är genom att skicka den till funktionen **count**

```
$length = count($myArray);
```

Dagens långövning

- Utgå från samma fil igen 🙂
- Skriv en funktion som heter minsta och returnerar det minsta av två tal.
- Skapa i globalscope en lista över bra låtar 🎶 (minst 10 stycken)
- Nu vill vi att funktionen myCoolLoop ska ha tre argument, en start, stop och en array
- myCoolLoop ska skriva ut innehållet i arrayen mellan start och stop
- Om stop sätts till en siffra som är större än vad arrayen är lång ska sidan fungera ändå, använd minsta()

foreach

Det finns ett keyword i PHP som heter foreach

det är en variant på en vanlig for loop som alltid går igenom alla värden i en hel array.

Uppfunnen på senare tid för att det är ett vanligt sätt att skriva for loopar.

foreach

```
$simpsons = array("lisa", "homer", "marge",
"bart");

foreach ($simpsons as $firstName) {

    echo fullName($firstName, "simpson");

}
```

foreach

```
$simpsons = array("lisa", "homer", "marge",
"bart");

for($i = 0; $i < count($simpsons); $i++) {

    $firstName = $simpsons[0];
    echo fullName($firstName, "simpson");

}
```

Tack för idag!