

# PHP INTRO

PHP Hypertext Processor

VAD ÄR PHP?

# PHP



Ref: 7php - Why Elephant?

```
<!doctype html>
<html>
  <head>
    <title>index.php</title>
  </head>
  <body>

  </body>
</html>
```

- Server-side Scripting
- All purpose: vi kan skriva t.ex. CLI-program
- PHP är inte servern, PHP körs PÅ servern.

- Apache/NGINX är programmet som levererar våra PHP-filer
- Våra PHP-filer tolkas på servern
- Servern levererar HTML, CSS & JS
- PHP är INTE det användaren ser men det som servern tolkar

# <?PHP ?>

```
<h1>This is PHP!</h1>
```

```
<?php  
    echo '<h1>This is also PHP!</h1>'  
?>
```

```
<h1> <?php echo "This is also PHP!" ?> </h1>
```

Allting utanför `<?php ?>` tolkas som **HTML**

Allting innanför `<?php ?>` tolkas som **PHP**



**EXEMPELKOD**

# MÅSTE JAG...?

Om din fil innehåller **PHP** behöver du inte en sluttagg

Om din fil innehåller både **PHP** och **HTML** måste du använda sluttagg

Du behöver inte skriva ut **php**

```
<? ?> == <?php ?>
```

Men det är att föredra, failsafe.

Använd alltid semikolon;

# SYNTAX & FUNKTIONALITET

**VARIABLER**

# DYNAMISKT TYPAT

Liknande JavaScript har vi bara en typ av variabel `var`

Vi behöver dock inte skriva `var` men vi måste skriva `$`

```
<?php
$name = 'Jesper';
$money = 30;
echo $name;
echo $money;
```

# CASTING

```
<?php
$money = '30';           //type: string
$cash = (int)$money;      //type: int
```

## CASTING

- `(int), (integer)` - cast to integer
- `(bool), (boolean)` - cast to boolean
- `(float), (double), (real)` - cast to float
- `(string)` - cast to string
- `(array)` - cast to array
- `(object)` - cast to object
- `(unset)` cast to NULL



```
<?php
```

```
$foo = 1 + "Whaddup10"; // $foo is 1 (int)  
$foo = 1 + "10Waddups"; // $foo is 11 (int)  
$foo = 1 + "10.5Waddups"; // $foo is 11.5 (float)  
$foo = "10.5Waddups" + 1; // $foo is 11.5 (float)
```

Strängar castas till float/int **om första karaktären är en siffra** annars ignoreras strängen.

It's all the same to me

```
<?php
$foo = 10;           // $foo is an integer
$str = "$foo";       // $str is a string
$fst = (string) $foo; // $fst is also a string
```

# STRING CONCATENATION

```
<?php
$first = 'Jesper';
$last = 'Orb';
$full = $first . $last;
$fullWithBraces = "{$first} {$last}";
```

Strängar sätts ihop med `.` istället för `+` som i JS

# """"QUOTES?!""""

Single & double quotes matter

```
<?php
$first = "Jesper";
$last = "Orb";
$full = "$first $last";           //It works
$fullWithDouble = "{$first} {$last}"; //It works
$fullWithSingle = '{$first} {$last}'; //Does not work
```

**DEBUGGING**

**JavaScript**: `console.log( );`

**PHP**: `var_dump( );`

Om inget syns, lägg till detta i början på filen

```
ini_set('display_errors', 1);  
ini_set('display_startup_errors', 1);  
error_reporting(E_ALL);
```

```
<?php
$name = 'Jesper';
var_dump($name); //Jesper
```

Man kan också döda hela processen

```
<?php
$name = 'Jesper';
die(var_dump($name)); //Jesper
```

All kod slutar köras efter `die();`

Ibland jobbar man med mycket data.

```
<?php
$huge = 10000000000000000;
die(var_dump($huge)); //I just took a huge dump and died :0
```



# ARRAYS

En del skillnader, samma tänk

```
<?php
```

```
$array = array(); //Empty array
```

```
$array2 = array("first index", "second index"); //With values
```

Ref: PHP.net - Arrays

## EFTER PHP 5.4

Vi använder troligtvis PHP 7.0/7.1

Lite som med ES6, oftast OK att använda men kan ställa till på äldre servrar.

```
<?php
$array = []; //Empty array
$array2 = ["first index", "second index"]; //With values
```

# PHP.NET - ARRAY FUNCTIONS

```
<?php
$array = array("koala", "elephant");
array_push($array, "sloth"); // array_push(array, item);
```

```
<?php
$array = ["koala", "elephant"];
$array[] = "sloth";
```

# ASSOCIATIVE ARRAYS

PHPs svar på JS Object

## KEY/VALUE

```
<?php
$assoc = array(
    "key" => "value"
);
```

```
<?php
$person = array(
    "name" => "Jesper",
    "age" => "Infinite"
);
```

Separeras med **=>** istället för **:**

# CONDITIONALS

if/elseif/else/switch/while/dowhile/for



**INGET NYTT Egentligen**

Programmering som programmering

PHP.net: Control Structures

# FOREACH

Väl använd for-loop

```
<?php
$animals = array("Sloth", "Panda");
foreach($animals as $animal){
    echo $animal;
}
```

## FOREACH ASSOCIATIVE

```
<?php
$panda = array(
    "name" => "Panda",
    "weight" => "3000"
);
foreach($panda as $key => $value){
    echo $key . $value;
}
```

Vi kan använda oss utav både **key** och **value** om vi vill.

# PRAKTISK TILLÄMPNING I KOD

samt **Alternative syntax**

**FUNCTIONS**

```
<?php
function foo(){
    echo "Foo!";
}
```

## RETURN VS. ECHO

```
<?php
function foo(){
    echo "Foo!"; //public broadcasting
}
```

```
<?php
function foo(){
    return "Foo!"; //just returns
}
```

**echo** SKRIKER UT DET. **return** returnerar bara tyst.

SCOPE



PHP utgår alltid ifrån att variablerna är lokala till scopet.

Vi har bara function scope

```
let foo = 42;  
function bar(){  
  console.log(foo); //42  
}
```

Allting i det yttre scopet finns i det inre

```
<?php
$foo = 42;
function bar(){
    echo $foo;    //No, nein, nothing
}
```

**Variabler finns enbart i det scope de är deklarerade i.**

```
<?php
$foo = 42;
function bar(){
    global $foo;
    echo $foo; //42
}
```

Alternativt referera till den globala variabeln. Oftast en dålig idé.

# ÖVNINGAR PÅ GITHUB

Göra sig bekväm med språket