

OBJECT ORIENTED PHP

PHP

Ref: 7php - Why Elephant?

OBJECTS

PHP: OBJECTS INSTANTIATED FROM CLASSES

När vi vill ha samma beteende för en rad olika saker

Vi är alla människor, men vi har olika namn, ålder etc.

Alla produkter har olika pris men vi har ändå
samlingsnamnet "produkt"

```
$products = array(  
    array(  
        "name" => "A thing",  
        "price" => "500"  
    ),  
    array(  
        "name" => "Another",  
        "price" => "1000"  
    )  
);
```

Upprepande samt vi kan lätt skriva fel

Struktur - var sak på sin plats

Modularitet - återanvänd kod

DRY - mindre upprepning



OBJECTS

When talking about objects, you refer to variables belonging to these objects as properties (or attributes or fields), and functions are called methods.

Varje gång man använder **new** så skapar man ett nytt objekt utifrån en klass

```
$pdo = new PDO( "", "", "" );
```

Vi kan skapa flera kopplingar till flera databaser, men funktionaliteten är densamma

```
$remote_database = new PDO();  
$local_database = new PDO();
```


Klassen är som en ritning över hur objektet ska fungera

Objekt är det som skapas utifrån ritningen

```
class Elephant
{
    //empty class
}
```

```
$hathi = new Elephant;
```

I `$hathi` har vi skapat ett nytt objekt utifrån klassen

CLASS METHOD


En funktion som är bunden till varje objekt

```
<?php
class Elephant
{
    function speak(){
        echo "ERRRRUUUUH!!!!";
    }
}
```

INSTANTIERING

Vi måste skapa ett objekt för att kunna använda metoderna i klassen

```
<?php  
$hathi = new Elephant; //new keyword  
$hathi->speak(); //ERRRRUUUUH!!!!
```

 används för att komma åt egenskaper och metoder i objekt

Properties

```
<?php
class Elephant
{
    public $name = "Hathi";
    function speak(){
        echo "ERRRRUUUUH!!!!";
    }
}
```

```
$hathi = new Elephant;
echo $hathi->name;
```

`public` har med åtkomst att göra.

Vi kan även sätta: `private` eller `protected`

Men vi återkomma till de imorgon mer när vi pratar
om `arv`.

```
<?php
class Elephant           //Class
{
    public $name = "Hathi"; //property
    function speak(){      //method
        echo "ERRRRUUUUH!!!!";
    }
}
```

```
//Instansiate new object from class
$hathi = new Elephant;
echo $hathi->name;           //Use class methods
```

\$THIS

Nyckelordet `$this` pekar på det egna objektet

```
$hathi = new Elephant;  
$mowgli = new Elephant;
```

Hathi & Mowgli är två olika elefanter, två olika

`$this`

De delar dock gemensamma egenskaper som inte är
kopplade till `$this`: `speak () ;`

```
<?php
class Elephant
{
    public $name = "Hathi";
    function speak(){
        echo $this->name;
    }
}
```

```
$hathi = new Elephant;
$hathi->speak();    //"Hathi"
```

```
<?php
class Elephant
{
    public $name = "Hathi";
    function speak(){
        echo $this->name;
    }
    function setName($new_name){
        $this->name = $new_name;
    }
}
```

```
$elephant = new Elephant;
$elephant->speak();    //"Hathi"
$elephant->setName("Mowgli");
$elephant->speak()    //"Mowgli"
```

STATIC

En metod kan vara bunden till klassen och inte till objektet.

Metoden kan användas utan att instatiera ett objekt

STATIC KEYWORD

```
<?php
class Elephant
{
    static function speak()
    {
        echo "ERRUUH!";
    }
}
```

```
<?php
Elephant::speak(); // 'ERRUUH!'
```

Kan tala utan instans

```
<?php
class Elephant
{
    static public $name = "Hathi";
}
```

```
<?php
Elephant::$name; // 'Hathi'
```

Kan ha ett namn utan instans och hör till klassen

"Hathi" hör till alla elefanter, inte de enskilda elefanterna.

Använd klasser när du:

- Ska skapa flera objekt med likadant beteende
- När du vill gruppera funktionalitet som hör ihop
- När du upptäcker att du upprepar dig mycket
- För att "säkra upp" din kod, mindre risk för att skriva fel