OBJECT ORIENTED PHP

OBJECTS

JS: OBJECTS LINKED TO OTHER OBJECTS

PHP: OBJECTS INSTANTIATED FROM CLASSES

VARFÖR?

Struktur

Modularitet

I JavaScript skapar vi objekt utifrån andra objekt

```
let elephant = {};
```

```
let elephant = Object.create({});
```

Allting är alltid kopplat till Object

I PHP skapar vi objekt utifrån `classes`

Liknar ES6-classes

```php
<?php
class Elephant
{


}
```

OBJECTS

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

CLASS METHOD

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

INSTANTIERING

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

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

`->` används för att komma åt egenskaper och metoder i objekt

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

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

CONSTRUCTOR

```php
<?php
class Elephant
{
  public $name; //public property

  public function __constructor($name){
      $this->name = $name; //JS: this.name = name;
  }
}
```

Man måste inte ha **public** på konstruktorn

```php
<?php
$hathi = new Elephant('Hathi');
$hathi->name; //Hathi
```

PRIVATE, PROTECTED, PUBLIC

PRIVILEGIER

I PHP har vi olika nyckelord för åtkomst.

Principle of Least Privilege

ENCAPSULATION

Inkapsling

Lägger samman sådant som hör ihop till samma objekt, minskar antalet globala variabler etc.

Information hiding: vi gömmer så mycket som möjligt, koden ska inte veta för mycket.

```php
<?php
class Elephant
{
  public $name; //public property

  public function __constructor($name){
      $this->name = $name; //Always public
  }
}
```

Kan kommas åt hur som helst, helgalet

```php
<?php
class Elephant
{
  private $name; //private property

  public function __constructor($name){
      $this->name = $name; //Always public
  }
}
```

Kan endast kommas åt från inuti klassen

Men vi måste komma åt `$name`

Oftast har man getter/setter

Funktioner som hämtar värden åt en

```php
class Elephant
{
  private $name; //private property

  public function __constructor($name){
      $this->name = $name; //Always public
  }
  public function getName(){
    return $this->name; //Only returns name
  }
}
```

```php
//...
private $name;
public function getName(){
    return $this->name; //Only returns name
}
public function setName($name){
  if($name == ''){
    return 'No empty name!';
  }
  $this->name = $name;
}
```

Vi bestämmer vem som kan komma åt värdena och hur värden kan sättas

- `private` : bara egna klassen kan komma åt
- `public` : alla kan komma åt
- `protected` : klassen och alla klasser som `ärver` av klassen kan komma åt (återkommer till den imorgon)

STATIC

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

Metoden kan användas utan att instatiera ett objekt

"Statiska" metoder i JavaScript

```
function Foo() {}; //constuctor
Foo.speak = function() { //Not bound to object
  console.log('Foooo!');
};
```

```
Foo.prototype.speak = function() { //Bound to object
  console.log('Foooo!'); //can reference 'this'
};
```

STATIC KEYWORD

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

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

Kan tala utan instans

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

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

Kan ha ett namn utan instans

IMORGON: INHERITANCE

IDAG: OOP ÖVNINGAR