More in PHP and Web
“OOP in PHP starts in PHP 5”

-php.net
What is an object

The fundamental idea behind an object-oriented language is to enclose a bundle of variables and functions into a single unit and keep both variables and functions safe from outside interference and misuse. An object contains:

- Method (function)
- Property (variable)
Definition in OOP

- **Class** – This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.

- **Object** – An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.

- **Member Variable** – These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.

- **Member function** – These are the function defined inside a class and are used to access object data.

- **Inheritance** – When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.

- **Parent class** – A class that is inherited from by another class. This is also called a base class or super class.

- **Child Class** – A class that inherits from another class. This is also called a subclass or derived class.
Definition in OOP

• Polymorphism – This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it take different number of arguments and can do different task.

• Overloading – a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.

• Data Abstraction – Any representation of data in which the implementation details are hidden (abstracted).

• Encapsulation – refers to a concept where we encapsulate all the data and member functions together to form an object.

• Constructor – refers to a special type of function which will be called automatically whenever there is an object formation from a class.

• Destructor – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.
Class

- Basic class definitions begin with the keyword `class`, followed by a class name, followed by a pair of curly braces which enclose the definitions of the properties and methods belonging to the class.

- The class name can be any valid label, provided it is not a PHP reserved word. A valid class name starts with a letter or underscore, followed by any number of letters, numbers, or underscores.

- **keywords:** `class`, `const`, `static`, `public`, `private`, `protected`, `$this`, `::`, `->`, `.self`
Class

• **class**: keyword to define class

• **const**: keyword to create a constant variable

• **static**: keyword to create a static variable

• **Variable visibility (accessibility):**
  
  • **private**: visible (accessible) only inside the parent, no access is granted from outside the class.
  
  • **protected**: visible (accessible) in the parent and child class, No access is granted from outside the class except a class that’s a child of the class with the protected property or method.
  
  • **public**: visible (accessible) to all, can be accessed from outside the class.

• **$this**: keyword to access current class

• **:: and ->**: to access class’s member
class Car{
    const constant_1 = "hello PHP";
    private $wheel = 4;
    protected $topSpeed = 200;
    public function __construct(){
        echo "Constructor <br/>";
    }
    public function __destruct(){
        echo "Destructor <br/>";
    }
    public function setTopSpeed($newTopSpeed){
        $topSpeed = $newTopSpeed;
    }
    public function showTopSpeed(){
        echo $this->topSpeed . "<br/>";
    }
    public function printCons(){
        echo self::constant_1 . "<br/>";
    }
}

$cars = new Car();
$cars->printCons();
$cars->showTopSpeed();
// error, protected property
// echo $cars->topSpeed;

class Mercedes extends Car{
    public function __construct(){
        $this->topSpeed = 300;
        echo "Mercedes constructor <br/>";
    }
}
$typeCMercy = new Mercedes();
$typeCMercy->showTopSpeed();
Session

- A PHP session solves this problem by allowing you to store user information on the server for later use (i.e. username, shopping cart items, etc).

- Session information is temporary and is usually deleted very quickly after the user has left the website that uses sessions.

- Sessions work by creating a unique identification (UID) number for each visitor and storing variables based on this ID. This helps to prevent two users' data from getting confused with one another when visiting the same webpage.

- To start session you must call: `session_start()` before you create a session variable.

- To destroy all session just call: `session_destroy()`.
Cookies

• A cookie is often used to identify a user.

• A cookie is a small file that the server embeds on the user's computer.

• Each time the same computer requests a page with a browser, it will send the cookie too.

• You can both create and retrieve cookie values.
• Syntax:
  setcookie(name, value, expire, path, domain, secure, httponly);

• Only name parameter is required, all others is optional
PHP and Form

• action: filename (url) that will handle form data when it submitted.

• method: post or get.

  • Get method: is an array of variables passed to the current script via the URL parameters.

  • Post method: is an array of variables passed to the current script via the HTTP POST method.

• Start with <form> and end with </form>

<html>
<body>
<form action="destination.php" method="post">
   Name: <input type="text" name="name"><br>
   E-mail: <input type="text" name="email"><br>
   <input type="submit">
</form>
</body>
</html>
Form Elements

- Text fields
- Radio buttons
- Checkbox
- File
- Combobox
Upload file

```html
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" enctype="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id="fileToUpload">
    <input type="submit" value="Upload Image" name="submit">
</form>

</body>
</html>
```