Php

class

```php
<?php
class Fruit {
  // code goes here...
}
?>
```

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}
?>
```

```php
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit();
$banana = new Fruit();


$banana->set_name('Banana');

echo $apple->get_name();
echo "<br>";
echo $banana->get_name();
?>
<?php
class Fruit {
  // Properties
  public $name;
  public $color;

  // Methods
  function set_name($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
  function set_color($color) {
    $this->color = $color;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit();
$apple->set_name('Apple');
$apple->set_color('Red');
echo "Name: " . $apple->get_name();
echo "<br>";
echo "Color: " . $apple->get_color();
?>
```

```
$apple = new Fruit();

var_dump($apple instanceof Fruit);

?>
```

Constructor

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function get_name() {
    return $this->name;
  }
}

$apple = new Fruit("Apple");
echo $apple->get_name();
?>
```

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  function get_name() {
    return $this->name;
  }
  function get_color() {
    return $this->color;
  }
}

$apple = new Fruit("Apple", "red");
echo $apple->get_name();
echo "<br>";
echo $apple->get_color();
?>
```

__destruct

__destruct() function that is automatically called at the end of the script

```
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name) {
    $this->name = $name;
  }
  function __destruct() {
    echo "The fruit is {$this->name}.";
  }
}

$apple = new Fruit("Apple");
?>
```

Access Modifiers

- public - the property or method can be accessed from everywhere. This is default
- protected - the property or method can be accessed within the class and by classes derived from that class
- private - the property or method can ONLY be accessed within the class

```
<?php
class Fruit {
  public $name;
  protected $color;
  private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '۳۰۰'; // ERROR
?>
```

OOP – Inheritance

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

// Strawberry is inherited from Fruit
```

```
class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}
$strawberry = new Strawberry("Strawberry", "red");
$strawberry->message();
$strawberry->intro();
?>
```

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  protected function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

class Strawberry extends Fruit {
  public function message() {
    echo "Am I a fruit or a berry? ";
  }
}

// Try to call all three methods from outside class
$strawberry = new Strawberry("Strawberry", "red");  // OK. __construct()
is public
$strawberry->message(); // OK. message() is public
$strawberry->intro(); // ERROR. intro() is protected
?>
```

```
<?php
class Fruit {
  public $name;
  public $color;
  public function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  public function intro() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

class Strawberry extends Fruit {
  public $weight;
  public function __construct($name, $color, $weight) {
    $this->name = $name;
    $this->color = $color;
    $this->weight = $weight;
  }
  public function intro() {
    echo "The fruit is {$this->name}, the color is {$this->color}, and the
weight is {$this->weight} gram.";
  }
}

$strawberry = new Strawberry("Strawberry", "red", ۵۰);
$strawberry->intro();
?>
```

OOP - Class Constants

Class constants can be useful if you need to define some constant data within a class.

```
<?php
class Goodbye {
  const LEAVING_MESSAGE = "Thank you for visiting
W۳Schools.com!";
}

echo Goodbye::LEAVING_MESSAGE;
?>
```

PHP - What are Abstract Classes and Methods?

```
<?php
// Parent class
abstract class Car {
  public $name;
  public function __construct($name) {
    $this->name = $name;
  }
  abstract public function intro() : string;
}

// Child classes
class Audi extends Car {
  public function intro() : string {
    return "Choose German quality! I'm an $this->name!";
  }
}

class Volvo extends Car {
  public function intro() : string {
    return "Proud to be Swedish! I'm a $this->name!";
  }
}

class Citroen extends Car {
  public function intro() : string {
    return "French extravagance! I'm a $this->name!";
  }
}

// Create objects from the child classes
$audi = new audi("Audi");
echo $audi->intro();
echo "<br>";

$volvo = new volvo("Volvo");
echo $volvo->intro();
echo "<br>";

$citroen = new citroen("Citroen");
echo $citroen->intro();
?>
```

OOP – Interfaces

Interfaces allow you to specify what methods a class should implement.

- Interfaces cannot have properties, while abstract classes can
- All interface methods must be public, while abstract class methods is public or protected

- All methods in an interface are abstract, so they cannot be implemented in code and the abstract keyword is not necessary
- Classes can implement an interface while inheriting from another class at the same time

:Commented [sb1]

```
<?php
interface Animal {
  public function makeSound();
}

class Cat implements Animal {
  public function makeSound() {
    echo "Meow";
  }
}

$animal = new Cat();
$animal->makeSound();
?>
```

OOP – Traits

PHP only supports single inheritance: a child class can inherit only from one single parent.

So, what if a class needs to inherit multiple behaviors? OOP traits solve this problem.

```
<?php
trait message۱ {
  public function msg۱() {
    echo "OOP is fun! ";
  }
}

trait message۲ {
  public function msg۲() {
    echo "OOP reduces code duplication!";
  }
}

class Welcome {
  use message۱;
}

class Welcome۲ {
  use message۱, message۲;
}

$obj = new Welcome();
$obj->msg۱();
echo "<br>";

$obj۲ = new Welcome۲();
```

```
$obj۲->msg۱();
$obj۲->msg۲();
?>
```

Static Methods

Static methods can be called directly - without creating an instance of the class first.

Static methods are declared with the static keyword:

```
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }
}

// Call static method
greeting::welcome();
?>
```

PHP Namespaces

Namespaces are qualifiers that solve two different problems:

۱. They allow for better organization by grouping classes that work together to perform a task

۲. They allow the same name to be used for more than one class

```
<?php
$table = new Html\Table();
$row = new Html\Row();
?>
```

Sql

Pdo

```
<?php
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
?>
```

```
<?php
try {
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
} catch (PDOException $e) {
// attempt to retry the connection after some timeout for example
}
```

```
$stmt = $pdo->query("SELECT * FROM users ORDER BY id DESC
LIMIT ١");
```

```
$user = $stmt->fetch();
```

```
// select a particular user by id
```

```
$stmt = $pdo->prepare("SELECT * FROM users WHERE id=?");
```

```
$stmt->execute([$id]);
```

```
$user = $stmt->fetch();
```

Pdo defaults

```
$host = '۱۲۷.۰.۰.۱';

$db   = 'test';

$user = 'root';

$pass = '';

$charset = 'utf۸mb۴';

$dbh = new PDO("mysql:host=$host;dbname=$db",$user, $pass);

$dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);


$dbh->query("create database newdatabase");

$dbh->query("use newdatabase");
```

```
$dsn = "mysql:host=$host;dbname=$db;charset=$charset";
$options = [
    PDO::ATTR_ERRMODE            => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES   => false,
];
$pdo = new PDO($dsn, $user, $pass, $options);
```

insert

```
$sql = "INSERT INTO users (name, surname, sex) VALUES (?,?,?)";
$stmt= $pdo->prepare($sql);
$stmt->execute([$name, $surname, $sex]);
```

```
$stmt = $pdo->prepare("SELECT * FROM auction WHERE name LIKE ?")
$stmt->execute(array("%$query%"));
```

```
// iterating over a statement
foreach($stmt as $row) {
    echo $row['name'];
}
```

UPDATE

```
$sql = "UPDATE users SET name=?, surname=?, sex=? WHERE id=?";
$stmt= $pdo->prepare($sql);
$stmt->execute([$name, $surname, $sex, $id]);
```

```
$sql = "DELETE FROM users WHERE id=?";
$stmt= $pdo->prepare($sql);
$stmt->execute([$id]);
```

Select:

```
// select a particular user by id
$stmt = $pdo->prepare("SELECT * FROM users WHERE id=?");
$stmt->execute([$id]);
$user = $stmt->fetch();
```

```
$stmt = $pdo->prepare("SELECT * FROM users LIMIT ?, ?");
$stmt->execute([$limit, $offset]);
```

```
while ($row = $stmt->fetch()) {
    echo $row['name']."<br />\n";
}
```

```
$stmt = $pdo->prepare("SELECT * FROM users LIMIT :limit, :offset");
$stmt->execute(['limit' => $limit, 'offset' => $offset]);
$data = $stmt->fetchAll();
// and somewhere later:
foreach ($data as $row) {
    echo $row['name']."<br />\n";
}
```

DELETE table

DROP TABLE *table_name*;

Pdo delele

```
$sql = "DELETE FROM users WHERE id=?";
$stmt= $pdo->prepare($sql);
$stmt->execute([$id]);
```

TRUNCATE

TRUNCATE TABLE *table_name*;

ADD Column

```
ALTER TABLE table_name
ADD column_name datatype;
```

```
ALTER TABLE Customers
ADD Email varchar(۲۵۵);
```

DROP COLUMN

```
ALTER TABLE Customers
DROP COLUMN Email;
```

RENAME COLUMN

```
ALTER TABLE table_name
RENAME COLUMN old_name to new_name;
```

DROP COLUMN

```
ALTER TABLE Persons
DROP COLUMN DateOfBirth;
```

PRIMARY KEY

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(۲۵۵) NOT NULL,
   FirstName varchar(۲۵۵),
   Age int,
   PRIMARY KEY (ID)
);
```

```
alter table Person add primary key (personId,Pname,PMID)
```

CHECK

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(۲۵۵) NOT NULL,
    FirstName varchar(۲۵۵),
    Age int,
    CHECK (Age>=۱۸)
);
```

```
ALTER TABLE Persons
ADD CHECK (Age>=۱۸);
```

CREATE INDEX

```
CREATE INDEX idx_lastname
ON Persons (LastName);
```

AUTO_INCREMENT

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(۲۵۵) NOT NULL,
    FirstName varchar(۲۵۵),
    Age int,
    PRIMARY KEY (Personid)
);
```

Injection

```
$stmt = $dbh->prepare("INSERT INTO Customers
(CustomerName,Address,City)
VALUES (:nam, :add, :cit)");
$stmt->bindParam(':nam', $txtNam);
$stmt->bindParam(':add', $txtAdd);
$stmt->bindParam(':cit', $txtCit);
$stmt->execute();
```

.htaccess

```
DirectoryIndex home.html

DirectoryIndex index.html home.html config.php
```

Block a specific IP or range of IPs:

```
Order Deny,Allow

Deny from ۱۹۲.۲۰۶.۲۲۱.۱۴۰
```

(Here ۱۹۲.۲۰۶.۲۲۱.۱۴۰ is a specific IPv۴ Address)

```
Order Allow,Deny

Deny from ۱۹۲.۱۹۲.*.*

Allow from all
```

---

۳۰۱ Permanent Redirect

```
Redirect ۳۰۱ / http://domain.com

RewriteEngine on
RewriteCond %{HTTP_HOST} ^geeksforgeeks.com [NC]
RewriteRule ^(.*)$ http://www.geeksforgeeks.com/$۱ [L,R=۳۰۱,NC]
```