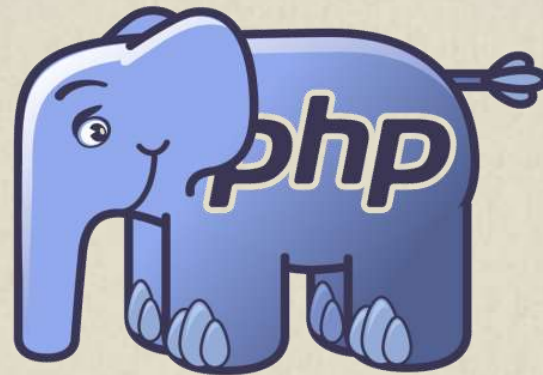


PHP: ВВЕДЕНИЕ В ПРОФЕССИЮ

УРОК 6. КЛАССЫ И ОБЪЕКТЫ. ВВЕДЕНИЕ В ООП.



КЛАССЫ И ОБЪЕКТЫ

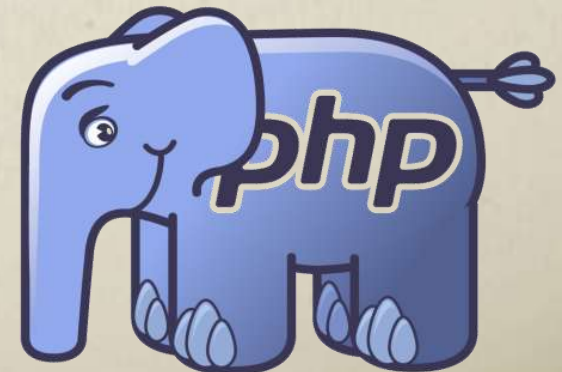
Класс – это описание будущих объектов

- Представим, что у нас есть столы. Разные. Большие и маленькие:



- Однако у всех этих столов есть нечто общее:
 - Ножки
 - Плоская столешница
- Мы может записать, что
Стол – это нечто, что обязательно имеет ножки и столешницу и обладает дополнительными свойствами: материал, цвет, размер, раскладывается ли (да/нет)
- Мы только что написали класс!

КЛАСС

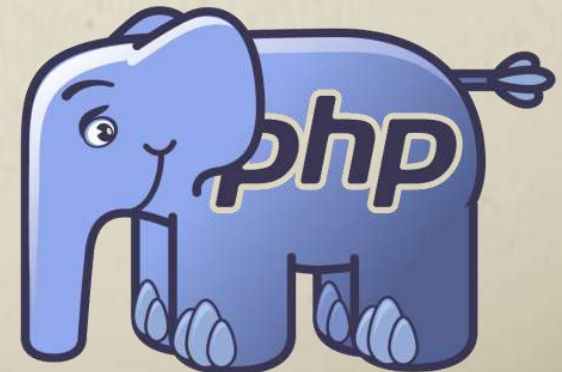


Как это будет выглядеть в PHP?

```
class Table {  
    public $color;  
    public $legs;  
}
```

- Класс начинается с ключевого слова **class**
- Название класса принято писать с большой буквы, в стиле **CamelCase**
- Далее, в фигурных скобках, пишется определение класса – список его свойств и методов (функций).
Мы указали, что объекты класса будут иметь два публичных свойства – **\$color** и **\$legs**
- Слово **public** обозначает, что свойство будет публичным, то есть доступным всем пользователям наших объектов
- Указанный выше код не производит никаких активных действий! Просто определяет класс!

КЛАСС



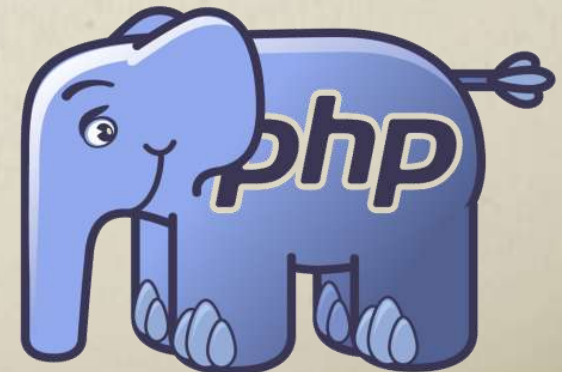
И как же создать объект?

```
class Table {  
    public $color;  
    public $legs;  
}
```

```
$table = new Table;  
$table->color = 'black';  
$table->legs = 4;
```

- Для создания объекта нужного нам класса применяется слово **new**
- Объект – это с одной стороны, обычная переменная. Ее можно передать в функцию или использовать еще как-то.
- С другой стороны, объект имеет свойства. Те, которые описаны в классе!
- Доступ к свойствам мы получаем через конструкцию **->**
- **Обратите внимание – знак «\$» пишется только один!**

ОБЪЕКТЫ



Что же такое «метод»?

Это функция. Определенная в классе. И связанная с объектами этого класса.

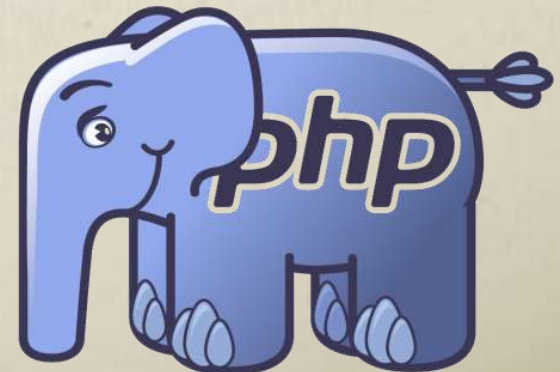
```
class Table {  
    public $color;  
    public $legs;  
    public function show() {  
        echo 'Привет, я стол!';  
        echo '0 ' . $this->legs . 'ногах';  
    }  
}
```

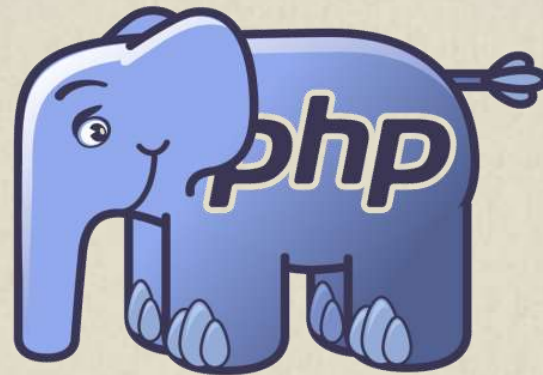
```
$table = new Table;  
$table->color = 'black';  
$table->legs = 4;
```

```
$table->show();
```

- Внутри метода мы можем обращаться к свойствам этого объекта с помощью псевдопеременной **\$this**
- Сам же метод вызывается через **->**
- «Сбор» свойств и методов в классе называется **«ИНКАПСУЛЯЦИЯ»**

МЕТОДЫ ОБЪЕКТОВ





ВИДИМОСТЬ

Мы можем определить, доступно ли свойство или метод «извне» или нет

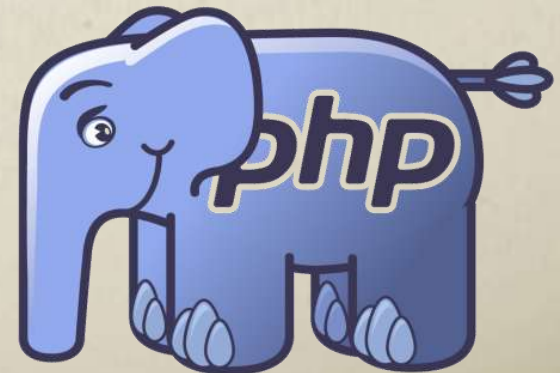
Например:

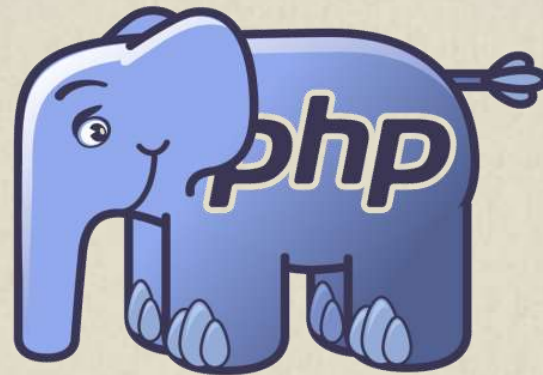
```
class Table {  
    public $color;  
    protected $legs;  
}
```

```
$table = new Table;  
$table->color = 'black';  
$table->legs = 4; // ОШИБКА!
```

- Ключевое слово **protected** вместо **public** приводит к тому, что обращение к свойству (или методу!) напрямую вызывает ошибку
- Однако «внутри», то есть в методах объекта, через **\$this**, такое свойство/метод по-прежнему доступно
- Это тоже часть «ИНКАПСУЛЯЦИИ»
- Совокупность всех публичных свойств и методов называется «интерфейс»

ВИДИМОСТЬ





НАСЛЕДОВАНИЕ

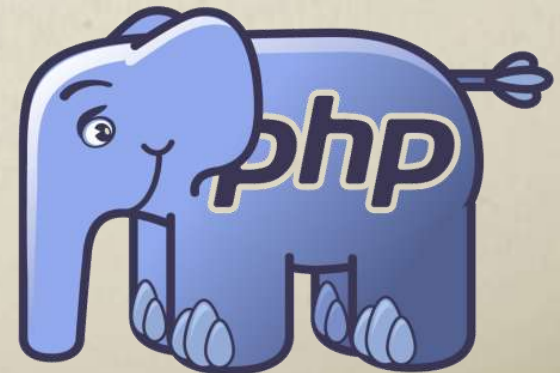
Наследование – это передача «по наследству» свойств и методов

- Все столы – это мебель. И стулья – тоже мебель. А еще шкафы.
- Что есть общего у всей мебели?
 - Материал
 - Размеры
- А что у каждого конкретного класса мебели?
 - У столов – число ножек
 - У стульев – мягкий или жесткий
 - У шкафов – количество дверей

```
class Item {  
    public $color;  
    public $material;  
}
```

```
class Table extends Item {  
    public $legs;  
}
```

НАСЛЕДОВАНИЕ



Наследование – это передача «по наследству» свойств и методов

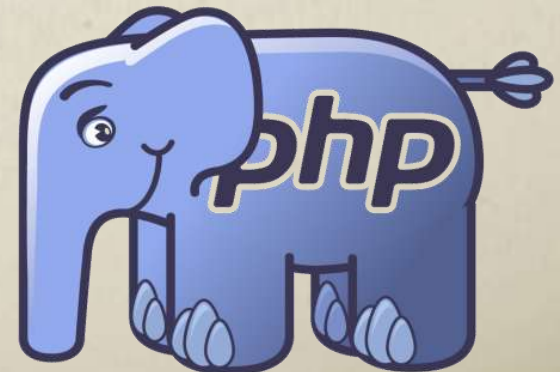
- Наследование реализуется указанием класса-родителя у класса-наследника (потомка) через ключевое слово **extends**.
- Родитель может быть только один!
- Классы-потомки получают в наследство все свойства и методы от классов-родителей
- Это значит, что нам придется писать код только один раз!

```
class Item {  
    public $color;  
    public function showColor() {  
        echo 'Мой цвет: ' . $this->color;  
    }  
}
```

```
class Table extends Item {  
    public $legs;  
}
```

```
$table = new Table;  
$table->color = 'red';  
$table->showColor();
```

НАСЛЕДОВАНИЕ



Наследование – это передача «по наследству» свойств и методов

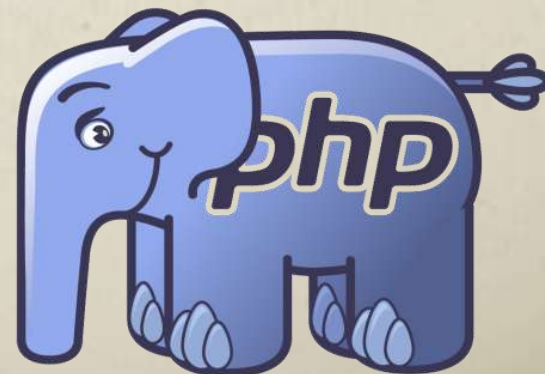
```
class Item {  
    public $color;  
    public function show() {  
        echo 'Мой цвет: ' . $this->color;  
    }  
}
```

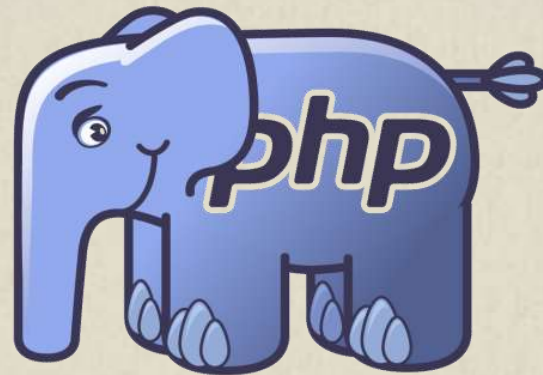
```
class Table extends Item {  
    public function show() {  
        echo 'Я стол. Мой цвет: ' .  
            $this->color;  
    }  
}
```

```
$table = new Table;  
$table->color = 'red';  
$table->show();
```

- Унаследованные методы можно переписать заново в классах-потомках
- Если метод не переопределять, то вполне будет работать родительский.
- К методу родителя можно обратиться с помощью конструкции **parent::method()**

НАСЛЕДОВАНИЕ





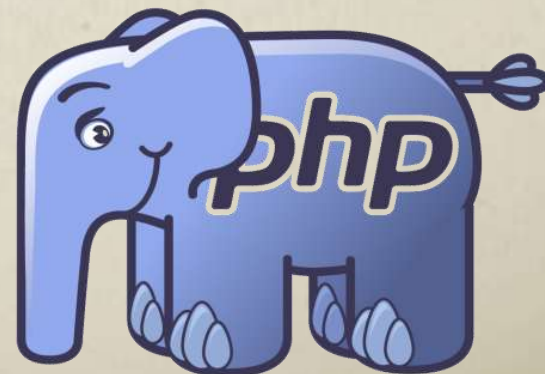
КОНСТРУКТОР

Конструктор – это «магический» метод, который выполняется АВТОМАТИЧЕСКИ при создании объекта класса

```
class Table extends Item {  
  
    public function __construct($color) {  
        $this->color = $color;  
    }  
  
    public function show() {  
        echo 'Я стол. Мой цвет: ' .  
            $this->color;  
    }  
}  
  
$table = new Table('red');  
$table->show();
```

- Метод-конструктор всегда называется **__construct**
- Конструктор может и не иметь аргументов (как и любая другая функция!)
- В конструкторе доступна переменная **\$this**, как и в любом другом методе

КОНСТРУКТОР



ДО ВСТРЕЧИ НА СЛЕДУЮЩЕМ УРОКЕ!

**ВИДЕОЗАПИСЬ, СЛАЙДЫ, ПРЕЗЕНТАЦИЯ
И ДОМАШНЕЕ ЗАДАНИЕ
БУДУТ ВЫЛОЖЕНЫ ДО 10 УТРА СЛЕДУЮЩЕГО ДНЯ**

