## PHP 中创建对象

$runoob = new Site;

$taobao = new Site;

$google = new Site;

### **调用成员方法**

// 调用成员函数，设置标题和URL

$runoob->setTitle( "菜鸟教程" );

## PHP 构造函数

function \_\_construct( $par1, $par2 ) {

$this->url = $par1;

$this->title = $par2;}

## 析构函数

void \_\_destruct ( void )

PHP 使用关键字 **extends** 来继承一个类，PHP 不支持多继承，格式如下：

class Child extends Parent {

// 代码部分}

## 方法重写

如果从父类继承的方法不能满足子类的需求，可以对其进行改写，这个过程叫方法的覆盖（override），也称为方法的重写。

实例中重写了 getUrl 与 getTitle 方法：

function getUrl() {

echo $this->url . PHP\_EOL;

return $this->url;}

function getTitle(){

echo $this->title . PHP\_EOL;

return $this->title;}

## 访问控制

PHP 对属性或方法的访问控制，是通过在前面添加关键字 public（公有），protected（受保护）或 private（私有）来实现的。

* **public（公有）：**公有的类成员可以在任何地方被访问。
* **protected（受保护）：**受保护的类成员则可以被其自身以及其子类和父类访问。
* **private（私有）：**私有的类成员则只能被其定义所在的类访问。

## 接口

使用接口（interface），可以指定某个类必须实现哪些方法，但不需要定义这些方法的具体内容。

// 声明一个'iTemplate'接口

interface iTemplate{

public function setVariable($name, $var);

public function getHtml($template);

}

## 常量

class MyClass{

const constant = '常量值';

function showConstant() {

echo self::constant . PHP\_EOL;

}}

## 抽象类

abstract class AbstractClass{

// 强制要求子类定义这些方法

abstract protected function getValue();

abstract protected function prefixValue($prefix);

// 普通方法（非抽象方法）

public function printOut() {

print $this->getValue() . PHP\_EOL;

}

}

## Static 关键字

声明类属性或方法为 static(静态)，就可以不实例化类而直接访问。

class Foo {

public static $my\_static = 'foo';

public function staticValue() {

return self::$my\_static;

}

}

## Final 关键字

PHP 5 新增了一个 final 关键字。如果父类中的方法被声明为 final，则子类无法覆盖该方法。如果一个类被声明为 final，则不能被继承。

class BaseClass {

public function test() {

echo "BaseClass::test() called" . PHP\_EOL;

}

final public function moreTesting() {

echo "BaseClass::moreTesting() called" . PHP\_EOL;

}

}

## 调用父类构造方法

PHP 不会在子类的构造方法中自动的调用父类的构造方法。要执行父类的构造方法，需要在子类的构造方法中调用 **parent::\_\_construct()** 。

class SubClass extends BaseClass {

function \_\_construct() {

parent::\_\_construct(); // 子类构造方法不能自动调用父类的构造方法

print "SubClass 类中构造方法" . PHP\_EOL;

}

}