模組化程式設計

載入 php 網頁

- require() 建議
- require_once()
- include()
- include_once()

require：載入錯誤停止
include：載入錯誤繼續

函數

定義方式與呼叫

- 函數名稱不分大小寫，但是變數有分大小寫

```
function add($x, $y) {
	return $x + $y;
}

echo add(5, 3);
```

參數預設值

- 若參數有預設值，呼叫若省略該參數時，會以預設值取代

```
function add($x, $y=10) {
        return $x + $y;
}

echo add(5);
```

傳值呼叫 – call-by-value

- 函數中取得的值是傳進去的複製品

```
function swap($a, $b) {
        $tmp = $a;
        $a = $b;
        $b = $tmp;
}

$a = 10;
$b = 20;
swap($a, $b);
echo "a = $a";
echo "b = $b";
```

傳址呼叫 – call-by-address

- 函數中取得的值是本尊

```
function swap(&$a, &$b) {
        $tmp = $a;
        $a = $b;
        $b = $tmp;
}

$a = 10;
$b = 20;
swap($a, $b);
echo "a = $a";
echo "b = $b";
```

變數範圍

- 區域變數優先於全域變數

```
function f() {
        global $n;
        $n = 20;
}

$n = 10;
f();
echo $n;
```

靜態變數

- 雖然是區域變數但只初始化一次，並且記憶體不會被回收

```
<?php
function f() {
    static $n = 0;
    $n += 1;
    echo $n . "\n";
}


for ($i = 0; $i < 10; $i++) {
    f();
}
?>
```

Nullable 型態

```php
function f(int $x) {
    var_dump($x);
}

f(20);      // 正確
f(null);    // 執行錯誤
```

```php
function f(?int $x) {
    var_dump($x);
}
```

Closure

- Closure 又稱為匿名函數，也就是將函數放到變數中

```
$x = function() {
        echo "Hello, World!";
};
$x();
```

模組化程式設計

- 用 require() 或 include() 載入另外一個 php 檔案
  - *require 在遇到錯誤時會停止執行*
  - *include在遇到錯誤時會產生錯誤訊息後繼續執行*
  - *例如載入不存在的檔案或呼叫不存在的函數*

物件導向

基本語法

- class 定義類別與 new 實體化
- 屬性代表物件的特徵，透過變數來實踐
- 方法代表如何操作物件，透過函數來實踐

```php
<?php
class Person {
    public $name = 'David';
    public function info() {
        echo $this->name;
    }
}

$p = new Person();
$p->info();
?>
```

靜態成員

- 也稱類別屬性與類別方法
- 全類別只有一個，不需實體化

```php
<?php
class Person {
    public static $title;
    public static function f() {
        return 'Hello, World!';
    }
}

Person::$title = '20';
echo Person::f();
?>
```

建構子與解構子

- 建構子，可加參數但無傳回值
  - *function __construct() {...}*
  - *function __construct($name) {...}*
- 解構子，沒有參數與傳回值
  - *function __destruct() {...}*

== V.S. ===

- == 只管內容是否一樣
- === 內容要一樣，記憶體位置也要一樣

```php
<?php
class Person {
    public $name;
    function __construct($name) {
        $this->name = $name;
    }
}

$p1 = new Person('Sonia');
$p2 = new Person('Sonia');
$p3 = $p1;

var_dump($p1 == $p2);   // true
var_dump($p1 === $p2);  // false
var_dump($p1 === $p3);  // true
?>
```

繼承

- 父類別有的東西，子類別相當於複製貼上，但不是真的複製貼上

```php
<?php
class Person {
    public $name;
}

class NextPerson extends Person {
    public $engName;
}

$p = new NextPerson();
$p->name = '大衛';
$p->engName = 'David';
?>
```

覆寫

- 子類別重新改寫父類別的函數

```php
<?php
class Person {
    public function display() {
        echo 'Hello, World!';
    }
}

class NextPerson extends Person {
    public function display() {
        echo 'Hi';
    }
}

$p = new NextPerson();
echo $p->display();
?>
```

存取等級

- public、var：公有等級
- private：私有等級
- protected：可繼承