


Corso PHP

appunti

La "storia" di PHP e lo stato di avanzamento del linguaggio

appunti

Costanti e Variabili

Le variabili in PHP sono identificate da un `$` seguito dal nome, mentre le costanti sono valori immutabili definiti con `define()` o usando la parola chiave `const`. Le costanti non cambiano durante l'esecuzione dello script.

```php
define("PI", 3.14); // Costante
$radius = 5;        // Variabile
```

[appunti]
[codice]

Tipologie di Dato

PHP supporta vari tipi di dato: scalari (booleani, interi, numeri in virgola mobile, stringhe), compositi (array, oggetti), e speciali come `null`.

```php
$boolean = true;              // Boolean
$integer = 42;                // Integer
$float = 3.1416;              // Float
$string = "Hello, World!";    // String
$array = [1, 2, 3];           // Array
```

appunti

Valori Booleani

I valori booleani (`true` e `false`) sono usati per rappresentare uno stato binario e vengono spesso utilizzati nelle espressioni condizionali.

```php
$isTrue = true;
if ($isTrue) {
    echo "It's true!";
}
```

appunti
codice

Numeri Interi

PHP supporta i numeri interi, che sono numeri senza parte decimale. Gli interi possono essere rappresentati in base decimale, ottale o esadecimale.

```
$decimal = 42;     // Base decimale
$octal = 052;      // Base ottale
$hex = 0x2A;       // Base esadecimale
```

appunti
codice

Numeri in Virgola Mobile

I numeri in virgola mobile (o numeri decimali) rappresentano numeri con una parte frazionaria e sono utili per calcoli matematici più complessi.

```php
$float = 3.14;
echo $float * 2; // Output: 6.28
```

appunti
codice

Caratteri e Stringhe

Le stringhe sono sequenze di caratteri. PHP fornisce molte funzioni per manipolare stringhe, come `strlen()` e `str_replace()`.

```php
$string = "PHP";
echo strlen($string);  // Output: 3
echo str_replace("P", "J", $string);  // Output: JHJ
```

appunti
codice

Valore Nullo

Il tipo `null` rappresenta una variabile senza valore. Una variabile è `null` se è stata dichiarata ma non ha ancora ricevuto un valore.

```php
$variable = null;
if (is_null($variable)) {
    echo "This is null";
}
```

appunti
codice

Liste e Arrays

Gli array in PHP sono strutture dati che possono contenere valori multipli. Possono essere indicizzati numericamente o associativamente.

```php
$array = ["apple", "banana", "cherry"];
echo $array[1];  // Output: banana
```

appunti
codice

Espressioni

Un'espressione in PHP è tutto ciò che restituisce un valore, come operazioni matematiche o assegnazioni.

```php
$result = 5 + 3;  // L'espressione restituisce 8
```

appunti

Operatori

PHP offre vari tipi di operatori, inclusi aritmetici (`+`, `-`), di confronto (`==`, `!=`), logici (`&&`, `||`), e di assegnazione (`=`).

```php
$a = 5;
$b = 10;
echo $a + $b;  // Operatore aritmetico
echo $a == $b; // Operatore di confronto
```

appunti

codice

Espressioni Condizionali (if, elseif, else)

Le espressioni condizionali controllano il flusso del programma in base a condizioni logiche. `if`, `elseif`, e `else` eseguono blocchi di codice diversi a seconda dei risultati delle condizioni.

```
$score = 85;
if ($score >= 90) {
    echo "A";
} elseif ($score >= 80) {
    echo "B";
} else {
    echo "C";
}
```

appunti

Cicli (foreach, for, do, while)

I cicli permettono di ripetere l'esecuzione di un blocco di codice. `foreach` itera su array, mentre `for`, `do`, e `while` sono cicli generici.

```php
$fruits = ["apple", "banana", "cherry"];
foreach ($fruits as $fruit) {
    echo $fruit . " ";
}
```

appunti

codice

codice

Deviatore (switch)

Il costrutto `switch` permette di eseguire diversi blocchi di codice in base al valore di un'espressione, facilitando il controllo condizionale multiplo.

```
$day = 3;
switch ($day) {
    case 1:
        echo "Monday";
        break;
    case 2:
        echo "Tuesday";
        break;
    case 3:
        echo "Wednesday";
        break;
}
```

Corrispondenza (match)

Il costrutto `match` introdotto in PHP 8 è simile a `switch`, ma più potente e pulito, poiché può restituire direttamente un valore e gestisce il confronto con identità stretta.

```php
$day = 3;
echo match ($day) {
    1 => "Monday",
    2 => "Tuesday",
    3 => "Wednesday",
};
```

appunti

Return, Declare, Goto

`return` restituisce un valore da una funzione, `declare` modifica alcune direttive di esecuzione del codice, e `goto` permette di saltare a un'istruzione specificata (è usato raramente, tu non usarlo MAI).

```php
function sum($a, $b) {
    return $a + $b;
}
```

appunti

codice

Includere altri Scripts

PHP consente di includere file esterni usando `include` o `require`, utili per organizzare il codice in più file.

```php
include 'header.php';
```

appunti
codice

Funzioni

Le funzioni sono blocchi di codice riutilizzabili che accettano parametri e possono restituire valori. Sono dichiarate con la parola chiave `function`.

```php
function greet($name) {
    echo "Hello, " . $name;
}
greet("John");
```

[appunti]
[codice]

Parametri e Argomenti

I parametri sono variabili dichiarate in una funzione, mentre gli argomenti sono i valori passati alla funzione durante la chiamata.

```php
function add($a, $b) {
    return $a + $b;
}
echo add(3, 4);  // Output: 7
```

appunti
codice

Visibilità delle Variabili

La visibilità delle variabili dipende dal contesto in cui sono dichiarate: globale, locale o di una classe (proprietà).

```php
$globalVar = "Global";

function test() {
    global $globalVar;
    echo $globalVar;
}
```

appunti
codice

Tipologie di Funzioni

Le funzioni possono essere definite dall'utente o predefinite. PHP offre numerose funzioni integrate per manipolare stringhe, array, file e altro.

```php
// Funzione definita dall'utente
function square($n) {
    return $n * $n;
}
echo square(4);  // Output: 16
```

appunti
codice

Funzioni degli Array

PHP offre diverse funzioni per la manipolazione degli array, come `array_merge()`, `array_push()`, e `array_filter()`.

```php
$array = [1, 2, 3, 4, 5];
array_push($array, 6);
print_r($array);  // Output: Array ( [0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5 [5] => 6 )
```

appunti

codice

Gestione dei Files

PHP permette di leggere, scrivere, modificare e eliminare file usando funzioni come `fopen()`, `fwrite()`, e `fclose()`.

```php
$file = fopen("test.txt", "r");
$content = fread($file, filesize("test.txt"));
fclose($file);
```

appunti

Date, ore e fusi orari

PHP gestisce date e ore tramite la classe `DateTime` e le funzioni `date()` e `strtotime()`. È possibile configurare il fuso orario usando `date_default_timezone_set()`.

```
date_default_timezone_set('Europe/Rome');
echo date('Y-m-d H:i:s');
```

appunti

Gestione degli Errori

PHP gestisce errori con varie modalità: errori fatali, avvisi e notifiche. Si possono utilizzare blocchi `try-catch` per gestire le eccezioni.

```php
try {
    $result = 5 / 0;
} catch (DivisionByZeroError $e) {
    echo "Error: Division by zero.";
}
```

appunti

Configurazione del file php.ini

`php.ini` è il file di configurazione principale di PHP, che definisce comportamenti come la dimensione massima dei file caricati e il limite di memoria.

Modifica `php.ini` per impostare, ad esempio, il limite di memoria:

```
memory_limit = 128M
```

appunti

Configurazione di Apache

Per eseguire PHP con Apache, bisogna configurare il modulo `mod_php`. Le impostazioni di PHP possono essere modificate attraverso il file `.htaccess`.

Aggiungi nel file `.htaccess`:

```
php_value upload_max_filesize 10M
```

appunti

Sviluppare con Docker

Docker permette di creare ambienti di sviluppo consistenti e isolati.
Si può utilizzare un'immagine Docker con PHP e altri servizi necessari per l'applicazione.

```
docker run -d -p 8080:80 -v $(pwd):/var/www/html php:7.4-apache
```

appunti

Integrazione di Composer

Composer è un gestore di dipendenze per PHP che permette di includere librerie di terze parti nel progetto tramite il file `composer.json`.

```
composer require guzzlehttp/guzzle
```

appunti

Programmazione Orientata agli Oggetti

La OOP in PHP permette di organizzare il codice in classi e oggetti, promuovendo la riusabilità e la manutenibilità del codice.

```php
class Person {
    public $name;
    public function greet() {
        echo "Hello, " . $this->name;
    }
}
```

appunti

Classi ed Oggetti

Una classe è un modello per creare oggetti. Gli oggetti sono istanze delle classi e possono contenere proprietà e metodi.

```php
class Car {
    public $color;
}

$myCar = new Car();
$myCar->color = "Red";
```

appunti

Promozione dei Parametri del Costruttore

Introdotta in PHP 8, la promozione dei parametri permette di dichiarare e inizializzare le proprietà della classe direttamente nel costruttore, riducendo il codice boilerplate.

```
class Point {
    public function __construct(public int $x, public int $y) {}
}
```

appunti

Operatore a Protezione dei Valori Nulli

L'operatore nullsafe (`?->`) in PHP 8 consente di evitare errori quando si accede a proprietà o metodi di un oggetto che può essere nullo.

```
$person = null;
echo $person?->name;
```

appunti

Spazi dei Nomi

Gli spazi dei nomi (`namespace`) permettono di organizzare il codice evitando conflitti tra classi, funzioni o costanti con lo stesso nome.

```php
namespace MyApp\Utils;
class Helper {
    public static function greet() {
        echo "Hello!";
    }
}
```

appunti
codice

Caricamento Automatico

Il caricamento automatico (autoloading) permette di caricare automaticamente le classi quando sono necessarie, senza richiedere manualmente i file.

```php
spl_autoload_register(function ($class_name) {
    include $class_name . '.php';
});
```

appunti

Costanti delle Classi

Le costanti di classe sono valori immutabili definiti all'interno di una classe con la parola chiave `const`, accessibili tramite `self::` o il nome della classe.

```php
class Math {
    const PI = 3.14;
}
echo Math::PI;
```

appunti

Proprietà e Metodi Statici delle Classi

Le proprietà e i metodi statici appartengono alla classe e non alle istanze. Possono essere accessibili senza creare un oggetto.

```php
class Counter {
    public static $count = 0;
    public static function increment() {
        self::$count++;
    }
}
Counter::increment();
echo Counter::$count;
```

appunti

Quattro Principi Fondamentali dell'OOP

I quattro principi dell’OOP sono: astrazione, incapsulamento, ereditarietà e polimorfismo, che permettono una struttura modulare e flessibile.

```php
// Esempio di Incapsulamento
class BankAccount {
    private $balance = 0;
    public function deposit($amount) {
        $this->balance += $amount;
    }
    public function getBalance() {
        return $this->balance;
    }
}
```

Classi Astratte e Metodi Astratti

Le classi astratte non possono essere istanziate e possono contenere metodi astratti, che devono essere implementati nelle sottoclassi.

```
abstract class Animal {
    abstract public function sound();
}

class Dog extends Animal {
    public function sound() {
        echo "Woof!";
    }
}
```

appunti

Interfacce e Polimorfismo

Le interfacce definiscono un contratto per le classi che le implementano. Il polimorfismo permette di trattare oggetti di diverse classi in modo uniforme.

```
interface Vehicle {
    public function drive();
}

class Car implements Vehicle {
    public function drive() {
        echo "Driving a car.";
    }
}
```

Gerarchie ed Ereditarietà

L'ereditarietà permette a una classe di ereditare proprietà e metodi da un'altra classe, creando una gerarchia di classi.

```php
class Animal {
    public $name;
    public function speak() {
        echo "Animal sound";
    }
}

class Dog extends Animal {
    public function speak() {
        echo "Woof!";
    }
}
```

Ereditarietà Multipla tramite Traits

I `traits` permettono di riutilizzare il codice in più classi, simulando l'ereditarietà multipla, che non è supportata direttamente in PHP.

```php
trait Logger {
    public function log($message) {
        echo $message;
    }
}

class Application {
    use Logger;
}

$app = new Application();
$app->log("Starting application");
```

Classi Anonime

Le classi anonime sono classi senza nome che possono essere create dinamicamente e utilizzate per scopi specifici o temporanei.

```php
$greeting = new class {
    public function sayHello() {
        echo "Hello, World!";
    }
};

$greeting->sayHello();
```

appunti

Metodi Magici

PHP include metodi speciali detti "magici", come `__construct()`, `__get()`, e `__set()`, che svolgono operazioni speciali o automatizzano comportamenti degli oggetti.

```php
class Person {
    private $name;
    public function __get($property) {
        return "Getting " . $property;
    }
}
```

appunti