PHP 8 関数の解説ハイライトと教え方のコツ

2023年4月17日

一般社団法人BOSS-CON JAPAN
PHP技術者認定機構
エバンジェリスト 三雲 勇二

アジェンダ

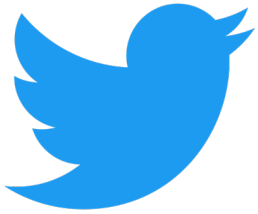
1. 自己紹介

2. PHP 8 関数の解説ハイライトと教え方のコツ

    1. 新しい関数
    2. 型プログラミング
    3. 今までの関数を PHP 8 を見据えて使うポイント

3. 継続して情報を集めるために

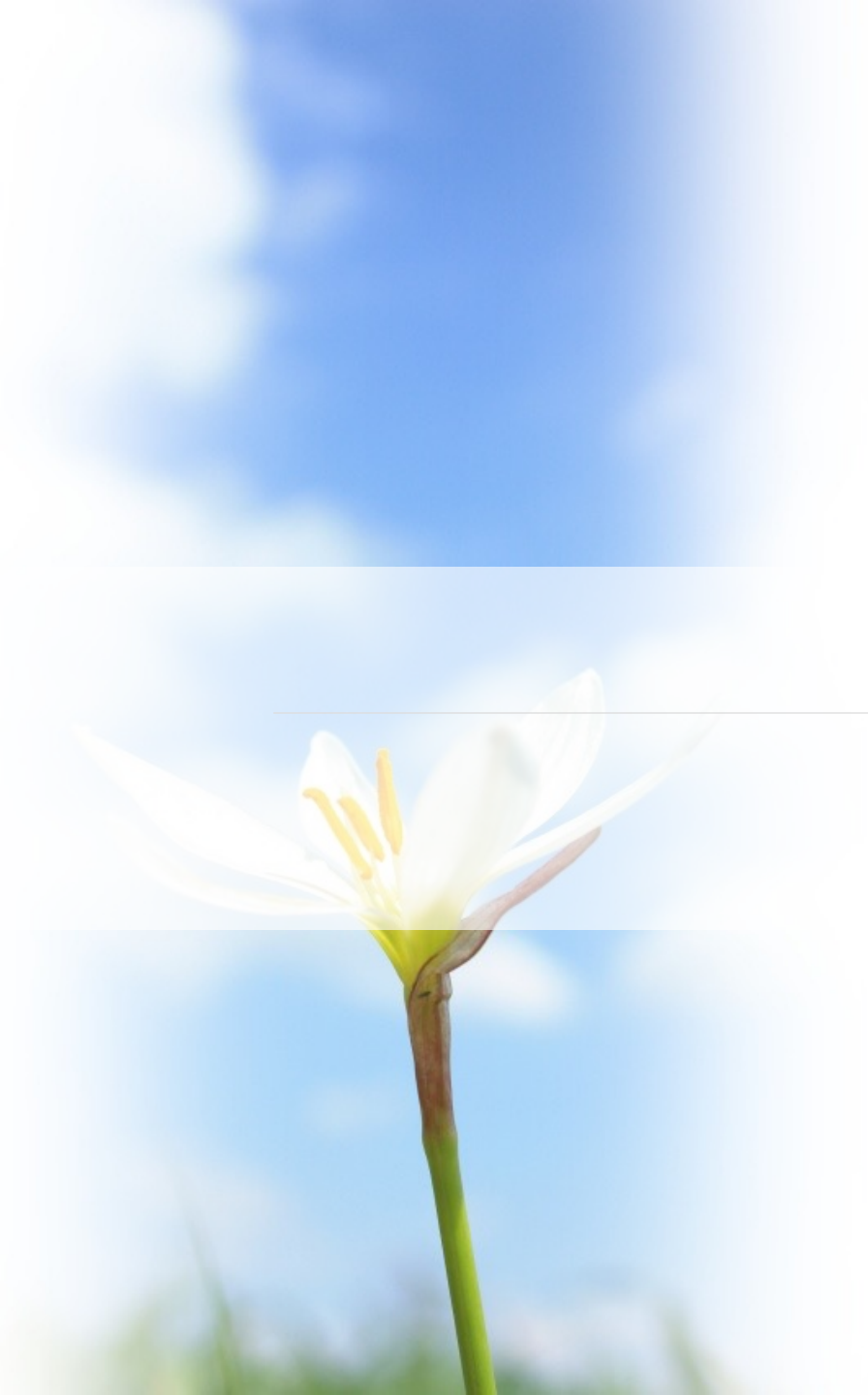
アジェンダ



Twitter でツイートされるときは、
ハッシュタグ

#PHP試験

をつけてください！
後ほど私がツイート見ます！



1. 自己紹介

自己紹介

- 三雲 勇二 （みくも ゆうじ）

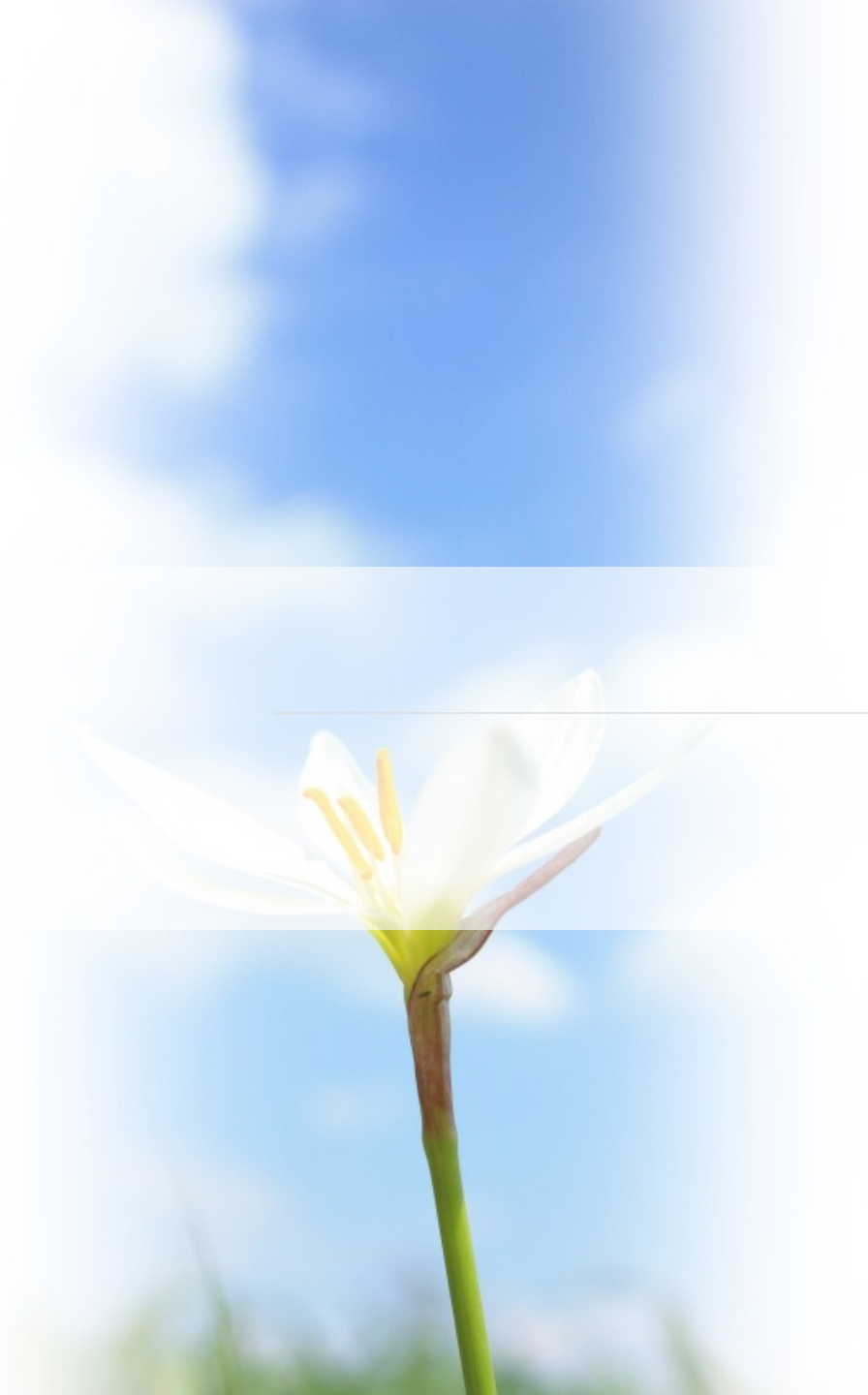
  三雲 勇二 / yuji.mikumo

  mikumo [@maikeru]

- 所属：
  - プライム・ストラテジー株式会社
    WordPress などCMS高速化技術
    KUSANAGI の会社です😊
  - PHP技術者認定協会 エバンジェリスト

- 出身地：宮崎県延岡市
- PHPの7試験に合格してます

超高速CMS実行環境
KUSANAGI
Powered by Prime Strategy



2. PHP 8 関数の解説
ハイライトと教え方のコツ

PHP 8 関数の解説 ハイライトと教え方のコツ

1. 新しい関数
2. 型プログラミング
3. 今までの関数を PHP 8 を見据えて使うポイント

PHP 8 関数の解説 ハイライトと教え方のコツ

1. 新しい関数
2. 型プログラミング
3. 今までの関数を PHP 8 を見据えて使うポイント

PHP 8 の環境ではない方も、PHP 8 に向けたコーディングのポイントを押さえていただければ、と考えております。

新しい関数

PHP 8 で使用できる新しい関数

新しい関数の前に…

『名前付き引数』が導入されました。

```
array_sample ( array: $value );
```

勘違いされやすいポイントですが、

こちらは、既存の関数を呼び出す際に必ずしも使えるものではないので気をつけてください。

https://www.php.net/manual/ja/functions.arguments.php#functions.named-arguments

新しい関数

- str_contains()
- str_starts_with()
- str_ends_with()
- array_is_list() [PHP 8.1]

新しい関数

- str_contains()
- str_starts_with()
- str_ends_with()
- array_is_list() [PHP 8.1]

追加された関数には他にも細かいものもあるのですが、
使いそうなものをピックアップしました。

str_contains()

needle が haystack に含まれるかを調べます。 大文字小文字は区別されます。

```
str_contains(string $haystack, string $needle): bool
```

str_contains()

needle が haystack に含まれるかを調べます。 大文字小文字は区別されます。

PHP でよく見る引数の名前です。

意味を覚えてしまいましょう。

- haystack → 干し草の山
- needle → 針

str_starts_with()

haystack が needle で始まるかを調べます。 大文字小文字は区別されます。

```
str_starts_with(string $haystack, string $needle): bool
```

str_ends_with()

haystack が needle で終わるかを調べます。 大文字小文字は区別されます。

```
str_ends_with(string $haystack, string $needle): bool
```

array_is_list() [PHP 8.1]

指定された array がリストかどうかを判定します。

配列のキーが連続した数値

(0 から count($array)-1) である場合、

リストと判定されます。

```
array_is_list(array $array): bool
```

array_is_list() [PHP 8.1]

指定された array がリストかどうかを判定します。

配列のキーが連続した数値

(0 から count($array)-1) である場合、

リストと判定されます。

[0, 1, 2, 3, 4, 5]

このような配列がリストと判定されます。

array_is_list() [PHP 8.1]

指定された array がリストかどうかを判定します。

配列のキーが連続した数値

(0 から count($array)-1) である場合、

リストと判定されます。

[0, 1, '2', 4, 5]

つまりこのような表記はリストではない、と判定されます。

新しい関数とあわせて、便利な構文も

- アトリビュート
- union 型
- match 式
- nullsafe 演算子

型プログラミング

文字の「'1'」と、数字の「1」、

同じ扱いではダメですか？

型プログラミング

内部では別物です。

16進数バイトという表記で表すと、

- 数値 1 →「1」
- 文字 1 →「31」(ASCII)

型プログラミング

内部では別物です。

16進数バイトという表記で表すと、

- 数値 1 →「1」
- 文字 1 →「31」(ASCII)

「1」と「31」という内部のデータが違うものを、PHPが気を利かせて同じ「1」として取り扱っていてくれていたのです。

型プログラミング

```
// 型の自動変更を行ったうえで比較
1 == '1'; // true

// そのまま比較
1 === '1'; // false
```

型プログラミング

```
// 型をそろえる事が重要
1 == '1'; // true

// そのまま比較
1 === '1'; // false
1 === (int)'1'; // true
```

下位互換性のない変更点 – 文字列と数値の比較

(厳密でないやり方で)数値と非数値文字列を比較する場合、 数値を文字列にキャストし、文字列と比較するようになりました。 数値と数値形式の文字列の比較は、以前と同じ振る舞いをします。

緩やかな比較 (==)	変更前 (PHP 7.4 以前)	変更後 (PHP 8.0)
`0 == "0"`	true	true
`0 == "0.0"`	true	true
`0 == "foo"`	true	false
`0 == ""`	true	false
`0 == "not-a-number"`	true	false
`42 == " 42"`	true	true
`42 == "42foo"`	true	false

下位互換性のない変更点 – 文字列と数値の比較

(厳密でないやり方で)数値と非数値文字列を比較する場合、 数値を文字列にキャストし、文字列と比較するようになりました。 数値と数値形式の文字列の比較は、以前と同じ振る舞いをします。

厳密な比較 (===)	変更前 (PHP 7.4 以前)	変更後 (PHP 8.0)
`0 === "0"`	false	false
`0 === "0.0"`	false	false
`0 === "foo"`	false	false
`0 === ""`	false	false
`0 === "not-a-number"`	false	false
`42 === " 42"`	false	false
`42 === "42foo"`	false	false

型プログラミング

下位互換性のない変更点 – 文字列と数値の比較

対策：できる限り、キャスト演算子で比較する型を合わせ、
緩やかな比較（==）ではなく、 厳密な比較（===）を使いましょう。

厳密な比較 (===)	変更前 (PHP 7.4 以前)	変更後 (PHP 8.0)
`0 === "0"`	false	false
`0 === "0.0"`	false	false
`0 === "foo"`	false	false
`0 === ""`	false	false
`0 === "not-a-number"`	false	false
`42 === " 42"`	false	false
`42 === "42foo"`	false	false

型プログラミング

厳密な比較のメリット [===、!==]

- 高速
- （文字列型以外の場合）インジェクション系のセキュリティ対策

  SQLインジェクション例、`3 === '3 OR 1=1'` のような攻撃を防ぐ。
- Mixed な戻り値を持つ関数の動作を正しく処理できる

  例: strpos の戻り値 `0`(先頭) と `false`(文字列なし) が同一視されない。

緩やかな比較メリット [==、!=、<>]

- 型を意識する必要がない

  初心者など型の概念がわからない場合、とりあえず動くコードが書けます。
- 同一クラスの比較

  同一クラスの別インスタンスを同等と扱うことができます。
- キャストする手間がない

型プログラミング

型を意識したプログラミングするために、

```
declare(strict_types=1);
```

PHPの暗黙の型変換を禁止して、型指定に厳密にチェックする設定

今までの関数を PHP 8 を見据えて使うポイント

今までの関数を PHP 8 を見据えて使うポイント

型プログラミングを意識しましょう。

比較演算子以外にも影響があるものがありますので、

それらの考慮も必要です。

- switch 文 → match() 式
- in_array() 関数 → 第3引数を true に指定

今までの関数を PHP 8 を見据えて使うポイント

インクリメントの仕様

PHP は、算術演算子で文字変数を扱った場合に C ではなく Perl の慣習に 従います。例えば、PHP や Perl では $a = 'Z'; $a++; の結果として $a が 'AA' になりますが C では a = 'Z'; a++; の結果として a は '[' になります ('Z' の ASCII 値は 90、そして '[' の ASCII 値は 91 です)。 文字変数はインクリメントされることは可能ですがデクリメントは不可能であるということ、 またプレーンな ASCII 文字と数字 (a-z、A-Z、そして 0-9) のみがサポートされるということに注意しましょう。 その他の文字変数のインクリメント/デクリメントは何の効果もなく、元の文字列は変更されません。

今までの関数を PHP 8 を見据えて使うポイント

インクリメントの仕様

```
$a = 'Z';
$a++;
// $a が 'AA' になります。
```

今までの関数を PHP 8 を見据えて使うポイント

in_array()

PHP 8.0.0 より前のバージョンでは、 strict フラグが指定されていない場合に、配列の値が 0 の場合でも文字列にマッチしてしまっていました。逆も同じです。 これにより、望ましくない結果が生じる可能性があります。 似たようなエッジケースは他の型でも存在します。 値の型が完全にわからない場合には、 予期せぬ振る舞いを避けるために常に strict フラグを使うようにして下さい。

今までの関数を PHP 8 を見据えて使うポイント

in_array()

```
$nums = [0, 1, 2, 3, 4, 5];
in_array(3, $nums));          // true
in_array('Name', $nums));     // true
```

今までの関数を PHP 8 を見据えて使うポイント

in_array() 第3引数 true を意識する

```
$nums = [0, 1, 2, 3, 4, 5];
in_array(3, $nums, true);          // true
in_array('Name', $nums, true);     // false
```

今までの関数を PHP 8 を見据えて使うポイント

get_debug_type()

PHP の変数 value の、解決済みの名前を返します。 この関数は、オブジェクトをクラス名に、 リソースをリソース型の名前に、 そしてスカラー型の値を型宣言で使われている通常の名前に解決します。

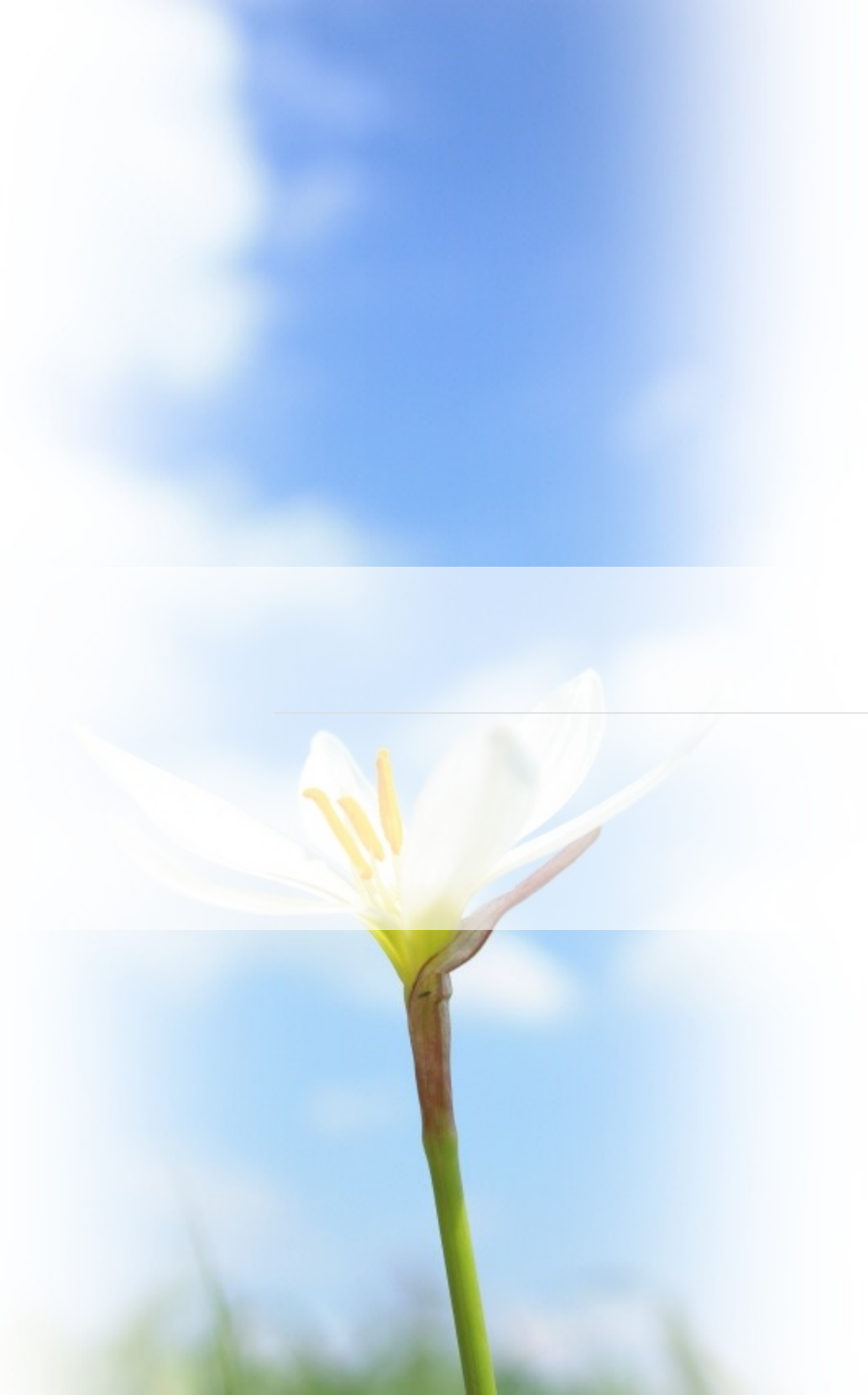
gettype() は、歴史的な理由で決まった型の名前を返します。 この関数は、それよりも実際に使われている型により近い名前を返すという点が異なります。

今までの関数を PHP 8 を見据えて使うポイント

get_debug_type()

```
get_debug_type(10.2);  // float

// 歴史的な理由により float の場合は、"float" ではなく、 "double" が返されます)
gettype(10.2);             // double
```



3. 継続して情報を集めるために

継続して情報を集めるために

公式ウェブサイトコラム / PHP Open Textbook で公開していきます。

一般社団法人BOSS-CON JAPAN
PHP 技術者認定機構

PHP Open Textbook

- 各種カンファレンスや勉強会に参加
  - PHP カンファレンス、PHPerKaigi、PHP勉強会@東京 など

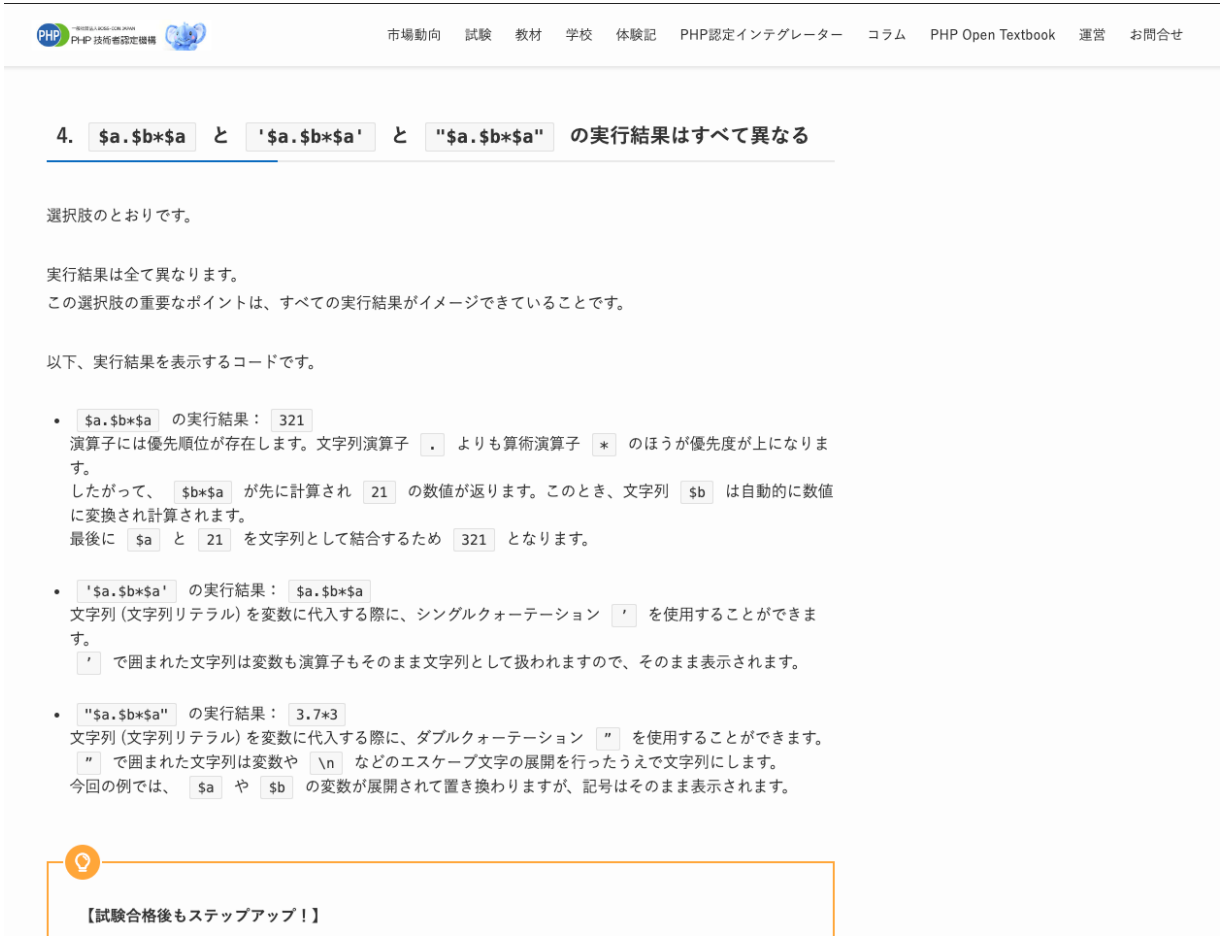
Web コラムのご紹介

PHP 技術者認定機構 公式サイトで

PHP 7 初級 / PHP 8 初級 受験者向けコラムを連載しています。

Web コラムのご紹介

掲載している模擬問題の解説例です。



PHP 技術者認定機構

市場動向　試験　教材　学校　体験記　PHP認定インテグレーター　コラム　PHP Open Textbook　運営　お問合せ

4. `$a.$b*$a` と `'$a.$b*$a'` と `"$a.$b*$a"` の実行結果はすべて異なる

選択肢のとおりです。

実行結果は全て異なります。
この選択肢の重要なポイントは、すべての実行結果がイメージできていることです。

以下、実行結果を表示するコードです。

- `$a.$b*$a` の実行結果： `321`
  演算子には優先順位が存在します。文字列演算子 `.` よりも算術演算子 `*` のほうが優先度が上になります。
  したがって、 `$b*$a` が先に計算され `21` の数値が返ります。このとき、文字列 `$b` は自動的に数値に変換され計算されます。
  最後に `$a` と `21` を文字列として結合するため `321` となります。

- `'$a.$b*$a'` の実行結果： `$a.$b*$a`
  文字列 (文字列リテラル) を変数に代入する際に、シングルクォーテーション `'` を使用することができます。
  `'` で囲まれた文字列は変数も演算子もそのまま文字列として扱われますので、そのまま表示されます。

- `"$a.$b*$a"` の実行結果： `3.7*3`
  文字列 (文字列リテラル) を変数に代入する際に、ダブルクォーテーション `"` を使用することができます。
  `"` で囲まれた文字列は変数や `\n` などのエスケープ文字の展開を行ったうえで文字列にします。
  今回の例では、 `$a` や `$b` の変数が展開されて置き換わりますが、記号はそのまま表示されます。

【試験合格後もステップアップ！】

Web コラムのご紹介

コラムの中で、PHP7初級 試験合格者向けの情報として PHP 8 に関する情報も多数掲載してあります。こちらもご確認ください。

【試験合格後もステップアップ！】

`php.ini` の `error_reporting` の初期値は `E_ALL & ~E_NOTICE & ~E_STRICT & ~E_DEPRECATED` です。
これは、Notice と Strict、Deprecated を表示しない設定になっているという意味です。

PHP 8.0 以降では `php.ini` の `error_reporting` の初期値が `0` となり、すべてのエラーが表示されるようになりました。

PHP 8.0 より前のバージョンから PHP 8.0 にアップグレードする前に、こちらのエラーを表示されるように一度設定を変更の上、潰せるエラーは潰しておくことをお勧めします。

PHP Open Textbook の一部ご紹介

無料で使用できるPHP教材「PHP Open Textbook」の一部として、PHP試験の自主学習ができるよう準備中です。

PHP オンライン活動のご紹介

- PHP勉強会@東京：次回 5月17日
- PHPerKaigi 2023: 3月23日～25日 開催されました。
- PHP カンファレンス：PHP Confarence 2023 秋。

他、多数のカンファレンスやミートアップなど
PHP関連イベントが各地で開催されています。

PHP技術者認定機構