



PHP

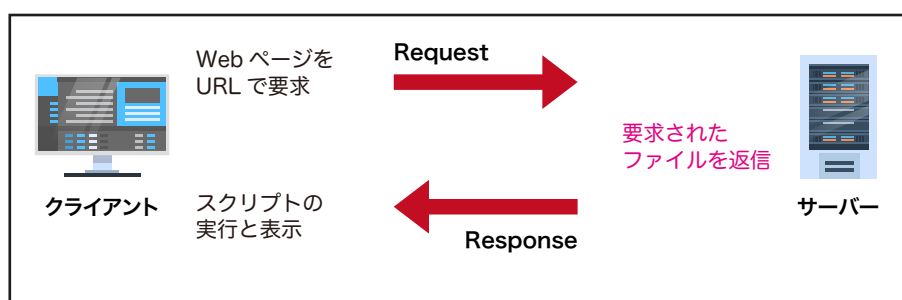
PHP の概要

クライアントサイドスクリプトとサーバーサイドスクリプト

クライアントが Web ページをサーバーから受け取る流れは、リクエスト & レスポンス方式で行います。クライアントから 1 回のリクエストに対して、サーバーは 1 回のレスポンスで答え、サーバーとの通信が終了するというものです。

クライアントサイドスクリプト

クライアントのコンピュータにある ブラウザ上で処理を行うスクリプトのことで、代表的なものに JavaScript があります。サーバーは、クライアントのリクエストに対して HTML やそのファイルに読み込まれている関連ファイルの送信だけをおこない、ダウンロードされたスクリプトの処理はブラウザが行います。



メリット

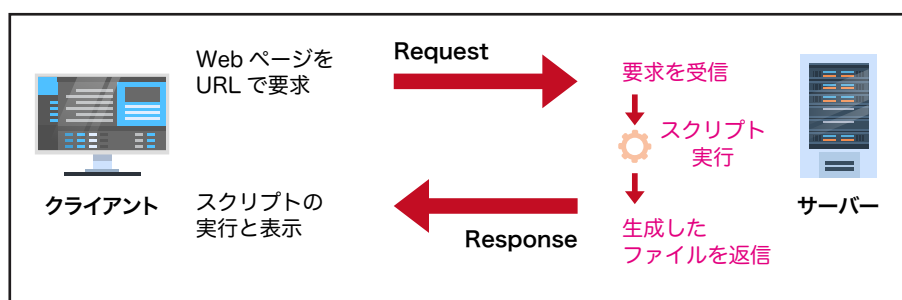
- ・サーバーへの負担が少ない
- ・無駄な通信によるユーザー体感の低下を軽減

デメリット

- ・ユーザーにコードの内容が見られてしまう
- ・データベース (DB) へのアクセスが難しい
- ・ブラウザの環境に依存する

サーバーサイドスクリプト

サーバーが処理を行うスクリプトのことで、PHP、Ruby、JSP、ASP.NET、Per などがあります。サーバーは、クライアントからのリクエストを受け取り、スクリプトを実行した結果のファイルをクライアントへ返信します。



メリット

- ・ユーザーにコードの内容が見られない
- ・データベース (DB) へのアクセスが容易
- ・ブラウザの環境に依存しない

デメリット

- ・スクリプトが処理できる環境が必要
- ・過度の通信は、ユーザーの体感を低下させる

PHP とは


PHP は Hypertext Preprocessor の再帰的な略語で、オープンソースの汎用スクリプト言語で、サーバーサイドで動的な Web ページを作成するための多くの機能を備えていることが特徴になります。

PHP は、サーバーサイドスクリプト言語とも呼ばれ、クライアントサイドの JavaScript と異なる点として、プログラムコードがサーバーで実行され、その結果がクライアントに送信されます。そのためクライアントは、PHP のコードがどんなものなのかを知ることができません。

PHP の動作環境

PHP のようなサーバーサイドスクリプトを動作させるためには、動作環境を構築が必要となります。


<div><ul style="list-style-type: none">・ Linux・ Windows・ Mac OS X<div>など</div></div> <div>OS</div>	<div><ul style="list-style-type: none">・ Apache・ Nginx・ IIS<div>など</div></div> <div>Web サーバー</div>	<div><ul style="list-style-type: none">・ 実行エンジン・ 設定ファイル・ 各種ライブラリ<div>など</div></div> <div>PHP コア</div>
--	--	---



Linux
Apache

✗

OS、Web サーバーのみサーバーに設置されているだけでは、PHP を利用した動的な Web サイトを構築することはできません。



Linux
Apache
PHP

◯

OS、Web サーバー、PHP がサーバーに用意されていて初めて、PHP を利用した動的な Web サイトを構築することができます。

PHP のプログラムを開発するには、PHP が用意された Web サーバー上にファイルを配置する必要があるので、作成したファイルを FTP クライアントソフトなどを利用して、PHP が稼働するサーバーへファイルをアップロードする必要があります。

PHP の動作確認

PHP ファイルの拡張子

Web サーバーで PHP のスクリプトを実行させるには、ファイルの拡張子を「.php」にする必要があります。Web サーバーの設定で、「.html」ファイルでも PHP のスクリプトを実行させる設定を行わない限り、拡張子が「.php」のファイル以外に記述した PHP スクリプト命令は、実行されません。

PHP の設定ファイル

PHP の設定は、PHP コアにある「php.ini」ファイルに記述されています。PHP コアファイル群は、一般ユーザーでは、閲覧、編集することができない領域に配置されているため、「php.ini」ファイルはサーバー管理者のみ閲覧、編集することができます。

PHP の設定確認

PHP は、ブラウザから一般ユーザーが PHP の設定がどうなっているかを確認できる命令を用意してくれています。PHP の動作と設定をチェックするために、PHP の設定情報を出力する命令で確認してみましょう。

サンプル

```
<?php
    phpinfo(); // PHPの設定情報を出力
?>
```

info.php

PHPマニュアル:<https://www.php.net/manual/ja/function.phpinfo.php>

PHP の基本書式

PHP タグ

PHP は、ファイルの中から PHP の開始タグと終了タグ（`<?php` と `?>`）を探して、PHP タグの中に記述されたコードを実行します。逆に PHP タグで囲われていない部分を無視するので、PHP を HTML に組み込んで、テンプレートファイルを作成することなども可能になります。

```
<?php print "Hello PHP Wold"; ?>
```

↑
開始タグ

↑
PHP コード

↑
終了タグ

```
<h1> はじめての PHP プログラミング </h1>
```

```
<p><?php print "Hello PHP Wold" ?></p>
```

HTML と PHP を組み合わせた記述

PHP は、ステートメントの区切りには、「;」（セミコロン）が必要となりますが、終了タグには、自動的にセミコロンが含まれていると認識されるため、最終行はセミコロンを省略することができます。

PHPマニュアル：<https://www.php.net/manual/ja/language.basic-syntax.phptags.php>

PHPマニュアル：<https://www.php.net/manual/ja/language.basic-syntax.phpmode.php>

PHPマニュアル：<https://www.php.net/manual/ja/language.basic-syntax.instruction-separation.php>

コメント

PHP は、単一行と複数行のコメントを使うことができます。

単一行コメントは、「//」（スラッシュ x 2）または、「#」（ハッシュマーク）で改行か PHP のコード終了が来るまでをコメントアウトにします。

複数行コメントは、「/*」（スラッシュ アスタリスク）から次に現れる「*/」（アスタリスク スラッシュ）までをコメントアウトにします。

PHPマニュアル：<https://www.php.net/manual/ja/language.basic-syntax.comments.php>

文字列

PHP は文字列を 4 つの異なる方法で指定することができます。

・引用符 ・二重引用符 ・ヒアドキュメント構文 ・Nowdoc 構文
ヒアドキュメント構文と Nowdoc 構文は今回は割愛させていただきます。

引用符

「'」（シングルクォーテーション）で括ることで、文字列を指定することができます。

「'」で括られた文字列はエスケープシーケンスされず、書いたそのまま出力されます。

二重引用符

「"」（ダブルクォーテーション）で括ることで、文字列を指定することができます。

「"」で括られた文字列では、エスケープシーケンスや変数展開などが可能になります。

PHPマニュアル:<https://www.php.net/manual/ja/language.types.string.php>

文字列の出力

print

文字列を出力します。

print \$expression

expression を出力します。print は関数ではなく、言語構造のため「(」（少括弧、丸括弧）で括る必要はありません。expression が文字列でない場合は、強制的に文字列へ変換されます。

文字列の出力には、echo という命令もありますが、echo との違いは、print は単一の引数のみを受け付けます。

```
<?php
print "printに括弧は不要です";
print "
複数行に分けて書くこともできますが、
出力結果に改行やインデントは再現されません
";
?>
```

PHPマニュアル:<https://www.php.net/manual/ja/function.print.php>

echo

1 つ以上の文字列を出力します。

echo ...expressions

1 つ以上の文字列を出力します。print との違いは、echo は出力したい文字列を「,」（カンマ）で区切ることで、1 つ以上の文字列を指定することができます。expressions が文字列でない場合は、強制的に文字列へ変換されます。

```
<?php
echo "echoに括弧は不要です";
echo "echoは", " 1 つ以上の", "文字列を", "指定できます";
?>
```

PHPマニュアル: <https://www.php.net/manual/ja/function.echo.php>

ショートタグ

echo には短縮構文も用意されており、短縮構文を利用することで、HTML などの異なるドキュメント内で文字列を出力する際に、コードの可読性の低下を軽減させることができます。

この短縮構文は、ショートオープンタグの設定がオフに設定されていても使用することが可能です。

```
<p> はじめての <?php echo "PHP" ?> プログラミング </p>
<p> はじめての <?= "PHP" ?> プログラミング </p>
```

変数

PHP の変数は、緩い動的型付けが特徴となります。変数を利用する際に、変数の宣言を行わなくても、変数に保存したデータによってデータ型が自動で決められます。

逆にプログラムを実行しないとバグの存在に気づけないデメリットもあるので、注意が必要です。

変数のルール

- ・「\$」（ドルマーク）の後に変数名が続く形式である必要がある
- ・変数名には、任意の数の半角英数と「_」（アンダーバー）の組み合わせで付ける
- ・数字から始まる変数名は付けられない

```
<?php

$siteName = "Web演習 1 ";
$pageTitle = "変数について";

?>
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title><?= $pageTitle ?> | <?= $siteName ?></title>
</head>
```

文字列内で変数の展開

「"」（ダブルクォーテーション）やヒアドキュメント構文内は、変数の展開が可能になります。

連想配列などの「"」の組み合わせに考慮するのを回避するために、文字列内で展開する変数名を「{」（中括弧）で括ってあげましょう。

```
<?php

$siteName = "Web演習 1 ";
$pageTitle = "変数について";

?>
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title><?= "{$pageTitle}" | "{$siteName}" ?></title>
</head>
```


定数

PHP では、define 関数か、const キーワードを使うことで、定数を宣言することができます。

define 関数を使えば、任意の式を使って定数を定義することができますが、const キーワードを使う場合には、いくつかの制約があります。一度定義された定数は、変更や未定義にすることはできません。

define

名前を指定して定数を定義します。

define(\$name, \$value)

名前を指定して定数を定義します。name に定数の名前を指定します。value に定数の値を指定します。

value に指定できる値は、スカラー値（int、float、string、bool あるいは null）以外に配列も指定することができます。

定数名のルール

- ・ 変数名には、任意の数の半角英数と「_」（アンダーバー）の組み合わせで付ける
- ・ 数字から始まる定数名は付けられない
- ・ 慣習的に、定数名は大文字で表記する
- ・ 「_」から始まる定数名は、将来予約語に追加されるかもしれないので、避けるべき

```
<?php

define( "SITE_NAME", "Web演習 1 " );
define( "CHARSET", "UTF-8" );
define( "WEB_ROOT", "http://example.ecc.ac.jp/web1/" );

?>
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="<?= CHARSET ?>">
  <title><?= SITE_NAME ?></title>
</head>
<body>
  <h1><a href="<?= WEB_ROOT ?>">Web演習 1 </a></h1>
```

演算子

PHP で使用する演算子の一部を紹介します。

代入演算子

「 = 」

結合演算子

「 . 」

算術演算子

「 + 」 加算

「 - 」 減算

「 * 」 乗算

「 / 」 除算

「 % 」 余り

加算子 / 減算子

「 ++ 」 1 を加える (インクリメント)

「 -- 」 1 を引く (デクリメント)

比較演算子

「 == 」 値が等しい

「 === 」 値が等しく、かつ型も等しい

「 != 」 値が等しくない

「 !== 」 値が等しくない、かつ型も等しくない

「 > 」 大きい

「 < 」 小さい

「 >= 」 大きい、または等しい (以上)

「 <= 」 小さい、または等しい (以下)

論理演算子

「 and 」 論理積 (かつ)

「 && 」 論理積

「 or 」 論理和 (または)

「 || 」 論理和

「 ! 」 否定

条件分岐

if

条件式は論理値（boolean）で評価され、条件式が true と評価された場合は、処理命令を実行します。false と評価された場合は、実行しません。

```
if ( 条件式 ) {  
    処理命令  
}
```

```
if ( 条件式 ) :  
    処理命令  
endif;
```

{ } でのグループ化を使わない記述方法

論理値への自動変換

PHP で false とみなされる値（boolean へのキャスト変換、自動変換）

- ・ 0 や 0.0
- ・ "0" や " "（空文字）
- ・ []（要素を持たない空配列）
- ・ NULL（初期化されていない変数も含みます）
- ・ 属性がない空要素から作成された SimpleXML オブジェクト

PHPマニュアル:<https://www.php.net/manual/ja/control-structures.if.php>

else

if の条件式が false と評価された時に、ある処理命令を実行するように if 文を拡張します。

```
if ( 条件式 ) {  
    処理命令  
}  
else {  
    処理命令  
}
```

```
if ( 条件式 ) :  
    処理命令  
else :  
    処理命令  
endif;
```

{ } でのグループ化を使わない記述方法

PHPマニュアル:<https://www.php.net/manual/ja/control-structures.else.php>

elseif / else if

if と else を組み合わせたもので、if の条件式が false と評価された時に、else の前に elseif の条件式が評価されます。

```
if ( 条件式 ) {  
    処理命令  
}  
elseif( 条件式 ) {  
    処理命令  
}  
else {  
    処理命令  
}
```

```
if ( 条件式 ) :  
    処理命令  
elseif( 条件式 ) :  
else :  
    処理命令  
endif;
```

{ } でのグループ化を使わない記述方法

PHPマニュアル:<https://www.php.net/manual/ja/control-structures.elseif.php>

switch

同じ変数を異なる値と比較して、値に応じた処理命令を実行したい際に switch を利用します。

swtich は式の値と同じ値を持つ case を見つけられた際に、case 内に定義されている処理命令を実行します。

```
switch( 評価する値 ) {  
    case ● : 処理命令 ; break;  
    case ▲ : 処理命令 ; break;  
    default : 処理命令 ;  
}
```

```
switch( 評価する値 ) :  
    case ● : 処理命令 ; break;  
    case ▲ : 処理命令 ; break;  
    default : 処理命令 ;  
endswitch;
```

{ } でのグループ化を使わない記述方法

break で switch の分岐処理を終了させなかった場合、該当した case 以下の全ての case 内の処理命令が実行されてしまうので、必ず case 内の処理命令の終わりには、break を使って switch を終わらせましょう。

PHPマニュアル:<https://www.php.net/manual/ja/control-structures.switch.php>

配列

PHP の配列は、順番付けされたマップになり、値をキーに関連付けします。配列に保存する値として、他の配列も保存することができるため、簡単にツリー構造を表現することが可能になります。

array

配列を生成します。

```
$array = array( $key => $value, $key => $value... )
```

```
$array = [ $index => $value, $index => $value... ]
```

index => value を「,」（カンマ）で区切った構文で、インデックスと値を定義します。インデックスは文字列または、数値を指定することが可能で、インデックスを文字列でしていた配列のことを連想配列とも呼びます。

インデックスの指定を省略した場合は、0 から始まる整数インデックスが自動的に生成されます。整数インデックスは連番になります。

```
<?php
    $array1 = [ 1, 2, "HTML", "CSS", "PHP" ];
    print_r( $array );
?>
```

```
Array ( [0] => 1 [1] => 2 [2] => HTML [3] => CSS [4] => PHP )
```

出力結果

配列の追加と修正

「[]」（角括弧）の中にキーを指定し、値を代入することにより、既存の配列要素の場合は、キーで指定した配列要素内の値を修正します。キーで指定した要素が存在しない場合は、最後尾に新規要素として追加されます。

「[]」の中のキーを省略することも可能で、省略した際は、配列の最後尾に整数インデックスの要素を新規作成して、代入した値を保存します。

```
$array = [ 1, 2, "HTML", "CSS", "PHP" ];
$array[ 4 ] = "JS"; // $array内のキーが 4 の値を修正
$array[] = "PHP";   // $arrayに新規要素としてデータを追加
print_r( $array );
```

```
Array ( [0] => 1 [1] => 2 [2] => HTML [3] => CSS [4] => JS [5] => PHP )
```

出力結果

配列と連想配列の組み合わせ

配列はキーは、整数インデックスと文字列が混ざっていてもエラーにはならないですが、実際に処理を行っていくさいに不都合が多くなるので、配列の要素内に連想配列が保存されていて、連想配列の要素内に配列が保存されていると、配列構造をしっかりと考えてデータのツリー構造を構築するように心がけましょう。

```
$ecc = [  
    [  
        "collage" => "IT",  
        "description" => "ECCコンピュータでITを学ぼう！技術に挑もう！それがプロになる一番の近道だ。",  
        "courses" => [  
            [  
                "name" => "Webデザイン",  
                "annual" => 3,  
                "description" => "デザインの基礎からWeb制作の全工程までを学び、問題をWebデザインで解決する能力を習得します。"  
            ],  
            [  
                "name" => "IT開発エキスパート",  
                "annual" => 4,  
                "description" => "スペシャリストになるために必要なスキルを幅広く学習できます。"  
            ]  
        ]  
    ]  
];  
print_r( $ecc );
```

```
Array  
(  
    [0] => Array  
    (  
        [collage] => IT  
        [description] => ECC コンピュータで IT を学ぼう！技術に挑もう！それがプロになる一番の近道だ。  
        [courses] => Array  
        (  
            [0] => Array  
            (  
                [name] => Web デザイン  
                [annual] => 3  
                [description] => デザインの基礎から Web 制作の全工程までを学びます。  
            )  
            [1] => Array  
            (  
                [name] => IT 開発エキスパート  
                [annual] => 4  
                [description] => スペシャリストになるために必要なスキルを幅広く学習できます。  
            )  
        )  
    )  
)
```

出力結果

ループ処理

PHPで利用できるループ命令を一部紹介します。ループが終了せず、繰り返しを続けてしまうと、Webサーバーへの負担が増えてしまうため、ループ命令が終了するようになっているからを確認してから実行するようにしましょう。

while

while は、PHP でもっとも簡単なループです。式の値が true の間ループ処理を繰り返し実行します。はじめから式が false の場合は、1 回もループ処理は実行されません。

```
while( 式 ) {  
    反復処理命令  
}
```

```
while( 式 ) :  
    反復処理命令  
endwhile;
```

{ } でのグループ化を使わない記述方法

```
$array = [ "HTML", "CSS", "JS", "PHP" ];  
$length = count( $array );  
$i = 0;  
while( $i < $length ) {  
    print "{$i} : {$array[ $i ]}";  
    $i++;  
}
```

for

for は、PHP でもっとも複雑なループです。「何回ループ処理を実行するか」を3つの式を用いて表します。

```
for( 式1; 式2; 式3 ) {  
    反復処理命令  
}
```

```
for( 式1; 式2; 式3 ) :  
    反復処理命令  
endfor;
```

{ } でのグループ化を使わない記述方法

式1 ループ開始時に実行されます。

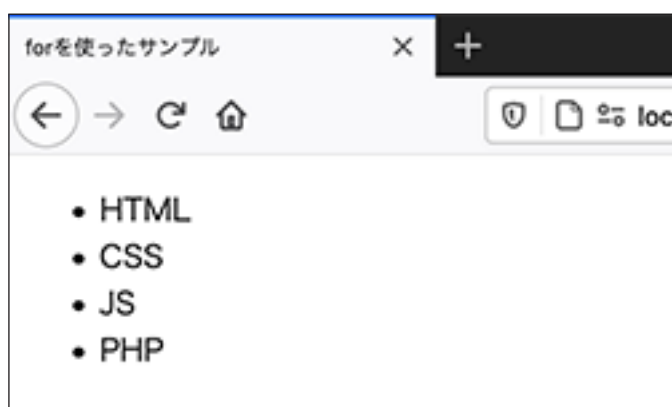
式2 各繰り返しの開始時に評価され、値が true の場合はループは継続され、false の場合はループを終了します。

式3 各繰り返しの後に、実行されます。

```
<?php  
$array = [ "HTML", "CSS", "JS", "PHP" ];  
$length = count( $array );  
?>
```

省略

```
<body>  
  <ul>  
    <?php for( $i = 0; $i < $length; $i++ ): ?>  
      <li><?= $array[ $i ] ?></li>  
    <?php endfor ?>  
  </ul>  
</body>
```



foreach

foreach は、拡張 for 文と呼ばれ、配列に対してループ処理を行うのに特化しており、配列を使ったループ処理に大変便利な命令となります。

foreach が使えるのは、配列とオブジェクトのみで、他のデータ型に対して使うとエラーとなります。

foreach の繰り返しの回数は、foreach で使用する配列のデータ数（要素数）回となり、各繰り返しの開始時に、配列の先頭要素から順次アクセスしていき、as キーワードの右辺へ要素のデータを代入します。

```
foreach( 配列データ as value ) {  
    反復処理命令  
}
```

各要素から値のみ取り出したいとき

```
foreach( 配列データ as value ) :  
    反復処理命令  
endforeach;
```

{ } でのグループ化を使わない記述方法

配列要素のキーと値の両方を取得する形式も用意されています。

```
foreach( 配列データ as key => value ) {  
    反復処理命令  
}
```

各要素からキー（要素名）も取り出したいとき

```
foreach( 配列データ as key => value ) :  
    反復処理命令  
endforeach;
```

{ } でのグループ化を使わない記述方法

```
<?php  
$collages = [  
    [  
        "name" => "IT",  
        "description" => "ECCコンピュータでITを学ぼう！技術に挑もう！それがプロになる一番の近道だ。",  
        "courses" => [  
            [  
                "name" => "Webデザイン",  
                "annual" => 3,  
                "description" => "デザインの基礎からWeb制作の全工程までを学び、問題をWebデザインで解決する能力を習得します。"  
            ],  
            [  
                "name" => "IT開発エキスパート",  
                "annual" => 4,  
                "description" => "スペシャリストになるために必要なスキルを幅広く学習できます。"  
            ]  
        ]  
    ]  
];  
?>
```

省略

省略

```
<body>
<h1>ECCコンピュータ専門学校</h1>
<ul>
<?php foreach( $collages as $collage ): ?>
  <li>
    <h2><?= $collage[ "name" ] ?>カレッジ</h2>
    <p><?= $collage[ "description" ] ?></p>
    <h3>コース一覧</h3>
    <ul>
    <?php foreach( $collage[ "courses" ] as $course ): ?>
      <li>
        <h4><?= $course[ "name" ] ?>コース ( <?= $course[ "annual" ] ?>年制 ) </h4>
        <p><?= $course[ "description" ] ?></p>
      </li>
    <?php endforeach ?>
    </ul>
  </li>
<?php endforeach ?>
</ul>
</body>
```

foreachを使ったサンプル

×

+

← → ↺ 🏠

localhost/2021_html_css/document/sample/sample-php-foreach.php

ECCコンピュータ専門学校

・ITカレッジ

ECCコンピュータでITを学ぼう！技術に挑もう！それがプロになる一番の近道だ。

コース一覧

- Webデザインコース（3年制）

デザインの基礎からWeb制作の全工程までを学び、問題をWebデザインで解決する能力を習得します。

- IT開発エキスパートコース（4年制）

スペシャリストになるために必要なスキルを幅広く学習できます。

スーパーグローバル変数

PHP で定義済み変数として用意されており、スクリプト全体のすべてのスコープで 사용할 수 있는 변수になります。

スーパーグローバル変数には、次のようなものがあります。

名前	説明
\$_GET	HTTP GET 変数
\$_POST	HTTP POST 変数
\$_FILES	HTTP ファイルアップロード変数
\$_SESSION	セッション変数
\$_COOKIE	HTTP クッキー変数
\$_SERVER	サーバー情報および実行時の環境情報
\$_ENV	環境変数
\$_REQUEST	HTTP リクエスト変数
\$GLOBALS	グローバルスコープで使用可能なすべての変数への参照

\$_GET

URL パラメータで、現在のスクリプトに渡された変数の連想配列になります。HTTP GET メソッドのデータだけでなく、クエ리스트リングも含まれます。

\$_POST

HTTP POST メソッドから現在のスクリプトに渡された変数の連想配列です。

\$_SERVER

ヘッダ、パス、HTTP リクエストメソッドなどの情報を持った連想配列になります。この配列は Web サーバーにより作成され、すべ Web サーバーがこれらを全て提供する保証はありません。

関数の紹介

isset

変数が宣言されていること、そして null ではないことを調べます。

```
isset( $var )
```

変数 var が宣言されており、その値が null とは異なる場合は、true を返し、そうでなければ false を返します。

PHPマニュアル:<https://www.php.net/manual/ja/function.isset.php>

trim

文字列の先頭および末尾にあるホワイトスペースを取り除きます。

```
trim( $string )
```

文字列の先頭および末尾にあるホワイトスペース（半角・全角スペース）を取り除き、取り除いた文字列を返します。

PHPマニュアル:<https://www.php.net/manual/ja/function.trim.php>

nl2br

改行文字の前に HTML の改行タグを挿入します。

```
nl2br( $string, $use_xhtml )
```

string に含まれる改行コードの前に
（XHTML 準拠）または、
（HTML 準拠）を挿入した文字列を返します。

use_xhtml は初期値が true に設定されており、XHTML 準拠の br タグが挿入されます。HTML 準拠の br タグを使いたい場合は、false を指定しましょう。

PHPマニュアル:<https://www.php.net/manual/ja/function.nl2br.php>

htmlspecialchars

特殊文字を HTML 特殊文字エンティティへ変換します。

htmlspecialchars(\$string, \$flag, \$encoding)

string 内の文字列から「&」や「<」や「>」のような HTML において、特殊な意味を持つ特定の文字を HTML の特殊文字形式に変換した文字列を返します。

flag で変換対象の文字を切り替えます。

encoding で文字列を変換する際の文字エンコーディングを指定します。省略された場合は、php.ini で指定されたデフォルト値を使います。

XSS 対策のために、ユーザーからのデータを HTML へ描画する際は、データを変換しましょう。

変換対象の文字

変換前	変換後
&	&
"	"; (ENT_NOQUOTES の場合は、変換されません)
'	'; (ENT_HTML401) '; (ENT_HTML5)(ENT_QUOTES の場合のみ変換されます。)
<	<;
>	>;

flag 定数

定義名	説明
ENT_COMPAT	ダブルクォーテーションは変換しますが、シングルクォーテーションは変換しません。
ENT_QUOTES	ダブルクォーテーションとシングルクォーテーションを共に変換します。
ENT_NOQUOTES	ダブルクォーテーションとシングルクォーテーションを共に変換しません。
ENT_HTML401	コードを HTML 4.01 として処理します。
ENT_XML1	コードを XML 1 として処理します。
ENT_XHTML	コードを XHTML として処理します。
ENT_HTML5	コードを HTML 5 として処理します。