ソフトウェア開発
第 10 回目授業

平野 照比古

2016/12/2

関数の特徴

- 関数はキーワード function に引き続いて () 内に、仮引数のリストを書く。そのあとに {} 内にプログラム本体を書く。
- 引数は値渡しである。参照渡しをするときは仮引数の前に&を付ける。
- 関数のオーバーロードはサポートされない。
- 関数の宣言を取り消せない。
- 仮引数の後に値を書くことができる。この値は引数がなかった場合のデフォルトの値となる。デフォルトの値を与えた仮引数の後にデフォルトの値がない仮引数を置くことはできない。
- 関数は使用される前に定義する必要はない。
- 関数内で関数を定義できる。関数内で定義された関数はグローバルスコープに存在する。ただし、外側の関数が実行されないと定義はされない。
- 関数の戻り値は return 文の後の式の値。複数の値を関数の戻り値にしたいときは配列にして返す。

ユーザー定義関数の使用例

```
<?php
function example($a, $as, &$b, $f=false) {
  print "\$a = $a\n";
  print_r($as);
  print "\$b = $b\n";
  if(!isset($x)) $x = "defined in function";
  print "\$x = $x\n";
  if($f) {
    print "\$GLOBALS['x'] = ". $GLOBALS['x']."\n";
  }
  $a = $a*2;
  $as[0] += 10;
  $b = $b*2;
  return array($a,$as);
}
```

ユーザー定義関数の使用例—解説

- 1 行目での関数では 3 番目の引数が参照渡し、4 番目の引数がデフォルトの値が設定されている。
- 6 行目の関数 `isset()` は与えられた変数に値がセットされているかどうかを確かめるものである。
- ここでは`$x` は仮引数ではないのでローカルな変数となり、`isset($x)` は `false` となる。`!isset($x)` は `true` となるので変数 `$x` には`"defined in function"`が代入される。
- 9 行目ではデフォルトの引数のチェックのための部分である。
- 11 行目から 13 行目までは変数に値を代入して、呼び出し元の変数が変わるかどうかのチェックをする。
- 14 行目は初めの二つの仮引数を配列にして戻り値としている。

関数の動作チェック

```
$a = 10;
$as = array(1,2);
$b = 15;
$x = "\$x is defined at top level";
```

ここでは、関数に渡す引数の値を設定している。

```
example($a, $as, $b, true);
print "\$a = $a\n";
print_r($as);
print "\$b = $b\n";
```

実行結果

20 行目で呼び出した関数内での出力結果は次のようになる。

```
$a = 10
Array
(
    [0] => 1
    [1] => 2
)
$b = 15
$x = defined in function
$GLOBALS['x'] = $x is defined at top level
```

関数の動作チェック—解説

- 仮引数の値は正しく渡されている。
- 6 行目は判定が `true` になるのでここで新たに値が設定され、それが 7 行目で出力される。
- デフォルトの仮引数に対して `true` が渡されているので、9 行目が実行される。スーパーグローバル`$GLOBAL` にはグローバルスコープ内の変数が格納されている。ここでは 19 行目に現れる変数の値が表示される。

関数の動作チェック—続き

21 行目から 23 行目の出力は次のようになる。

```
$a = 10
Array
(
    [0] => 1
    [1] => 2
)
$b = 30
```

関数の動作チェック—続き (解説)

- 初めの 2 つの引数は配列であっても書き直されていない。11 行目と 12 行目の設定は戻り値にしか反映されない。
- 参照渡しの変数$b は書き直されている。

関数の動作チェック—続き

```
list($resa) = example($a, $as, $b);
print "\$resa = $resa\n";
?>
```

- 24 行目の関数呼び出しはデフォルトの引数がない。
- したがって、引数$f の値が false に設定され、9 行目は実行されない。
- list() は配列である右辺の値のうち先頭から順に指定された変数に代入
- ここでは関数内の 11 行目で計算された値を変数$resa に代入

関数の動作チェック—続き

```
$a = 10
Array
(
    [0] => 1
    [1] => 2
)
$b = 30
$x = defined in function
$resa = 20
```

可変関数

文字列が代入された変数の後に () をつけると、その文字列の関数を呼び出すことができる。

```
<?php
function add($a, $b) { return $a+$b;}
function sub($a, $b) { return $a-$b;}

$a = 5;
$b = 2;
$f = "add";
print "$f($a,$b) = " . $f($a,$b) ."\n";// add(5,2) = 7

$f = "sub";
print "$f($a,$b) = " . $f($a,$b) ."\n";// sub(5,2) = 3
?>
```

可変関数–解説

- 2 行目と 3 行目で 2 つの関数 add() と sub() を定義
- 7 行目で変数$f に文字列"add"を代入して、8 行目で可変関数として呼び出すと、関数 add が呼び出されている。
- 同様に、変数$f に文字列"sub"を代入して、可変関数として呼び出すと、関数 sub が呼び出されている。

サーバーとのデータ交換の基本

Web ページにおいてサーバーにデータを送る方法には POST と PUT の 2 通りの方法がある。

POST による送信

windows.onload =function() 内に次のコードを追加する。

```
var Form = document.getElementsByTagName("form")[0];
Form.setAttribute("method","POST");
Form.setAttribute("action","09sendData.php");
```

HTML の要素に対しては次のことを行う。

- <select>要素の属性に name="select" を追加する。
- id が"colorName"であるテキストボックスに name="colorName" を追加する。
- 「設定」ボタンの要素の後に次の要素を追加する。

```
<input type="submit" value="送信" id="Send"></input>
```

一時期は name 属性を指定しておくと、id 属性を兼ねていた時期もあったが、最近では両者は厳密に区別されている。

POST による送信 (解説)

- このページでは「送信」ボタンを押すと<form>の action 属性で指定されたプログラムが呼び出される。
- ここでは Web ペー ジと同じ場所にある 09sendData.php が呼び出される。

サーバープログラムのリスト

```
<?php
print <<<_EOL_
<!DOCTYPE html>
<head>
<meta charset="UTF-8"/>
<title>サーバーに送られたデータ</title>
</head>
<body>
<table>
_EOL_;
foreach($_POST as $key=>$value) {
 print "<tr><td>$key</td><td>$value</td></tr>\n";
}
print <<<_EOL_
</table>
</body>
</html>
_EOL_;
?>
```

サーバープログラムのリスト (解説)

- 2 行目から 10 行目の間はヒアドキュメント形式で HTML 文書の初めの部分を出力させている。
- `method="POST"`で呼び出されたときには `form` 要素内の `name` 属性が指定されたものの値がスーパーグローバル`$_POST` 内の連想配列としてアクセスができる。
- 11 行目から 13 行目でそれらの値を `table` 要素内の要素として出力している。

サーバープログラムのリスト (ソース)

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8"/>
<title>サーバーに送られたデータ</title>
</head>
<body>
<table><tr><td>select</td><td>yellow</td></tr>
<tr><td>color</td><td>green</td></tr>
<tr><td>colorName</td><td>gray</td></tr>
</table>
</body>
</html>
```

GET による通信

- method="PUT"で呼び出した場合にはスーパーグローバル$_GET を用いる。
- スーパーグローバル$_REQUEST は method="POST"でも method="PUT"で呼び出された場合の$_POST や$_GET の代わりに使用できる。

通信に関する注意

- type="submit"の input 要素は、ボタンが押されたときに直ちに、action 属性で指定された処理が呼び出される。
- サーバーにデータを送る前に最低限のエラーチェックを行い、エラーがない場合にだけサーバーと通信するのが良い。

スパーグローバルの補足

- $_SERVER
  サーバーにアクセスしたときのクライアントの情報などを提供。具体的な内容はクライアントごとに異なる。
- $_COOKIE
  COOKIE とは Web サーバー側からクライアント側に一時的にデータを保存させる仕組み。すでに訪問したことがあるサイトに対して情報を開始時に補填する機能などを実現できる。
- $_SESSION
  セッションとはある作業の一連の流れを指す。たとえば会員制のサイトではログイン後でなければページを見ることができない。情報のページに直接行くことができないような仕組みが必要

セッション

- HTTP 通信はセッションレスな通信である（各ページが独立して存在し、ページ間のデータを直接渡せない）
- セッションを確立するためには、クライアント側から情報を送り、それに基づいてサーバー側が状況を判断するなどの操作を意識的にする必要
- PHP ではセッションを開始するための関数 `session_start()` とセッションを終了させる `session_destroy()` が用意されている。
- セッションを通じで保存させておきたい情報はこの連想配列に保存
- セッションの管理はサーバーが管理
- この機能は COOKIE の機能を利用して実現

Web Storage

- localStoarage と sessionStorage の 2 種類
- localStorage は文字列をキーに、文字列の値を持つ Storage オブジェクト
- 同一の出身 (プロトコルやポート番号も含む) のすべてのドキュメントがおなじ localStorage を共有
- このデータは意識的に消さない限り存在
- sessionStorage はウインドウやブラウザが閉じられると消滅
- セッション間の情報の移動を可能にしている。

Web Storage の補足

- いくつかのサイトではこの機能を用いており、その開発者ツールで見ることが可能
- データの形式は文字列である。構造化されたデータは JSON 形式で保存するのがよい

WebStrage の例 (1)

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>WebStorage --- localStorage</title>
```

WebStrage の例 (2)

```
<script type="text/javascript">
//<![CDATA[
var Storage = window.localStorage;
//var Storage = window.sessionStorage;
window.onload = function() {
  var AccessList, Message = document.getElementById("message");
  var D = new Date();
  if(Storage["access"]) {
    AccessList = JSON.parse(Storage["access"]);
  } else {
    AccessList = [];
    appendMessage(Message, "初めてのアクセスです");
  }
  AccessList.unshift(D.getTime());
  Storage["access"] = JSON.stringify(AccessList);
  appendMessage(Message, "今までのアクセス時間です");
  AccessList.forEach(function(D, i, A) {
    appendMessage(Message, new Date(D));
  });
}
```

WebStrage の例 (3)

```
function appendMessage(P, Mess) {
  var div = document.createElement("div");
  P.appendChild(div);
  div.appendChild(document.createTextNode(Mess));
}
//]]
</script>
</head>
```

WebStrage の例 (4)

```
<body>
  <form action="10next.html">
    <input type="submit" value="次のページ"></input>
  </form>
  <div  id="message"/>
</body>
</html>
```

レポート問題