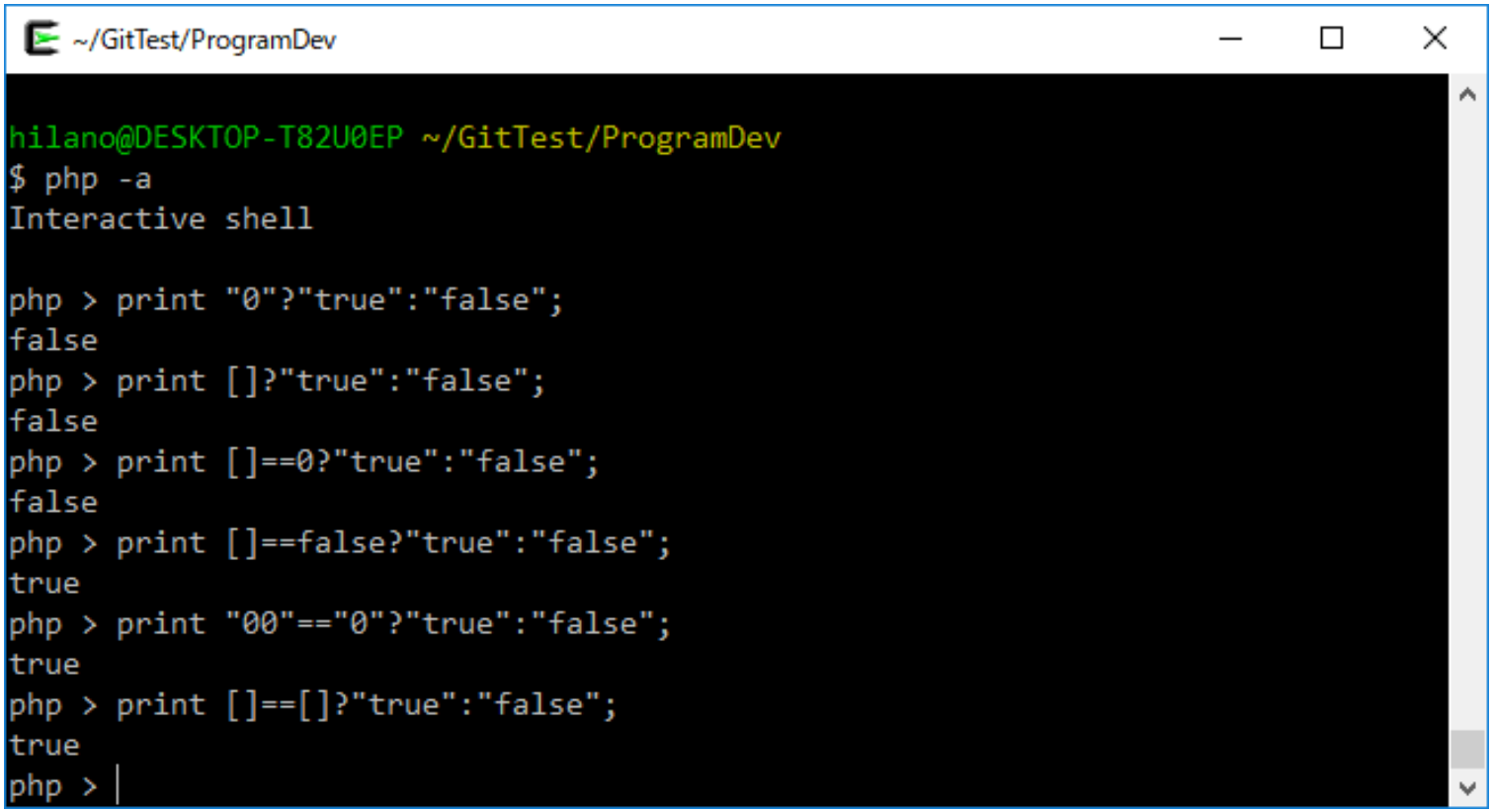
ソフトウェア開発
第 10 回目授業

平野 照比古

2016/12/2

問題 1

ルーブリックに記載したもの以外にもたくさんある。



問題 2

```
$a = array(1,2,3,"4");
```

とおいて違いを見るとよい。
ブラウザで実行した場合には、表示された画面のソースを見るほうが結果が見やすい。

問題 3–array_splice() の仕様

```
array array_splice ( array &$input ,
int $offset [, int $length = count($input)
[, mixed $replacement = array() ]] )
```

- input 入力の配列。
- offset offset が正の場合、削除される部分は 配列 input の最初から指定オフセットのぶんだけ進んだ位置
  offset が負の場合、削除される部分は、 input の末尾から数えた位置
- length length が省略された場合、 offset から配列の最後までが全て削除
  length が指定され、正の場合、複数の要素が削除
  負の length が指定された場合、削除される部分の末尾の位置は配列の末尾を基準にして計算されます。
  length にゼロを指定した場合は、どの要素も削除しません。
- replacement 配列 replacement が指定された場合、 削除された要素は、この配列の要素で置換

問題 2

```
 1 <?php
 2 $a = [1,2,3,4,5];
 3 print array_splice($a, -1, 1)[0]."\n";//POP
 4 print_r($a);
 5 array_splice($a, count($a), 0, 10);//PUSH
 6 print_r($a);
 7 print array_splice($a, 0, 1)[0]."\n";//SHIFT
 8 print_r($a);
 9 array_splice($a, 0, 0, [-1, -2]);//UNSHIFT
10 print_r($a);
11 function pop(&$a) {
12     return array_splice($a, -1, 1)[0];
13 }
14 print pop($a)."\n";
15 print_r($a);
16 ?>
```

問題 3

```
<?php
$A = 2;                  //JavaScript のときは $ は取り除く。
$B = 5;

print "1:$A+$B\n";     //JavaScript のときは print は   console.log() になおす。
print "2:"+$A+$B+"\n";
print "3:".$A+$B."\n";//JavaScript のときは省略すること

print sum(10,20,30)."\n";
print sum(10,20)."\n";
print sum(10,20,30,40)."\n";

print sum2(10,20)."\n";

function sum($a,$b,$c){
  return $a+$b+$c;
}
function sum2($a,$b){
  return $A+$B;
}
?>
```

問題 3–結果

- PHP では+数として加法
- 文字列は数として正しいところまで変換
- 1:では内の変数は値に置き換えられる
- 2:では"2:"の部分は数の 2 に変換、その後加法が行われ、9 となる。改行コードは数の 0 に変換
- 3:では"3:".$A の部分は"3:2"になり、+で $3+5$ が計算される。
- PHP では引数が少ないと警告が出る。多い方は無視
- sum2() 内では$A などは関数内で値が定義されていないので警告が出る

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。

```
let 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() {
  let AccessList, Message = document.getElementById("message");
  let 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) {
  let 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>
```

Ajax[1] とは

- Asynchronous Javascript+XML の略
- 非同期 (Asynchronous) で Web ページとサーバーでデータの交換を行い、クライアント側で得られたデータをもとにその Web ページを書き直す手法

- Google Maps がこの技術を利用したことで一気に認知度が高まった。
- 検索サイトでは検索する用語の一部を入力していると検索用語の候補が出てくる。これも Ajax を使用している (と考えられる)。

[1] Ajax の名称を提案したブログ
http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

XMLHTTPRequest

Ajax の機能はこのオブジェクトを用いて実現

- 日付が変わったときに、その日の記念日をメニューの下部に示す
- 記念日のデータは http://ja.wikipedia.org/wiki/日本の記念日一覧の表示画面からコピーして作成

リスト (1)

```
42      let d2 = Form.children[2].value
43      d = new Date(Year.value, Month.value, 0).getDate();
44      if( d != Form.children[2].children.length) {
45        Form.replaceChild(Days[d],Form.children[2]);
46        d2 = Math.min(Form.children[2].length, d2);
47        Form.children[2].value = d2;
48      }
```

リスト—概要

- 以前のものとは 41 行目から 61 行目が異なっている。
- イベントハンドラーを関数として定義している。
- 42 行目から 48 行目は以前と同じプルダウンメニューの処理である。

リスト (2)— XMLHttpRequest オブジェクトの生成

```
49      let  xmlHttpObj = new XMLHttpRequest();
50      if(xmlHttpObj) {
51        xmlHttpObj.onreadystatechange = function(){
52          if(xmlHttpObj.readyState == 4 && xmlHttpObj.status == 20
53            document.getElementById("details").firstChild.nodeValu
54                xmlHttpObj.responseText;
55          }
56        }
57        xmlHttpObj.open("GET",
58          `./aniversary.php?month=${Month.value}&day=${d2}`,true);
59        xmlHttpObj.send(null);
60      }
61    }
62    Form.addEventListener("change", changePulldown,false);
63    changePulldown();
```

XMLHttpRequest の作成

49 行目でサーバーと通信をするための XMLHttpRequest オブジェクトを作成している。

リスト (3)— コールバック関数の登録

```
51          xmlHttpObj.onreadystatechange = function(){
52            if(xmlHttpObj.readyState == 4 && xmlHttpObj.status == 200) {
53              document.getElementById("details").firstChild.nodeValue =
54                  xmlHttpObj.responseText;
55            }
56          }
```

通信の開始

XMLHttRequest が生成できていたら (50 行目)、このオブジェクトが生成する onreadystatechange イベントのイベントハンドラーを登録する (51 行目から 56 行目)。

- XMLHttRequest の readyState は通信の状態を表す。4 は通信終了を意味する。
- 通信が終了しても正しくデータが得られたかを調べる必要がある。200 は正しくデータが得られたことを意味する。
- 得られたデータは responseText で得られる。この場合、得られたデータは文字列となる。このほかに responXML で XML データが得られる。
- 57 行目から 59 行目が通信の開始する。ここでは、GET で行うので、URL の後に必要なデータを付ける。
- GET では送るデータ本体がないので、通信の終了をのため null を送信する。POST のときはここでデータ本体を送る。
- プルダウンメニューが変化したときのイベントハンドラーを登録し (62 行目)、最後に現在の日付データをサーバーに要求する。、

リスト (4)—HTML 文書

```
63      changePulldown();
64    }
65 //]]>
66 </script>
67 </head>
68 <body>
69    <form id="menu"></form>
70    <p id="details"> </p>
71 </body>
```

送られてきたデータの処理

- 得られたデータは 70 行目の p 要素の中に入れる (53 行目から 54 行目)。
- この要素の `firstChild` を指定しているので 85 行目には と の間に空白を設けて、テキストノードが存在するようにしている。

Ajax で呼び出される PHP のプログラム

```
 1 <?php
 2 mb_internal_encoding("UTF8");
 3 //print mb_internal_encoding();
 4 $m = isset($_GET["month"])?$_GET["month"]: $argv[1];
 5 $d = isset($_GET["day"])?$_GET["day"]:$argv[2];
 6 $data = file("aniversary.txt",FILE_IGNORE_NEW_LINES);
 7 for($i=0;$i<count($data);$i++) {
 8   $data[$i] = mb_convert_encoding($data[$i],"UTF8");
 9   $mm = mb_split("\[",$data[$i]);
10   if(count($mm) >1) {
11     if(mb_convert_encoding($m."月","UTF8") === $mm[0]) break;
12   }
13 }
14 for($i++;$i<count($data);$i++) {
15   $data[$i] = mb_convert_encoding($data[$i],"UTF8");
16   $dd = mb_split("\s-\s",$data[$i]);
17   if(($d."日") === $dd[0]) break;
18 }
19 //  print mb_convert_encoding($dd[1],"SJIS");
20 print $dd[1];
21 ?>
```

Ajax で呼び出される PHP のプログラム–解説

- 2 行目で内部で処理をするエンコーディングを UTF8 にしている。関数、mb_internal_encoding 関数を引数なしで呼び出すと現在採用されているエンコーディングを得ることができる。
- 4 行目と 5 行目では月 ($m) と日 ($d) の値をそれぞれの変数に設定している。
  - ここではコマンドプロンプトからもデバッグできるように、スーパーグローバル$_GET 内に値があれば (isset()) が true になれば、その値を、そうでなければコマンドからの引数を設定している。
  - スーパーグローバル$argv はの先頭は呼び出したファイル名であり、その後に引数が順に入る [2]。

[2] C 言語の main 関数は通常、int main(int argc, char* argv[]) と宣言される。argc は argv の配列の大きさを表し、渡された引数のリストが argv[] に入っている。このとき、argv[0] は実行したときのファイル名が入る。

file() 関数

指定されたファイルを行末文字で区切って配列として返す関数

- 引数には URL も指定できる。
- この関数は 2 番目の引数をとることができる。

FILE_USE_INCLUDE_PATH	include_path のファイルを探す
FILE_IGNORE_NEW_LINES	配列の各要素の最後に改行文字を追加しない
FILE_SKIP_EMPTY_LINES	空行を読み飛ばす

- file_get_contents() はファイルの内容を一つの文字列として読み込む。Web ページの解析にはこちらの関数を使うとよい。

読み込むファイルの一部

1 月 [編集]
1 日 – 鉄腕アトムの日
2 日 – 月ロケットの日
[中略]
31 日 – 生命保険の日、愛妻家の日
2 月 [編集]
1 日 – テレビ放送記念日、ニオイの日
2 日 – 頭痛の日
[以下略]

- 月の部分の後には [がある。
- 日の情報は␣-␣で区切られている (␣は空白を表す)。
- すべての日の情報が入っている。

データの処理—月を探す

- 8 行目で念のためコードを UTF8 に変更
- 関数 mb_split() 関数は第 1 引数に指定された文字列パターンで第 2 引数で指定された文字列を分割して配列として返す関数
- 分割を指定する文字列には正規表現が使えるので、文字 [で分割するために、"\["としている (9 行目)。
- 指定された文字列があれば配列の大きさが 1 より大きくなる。その行に対して求める月と一致しているか判定し、等しければループを抜ける (11 行目)。

- 14 行目から 18 行目までは指定された月での指定された日の情報を探している。日を決定する方法も月と同じである。文字列の分割は"\s-\s"となっている[3]。
- 20 行目で得られた情報をストリームに出力している。

[3] これは"\s"ではうまく行かなかったためである。