

サーバーサイドプログラミング

Server Side Scripting

Chapter 01

- サーバサイドプログラミングとは
- ローカル開発環境
- PHPの基本的な記述方法
- 画面への出力

サーバーサイドプログラミングとは

サーバ側で処理を実行するプログラムの総称。
アクセスするたびに違う内容になる可能性があるサイトで
利用されている。

Ex. 掲示板、SNS、ショッピングサイト、ブログなど

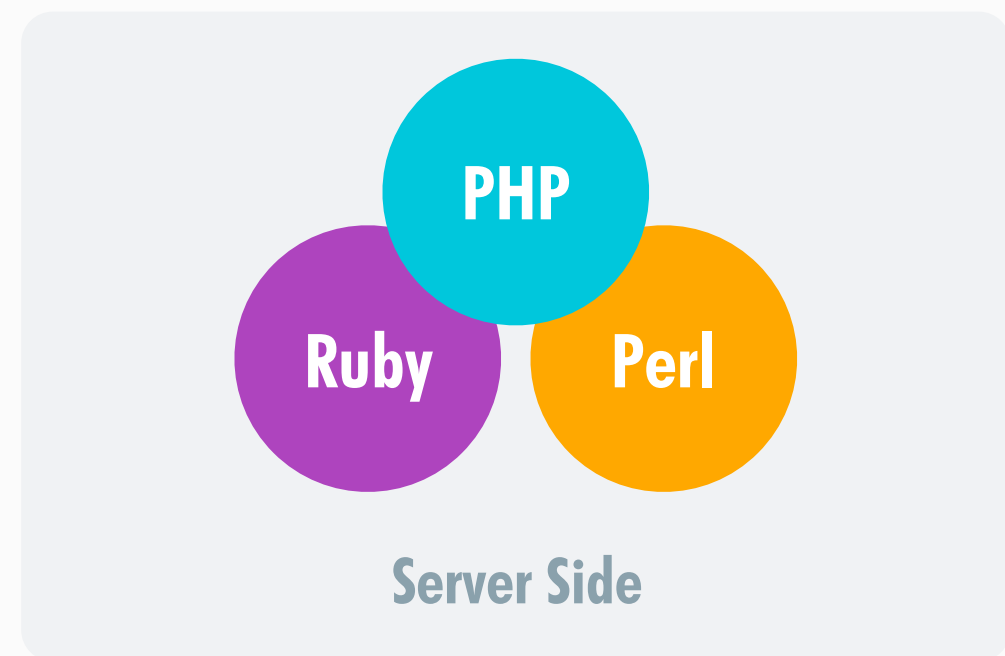


CGI (Common Gateway Interface)

WEBサーバと外部プログラムを連動して動的なWEBサイトを構築する仕組み

クライアントサイドとサーバーサイド

WEBシステムは「クライアント側」か「サーバ側」か
どちらが適しているか見極めることが重要。



Ajax (Asynchronous JavaScript + XML)

これらの技術を組み合わせて非同期通信を行なう仕組み。

PHP

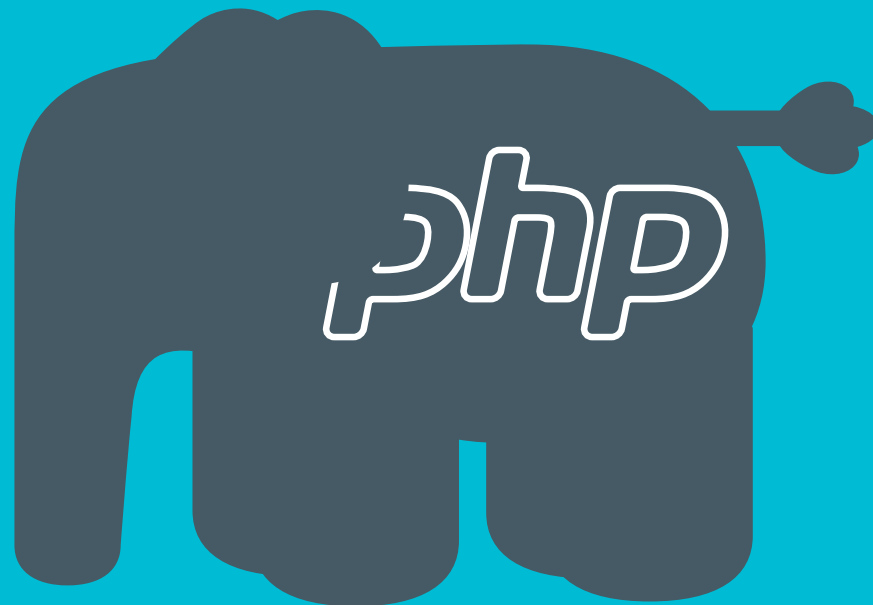
Hypertext Preprocessor

動的にHTMLを生成する
サーバーサイドプログラム
言語。

- 比較的習得が容易
- データベースとの連携が容易
- WEBアプリケーションに特化
- HTMLと混在して記述できる



1995



Contact

Confirm

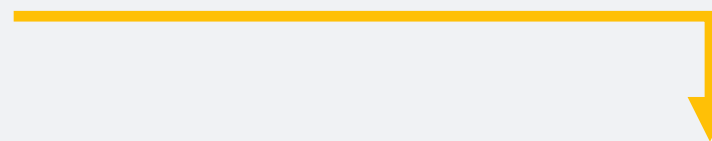
Confirm

Name : 山田 太郎

E-mail : taro@example.com

Message : ちょっと相談です。

Submit



Database

SQL



Confirm

PHP

Name :

E-mail :

Message :

Submit

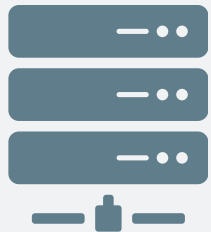


Mail



ローカル開発環境

サーバーサイドの開発には以下の環境を整える必要がある。



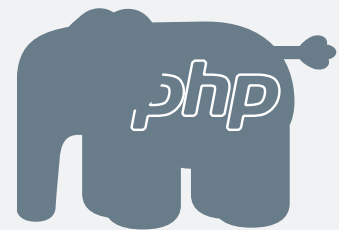
WEBサーバ

Apache



データベース

MySQL



プログラム言語

PHP

XAMPP (Apache + MySQL + PHP + Perl)

開発に必要なソフトをまとめた無料のパッケージ（仮想サーバ）

XAMPPを使った開発手順

HTMLとは異なり、ローカルのWEBサーバ内に保存し、
URLでアクセスして確認する必要がある。

- 1. XAMPP Control Panelを起動する
- 2. 必要なソフトを起動する (Apache、MySQLなど)
- 3. PHPファイルを作成
- 4. 公開用フォルダ (ドキュメントルート) 内に保存
※XAMPPの場合は「htdocs」 (C → xampp →htdocs)
- 5. URLを指定してアクセス
※XAMPPの場合は http://localhost/

PHPの基本文法

PHPは<?php から ?>の間に記述する。

それ以外の箇所に記述した内容は通常のHTMLとして出力される。

拡張子は「.php」

Ex

PHP

<?php ここからPHPで処理をする

print('Hello PHP');

?> ここでPHPの処理を終わる

<p>ここは通常のHTML</p>

ここにPHPの処理を記述
処理の区切りには「;」を付ける

PHPのコメント

ソースコード上にメモを残したい場合に利用する。

- `//` 1行分のコメント
- `/* ~ */` 複数行のコメント

Ex

PHP

`//` 一行分のコメント

`/*`

複数行のコメント

`*/`

画面に表示 (print)

画面に表示するには**print**という命令を使う。
文字列を表示させる際は、「**'**」か「**"**」で囲む。

構文

```
print(表示する内容)           // 画面に表示する命令（関数ではない）
```

Ex

PHP

```
// 画面に「Hello PHP」と表示せよ  
print('Hello PHP');
```

```
// 半角スペースを開けることで()は省略可能  
print 'Hello PHP';
```

画面に表示 (echo)

画面に表示するには**echo**という命令も存在する。
文字列を表示させる際は、「**'**」か「**"**」で囲む。

構文

echo(表示する内容) // 画面に表示する命令（関数ではない）

Ex

PHP

```
// 画面に「Hello PHP」と表示せよ  
echo('Hello PHP');
```

```
// 半角スペースを開けることで()は省略可能  
echo 'Hello PHP';
```

PHPの環境情報

開発前に**phpinfo関数**（命令）を用いてPHPのバージョンや設定内容等を確認する。

構文

phpinfo(**表示したい情報**)

// PHPの環境設定情報を表示する関数

関数名

引数 (関数に渡す値)

※引数はPHPマニュアル等を参照

Ex

PHP

```
phpinfo(); // 全ての情報を表示
```

関数

あらかじめ決められた処理を行い、その結果を返す仕組み。
関数には、組み込み関数と、独自関数がある。

構文

戻り値 = 関数名(引数 , 引数 ...)

結果

処理の名前

処理に必要な値

// 戻り値が不要な関数もある。

// 引数を複数指定する場合は、「,」で区切る。引数は省略できるものもある。

Ex

PHP

```
phpinfo(); // 戻り値も引数も必要ない。
```

```
$users = array('Tom', 'Bob'); // 引数が複数ある場合
```

Chapter 02

- 演算
- 変数
- データ型
- データの送受信

プログラムの基礎

プログラム言語と呼ばれているものは数多くあるが、理解すべきポイントは同じ。

- 演算 …… 数値を計算する
- 変数 …… 計算した数値を保存する
- 分岐 …… プログラム処理の流れを変える
- 反復 …… 同じ処理を繰り返す
- 関数 …… あらかじめ決められた処理を行ない結果を返す

この**5つのポイント**をしっかりと理解すれば、他の言語の理解もスムーズになる。

演算 (数値の計算)

算術演算子を用いて数値を計算する

主な算術演算子

- + 加算
- - 減算
- * 掛け算
- / 割り算
- % 余剰 (余り)

Ex

PHP

```
// 10+10の結果を画面に表示  
print(10+10); // 20
```

```
// 10×10の結果を画面に表示  
print(10*10); // 100
```

```
// 10÷2の余りを画面に表示  
print(10%2); // 0
```

変数 (データを入れる箱)

データに名前を付けて一時的に保存しておく事が出来る。



構文

```
$変数名 = 代入するデータ; // 「=」は代入演算子
```

Ex

PHP

```
// 変数msgに「Hello PHP」を代入
```

```
$msg = 'Hello PHP';
```

変数名のルール

変数名は以下のルールに基づいて命名する必要がある。

- 先頭には「\$ (ドル)」を付ける
- 使える文字は英数字と日本語
- 使える記号は「_ (アンダーバー)」のみ
- 先頭に数値は不可。
- 大文字と小文字は区別される

Ex

PHP

```
$msg = 'Hello'; // OK
$_msg = 'Hello'; // OK
$Msg = 'Hello'; // OK
msg = 'Hello'; // NG
$msg-1 = 'Hello'; // NG
$1msg= 'Hello'; // NG
```

データ型

PHPで扱えるデータ型には、主に以下のものがある。

- **string** 文字列
- **integer** 整数値
- **float** 浮動小数点
- **boolean** 論理値 (True / false)
- **array** 配列

Ex

PHP

```
$str = 'Hello'; // string
```

```
$int = 10; // integer
```

```
$float = 3.14; // float
```

```
$bool = true; // boolean
```

string型の注意点

文字列は「'」または「"」で囲む。

文字列の中にクォートを含む場合は以下のいずれかの対策を行なう。

- 文字列に含まない方のクォートを使う
- クォート文字の前に「¥」を付ける（**エスケープ処理**）

Ex

PHP

```
// 文字列に含まない方のクォートを使う。
```

```
echo "I'm Fine!";
```

```
// クォート文字をエスケープ処理。
```

```
echo 'I¥'m Fine!'
```

シングルクォートとダブルクォートの違い

文字列はクォートによって以下の違いがある。

- シングルクォート …… 入力したものをそのまま表示
- ダブルクォート …… 変数を展開、エスケープ文字を認識

Ex

PHP

```
$name = 'Shibata';
```

```
// シングルクォートは変数を展開しない。
```

```
echo 'Hello { $msg }'; // 出力結果：Hello { $msg }
```

```
// ダブルクォートは変数を展開する。（変数は{～}で囲んで明確にする）
```

```
echo "Hello { $msg }"; // 出力結果：Hello Shibata
```

文字列連結演算子

文字列の結合を行なう際は、「`.`」を用いる

Ex

PHP

```
$name = $_POST['name'];  
  
echo 'あなたの名前は' . htmlspecialchars($name,  
ENT_QUOTES, 'UTF-8') . 'ですね。';
```


データの送受信

データの送信はHTMLのみで送信できる。ただし受信は出来ない。
データの送信方法にはGETとPOSTがある。



構文

```
<form action="送信先" method="GET or POST">
```

```
...
```

```
</form>
```

データの送信

データの送信はHTMLのみで送信できる。ただし受信は出来ない。

Ex

HTML

```
<form action="conf.php" method="POST">
  <dl>
    <dt><label for="user">ユーザ名</label></dt>
    <dd><input type="text" name="user" id="user"></dd>
    <dt><label for="pass">パスワード</label></dt>
    <dd><input type="password" name="pass" id="pass"></dd>
  </dl>
  <p><input type="submit" value="ログイン"></p>
</form>
```

GETとPOST

HTTPで決められたデータの送信方法

	GET	POST
送信方法	URLの末尾にデータを付加 Ex http://example.com?val=3	HTTPのリクエストボディにデータを付加
メリット	フォームを利用せずに送信できる	データの中身が見えない
デメリット	<ul style="list-style-type: none">大量のデータは送信できないデータの中身が見える	送信後にデータは消える
用途例	<ul style="list-style-type: none">サイト内検索ページング	<ul style="list-style-type: none">お問合せフォーム商品購入フォーム

いつでもデータにアクセスしたい時はGET。
ユーザーの情報を送信するときはPOSTを使う。

GET（クエリ情報）による受信

GETはURLの末尾にデータを付加して情報を渡す。
受け取りにはスーパーグローバル変数の**`$_GET`**を用いる。

構文

`$_GET['キー']` // クエリ情報を受け取るスーパーグローバル変数

連想配列

キー・・・input要素のname属性に指定した属性値

Ex

PHP

```
// クエリ情報your_nameの値を変数nameに代入
```

```
$name = $_GET[ 'your_name' ];
```

POST（ポストデータ）による受信

POSTはHTTP通信の一部としてデータを渡す。

受け取りにはスーパーグローバル変数の**`$_POST`**を用いる。

構文

`$_POST['キー']` // ポストデータを受け取るスーパーグローバル変数

連想配列

キー・・・input要素のname属性に指定した属性値

Ex

PHP

```
// ポストデータyour_nameの値を変数nameに代入
```

```
$name = $_POST[ 'your_name' ];
```

クロスサイトスクリプティング (XSS) 対策

ユーザーから受け取ったデータは必ず安全とは限らない。
悪意のあるJavaScriptなどを記述される可能性がある。

構文

省略可能

`htmlspecialchars(文字列, 種類, 文字コード)`

エスケープ処理の関数

※引数はPHPマニュアル等を参照

Ex

PHP

```
// ポストデータyour_nameの値をエスケープ処理後表示
```

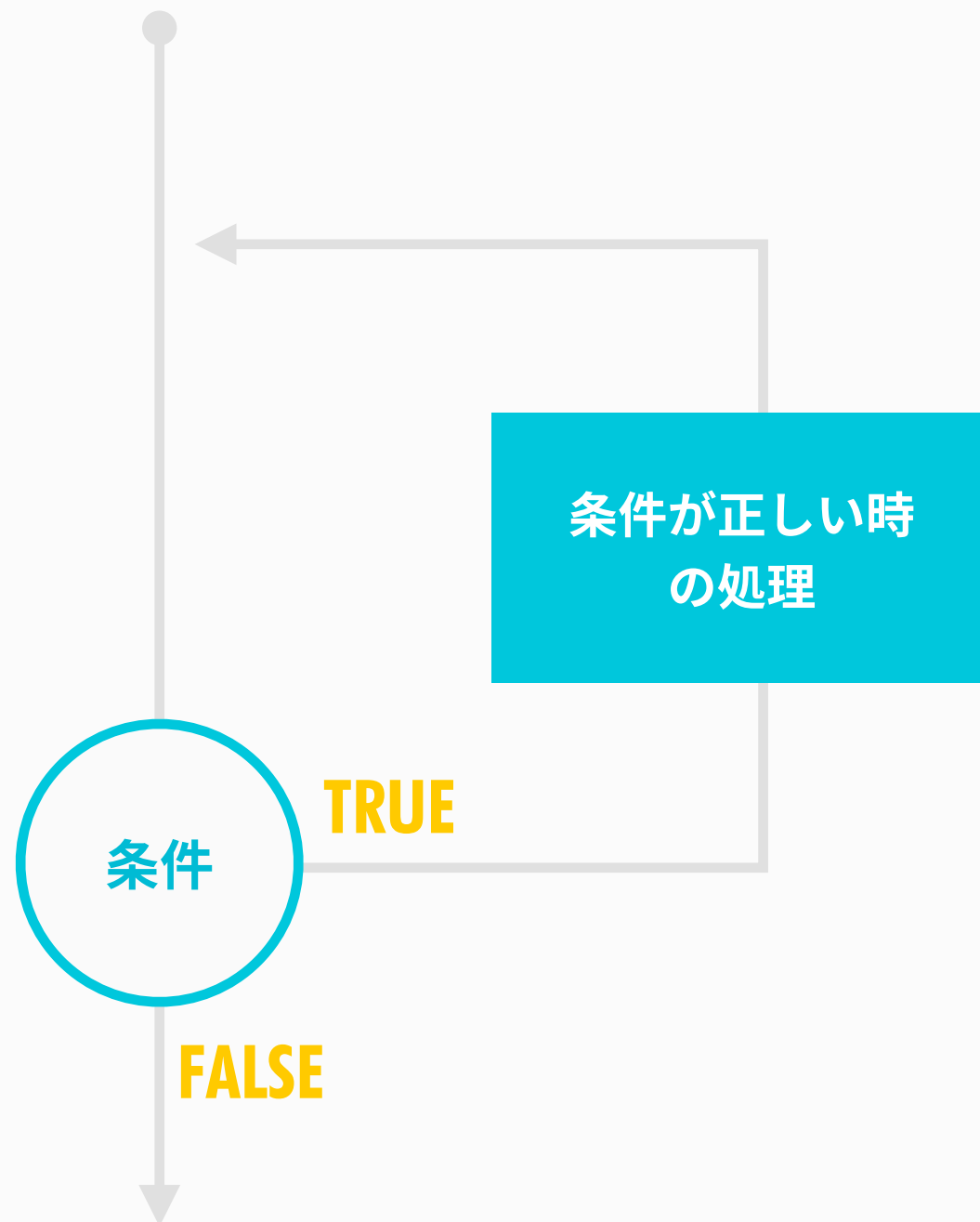
```
echo htmlspecialchars($_POST['your_name'], ENT_QUOTES, 'UTF-8');
```

Chapter 03

- 反復 (while)
- 反復 (for)
- 配列 (通常配列 / 連想配列)
- 反復 (foreach)
- 多次元配列

反復 (while)

条件を満たしている間は同じ処理を繰り返す。



構文

```
while (条件式) {
```

```
// 条件式がTRUEの間、繰り返す処理
```

```
}
```


反復の注意点

反復は、必ず終了するように注意する。

ミスがあると無限ループとなりプログラムが終了しない。

Ex

PHP

// ループ用変数の初期化

```
$i = 0;
```

// ループの継続条件

```
while($i < 10) {  
    echo 'Looping...';
```

// ループ変数の増減

```
$i++; // インクリメント・・・自身に1を加算する算術演算子。$i = $i + 1と同じ  
}
```

左記の3点に注意する。

ひとつでもミスがあると
無限ループになる。

反復 (for)

ループ変数の初期化、条件式、ループ変数の増減を一行で記述できる

構文

```
for (ループ変数の初期化; 条件式; ループ変数の増減) {  
    // 条件式がTRUEの間、繰り返す処理  
}
```

Ex

PHP

```
for($i = 0; $i < 10; $i++) {  
    echo 'Looping...';  
}
```

配列

複数のデータを格納できる保存方法で、変数に似た仕組みを持つ。
配列には「通常配列」と「連想配列」がある。



変数

100個のデータが必要な場合は、
100個の変数が必要になる。

複数のデータ



配列

1つの配列の中には、
複数のデータを保存できる。

配列の作成 (array)

新たな配列を生成するときはarray関数を使って配列を生成する。

構文

```
$配列名 = array(値1, 値2, 値3 ...);
```

※ インデックス番号は「0」から順番に割り振られる。

Ex

PHP

```
// 配列nameを新規に作成  
$name = array('Suzuki', 'Yamada', 'Tanaka');  
  
// 配列nameの1番目の要素を表示せよ  
print($name[1]);
```

配列の作成（ブラケット構文）

配列は部屋番号（インデックス番号）を指定して、中身（要素）に値をセットする。

構文


`$配列名[インデックス番号] = 値;`

※ インデックス番号を省略した場合は、末尾に要素が追加される（空の場合は「0」）。

Ex

PHP

```
$name[0] = 'Suzuki'; // 配列nameの0番目に「Suzuki」を代入
$name[1] = 'Yamada'; // 配列nameの1番目に「Yamada」を代入

print($name[0]); // 配列nameの0番目の要素を表示せよ
```

配列の要素数を調べる (count)

配列の要素数を調べる際は、count関数を用いる。

構文

省略可能

`count(調べたい配列, カウントの方法)`

※引数はPHPマニュアル等を参照

Ex

PHP

```
// 配列nameを新規に作成
```

```
$name = array('Suzuki', 'Yamada', 'Tanaka');
```

```
// 配列nameの要素数を表示せよ
```

```
echo count($name);
```

反復 (foreach)

P68

配列の要素をひとつずつ取り出す時に用いる反復

構文

```
foreach (配列 as 値を格納する変数) {  
    // 配列の要素の数だけ繰り返し実行する処理  
}
```

Ex

PHP

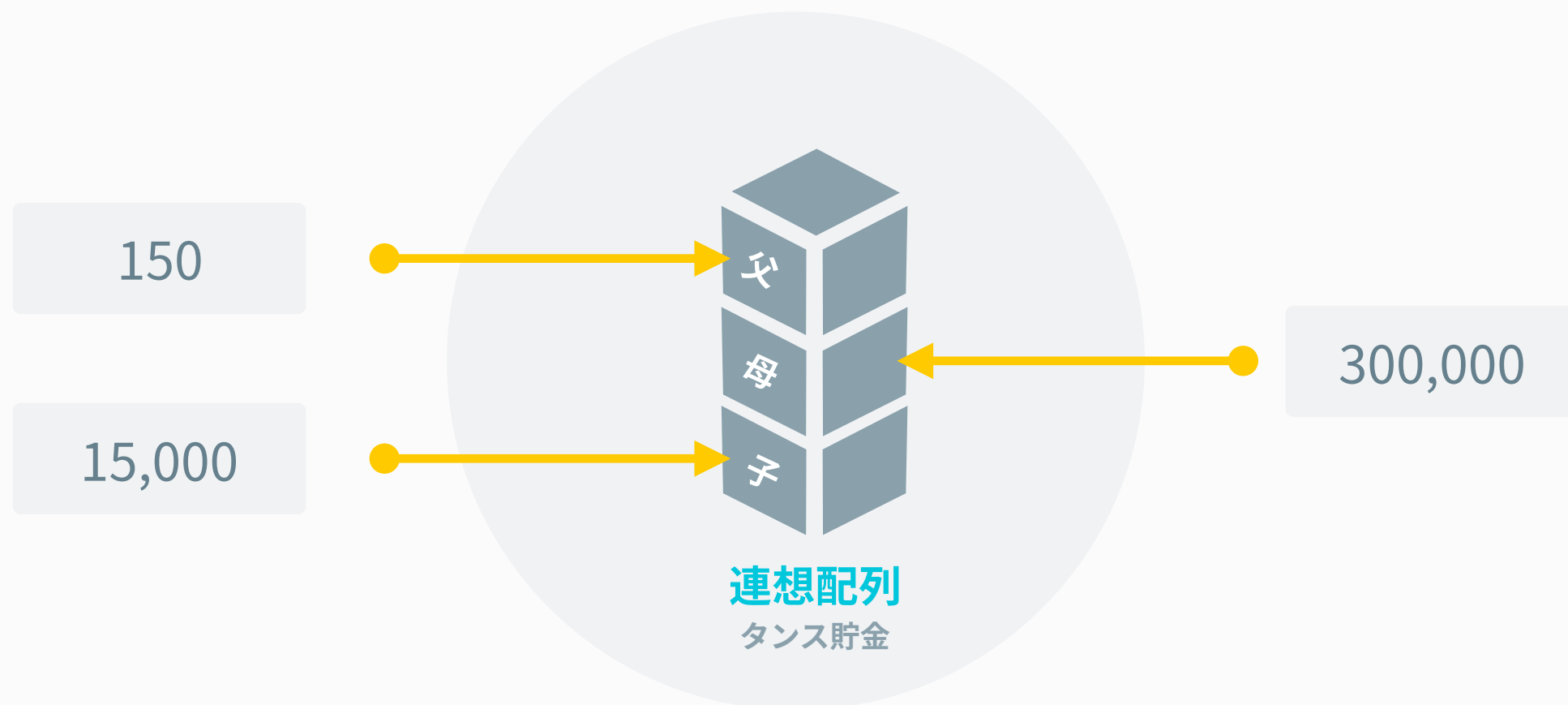
```
$names = array( 'Tanaka', 'Suzuki', 'Yamada' );  
foreach($names as $name) {  
    echo $name . 'さん';  
}
```

連想配列

P48

値を取り出すキーが文字列の配列。

タンスの引き出しにシールを貼るようなイメージ。



連想配列の作成 (array)

array関数によって連想配列を作成する際は、
「=> (ダブルアロー演算子)」を用いて値を代入する。

構文

```
$配列名 = array('キー1' => 値1, 'キー2' => 値2 ...);
```

Ex

PHP

```
$users = array(  
    'name' => 'Suzuki',  
    'pass' => '123abc'  
);  
  
print($users['name']);
```

連想配列の作成（ブラケット構文）

連想配列はキーに文字列を指定して、
中身（要素）に値をセットする。

構文

```
$配列名[ '文字列' ] = 値;
```

Ex

PHP

```
$savings[ 'father' ] = 150;  
$savings[ 'mother' ] = 300000;  
$savings[ 'child' ] = 15000;  
  
print( $savings[ 'mother' ] );
```

反復 (foreach)

配列の各要素をキーと値に分けてひとつずつ取り出すことも可能

構文

```
foreach (配列 as キーを格納する変数 => 値を格納する変数) {  
    // 配列の要素の数だけ繰り返し実行する処理  
}
```

Ex

PHP

```
$users = array( 'Yamada' => 30, 'Suzuki' => 28 );  
foreach($users as $key => $val) {  
    echo $key . 'さん (' . $val . '歳) ' ;  
}
```

多次元配列

配列の中に配列を格納すること。

2次元配列、3次元配列などを作成できる。

0	0	Takahashi
	1	Maeda
	2	Oshima
1	0	Yamamoto
	1	Watanabe
	2	Umeda
2	0	Sashihara
	1	Miyawaki
	2	Kodama

Ex

PHP

```
$members = array(  
    array('Takahashi', 'Maeda', 'Oshima'),  
    array('Yamamoto', 'Watanabe', 'Umeda'),  
    array('Sashihara', 'Miyawaki', 'Kodama')  
);  
  
print($members[2][0]);
```

多次元配列

配列の中に配列を格納すること。

2次元配列、3次元配列などを作成できる。

	name	password
0	Yamada	abc123
1	Tanaka	tantan
2	Suzuki	123123
3	Tom	tomtom
4	Bob	bbbb
5	Stave	stave

Ex

PHP

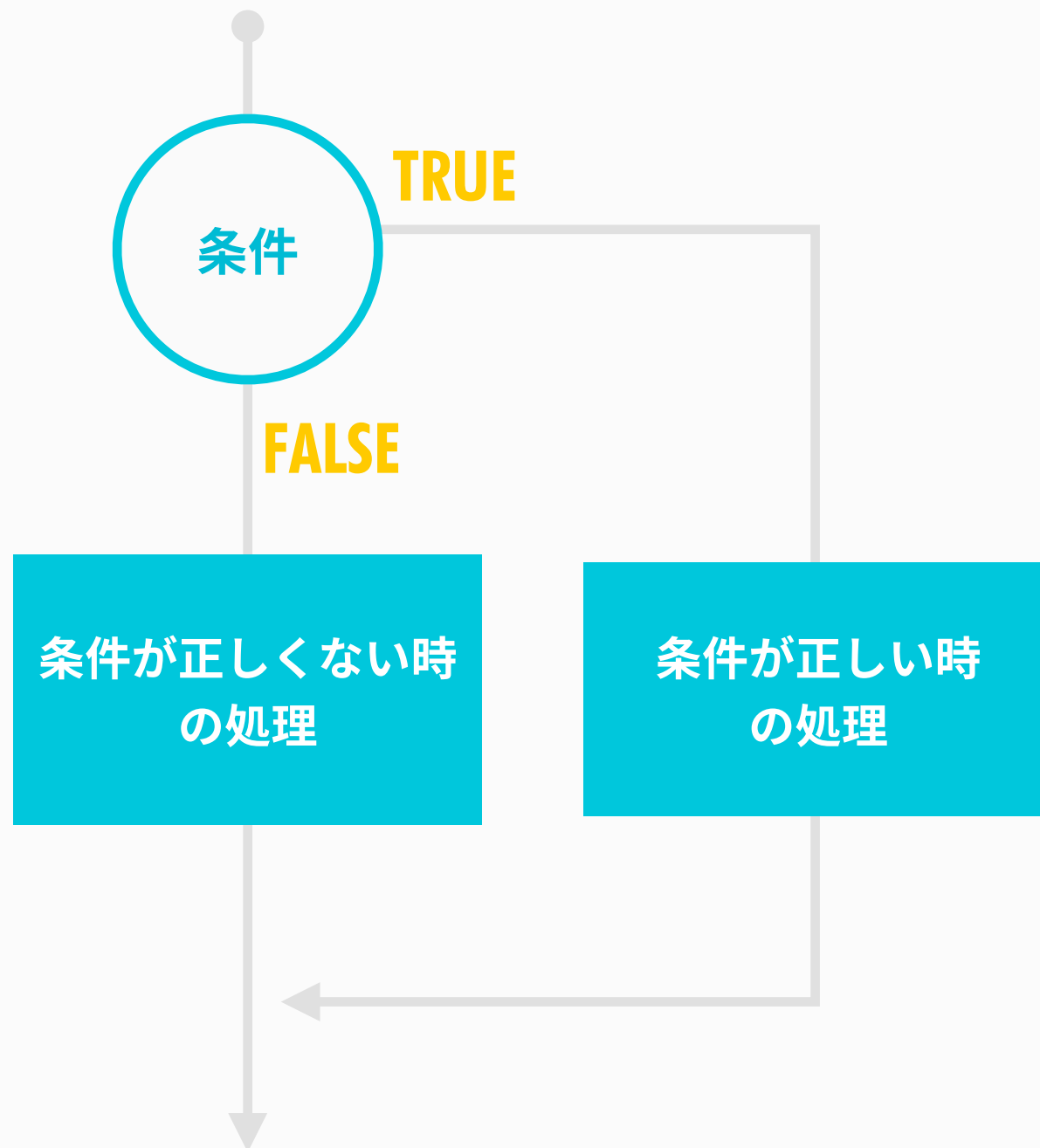
```
$users = array(  
    array(  
        'name' => 'Yamada',  
        'password', 'abc123'  
    ),  
    array(  
        'name' => 'Tanaka',  
        'password', 'tantan'  
    ),  
    ...  
);  
print($users[0]['name']);
```

Chapter 04

- 条件分岐 (if文)
- 独自関数

条件分岐 (if文)

条件式がTRUEかFALSEによって実行する処理を分岐する命令。



構文

```
if (条件式) {  
    // 条件式がTRUEの時に実行する処理  
} else {  
    // 条件式がFALSEの時に実行する処理  
}
```

比較演算子

2つの値を比較するときに用いる演算子。以下は主な比較演算子。

- **A > B** AはBより大きい
- **A < B** AはBより小さい
- **A >= B** AはB以上
- **A <= B** AはB以下
- **A == B** AとBは等しい
- **A != B** AとBは異なる

Ex

PHP

```
$i = 15;  
  
// 変数 i が10より大きい時  
if ($i > 10) {  
    print('10より大きい');  
} else {  
    print('10より小さい');  
}
```


文字列の長さを取得

文字列の長さ（文字数）を取得する

構文

省略可能

`mb_strlen(対象の文字列, 文字コード)`

※引数はPHPマニュアル等を参照

Ex

PHP

```
$str = 'Hello PHP';  
  
// 変数strの文字数を表示  
echo mb_strlen($str);
```

論理演算子

複数の条件式を結合する時に用いる演算子。以下は主な論理演算子。

- **A && B** A、B共にTRUEの時
- **A and B** A、B共にTRUEの時
- **A || B** AかBがTRUEの時
- **A or B** AかBがTRUEの時

Ex

PHP

```
$i = 15;

// 変数 i が10より大きく、20より小さい時
if ($i > 10 && $i < 20) {
    print('条件に一致');
} else {
    print('条件に一致しない');
}
```

独自関数

関数は自分で作る事ができる。よく使う処理は関数にする。

構文

```
function 関数名(引数, 引数) {  
    ...  
    return 戻り値;  
}  
                処理の名前      処理に必要な値  
                結果
```

Ex

PHP

```
function h($s) {  
    return htmlspecialchars($s, ENT_QUOTES, 'UTF-8' );  
}  
  
echo h($_POST['name']);
```

PHPマニュアルの構文表記の読み方

PHPマニュアルでの構文表記は以下のように記述されています。

(<http://php.net/manual/ja/index.php>)

構文

省略可能

戻り値のデータ型 関数名 (引数のデータ型 仮引数 [, 引数のデータ型 仮引数 = デフォルト値])

引数

Ex

PHP

戻り値のデータ型 引数のデータ型 引数のデータ型
bool sort (array \$array [, int \$sort_flags = SORT_REGULAR])
関数名 仮引数 仮引数 デフォルト値

// [~] は省略可能

Chapter 05

- リダイレクト
- HTTPはステートレス
- Cookie
- Session

リダイレクト処理

別のファイルへリダイレクトする際はheader関数を用いる。
header関数は必ず全ての出力の前に記述する必要がある。

構文

header(ヘッダ文字列) //HTTPのレスポンスヘッダを設定する関数

//ヘッダ文字列：リダイレクトの際は「'Location: [半角スペース]URL'」の形式で記述する

※引数はPHPマニュアル等を参照

Ex

PHP

```
// index.phpにリダイレクト
```

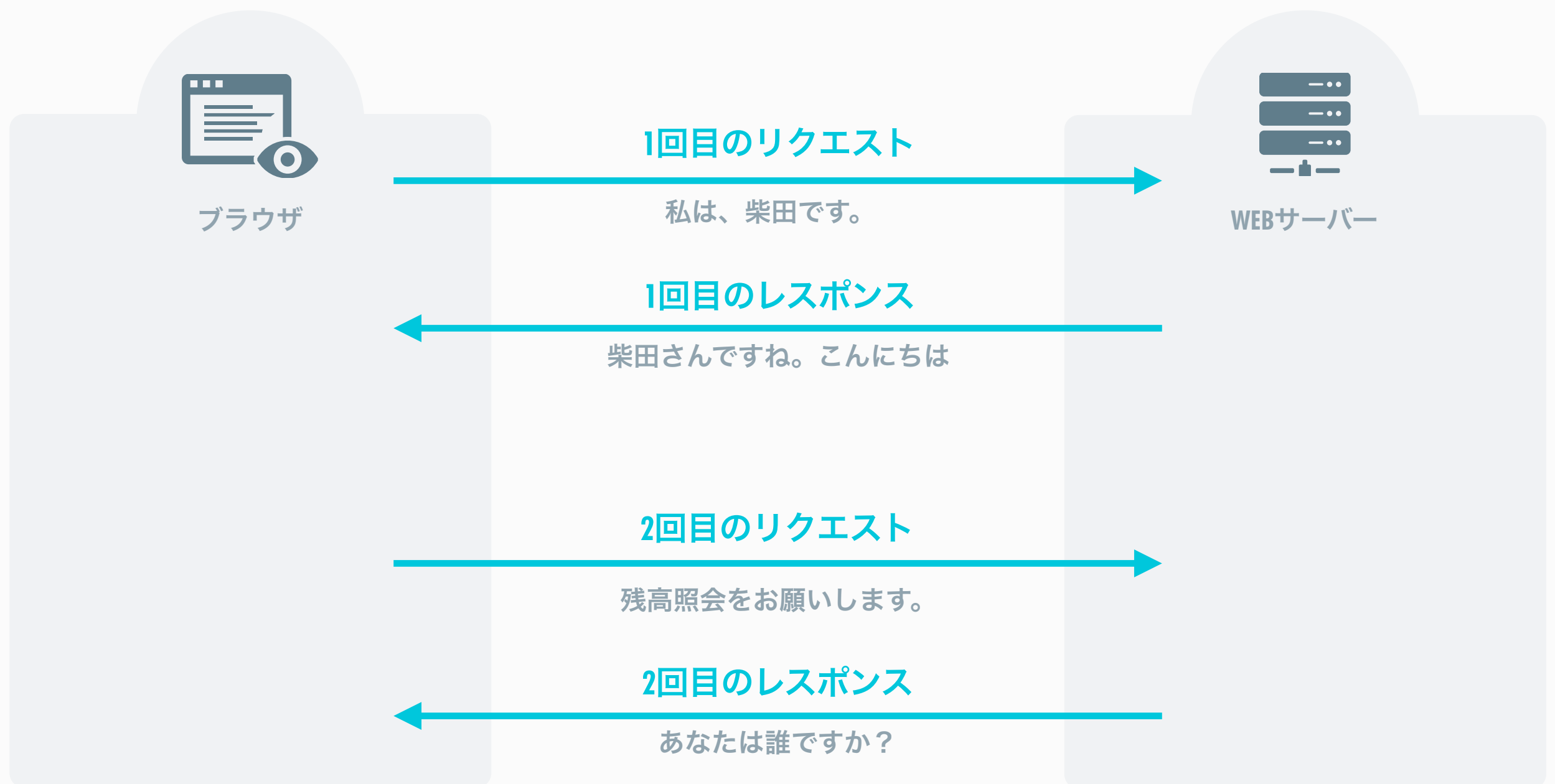
```
header('Location: index.php');
```

```
// 処理の終了
```

```
exit();    // exit関数    . . . スクリプトの実行を終了させる関数
```

HTTPはステートレス

HTTPはリクエストとレスポンスから成り立っている。
WEBサーバでは状態を管理する事はできない。



情報の引き継ぎ

複数のリクエスト間で情報を引き継ぐ方法に以下の方法などがある

- 1. URLのパラメータに引き継ぎたい情報を付ける
- 2. input要素のhiddenフィールドに引き継ぎたい情報付ける
- 3. Cookieに引き継ぎたい情報を保存する
- 4. SESSIONに引き継ぎたい情報を保存する

Cookie

サーバ側で引き継ぎたい情報をクライアント側に保存する仕組み



Cookieの書き込み

クッキーを発行するにはsetcookie関数を用いて値を保存する。
setcookie関数は全ての出力より先に記述する必要がある。

構文

省略可能

-----,

setcookie(クッキー名, 保存する値, 有効期限)

// 有効期限は1970年1月1日からの経過ミリ秒で指定

// 有効期限を省略するとブラウザを閉じるまで保存される

※引数はPHPマニュアル等を参照

Ex

PHP

// クッキー名nameにポストデータnameの値を90日間保存

```
setcookie(name, $_POST['name'], time() + (60 * 60 * 24 * 90));
```

// time()・・・現在の経過ミリ秒（1970年1月1日から）を返す関数

Cookieの読み込み

クッキーの読み込みはスーパーグローバル変数の、`$_COOKIE`を用いる。

構文

```
$_COOKIE[ 'クッキー名' ] // クッキー情報を受け取るスーパーグローバル変数
```

Ex

PHP

```
// クッキー名nameにポストデータnameの値を90日間保存
setcookie(name, $_POST['name'], time() + (60 * 60 * 24 * 90));

// クッキーnameの値を表示
echo htmlspecialchars($_COOKIE['name'], ENT_QUOTES, 'UTF-8');
```

Session

引き継ぎたい情報をサーバ側で保存する仕組み



Sessionの開始

セッションを利用するにはsession_start関数を用いる。
session_start関数は全ての出力より先に記述する必要がある。

構文

```
session_start()           //セッションを開始する為の関数
```

Ex

PHP

```
// セッションを開始  
session_start();
```

Sessionの読み書き

セッションの読み書きには、スーパーグローバル変数の\$_SESSIONを用いる

構文

```
$_SESSION[ 'キー' ]    // セッション情報を受け取るスーパーグローバル変数
```

Ex

PHP

```
// セッションを開始
session_start();
// セッションnameにポストデータnameの値を代入
$_SESSION[ 'name' ] = '$_POST[ 'name' ];
// セッションnameの値を表示
echo htmlspecialchars($_SESSION[ 'name' ], ENT_QUOTES, 'UTF-8');
```

Session情報の破棄

セッション情報を削除するにはsession_destroy関数を用いる。
ただし、セッションIDを受け渡すクッキーは削除されない。

構文

```
session_destroy()    //セッション情報を保存しているファイルを削除
```

Ex

PHP

```
// セッションを開始  
session_start();  
  
// セッション情報を保存しているファイルを削除  
session_destroy();
```

Chapter 06

- メールの送信

メールの送信

マルチバイトのメールを送信する場合は、mb_send_mail関数を用いる。

構文

```
mb_send_mail(宛先, 件名, 本文, 送り主情報) //メールを送信する関数
```

Ex

PHP

```
$to = $_POST['email'];  
$subject = $_POST['subject'];  
$body = $_POST['message'];  
  
// メールを送信  
mb_send_mail($to, $subject, $body, 'From: info@dummy.com');
```

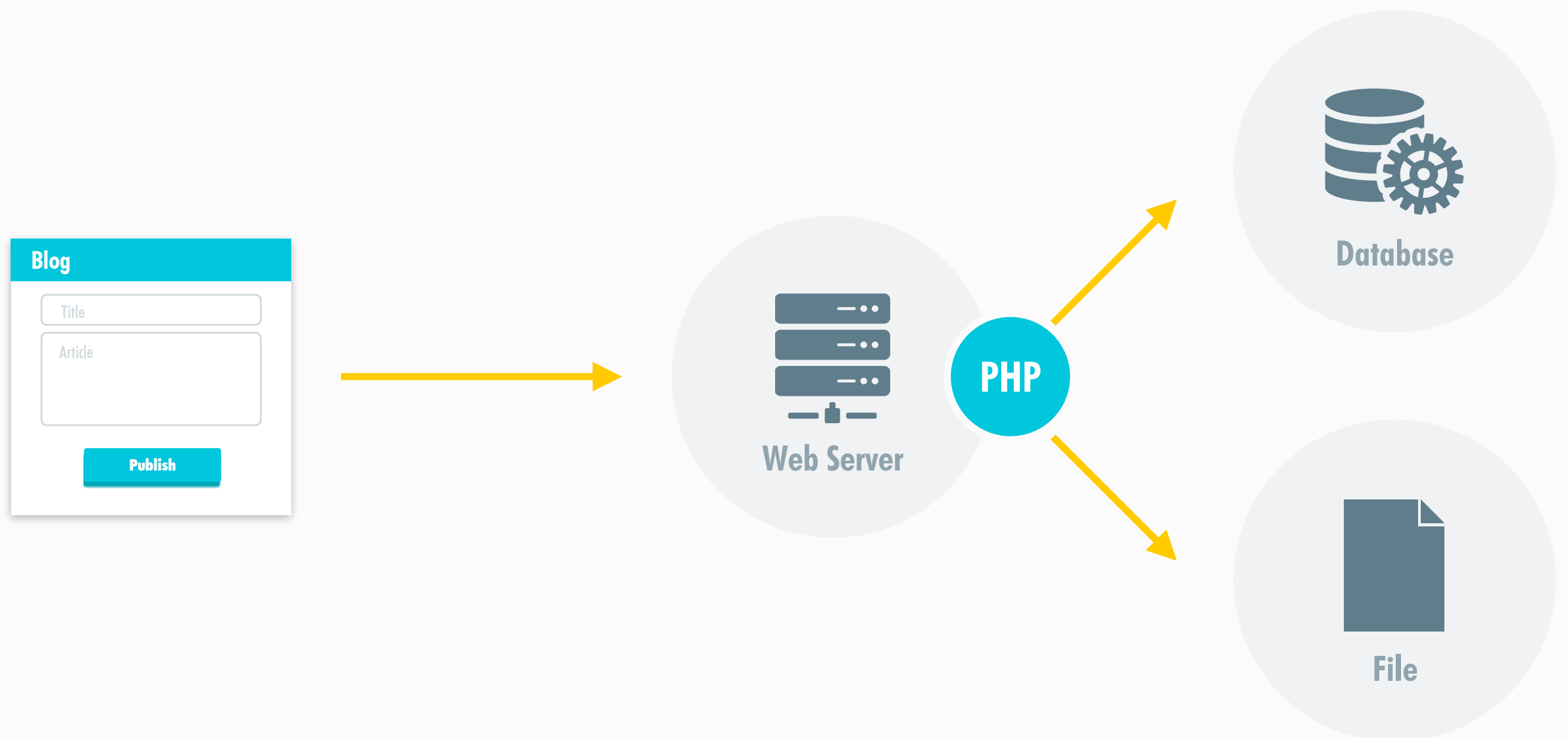
Chapter 08

- データの保存
- ファイル操作

データの保存

何度も利用するデータは保存する必要がある。

保存はデータベースやファイルなどに行なうことができる。



ファイル操作

PHPは簡単にファイルを操作する事ができる。

- 1. ファイルを開く
- 2. ファイルをロックする
- 3. ファイルからデータを読み込み・書き込みする
- 4. ファイルを閉じる（ロックの解除）

ファイルを開く

テキストに書き込む際は、まずそのテキストファイルを開く必要がある

構文

```
$ファイルハンドル = fopen(ファイルパス, 開くモード)
```

```
// ファイルハンドル・・・そのファイルを操作する為のキーとなる情報
```

※引数はPHPマニュアル等を参照

Ex

PHP

```
// data.txtを読み込みモードで開く
```

```
$fp = fopen('data.txt', 'rb');
```

主なオープンモード

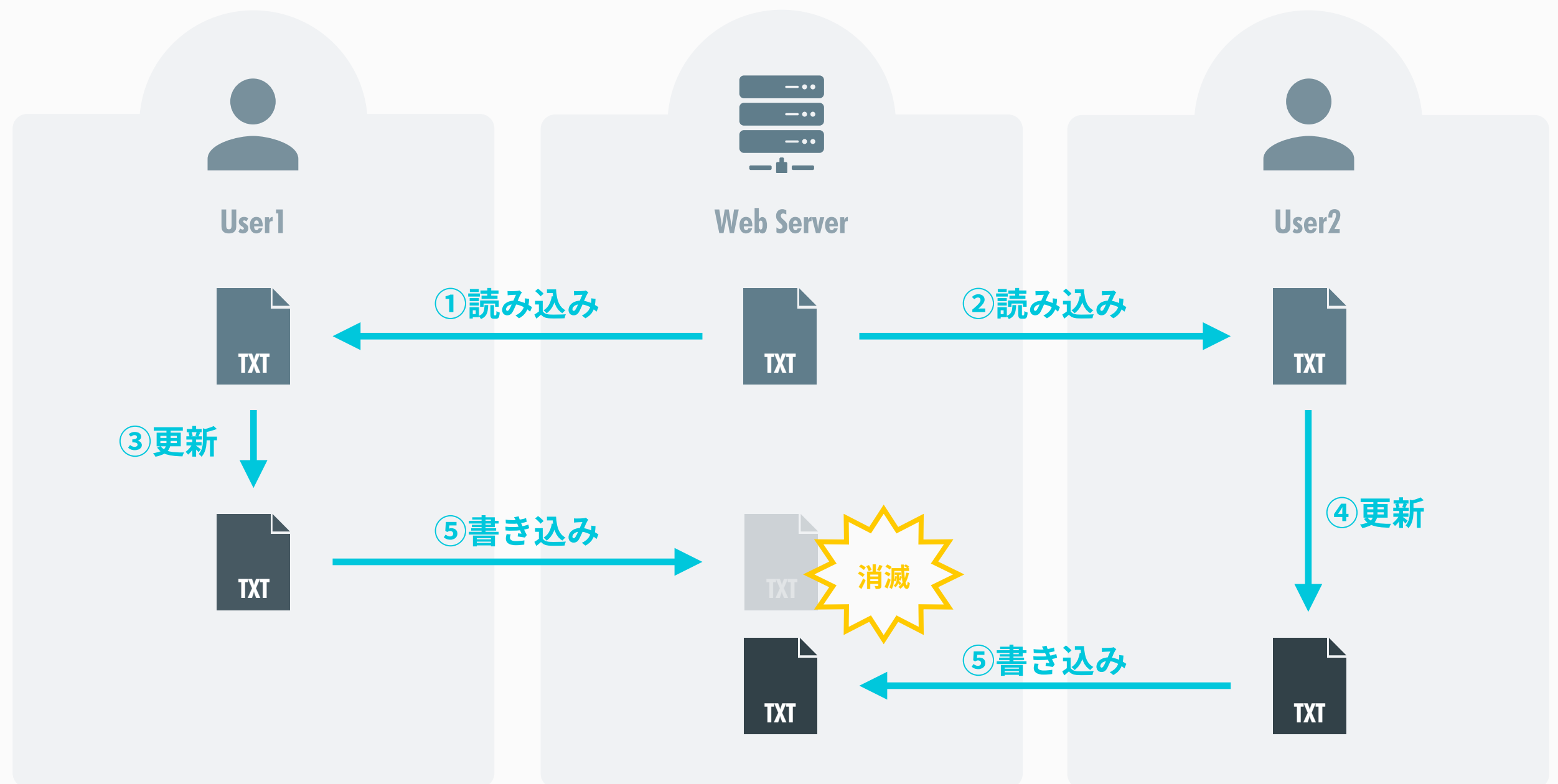
ファイルを開く際は以下のオープンモードを使い分ける

モード	読み／書き	ファイルが存在		ファイルポインタ
		しない	する	
r	読み込み専用	エラー	開く	先頭
r+	読み込み/書き込み可能	エラー	開く	先頭
w	書き込み専用	新規作成	中身を削して開く	先頭
w+	読み込み/書き込み可能	新規作成	中身を削して開く	先頭
a	書き込み専用	新規作成	開く	終端
a+	読み込み/書き込み可能	新規作成	開く	終端
x	書き込み専用	新規作成	エラー	先頭
x+	読み込み/書き込み可能	新規作成	エラー	先頭

バイナリファイルを扱う時は、モードの後ろに「b」を付ける。

ファイルをロックする

同じファイルを同じタイミングで書き込みされると、
正しく更新されない可能性がある。



ファイルをロックする

同じファイルを同じタイミングで書き込みされると、正しく更新されない可能性がある。

構文

`flock(ファイルハンドル, ロックモード)`

※引数はPHPマニュアル等を参照

Ex

PHP

```
// data.txtを読み込みモードで開く
$fp = fopen('data.txt', 'rb');
flock($fp, LOCK_SH);
```


ロックモード

以下のロックモードが利用できる

設定値	説明
LOCK_SH	共有ロック（他者による書き込みを禁止する） ロックされている場合は、ロック出来るまで待つ
LOCK_EX	排他的ロック（他者による読み書きを禁止する） ロックされている場合は、ロック出来るまで待つ
LOCK_UN	ロックの解除
LOCK_NB	ロックされている場合はfalseを返す（Windows環境非サポート） すでにロックされている場合は、あきらめる

LOCK_NBは単独では使わず、「LOCK_SH | LOCK_NB」などと使う。

ファイルの読み込み

ファイルの読み込みにはfgets関数を使います。

fgets関数はファイルポインタから1行分のデータを取得します

構文

省略可能

```
$取得したデータ = fgets(ファイルハンドル, 最大読み込みバイト数)
```

// 最大読み込みバイト数を省略すると行末まで読み込みます。

// 読み込むデータが無い場合 FALSE を返します。

Ex

PHP

```
$fp = fopen('data.txt', 'rb');
```

```
// データがある限り繰り返す。
```

```
while (($line = fgets($fp)) != false ) {
```

```
    echo htmlspecialchars($line, ENT_QUOTES, 'UTF-8'). '<br>';
```

```
}
```

ファイルポインタ

ファイルの現在読み書きしている位置を示す目印



ファイルを閉じる

ファイルを閉じるときは、fclose関数を用いる。

構文

fclose(ファイルハンドル)

※引数はPHPマニュアル等を参照

Ex

PHP

```
// ロックの解除
flock($fp, LOCK_UN);

// ファイルを閉じる
fclose($fp);
```

配列の逆順

配列の中身を逆順にするにはarray_reverse関数を用いる。

構文

```
array_reverse(逆順にしたい配列) //配列を逆順にする関数
```

Ex

PHP

```
$users = array('Suzuki', 'Tanaka', 'Yamada');
```

```
// 配列usersに 配列usersの中身を逆順にした結果を返す
```

```
$users = array_reverse($users);
```

ファイルの書き込み

ファイルの書き込みにはfwrite関数を使います。

構文

fwrite(ファイルハンドル, 書き込みたいデータ)

※引数はPHPマニュアル等を参照

Ex

PHP

```
// ファイルを書き込みモード開く  
$fp = fopen($fileName, 'wb');  
$msg = “こんなことがありました。 \n”;  
fwrite($fp, $msg);
```

改行コード

改行を表す特殊な文字。環境によって異なる

改行コード	特殊文字	説明
LF	¥n	ラインフィード Linux環境の改行コード
CR	¥r	キャリッジリターン Mac OS 環境の改行コード
CR + LF	¥r¥n	キャリッジリターン+ラインフィード Windows 環境の改行コード

CSV (comma separated value)

各項目を「,」カンマで区切ったテキストファイルの形式

id	name	mail
1	Tom	tom@example.com
2	Bob	<u>bob@example.com</u>
3	Stave	<u>stave@example.com</u>

Ex

CSV

```
id,name,mail
```

```
1,Tom,tom@example.com
```

```
2,Bob,bob@example.com
```

```
3,Stave,stave@example.com
```


CSVファイルの読み込み

カンマ区切りなどのテキストを読み込む際はfgetcsv関数を用いる。
指定した区切り文字で分割した値は、配列に格納される

構文

省略可能

```
$取得した配列 = fgetcsv(ファイルポインタ, 最大読込バイト数, 区切り文字)
```

```
// 区切り文字でフォルは「, (カンマ)」
```

※引数はPHPマニュアル等を参照

Ex

PHP

```
while ( ($rows = fgetcsv($fp) ) != FALSE) {  
    echo '<tr>';  
    foreach ($rows as $row) {  
        echo '<td>' . htmlspecialchars($row, ENT_QUOTES, 'UTF-8') . '</td>';  
    }  
    echo '</tr>';  
}
```

CSVファイルの書き込み

CSV形式の書き込みにはfputcsv関数が便利です。

構文

fputcsv(ファイルハンドル, 書き込む項目毎に分けた配列)

※引数はPHPマニュアル等を参照

Ex

PHP

```
$data = array('Tom', 'tom@example.com', '1234');  
// ファイルを書き込みモード開く  
$fp = fopen($fileName, 'wb');  
// CSV形式で配列dataの値を書き込む  
fputcsv($fp, $data); // 書き込み結果：Tom, tom@example.com, 1234  
  
fclose($fp);
```

Chapter 09

- SQLとは
- MySQLのログイン
- データベースの関連のSQL
- テーブル関連のSQL
- フィールド情報

データベースとは

検索や蓄積が容易にできるよう整理された情報の集まり。
データを格納する構造には様々なタイプがある。

階層型

ネットワーク型

リレーショナル

オブジェクト

カード型

最も普及している構造はリレーショナルデータベース

リレーショナルデータベースとは

表形式でデータを管理し、複数の表同士を関連付ける事ができる。

例：MySQL、PostgreSQL, SQL Server, Oracle



データベース

テーブル（商品マスタ）

ID	name	price	stock
0	製品A	500	3
1	製品B	1000	10
2	製品C	100	8

テーブル（顧客マスタ）

ID	name	email	pref
101	A社	a@a.jp	大阪
102	B社	b@b.jp	兵庫
103	C社	c@c.jp	京都

テーブル（受注管理）

ID	date	product	client
0	16/1/2	1	101
1	16/1/6	2	102
2	16/1/9	0	101

コマンドラインツール

SQLはコマンドラインツールを利用して操作する。

Windows

コマンドプロンプト

Mac

ターミナル

MySQLにログイン

cdコマンドを使って、mysqlファイルがあるディレクトリに移動し、mysqlコマンドでMySQLモニタにログインする。

■mysqlが「Cドライブ」→「xampp」→「mysql」→「bin」→「mysql」にある場合

Ex Linux

-- cd : ディレクトリの移動

```
cd C:\xampp\mysql\bin
```

■mysqlにログインするユーザ名が「root」の場合

Ex Linux

オプション

```
mysql -u root -p --default-character-set=utf8
```

ユーザ名 パスワード

文字コード

<https://dev.mysql.com/doc/refman/5.6/ja/mysql-command-options.html>

データベース関連のSQL

データベースの作成や削除などのコマンドは以下の通り。

SQL	説明
CREATE DATABASE データベース名	データベースの作成
SHOW DATABASES	データベースの一覧表示
USE データベース名	データベースの選択
DROP DATABASE データベース名	データベースの削除

テーブル関連のSQL

テーブルの作成や削除などのコマンドは以下の通り。

SQL	説明
CREATE TABLE テーブル名 (フィールド情報)	テーブルの作成
SHOW TABLES	テーブルの一覧表示
DESC テーブル名	テーブルのフィールド情報を表示
DROP TABLE テーブル名	テーブルの削除

データ型

MySQLで使える主なデータ型。

分類	書式	説明
整数	SMALLINT[（表示幅）][UNSIGNED][ZEROFULL]	2バイト整数 (-32,768 ～ 32,767)
	INT[（表示幅）][UNSIGNED][ZEROFULL]	4バイト整数 (約± 21億)
	BIGINT[（表示幅）][UNSIGNED][ZEROFULL]	8バイト整数 (約± 922京)
実数	DOUBLE[（合計桁数,小数点以下の桁数）] [UNSIGNED][ZEROFULL]	倍精度浮動小数点 (約小数第 15位まで有効)

UNSIGNED: マイナスの値を無効（整数型の場合、マイナス分がプラスに加算される）

ZEROFULL: 足りない桁数分 0 を先頭に埋める

データ型

MySQLで使える主なデータ型。

分類	書式	説明
文字列	CHAR[（最大文字数）][BINARY]	固定長文字列 （最大長 255バイト）
	VARCHAR[（最大文字数）][BINARY]	可変長文字列 （最大長 65535バイト）
	LONGTEXT	可変長文字列 （最大長 4294967295バイト）

BINARY: 大文字と小文字を区別する

データ型

MySQLで使える主なデータ型。

分類	書式	説明
日付	DATE	日付 (yyyy-mm-dd)
	TIME	時間 (hh:mm:ss)
	DATETIME	日時 (yyyy-mm-dd hh:mm:ss)

データ型

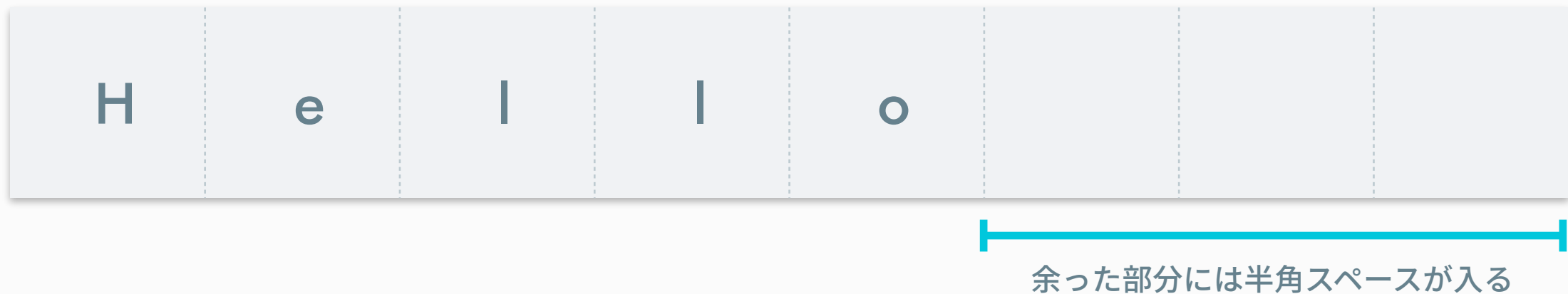
MySQLで使える主なデータ型。

分類	書式	説明
真偽	BOOL	真偽値 (true/false)
定数	ENUM(値1, 値2, 値3...)	列挙型 (最大 65535個の固有値)
真偽	BOOL	真偽値 (true/false)

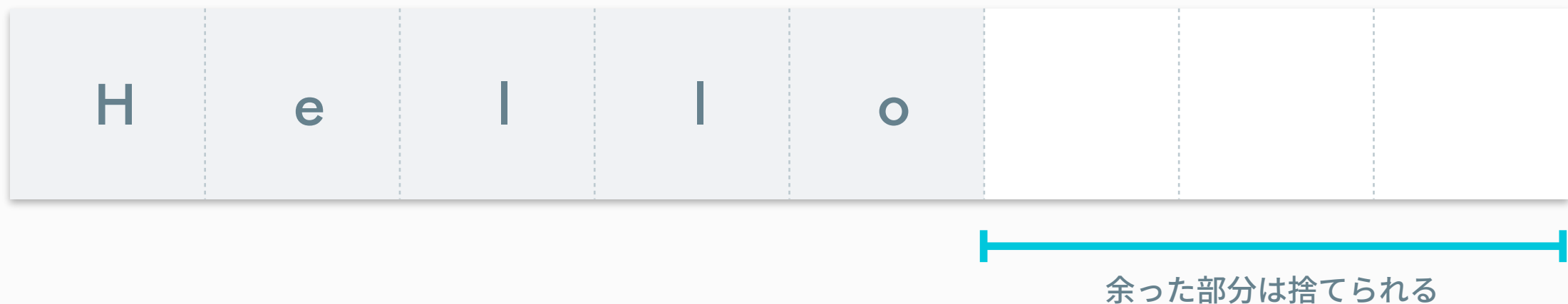
charとvarcharの違い

charは固定長文字、varcharは可変長文字。

■char(8)の場合



■varchar(8)の場合



charは電話番号や郵便番号など**文字数が決まっているもの**に使う。

STEP 10

- テーブルの編集
- レコードの挿入
- レコードの更新
- レコードの削除

テーブルの編集

テーブルのフィールド情報の編集はALTER TABLEコマンド。

分類	書式	説明
追加	ADD [COLUMN] フィールド情報 [追加場所]	フィールドの追加 [追加場所] <ul style="list-style-type: none">• 指定なし： 末尾• FIRST： 先頭• AFTER フィールド名： 指定フィールドの後
	ADD PRIMARY KEY フィールド名	主キーの追加
	ADD UNIQUE フィールド名	ユニークインデックスの追加
削除	DROP [COLUMN] フィールド名	フィールドの削除
	DROP PRIMARY KEY	主キーの削除
変更	CHANGE [COLUMN] 元フィールド名 新フィールド情報	フィールド情報の変更
	MODIFY [COLUMN] フィールド名 データ型 [制約]	データ型、制約の変更

レコードの挿入

レコードの挿入はINSERT文を用いる

構文

```
INSERT [INTO] テーブル名 (  
    フィールド名1, フィールド名2, フィールド名3...  
)  
  
VALUES (  
    値1, 値2, 値3...  
);
```

[～]：省略可

フィールド名：全フィールドに値を挿入する時は「()」ごと省略可能

レコードの更新

レコードの更新はUPDATE文を用いる

構文

UPDATE テーブル名

SET フィールド名1 = 値1, フィールド名2 = 値2, フィールド名3 = 値3..

[WHERE 条件式]

[ORDER BY フィールド名 並び順]

[LIMIT 最大更新数]

;

[～]：省略可

WHERE：省略すると全てのレコードが更新される

ORDER BY：並び順…DESC（降順）/ASC（昇順）

LIMIT：条件に一致したレコードの更新数の上限を指定できる

レコードの削除

レコードの削除はDELETE文を用いる

構文

DELETE FROM テーブル名

[WHERE 条件式]

[ORDER BY フィールド名 並び順]

[LIMIT 最大更新数]

;

[～]：省略可

WHERE：省略すると全てのレコードが削除される

ORDER BY：並び順… DESC（降順） / ASC（昇順）

LIMIT：条件に一致したレコードの削除数の上限を指定できる

STEP 11

- レコードの抽出

レコードの抽出

レコードの抽出はSELECT文を用いる

構文

```
SELECT 表示するフィールド名 FROM テーブル名  
[WHERE 条件式]  
[ORDER BY フィールド名 並び順]  
[LIMIT [開始位置,] 上限数];
```

[～]：省略可能

表示するフィールド名：「, (半角カンマ)」区切りで複数指定できる。

「* (アスタリスク)」を指定すると全てフィールドが表示される。

並び順：DESC (降順) / ASC (昇順) なお、「, (半角カンマ)」区切りで並び順を複数指定できる

STEP 12

- 集約関数

集約関数

レコードを集約することができる

関数	説明
COUNT(*)	レコード数を取得する
SUM(フィールド名)	指定したフィールドの合計値を取得する
AVG(フィールド名)	指定したフィールドの平均値を取得する
MAX(フィールド名)	指定したフィールドの最大値を取得する
MIN(フィールド名)	指定したフィールドの最小値を取得する

SNS

- データベースへの接続
- データベースとの接続を閉じる
- データベースの文字コードの設定
- SQLの実行
- プリペアドステートメント

データベースへの接続

データベースへはmysqli_connect関数で接続する

構文

```
$ハンドル = mysqli_connect(ホスト名, ユーザ名, パスワード, データベース名)
```

Ex

PHP

```
// 定数を定義
```

```
define('DB_HOST','localhost');  
define('DB_USERNAME','root');  
define('DB_PASSWORD','root');  
define('DB_NAME','db_sample');
```

```
// データベースに接続
```

```
$db = mysqli_connect(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_NAME);
```

データベースとの接続を閉じる

データベースはmysqli_close関数で接続を終了する

構文

```
mysqli_close(ハンドル)
```

Ex

PHP

```
// データベースに接続
$db = mysqli_connect(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_NAME);
// データベースを閉じる
mysqli_close($db);
```

データベースへの文字コードの設定

データベースの文字コードはmysqli_set_charset関数を使う

構文

```
mysqli_set_charset(ハンドル, 文字コード);
```

Ex

PHP

```
// データベースに接続
$db = mysqli_connect(DB_HOST, DB_USERNAME, DB_PASSWORD, DB_NAME);

// データベースで使う文字コードをUTF8に変更
mysqli_set_charset($db, 'utf8');
```

SQLの実行

SQLを実行するだけならばmysqli_query関数を使う

構文

```
mysqli_query(ハンドル, SQL);
```

Ex

PHP

```
// SQLクエリを変数に代入
$sql = 'insert into sample_table values(
    1, 'Tom', 'tom@tom.com'
)';

// SQLクエリを実行
mysqli_query($db, $sql);
```

結果の行を連想配列で取得する

結果の行はmysqli_fetch_assoc関数を使って連想配列で取得する。

構文

```
mysqli_fetch_assoc(結果セット);
```

Ex

PHP

```
// SQLクエリを変数に代入
$sql = 'select * from user_table';
// SQLクエリを実行
$result = mysqli_query($db, $sql);
// 結果の行を$rowに連想配列に取得
while ($row = mysqli_fetch_assoc($result)) {
    echo htmlspecialchars($row['name'], ENT_QUOTES);
}
```

プリペアドステートメント

SQLで変数などを使うときはプリペアドステートメントで！

構文

```
$ステートメント = mysqli_prepare(ハンドル, SQL文);
```

Ex

PHP

```
$name = 'Tom';  
// 挿入する  
$stmt = "insert into users(name) values( ? )";  
// ステートメントにパラメータをセット  
mysqli_stmt_bind_param($stmt, 's', $name );  
// クエリを実行  
mysqli_stmt_execute($stmt);
```

SNS

- POST送信されたかを判別する方法
- 日付のフォーマット
- 端数の切り上げ
- おまけ

POST送信されたかを判断する方法

現在のアクセスがPOST送信されたかどうかを判断するには
スーパーグローバル変数\$_SERVERのREQUEST_METHODで確認できる。

構文

\$_SERVER[キー] //サーバ情報のスーパーグローバル変数

//キー：リクエストのメソッド（POST／GET）を判別するには、「REQUEST_METHOD」を指定

※キーはPHPマニュアル等を参照

Ex

PHP

```
// POST送信されたアクセスかを判別
if ( $_SERVER[ 'REQUEST_METHOD' ] == 'POST' ) {
    echo 'POSTによるアクセスです。';
}
```


タイムゾーンを設定する

date関数などで取得した現在時刻がおかしい場合は、
PHP.iniファイルかスクリプトでタイムゾーンの設定を変更する

構文

```
date_default_timezone_set( タイムゾーン )    //タイムゾーンを設定する関数
```

Ex

PHP

```
// デフォルトのタイムゾーンを東京に設定
```

```
date_default_timezone_set( 'Asia/Tokyo' );
```

日付を書式化する

日付や時刻を書式化するにはdate関数を用いる

構文

省略可能

date(フォーマット, タイムスタンプ) //タイムゾーンを設定する関数

//タイムスタンプを省略すると現在の時刻が使われる

※引数はPHPマニュアル等を参照

Ex

PHP

```
// 現在の時刻を西暦4桁-月2桁-日2桁 時間2桁:分2桁:秒2桁で表示
```

```
echo date('Y-m-d H:i:s');
```

主なフォーマット

date関数などで利用する主なフォーマットは以下の通りです。

フォーマット		内容
年	Y	4桁の年。 Ex: 2016
	y	2桁の年。 Ex: 16
月	m	ゼロ詰めの月。 01 ~ 12
	n	ゼロなしの月。 1 ~ 12
	F	フルスペルの月。 Ex: January
	M	3文字形式の月。 Ex: Jan
日	d	ゼロ詰めの日。 01 ~ 31
	j	ゼロなしの日。 1 ~ 31
	z	年間の通算日。 0 ~ 365

フォーマット		内容
時	g	12時間単位の時（ゼロなし）。 1 ~ 12
	G	24時間単位の時（ゼロなし）。 0 ~ 23
	h	12時間単位の時。 01 ~ 12
	H	24時間単位の時。 00 ~ 23
分	i	ゼロ詰めの分。 00 ~ 59
秒	s	ゼロ詰めの秒。 00 ~ 59
曜日	l	フルスペルの曜日。 Ex: Monday
	D	3文字形式の曜日。 Ex: Mon
	w	数値の曜日。 0（日） ~ 6（土）

端数の切り上げ

端数を切り上げるにはceil関数を用いる。

構文

```
ceil(繰り上げたい値)           //端数を切り上げる関数
```

Ex

PHP

```
$int = 3.14;  
// 変数intの端数を切り上げて表示  
echo ceil($int);    // 4
```

ヒアドキュメント

指定した文字が出てくるまでの入力を全て文字列とする。
改行を含むような長い文字列を表すのに適している。

構文

\$変数 = <<<**任意の文字** **ここからヒアドキュメント**

変数に格納したい複数行の文字列を記述。

変数も展開される。

「”」や「'」も使える。

任意の文字; **ここまでがヒアドキュメント**

ヒアドキュメント

指定した文字が出てくるまでの入力を全て文字列とする。
改行を含むような長い文字列を表すのに適している。

Ex

PHP

```
$msg = <<<EOD
```

```
Hello PHP!
```

```
I'm Fine!
```

```
EOD;
```

```
echo $msg;
```

文字列の置換

文字列に含まれる特定の文字列を別の文字列に置換する。

構文

省略可能

```
str_replace(置換したい文字, 置換後の文字, 対象の文字, 置換した数を格納する変数)
```

※引数はPHPマニュアル等を参照

Ex

PHP

```
$str = 'Hello PHP';
```

```
// 文字列「PHP」を「World」に置き換えて画面に出力する
```

```
echo str_replace('PHP', 'World', $str);
```

文字列を部分的に取得

文字列から部分的に文字列を取り出す。

構文

省略可能

`mb_substr(対象の文字列, 取得開始位置, 取り出す文字数, 文字コード)`

※引数はPHPマニュアル等を参照

Ex

PHP

```
      0 1 2 3 4 5 6 7 8
$str = 'Hello PHP';

// 変数strの3文字目から3文字を画面に表示
echo mb_substr($str, 2, 3);
```