Webデザイン実習3B

2019/9/02

Kazuma Sekiguchi
class@cieds.jp

JS課題講評

- HTMLタグの間違い（とも言えないが）が多い
  - mainタグは記述する
  - タグへのstyle属性記述はできるだけ避ける
  - section,articleタグには中に見出しタグが必要
  - span,a,imgはブロックレベル要素内に記述する（pタグとかliタグとか）
  - id属性の利用を避ける（ページ内リンクくらいにしか使わない）
  - 読み込みはCSSを先に、JSは後に。できれば、JSは</body>の直前に読み込むのがベター
  - pタグを3連続以上使わない。文章であれば、
などで改行を。3連続以上出るのであれば、普通は箇条書きになる
  - WebFont用などのCSSを読み込む場合は、自分のCSSよりも先に読み込む（自分のCSSを先に記述すると結果として表示されないこともある）
  - headerタグはbodyタグの中に記述する
  - グローバルナビゲーションにはnavタグを利用して記述
  - リンクすべてがnavタグではないので、navタグはあくまでもグローバルナビゲーションのときにだけ利用する（数カ所出てきたらおかしい）

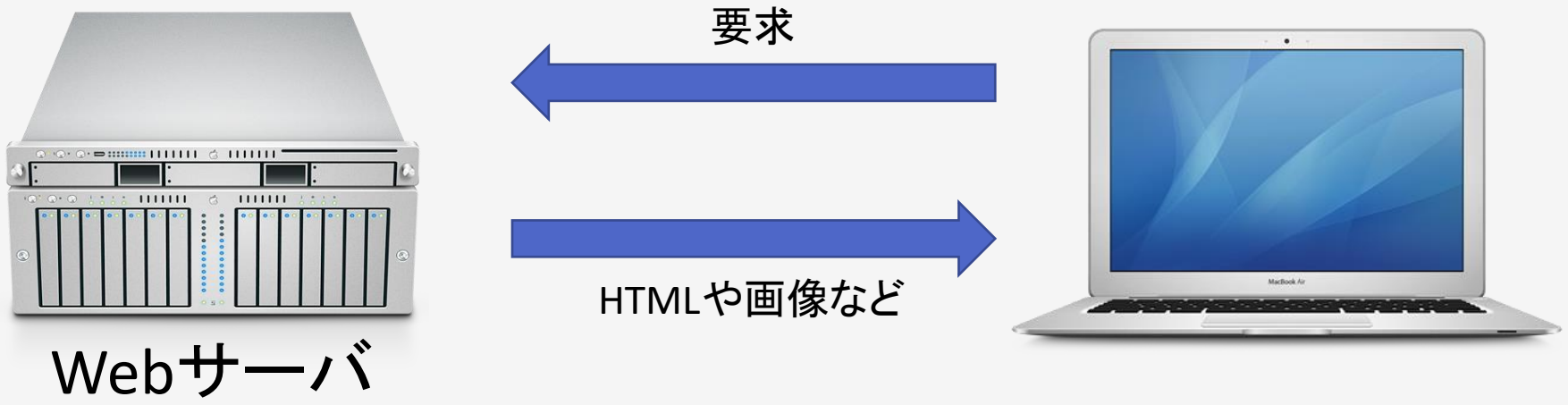
JS課題講評

- 画像として小さいサイズのものは使わない
  - 大きいものを小さくするのは画質低下しないが逆は低下する
- jQueryの初期化を1つにする
  - $(function(){とか

こんな場合どうするか

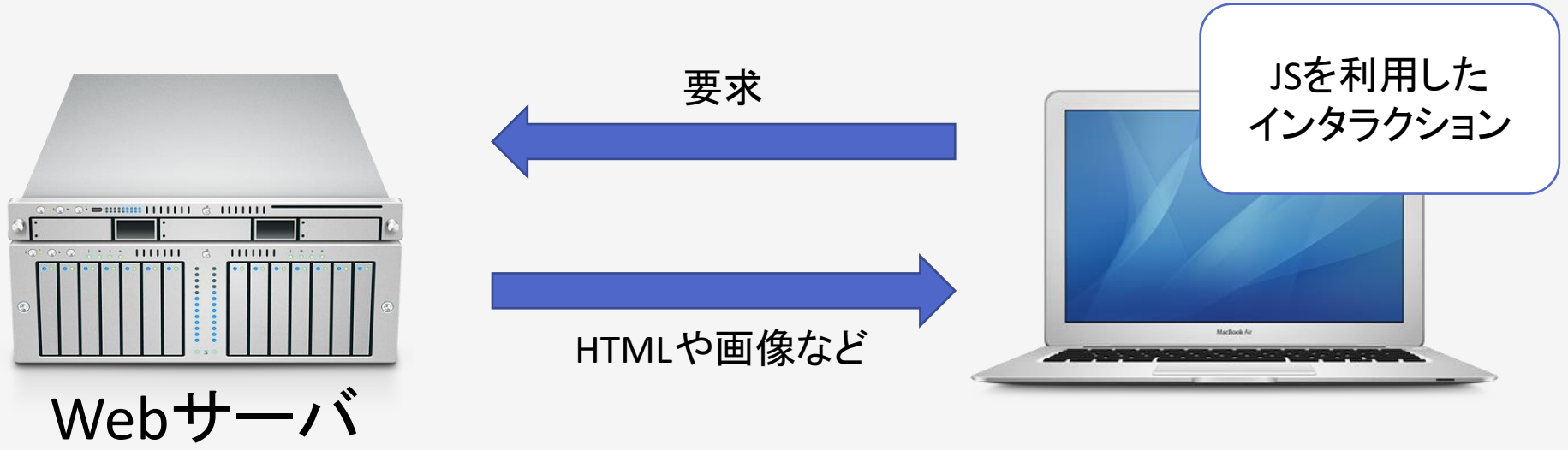
- フォームで入力してもらい確認画面を表示
  - ユーザが入力するものはマチマチなためあらかじめ作成しておくことは不可能
- ページ数が膨大なサイト
  - 表示のフォーマットが決まっているなら、動的に生成して出力してあげた方が効率的
- 検索ページ
  - どんな言葉が検索されるか分からない

サーバサイドとクライアントサイド



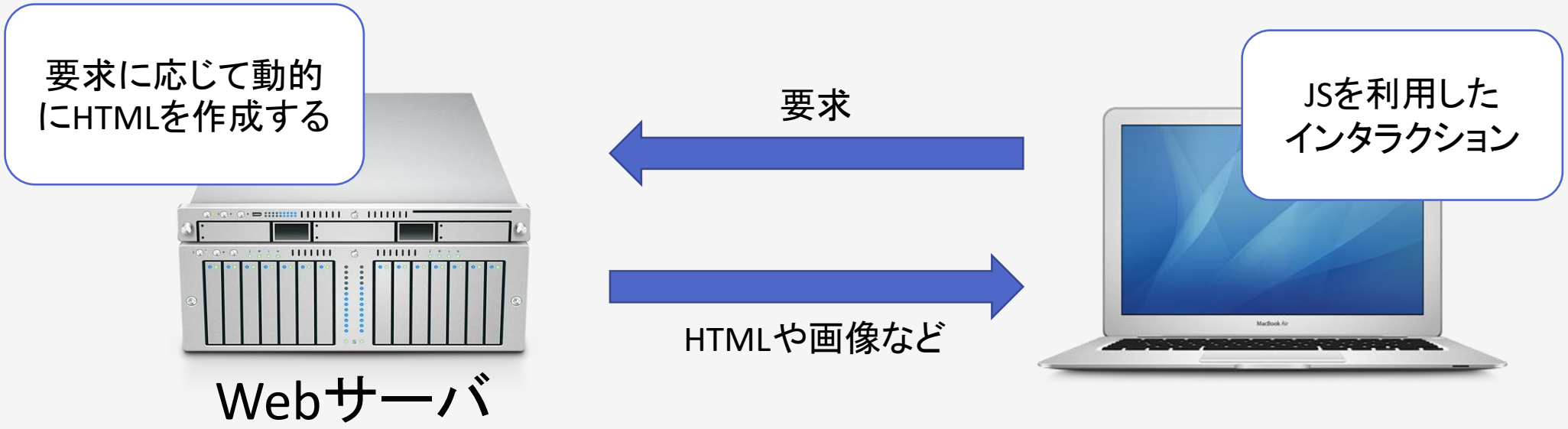
- URLを入力してアクセス
  - WebサーバというPCが要求に応じてファイルをクライアントに送り出す
  - （ファイルを送り出す）サービスをする＝サーバー

サーバサイドとクライアントサイド



- ファイルにJSが合った場合、クライアント側で、JSが実行される
  - クライアント側で実行される＝クライアントサイドプログラム

サーバサイドとクライアントサイド



- 要求に応じてHTMLを生成する
  - HTMLを生成→何らかのプログラミング言語を用いる
  - サーバー側でHTMLを生成する＝サーバーサイド
    - 生成するのはHTMLだけとは限らない

動的と静的

- 動的
  - アクセスがあるごとにHTMLをプログラムに基づいて作成し、クライアントに対して送信する
- 静的
  - あらかじめPCなどでHTMLを作成し、FTPなどでアップロードをしておき、クライアントのアクセスがあったら、作成済みのHTMLを送信する

サーバサイドとクライアントサイド

さまざまなデータを保存しておく

DBサーバ

MySQL、PostgreSQL、SQLite、Oracle、SQL Serverなど

Webサーバ

Apache、IIS、nginxなど

PHP , Python , Ruby
Perl , JavaScript
C , Java , Go , Scala

要求

HTMLや画像など

JavaScript

動的生成

- ユーザの操作や動きに合わせてその場その場でHTMLを生成する
  - 検索結果画面などを用意しておく必要がない
  - リアルタイムに結果を反映可能
- プログラミング言語を用いて作成する
- 表示するデータは予めデータベースに保存しておく
  - そこから必要なデータを取り出してくる

JSとの比較

	JavaScript	PHP
実行タイミング	ユーザーイベント連動	読み込み時に実行
動作	ブラウザー、Nodeなど	PHP上で実行
変数	var , let , constを付けて宣言	$を付ける。宣言は不要
データ保存	localstorage , IndexedDb	何らかのサーバーDB、ファイル
ユーザーによる改変	可能	不可能

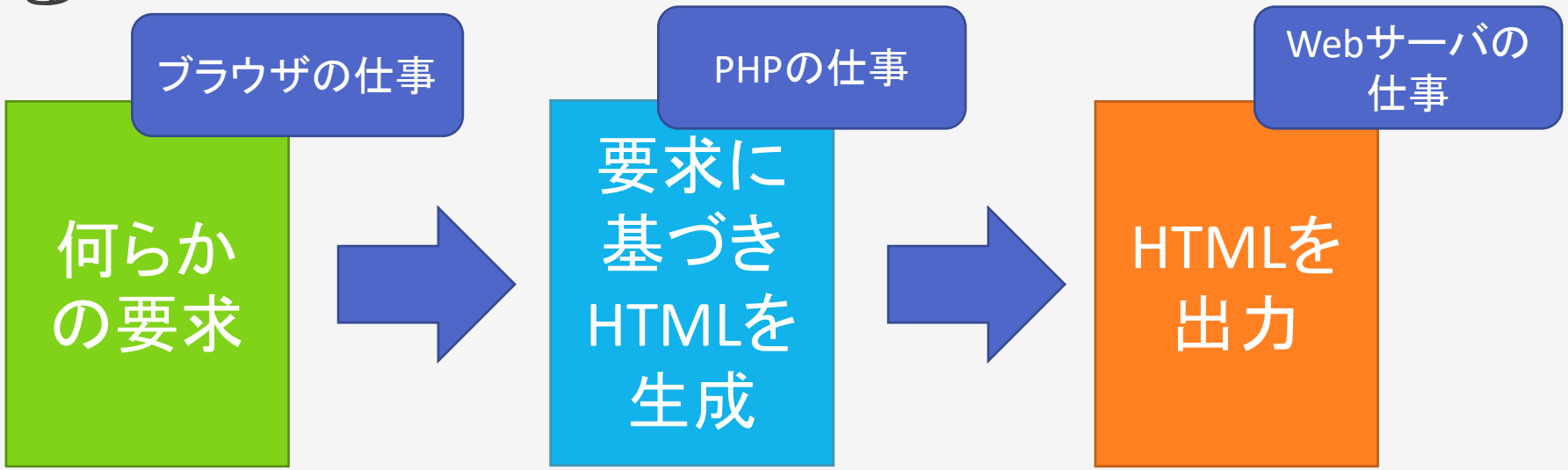
HTMLとPHP

- PHPはHTMLを生成可能
  - PHP構文内で無ければ、PHPファイル内に通常通りHTMLタグを記述していけばOK
- PHP構文内部で利用する場合は、print()関数などを用いて、シングルクォートで括ればそのまま出力される
- PHPを利用するためには、実行環境としてPHP（というソフト）が必要
  - 通常はWebサーバと組み合わせて利用する

```
<h1>PHPでのHTML</h1>
<?php
        print('<h2>ここに書くなら、print()関数を利用する');
?>
```

PHPの利用

- 通常はユーザから送られたデータを元にしてHTMLを動的に生成する



- データが送られてこない場合、静的なHTMLでもOK
  - データに応じて変化させる必要がある場合、PHPなどを利用する

Use PHP

How to make program with PHP

Webサーバ環境構築

- PHPとかMySQLを使うためには環境が必要
  - レンタルサーバで借りるのが手っ取り早いがテストなどもしないでそのままアップロードは危険
    - セキュリティホールになる可能性がある
  - ローカル環境でテストできるようにしておくことは必要
    - 速度の検証はできない

一番多い構成

- Linux（またはBSD）：OS
- Apache：Webサーバ
- PHP：アプリケーション実行環境
- MySQL：データベース

- 大体のレンタルサーバなどもこういう構成
- 採用例も非常に多いため、情報が多い
  - 安定度も非常に高い
  - ソフトが全て無料で手に入る
  - ある程度古いマシンでも動作する

テスト環境としてお薦め

- XAMPP（ザンプ）などの一括でインストールしてくれるソフトを利用する
  - WindowsなどにApache，PHP ,MySQLをインストールすることは可能
  - ただしかなり面倒（設定が）
  - しかも上手く動かない率が高い

- ローカル環境でウェブ上と同じWebサーバーを動かすことが可能
  - ほとんど無設定で可能

Apacheなどの設定

- Linuxなどで動作するアプリは設定を
  テキストファイルで行う
  - テキストファイルに所定の書式で記述することで設定完了
  - GUIでも設定できる場合があるが、基本はエディタで設定をしていく
  - 設定後は保存をして、ソフトを再起動する
    - 上手く起動しなかったら、ログを見て、エラーを確認する
    - ほとんどが書式エラーか文法エラー
    - 再起動するまで設定は反映されないので注意

サーバー経由での確認

- Xamppなどから確認するためには、Xamppフォルダー内にあるhtdocsフォルダー内に英数字でフォルダーを作り格納する
- 確認時にはブラウザーでhttp://127.0.0.1/フォルダー名を入力しアクセスすればOK
  - フォルダーに格納したファイルをダブルクリックして開いてもサーバーを通していないので注意
  - PHPプログラムはサーバーを通さないと実行されない
    - そのままソースが出てくるだけ

MAMP

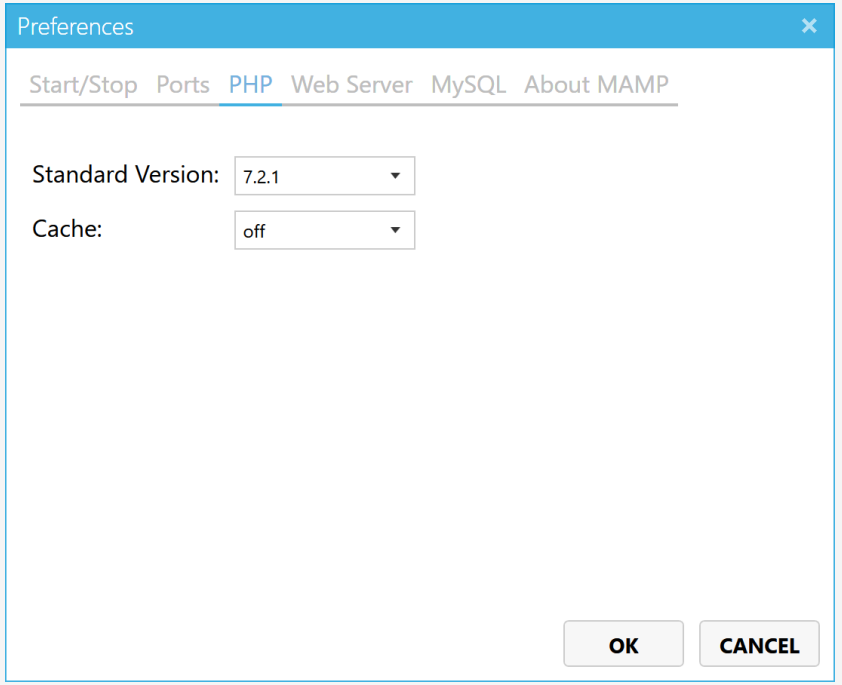
開発環境の設定（Windows,Mac）

設定変更（Macの場合）

- メニューバーからPreferencesを選択
- Portsを選択し、「Set Web & MySQL ports to 80 & 3306」をクリック
- 「Start Servers」をクリックし、Apache ServerとMySQL Serverの横に緑がともればOK

設定変更

- Macの場合
  「PHP」をクリックし、
  「Standard Version」で7.1.8を選択、「Cache」で「off」が選択されていることを確認してOKをクリック



php.iniの設定(Macの場合)

472行目付近

display_errors = Off → display_errors = On

665行目付近

post_max_size = 8M → post_max_size = 32M

データ量として受け入れる最大サイズ。ファイルサイズでは無く、パケットの総合計サイズになる

910行目付近

date.timezone = “Europe/Berlin” → date.timezone = Asia/Tokyo

時間を出すときに利用される。日本であれば、GMT+9なので、タイムゾーンを変えないと変な時間が使われる

php.iniの設定（Macの場合）

1597行目付近以下

文字コードの設定。変更しないと文字化けする

;mbstring.language = Japanese	mbstring.language = Japanese
;mbstring.internal_encoding = EUC-JP	mbstring.internal_encoding = UTF-8
;mbstring.http_input = auto	mbstring.http_input = pass
;mbstring.http_output = SJIS	mbstring.http_output = pass
;mbstring.encoding_translation = Off	mbstring.encoding_translation = Off
;mbstring.detect_order = auto	mbstring.detect_order = UTF-8,SJIS,EUC-JP,JIS,ASCII

php.iniの設定（Macの場合）

- 変更をしたら上書きして保存する
- ファイルを保存したらMAMPのパネルから「Stop Servers」を押してサーバーを停止
- 「Start Servers」を押して再度サーバーを起動
  - 問題なく起動すればOK
  - 起動しない場合、設定をどこか間違えている可能性があるため、再度確認する（セミコロンを外す、残しておくとか、スペルなど）

PHP

PHP Programming

PHP（1）

- PHP Hypertext Preprocessorの略
- HTML生成に特化したプログラミング言語
  - 作成したプログラムはほとんどブラウザで確認する
- ほとんどのサーバOS上で動作する
- 比較的簡単に書ける
- データベースとの連携が簡単
- 採用例が多い

PHP（2）

- デメリット
  - セキュリティが脆弱になりやすい
    - プログラミングの際に注意が必要
  - 関数の命名規則が統一されていない
    - 似たような関数が全然違う名前だったりする
  - ほかにもいろいろと批判も多い

PHP（3）

- 変数（配列を含む）
- 関数
- 制御
- クラス

の4つから構成される

書式

- <?phpから始まり?>で終わる
  - HTMLで言う<body>タグと同じ
- 空の行などは一切意味がない
  - 見やすいように適宜改行を行って良い
    - スクリプトによっては、ファイルサイズを小さくするため、改行、スペースを削除する場合もある
- 1行で一つの動きを書く
- 「；」で行の終わりを示す

```
<?php
    print('PHPで文字を表示します');
?>
```

変数

- 変数名には「＄」をつける

```
$X = 5;
```

  - この中のXが変数。Xには５が入る
  - 変数は自由に値を格納できる箱みたいなもの

- 変数名は自由に設定できる
  - 但し、予め別の役割が決められているものは使用不可（予約語）
  - 変数名の最初は英字で始まること。数字は文頭には使えない
  - 「_（アンダーバー）」から始まる変数は特殊な用途に利用される

変数（2）

- 変数は宣言せずに使用可能
  - 初期化できる（現在は初期化をすることが推奨）

```
$variables = “”;
```

- 型は一切なし
  - $txt ＝ “” ;とすれば文字列型
  - $txt ＝ 0;とすれば数値

文字列連結

- 変数同士をくっつける
- 「.（ドット）」を使う

```
$txt_first = “PHP”;
$txt_second = “Program”;
$moji = $txt_first . $txt_second;
```

  - $mojiにはPHPProgramが入る
  - 「.」を使う際には「” .” 」とする

制御

- スクリプトの流れを制御するもの
  - if文を使用
  - elseifのみ空白が無いので注意

```
if(x > 3){
    print(“Xは3よりデカイ”);
}
elseif(x > 2){
    print(“Xは2よりデカイ”);
}
else{
    print(“Xは小さい”);
}
```

コメント

- スクリプトの実行時には無視される
- プログラムの動きなどを記述しておき、後から見直しやすいようにする

```
//1行だけコメント

/*この挟まれた間がコメント*/
```

関数

- 多数の関数が存在
- 引数を与えることで任意の動作が可能
- PHPの場合、関数が大量にあるため、上手く組み合わせる

```
<?php
print(”今の時刻を10回表示します”);
for($i = 0; $i < 10; $i++){
    print(time());
}
?>
```

PHPの記述

- 単一のファイルに記述して保存
  - ダブルクリックしたら動作するようなものではないので注意
- 拡張子はphpとする
  - アクセスされるときもhttp://example.jp/example.phpとしてアクセスされる
- サーバ上で動作するため、サーバにFTPなどでアップして実際にアクセスして動作を確認する
  - ローカルで開発環境を作成して確認しておく
- 文字コードはUTF-8で作成する
  - Shift-JISでは作らないこと

サーバ上のPHP環境情報

PHP Version 5.1.6

System	Linux serverlx01 2.6.18-92.1.10.el5 #1 SMP Tue Aug 5 07:41:53 EDT 2008 i686
Build Date	Jul 16 2008 19:54:37
Configure Command	'./configure' '--build=i686-redhat-linux-gnu' '--host=i686-redhat-linux-gnu' '--target=i386-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--libdir=lib' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pear' '--with-bz2' '--with-layout=GNU' '--enable-exif' '--enable-ftp' '--enable-magic-quotes' '--enable-sockets' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--enable-sqlite' '--with-libxml-dir=/usr' '--with-xml' '--with-system-tzdata' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--without-odbc' '--dis
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
additional .ini files parsed	/etc/php.d/dbase.ini, /etc/php.d/dom.ini, /etc/php.d/eaccelerator.ini, /etc/php.d/gd.ini, /etc/php.d/ldap.ini, /etc/php.d/mbstring.ini, /etc/php.d/mcrypt.ini, /etc/
PHP API	20041225
PHP Extension	20050922
Zend Extension	220051025
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.1.0, Copyright (c) 1998-2006 Zend Technologies
with eAccelerator v0.9.5.2, Copyright (c) 2004-2006 eAccelerator, by eAccelerator

Powered By Zend Engine 2

- サーバで動いているPHPの情報を入手

```
<?php
    phpinfo();
?>
```

- 適当な名前で保存して実行する

ユーザからデータをもらう

- 正確には、ユーザがデータを送ってくる
  方法は2つ
  - GET送信、POST送信
- GET送信
  - URLの後ろに「?」を付けて特定のパラメータを送出する方法
    - 複数を送出する場合、2つ目以降のパラメータは「&」で繋ぐ
  - 手軽であるが、文字数制限がある（最近のブラウザには無い）

https://www.google.co.jp/search?q=google&ie=utf-8&oe=utf-8

ユーザからデータをもらう

- POST送信
  - 通常フォームで送信するときに使われるタイプの送信方法
  - URLにパラメータは表示されない
    - GET送信と組み合わせることもできるため、組み合わせた場合は、GET送信のパラメータは表示される
  - ファイル（バイナリデータ）を送信可能

フォームの形式

- ファイルをフォームからアップロードしてもらう場合、enctypeを変更する必要がある
  - 変更しないとファイルがアップロードされないので注意

```
<form action="confirm.php" method="post" enctype="multipart/form-data" name="form1" id="form1">
```

サニタイズ

- サニタイズ＝無毒化
- ユーザが送ってくるデータが安全なデータとは限らない
  - パスワードを表示するような命令を送る
  - JavaScriptで永遠にウィンドウを開く命令を記述する
  - などなど
- 送信されてきたデータが正しい形式（想定通り）かを検証する必要がある
  - そのまま利用しないこと
  - 基本的に送れてきたデータは信用しない（性悪説に基づく）

サニタイズ処理

- 数値かどうか判別する（強制的に数値にする）
  - 数値で判別することは多いため
  - intval()関数で数値に変換可能
- HTMLタグをタグとして利用できなくする
  - JavaScriptを動作させなくする
  - htmlspecialchars()関数で<と>を<や>に置き換える
- SQLインジェクション
  - DBに関係するが、DBのデータを不正に取得できないようにする（重要！）

データの受け取り

- PHPの場合、ユーザから送信されてくるデータは全て特殊な変数に格納される（このときの変数名は大文字！！）
  - POST送信：$_POST
  - GET送信：$_GET
- 配列として格納されるため、使う場合は、$_GET[“パラメータ名”]、$_POST[“パラメータ名”]で取得する
- 必要なデータだけ取得するようにする
  - 全部を取得することも可能だが、セキュリティ的にまずくなる

データの受け取り

- 受け取ったデータは変数に格納する
  - 格納時にサニタイズ処理を行っておく
  - 変数とパラメータ名は一致している必要は無い
  - htmlspecialchars関数はHTMLタグを単なる文字列に変換する関数

```
<?php
    $id = intval($_GET[“id”]);
    $age = intval($_POST[“age”]);
    $name = htmlspecialchars($_POST[“name”], ENT_QUOTES,”UTF-8“);
    $comment = htmlspecialchars($_POST[“comment”], ENT_QUOTES,”UTF-
    8“);
?>
```

フォーム

- HTMLで作成されるユーザが入力できる唯一の画面
- 主要なタグ
  - <form>：フォームの開始を示す
  - <input>：ほとんどのフォーム部品
  - <textarea>：長いテキストを入力して貰うときの部品
  - <label>：部品の説明（ex：氏名など）

- ほとんどはinputタグのtype属性で変化させる
- name属性に固有の値を設定しておく
- 送信先はformのaction属性で指定する

最小限のフォーム

```
<form action=“mail.php” name=“form1” method=“post”>
<input type=“text” name=“name”>
<input type=“submit”>
</form>
```

- formのmethod属性：postかgetを選択。通常はpost
- action属性：データの送信先を指定
- input type=“text”：テキストフィールドの作成
- input type=“submit”：送信ボタンの作成
  - クリックすることで、テキストフィールドに入力されたデータが送信される

フォームの注意

- 必須などのチェックを行う機能は無い
  - HTML5から必須チェックを行うことが可能（ブラウザ依存）
  - JSでチェックを行う
- 正しいデータが入っているかは調べられない
  - 形式に沿っているかどうかはJSで判別可能
    - 電話番号、メールアドレスなど
- 送信ボタンを押すと、データの送信先に遷移する
  - 確認画面などはデータの送信先のプログラム上で作成する必要あり
  - JSを利用して、データだけを送ることも可能（遷移無し）

フォームの部品

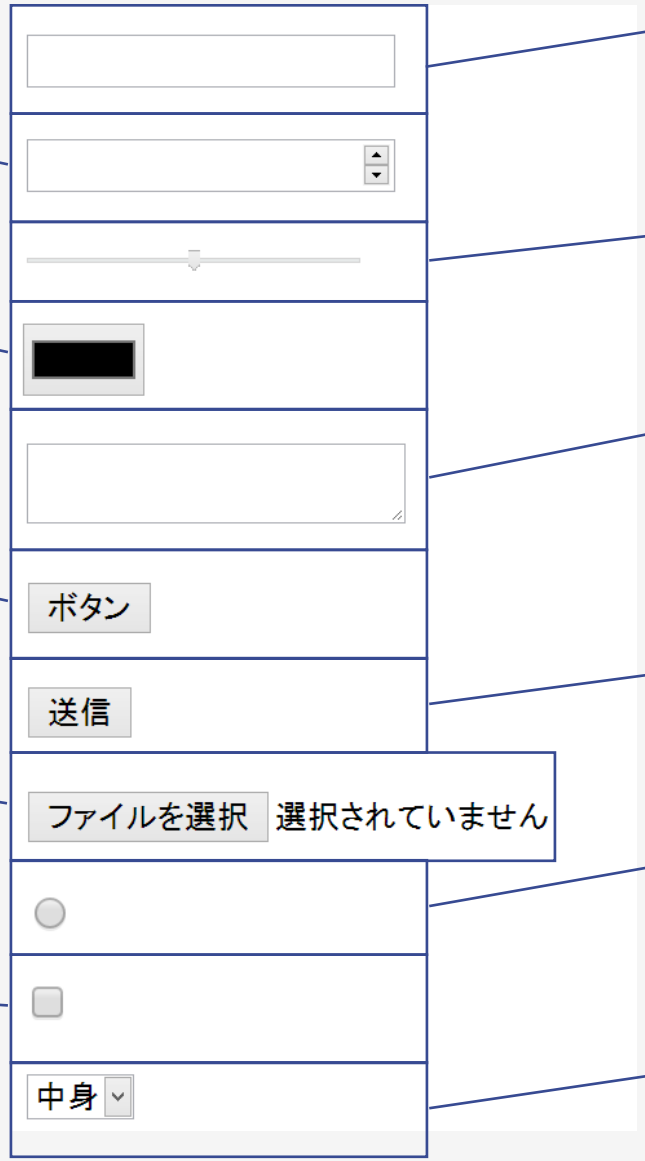
数値フィールド(HTML5)
<input type="number">

色選択(HTML5)
<input type="color">

ボタン
<input type="button">

ファイル
<input type="file">
保存されているファイルを
アップロードする場合に使用

チェックボックス
<input type="checkbox">



テキストフィールド
<input type="text">

範囲指定(HTML5)
<input type="range">

テキストエリア
<textarea>

送信ボタン
<input type="submit">

ラジオボタン
<input type="radio">
排他処理

セレクトボタン
<select>
中身は<option>で指定

フォームとCSS

- フォームの部品は全てCSSを適用可能
  - 色や枠線などもCSSで規定されているため、指定することで上書き可能
  - フォーム部品自体はインライン要素

- CSSで記述する場合は、属性値を含めた指定をすると指定しやすい

```
input[type=“text”]{
    background-color:#FFF;
}
```

テキストフィールドのみに適用

```
input[type=“submit”]{
    background-color:#FFF;
}
```

送信ボタンのみに適用