



G's ACADEMY
TOKYO

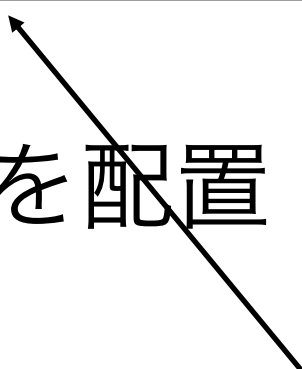
PHP/Ajax



本日専用のファイルとDBを用意します。

本日の配布ファイルは完成形に近いサンプルです。

今日の環境準備

1. DBを構築 → gs_db5
 2. サンプル[php05/SQL/DB.sql]をインポートします。
 3. htdocsと同じ階層に 「includes」 フォルダを作成
 4. htdocsに サンプル [php05] を配置
 5. ~~php05に入っているfuncs.phpを~~ 「includes」 に移動
- 


※funcs.phpをhtdocsの上に置くことでブラウザからアクセスできないファイルとします。

Ajaxとは

Ajaxとは

- AjaxのAである「Asynchronous（非同期）」は、非同期でのクライアント・サーバ間の通信を指します
- JavaScriptによりクライアント上での動作が主で必要なデータのみ受信することで通信量の負担を軽減
- 特殊なサーバの設定等は必要としない

Ajaxのメリット

- Webページのリンクをクリックした時のレスポンス待ち時間の体感時間が少ない。
- 必要な部分の情報のみを取得変更し、必要なときに更新可能のため高速に動作する。
- 例)
5画面 → 5HTMLファイル (通常の方法)

5画面 → 1HTMLファイル (Ajaxだとうちができる！)

Ajaxのデメリット

- SEO対策には不向き
- スクリプトの知識が必要
- 作りが複雑になりがち
- 更新履歴が残りません、ブラウザの[戻る]ボタンでは1つ前の状態には戻りません。
- 更新後でも再表示すると初期状態に戻ってしまいます。

Ajaxとは (Chrome:Yahooサイトで確認)

◇ChromeブラウザでYahooサイトを表示 → 右クリック(検証)

The screenshot shows the Yahoo! Japan homepage with the Chrome DevTools Network tab open. The following elements are annotated with red circles and numbers:

- ①: The "エンタメ" (Entertainment) category in the main navigation bar.
- ②: The "Network" tab in the DevTools toolbar.
- ③: The "XHR" filter in the DevTools Network panel.
- ④: The request URL: `?_m=Topics&a=getTopics&s...`.
- ⑤: The "Preview" tab in the DevTools Network panel.
- ⑥: The response data, which includes a timestamp "15時49分更新" and a list of news items.

A red arrow points from the "エンタメ" category to the "Preview" tab, indicating the flow of the demonstration.

Ajaxの構文（全部入り）

```
$.ajax({  
    type: "GET",                //送信方法：POST,GET  
    url: "*****",            //データを取得するURL  
    datatype: "jsonp",         //データ形式(html,text,xml,json...)  
    cache: false,              //cacheするしない  
    success: function(data){   //通信成功時にサーバーからdata変数に値が渡ります。  
        alert("通信成功");  
    },  
    error: function() {        //エラー時の処理  
        alert("エラー");  
    },  
    complete: function() {     //成功orエラーどちらでも最後に処理させたい場合  
        alert("完了");  
    }  
});
```

参考までに： New Ajaxの記述方法

//A.通信プロパティの設定(datatype:JSON)

var request = \$.ajax({

type: "GET", //送信方法：POST,GET

url: "demo_ajax_data.json", //データを取得するURL

datatype: "json", //データ形式(html,text,xml,json...)

data:{"id": "1", "name":"yamazaki"}, //URL先にデータを送る場合に使用

timeout: 3000 //接続待ち時間(1000で1秒,3000で3秒)

});

//B. 通信成功の処理

request.done(function(data) {

alert("通信成功"); //ここに通信成功時の処理を記述していく

});

//C. 通信エラーの処理

request.fail(function() {

alert("通信エラー"); //ここに通信エラー時の処理を記述していく

});

//D. 通信処理完了後の処理

request.always(function() {

alert("通信完了"); //通信完了時の処理を記述

});

POST/GET が送信されてる??を確認する方法

POST/GET送信データを確認する方法

The image shows a web browser window with search results for the keyword 'テスト' (Test). The search results include links to '全国統一高校生テスト' and '全国統一中学生テスト'. The browser's developer tools are open, showing the Network tab. A red arrow points from the 'Network' tab to the 'Headers' sub-tab. Another red arrow points from the 'Headers' sub-tab to the 'Query String Parameters' section, which lists the following parameters:

- p: テスト
- aq: -1
- oq:
- ei: UTF-8
- fr: top_ga1_sa
- x: wrt

The 'Query String Parameters' section is highlighted with a red box. The 'Network' tab is also highlighted with a red box. The 'Headers' sub-tab is highlighted with a red box. The 'Query String Parameters' section is highlighted with a red box.



G's ACADEMY
TOKYO

Fileupload



Form ・ Camera

◇ Camera/写真選択

```
<form method="post" action="送信先" enctype="multipart/form-data">  
  <input type="file" accept="image/*" capture="camera" name="upfile">  
  <input type="submit" value="Fileアップロード">  
</form>
```

① FILE選択できるようにする

```
<input type="file" .....>
```

※他の例<input type="text">

② File送信時はenctype属性を指定

```
enctype="multipart/form-data"
```

③ POST送信 (Action="送信先")

◇ Camera/写真選択

```
<form method="post" action="送信先" enctype="multipart/form-data">  
  <input type="file" accept="image/*" capture="camera" name="upfile">  
  <input type="submit" value="Fileアップロード">  
</form>
```

② カメラ起動＆画像選択可能

input accept="image/*" capture="camera"

※ 他の記述方法例) accept="image/jpeg, image/gif, image/png"

※ 他の記述方法例) accept="audio/*"

※ 他の記述方法例) accept="video/*"

※ 他の記述方法例) accept="text/comma-separated-values" //CSV

PHP:FileUpload

FileUpload : ①アップロードチェックの処理

◇ \$_FILES : パラメータチェック

ファイルアップロード パラメータ取得

- isset(パラメータ名) (パラメータがセットされているか? ※未入力チェックとは違う)
- \$_FILES["upfile"]["error"] == 0 (0は正常にアップロードしてることを意味する)

※参考URL : http://www.flatflag.nir87.com/move_uploaded_file-970#tablepress-21

```
if( isset( $_FILES["upfile"] ) && $_FILES["upfile"]["error"]==0 ) {  
    // echo 'アップロードしてきている';  
} else {  
    // echo 'アップロードしてきてない OR なにかしらのErrorが発生';  
}
```

◇ \$_FILES : ファイル名、アップロード先のPath取得

```
if( isset($_FILES["upfile"]) && $_FILES["upfile"]["error"]==0 ) {  
    $file_name = $_FILES["upfile"]["name"];           // "1.jpg"ファイル名取得  
    $tmp_path = $_FILES["upfile"]["tmp_name"];        // www/tmp/1.jpg : Tempフォル  
    ダPath取得  
} else {  
    // echo 'アップロードしてきてない OR なにかしらのErrorが発生';  
}
```

FileUpload:②アップロード処理

◇ アップロードに使用する関数: 3ステップ!

1. `is_uploaded_file` アップロードOK!?
2. `move_uploaded_file` Tempフォルダからimgフォルダへ移動
3. `chmod` ファイルに権限を付与する (ブラウザで見れるように)

```
// FileUpload [--Start--]
if ( is_uploaded_file( $tmp_path ) ) {
    if ( move_uploaded_file( $tmp_path, $file_dir_path ) ) {
        chmod( $file_dir_path, 0644 );

    } else {
        // echo "Error:アップロードできませんでした。";
    }
}
// FileUpload [--End--]
```

FileUpload : ③アップロード完成例

◇ fileupload処理の流れ（シンプルバージョン）

```
<?php
if ( isset($_FILES["upfile"]) && $_FILES["upfile"]["error"]==0 ) {
    $file_name  = $_FILES["upfile"]["name"];           //"1.jpg"ファイル名取得
    $tmp_path   = $_FILES["upfile"]["tmp_name"];       //"../www/tmp/1.jpg"などの一時フォルダ
    $file_dir_path = "upload/". $file_name;           //画像ファイル移動先とファイル名
    // FileUpload [--Start--]
    if ( is_uploaded_file( $tmp_path ) ) {
        if ( move_uploaded_file( $tmp_path, $file_dir_path ) ) {
            chmod( $file_dir_path, 0644 ); //ファイルアップ完了&アクセス権限付与
        } else {
            //$error= "Error:アップロードできませんでした。"; //Error文字
        }
    }
    // FileUpload [--END-]
}else{
    //$error = "画像が送信されていません"; //Error文字
}
```

◇ DBに保存するのはファイル名のみ

ファイルアップロード処理の下でSQL (PDO) を作成しましょう！

ファイル名を保持している変数 \$file_name をDBに保存！

例) \$stmt->bindValue(":img", \$file_name);

画像表示方法

表示方法 例)

```
">
```

上記は、imgカラムに画像ファイル名が保存されてる
場合の例で、\$resultにデータが入ってる想定です。

同一ファイル名問題の対処（例）

問題：アップロード後のファイル名をユニークにする方法を実装せよ



0a25f8206342584d358e9d...5851bae234418c80b8.html
0a25f8206342584d358e9d...5f9b49060ccb6b41b6.html
0a079bde318105ae37ad6...42dcacde8bdc3fbdc3a6.html
0a7943b51cb56b3e8d4c6...5a7dabb7dbb...
0a85106b9f23128b55b29...87a0b863904d891c65.html
0ae394dcca345cf4b74fa48...35bb1b78314c2e658f.html
0af77197eb0261f11f623f9...9083093aafaa076fdc6f.html
0b2d1b09818dd5ae7bd18...6d443e742fc97398472.html
0b14fe88ec673142870652...f996ed4bb5b8e3c570.html
0b491a059bab0a219b4ebb...9390bc5eb44fdc706a.html
0b965f4a435296296e266e...12a918b4c6250c0543.html
0b7662cc0bb01d22445c1...52c02f2ccc6d02c54f4.html

イメージはこんな感じ。

- ・ 工夫をすることで
同じ名前にならないように
することが出来る。

※結果が同じ名前にならない
ければ、どういう方法でも
OK！！

random, md5, passwd_hash, session_id, date, など様々な
ユニーク値を作れそうな関数があります。どうやったらなる
か？自分なりに考えて作ることも可能です。

FileUpload : アップロード時のファイル名 重複問題対応

◇ ファイル名重複問題の解決策の例

```
2  if (isset($_FILES["upfile"]) && $_FILES["upfile"]["error"] == 0) {
3      $file_name = $_FILES["upfile"]["name"]; // "1.jpg"ファイル名取得
4      $tmp_path = $_FILES["upfile"]["tmp_name"]; // "/usr/www/tmp/1.jpg"アップ
5      ...
6      /***File名の変更***/
7      $extension = pathinfo($file_name, PATHINFO_EXTENSION);
8      $file_name = date("YmdHis").md5(session_id()) . "." . $extension;
9
10     $file_dir_path = "upload/".$file_name; //画像ファイル保管先
11     $img="";
```

◆ 以下 2 行で対応を実現 !

```
/***File名の変更***/
$extension = pathinfo($file_name, PATHINFO_EXTENSION);
$file_name = date("YmdHis").md5(session_id()) . "." . $extension;
```

Ajax補足

PHP側でajax通信のみ受け入れる方法

※必要なら (PHP側に記述)

```
2  /**
3   * Ajaxによるリクエストかどうか
4   * @return boolean true or false
5   */
6  function isAjax(){
7      if(isset($_SERVER['HTTP_X_REQUESTED_WITH']) &&
8          strtolower($_SERVER['HTTP_X_REQUESTED_WITH']) == 'xmlhttprequest'){
9          return true;
10     }else{
11         return false;
12     }
13 }
14
15 //Ajaxアクセス以外は処理しない。
16 if( isAjax()==false ){
17     header("HTTP/1.0 404 Not Found");
18     exit();
```

後は通常通りの処理