

# file upload & Ajax



G's ACADEMY  
FUKUOKA



# アジェンダ

- ファイルアップロードの仕組み
- todoリストアプリにファイルアップロード機能を追加
- Ajax
- 課題発表→P2Pタイム

# 授業のルール

- 授業中は常にエディタを起動！
- 考えたことや感じたことはzoomチャットでガンガン発信！
- 質問はslackへ！ 他の人の質問にも目を通そう！（同じ質問があるかも）
- 演習時，できた人はスクショなどslackに貼ってアウトプット！
- まずは打ち間違いを疑おう！

{ } ' " ; など

- 書いたら保存しよう！（よく忘れる！）

command + s

ctrl + s

# PHPの準備

- XAMPPの起動確認
- <http://localhost/>のアクセス確認
- サンプルフォルダを「htdocs」フォルダに入れる

# ファイルのアップロード

# phpでファイルをサーバにアップロード！

## - ファイルアップロードの流れ

- ①フォームからアップロード
- ②tmp領域(一時保存場所)に保存
- ③サーバの保存領域に移動(サンプルでは「upload」ディレクトリ)
- ④(データベースに保存場所のパスを登録)



# ①フォームの準備

```
// <input type="file">を使用.  
// 使用時には「enctype="multipart/form-data"」が必須！！  
// methodはpostを使用！getだと容量不足の可能性が．．．！  
  
// コード  
<form action="file_upload.php" method="POST" enctype="multipart/form-data">  
  // ...  
  <input type="file" name="upfile" accept="image/*" capture="camera">  
  // ...  
</form>
```



## ② - ④ファイルの保存

- 準備: 送信時にエラー等ないかどうか確認.
  - ①送られてきたファイルの情報を取得(自動的にtmp領域に保管)
  - ②ファイル名を準備(他のファイルと被らないように)
  - ③サーバの保存領域に移動(サンプルでは「upload」)  
(ファイル名に保存ディレクトリも含めている点に注意!)
  - ④サンプルファイルではimgタグで表示

```
// ファイルが追加されていない or エラー発生の場合を分ける.  
// 送信されたファイルは$_FILES['...'];で受け取る！  
  
// コード  
if (isset($_FILES['upfile']) && $_FILES['upfile']['error'] == 0) {  
    // 送信が正常に行われたときの処理  
    ...  
} else {  
    // 送られていない, エラーが発生, などの場合  
    exit('Error:画像が送信されていません');  
}
```

```
// アップロードしたファイル名を取得.  
// 一時保管しているtmpフォルダの場所の取得.  
// アップロード先のパスの設定（サンプルではuploadフォルダ←作成！）  
  
// コード  
$uploadedFileName = $_FILES['upfile']['name']; //ファイル名の取得  
$tempPathName    = $_FILES['upfile']['tmp_name']; //tmpフォルダの場所  
$fileDirectoryPath = 'upload/'; //アップロード先フォルダ  
                                (↑自分で決める)
```

```
// ファイルの拡張子の種類を取得.  
// ファイルごとにユニークな名前を作成. (最後に拡張子を追加)  
// ファイルの保存場所をファイル名に追加.  
  
// コード  
$extension = pathinfo($uploadedFileName, PATHINFO_EXTENSION);  
$uniqueName = date('YmdHis').md5(session_id()) . "." . $extension;  
$fileNameToSave = $fileDirectoryPath.$uniqueName;  
  
// 最終的に「upload/hogehoge.png」のような形になる
```

- アップロード領域へファイルを移動.
- 権限の変更.
- <img>で出力.

※権限: <https://www.atmarkit.co.jp/ait/articles/1605/23/news020.html>

```
$img='';  
if (is_uploaded_file($tempPathName)) {  
    if (move_uploaded_file($tempPathName, $fileNameToSave)) {  
        chmod($fileNameToSave, 0644);           // 権限の変更  
        $img = ''; // imgタグを設定  
    } else {  
        exit('Error:アップロードできませんでした'); // 画像の保存に失敗  
    }  
} else {  
    exit('Error:画像がありません'); // tmpフォルダにデータがない  
}
```

### - 練習

- アップロード用のフォームを準備しよう！(file\_upform.php)
- アップロード処理を記述して画像をアップロードしよう！
- アップロードしたファイルを表示しよう！  
(file\_upload.phpで\$imgを出力！)

-> 画像が「upload」フォルダに保存されて画面に表示されていればOK！



# todoアプリに機能追加

# phpでファイルをサーバにアップロード！

## - ファイルアップロードの流れ

- ①フォームからアップロード
- ②tmp領域(一時保存場所)に保存
- ③サーバの保存領域に移動(サンプルでは「upload」ディレクトリ)
- ④データベースに保存場所のパスを登録



# phpでファイルをサーバにアップロード！

## - 準備①

- todo\_tableにカラムを追加する.
- 「image」を追加！
- 保存した画像のURLを登録する.

	#	名前	データ型	照合順序	属性	NULL	デフォルト値	コメント	その他	操作
<input type="checkbox"/>	1	id 	int(12)			いいえ	なし		AUTO_INCREMENT	 変更  削除  その他
<input type="checkbox"/>	2	todo	varchar(128)	utf8mb4_unicode_ci		いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	3	deadline	date			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	4	image	varchar(128)	utf8mb4_unicode_ci		はい	NULL			 変更  削除  その他
<input type="checkbox"/>	5	created_at	datetime			いいえ	なし			 変更  削除  その他
<input type="checkbox"/>	6	updated_at	datetime			いいえ	なし			 変更  削除  その他

```
// 準備②
// アップロードフォームの準備
// input type="file"の追加, actionの宛先変更, enctype属性の追加
// (流れはさっきやったものと同じ)

// コード例
<form method="post" action="create_file.php" enctype="multipart/form-data">
  // ...
  <div>
    <input type="file" name="upfile" accept="image/*" capture="camera">
  </div>
  // ...
</form>
```

- 準備:送信時にエラー等ないかどうか確認.
  - ①送られてきたファイルの情報を取得(自動的にtmp領域に保管)
  - ②ファイル名を準備(他のファイルと被らないように)
  - ③サーバの保存領域に移動(サンプルでは「upload」)  
(ファイル名に保存ディレクトリも含めている点に注意！)  
-- ここまで全く同じ --
  - ④DBに情報を登録
  - ⑤一覧画面に画像を表示

```
// ファイルが追加されていない or エラー発生の場合を分ける  
// 送信されたファイルは$_FILES['...'];で受け取る！
```

全く同じ！

```
// コード  
if (isset($_FILES['upfile']) && $_FILES['upfile']['error'] == 0) {  
    // 送信が正常に行われたときの処理  
    ...  
} else {  
    // 送られていない, エラーが発生, などの場合  
    exit('Error:画像が送信されていません');  
}
```

```
// アップロードしたファイル名を取得.  
// 一時保管しているtmpフォルダの場所の取得.  
// アップロード先のパスの設定（サンプルではupload/）
```

全く同じ！

```
// コード
```

```
$uploadedFileName = $_FILES['upfile']['name']; //ファイル名の取得  
$tempPathName    = $_FILES['upfile']['tmp_name']; //tmpフォルダの場所  
$fileDirectoryPath = 'upload/'; //アップロード先フォルダ  
                                (↑自分で決める)
```

```
// ファイルの拡張子の種類を取得.  
// ファイルごとにユニークな名前を作成. (最後に拡張子を追加)  
// ファイルの保存場所をファイル名に追加.
```

全く同じ!

```
// コード  
$extension = pathinfo($uploadedFileName, PATHINFO_EXTENSION);  
$uniqueName = date('YmdHis').md5(session_id()) . "." . $extension;  
$fileNameToSave = $fileDirectoryPath.$uniqueName;
```

```
// 最終的に「upload/hogehoge.png」のような形になる
```



- アップロード領域へファイルを移動.
- 権限の変更.
- 今回は表示しない！(imgタグを作成しない)

```
if (is_uploaded_file($tempPathName)) {  
    if (move_uploaded_file($tempPathName, $fileNameToSave)) {  
        chmod($fileNameToSave, 0644);  
    } else {  
        exit('Error:アップロードできませんでした');           // 画像の保存に失敗  
    }  
} else {  
    exit('Error:画像がありません');                             // tmpフォルダにデータがない  
}
```

今回は画像を表示しない！

```
// 他のデータと一緒にDBへ登録！
```

```
// INSERT文にimageカラムを追加！
```

```
$sql = 'INSERT INTO  
        todo_table(id, todo, deadline, image, created_at, updated_at)  
        VALUES(NULL, :todo, :deadline, :image, sysdate(), sysdate())';
```

```
// ...
```

```
$stmt->bindValue(':image', $fileNameToSave, PDO::PARAM_STR);
```

```
// ...実行, エラー処理, etc...
```

```
// 一覧画面で画像を表示
```

```
...  
$output .= "<td><img src='{$_record[\"image\"]}' height=150px></td>";  
...
```

# todoリストアプリからファイルアップロード

## - 練習

- アップロード用のフォームを準備しよう！(todo\_input.php)
  - アップロード処理を記述して画像をアップロードしよう！(create\_file.php)
  - アップロードしたファイルのURLをDBに保存しよう！(create\_file.php)
  - 一覧画面に画像を表示しよう！(todo\_read.php)
- >画像が「upload」フォルダに保存されて、パスがDBに保存されていればOK！
- >一覧画面で画像が表示されればOK！

# Ajax

- DBへの登録, 表示などの処理を実行するPHPファイルとのhttp通信を

# JavaScriptで

扱う手法！！

## - メリット

- データだけをやり取りするので速い&通信量が少ない！
- ファイル数が少なくできる！
- 通信時にリロードがない！

## - デメリット

- SEOに弱い(最近は大丈夫になってきている)
- 構造が複雑になりがち.
- ページを更新すると表示内容が初期状態に戻る.



## - JavaScriptでhttp通信するときの方法

名称	特徴
XMLHttpRequest	生JS / 一番昔からあるやつ
\$.ajax()	jQuery / これが出てきて流行った
fetch	生JS / 最近できたけど使いづらい
axios	ReactとかVueでも使われていて使い勝手が良い

# 今日やること

- Ajaxを使ってリアルタイム検索を実装！！

※JavaScriptとPHPが入り乱れるので都度ファイル名を確認！！！！

# DBに登録されている情報を取得する

- 必要なもの
  - JavaScriptのコード(`ajax_search.html`)
    - PHPファイルに対してリクエストを送る処理.
    - APIへのリクエストと同じく`axios.get()`を使用.
  - PHPのコード(`ajax_get.php`)
    - DBからデータを取得する処理.
    - 前回までの`todo_read.php`とほぼ同様.
    - 取得したデータをJSON形式で返す.

# DBに登録されている情報を取得する

## - 処理の流れ

- ①JavaScriptからPHPファイルにリクエスト(検索ワード)を送る。(JS)
- ②DBからデータを取得する。(PHP)
- ③取得したデータをJSON形式にして出力する。(PHP)
- ④JavaScriptでデータを受け取る。(JS)←ここまでつくろう！
- ⑤(受け取ったデータをブラウザに表示する)

## JS側の処理

```
// phpへリクエストを送って結果を出力する処理
検索フォーム : <input type="text" id="search">
// ...
$('#search').on('keyup', function () {
  console.log($(this).val());           // inputの内容をリアルタイムに取得
  const serchWord = $(this).val();
  const requestUrl = 'ajax_get.php';    // リクエスト送信先のファイル
  // ...続く
});
```

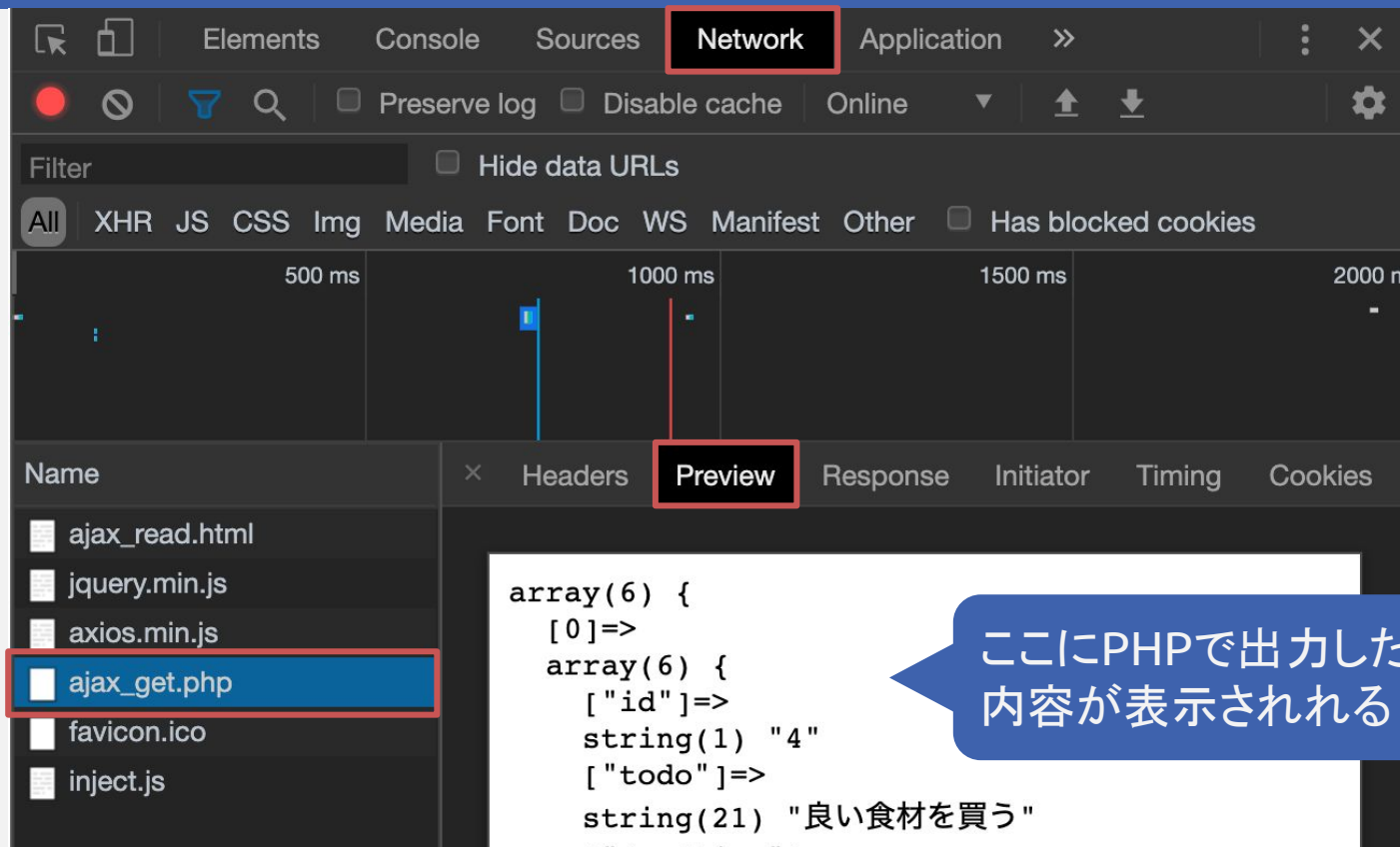
```
// phpへリクエストを送って結果を出力する処理
axios.get(`${requestUrl}?serchword=${serchWord}`) // ①リクエスト送信
  .then(function (response) {
    console.log(response); // ④受け取り→表示
    // できる人はここにブラウザに表示する処理を書こう！
  })
  .catch(function (error) {...})
  .finally(function () {...});
```

# PHP側の処理



```
// ...関数ファイル読み込み処理を記述（認証関連は省略でOK）
// ...DB接続の処理を記述
$search_word = $_GET["serchword"]; // GETのデータ受け取り
$sql = "SELECT * FROM todo_table WHERE todo LIKE '%{$search_word}%'";
// ...SQL実行の処理を記述
if ($status == false) {
    // ...エラー処理を記述
} else {
    $result = $stmt->fetchAll(PDO::FETCH_ASSOC);
    echo json_encode($result); // JSON形式にして出力
    exit();
}
```

# エラー / var\_dump()の確認にはNetworkタブを使え！！



The screenshot shows the Chrome DevTools Network tab. The 'Network' tab is selected at the top. Below it, the 'Filter' section shows 'All' selected. The 'Name' column lists several files, with 'ajax\_get.php' highlighted. The 'Preview' tab is selected for 'ajax\_get.php', showing a JSON response. A blue callout bubble points to the response content.

Network tab interface showing the response of `ajax_get.php`.

Response content (JSON):

```
array(6) {  
  [0]=>  
    array(6) {  
      ["id"]=>  
        string(1) "4"  
      ["todo"]=>  
        string(21) "良い食材を買う"  
    }  
}
```

ここにPHPで出力した内容が表示される！

- リアルタイム検索を実装しよう！
  - ①axios.get()でリクエストを送ろう！（ajax\_read.html）
  - ②DBからデータを取得しよう！（ajax\_get.php）
  - ③JSON形式にして出力しよう！（ajax\_get.php）
  - ④受け取ってconsoleでデータを確認しよう！（ajax\_read.html）
  - （できる人はブラウザにデータを表示しよう！）

# 課題

# PHPで自由に実装

- PHPとDBを使用したアプリケーションを実装
  - 卒制プロトタイプ
  - twitterみたいなもの
  - 投票管理アプリ
  - 写真共有アプリ
  - 自作wordpress

今回までの内容で大体のwebサービスは実装可能！！

提出は次回授業前木曜「23:59:59」まで！！

# P2Pタイム

まずはチーム内で解決を目指す！  
訊かれた人は苦し紛れでも応える！！