

File upload & Ajax

Contents

- 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

rules...

- 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slackのガヤチャンネルでガンガン発信！
- 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

以下3点ができているか確認しよう！

- 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

Goal

- ファイルの扱いを知る.
- ファイルとDBの組み合わせ方を知る.
- JavaScriptでデータ通信を行う！

エラー表示用の設定

エラー表示用の設定

PHPではデフォルトでエラーが表示されない設定となっている。

開発時、エラーメッセージを確認しやすいよう設定の上書きを行う。

1. プロジェクトのフォルダ内に`.htaccess`ファイルを作成する。
2. `.htaccess`ファイルに下記の内容を追記して保存する。

```
php_flag display_errors On
```

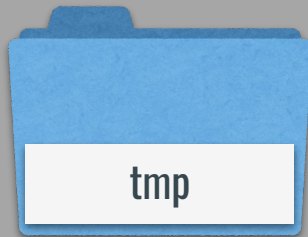
これで、PHPのコードが実行されてエラーが発生した場合に画面で確認することができる。

ファイルアップロード
(画像 / 音声など)

画像や音声などのファイルをサーバに保存

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
2. tmp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)



⚠ 注意点 ⚠

PHPからファイルにアクセスするときにハマることがあります！！

- macの人
 - 「htdocsに入れた講義フォルダ」で
メニュー表示⇒情報を見る⇒共有とアクセス権
 - 全て「読み/書き」に変更⇒歯車ボタン⇒内包している項目に適用
- winの人
 - 特になし

①フォームの準備

①フォームからアップロード(file_upform.php)

```
// <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

ファイル保存の流れ(file_upload.php)

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

if文が多いのでコードをどこに書くか確認しましょう!

準備:送信時にエラー等ないかどうか確認(file_upload.php)

```
// 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 (この後記述)  
    ...  
} 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');  
}
```

①送信されたファイルの情報を取得(file_upload.php)

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


②ファイル名の準備(file_upload.php)

```
// 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  
$extension = pathinfo($uploaded_file_name, PATHINFO_EXTENSION);  
$unique_name = date('YmdHis').md5(session_id()) . "." . $extension;  
$filename_to_save = $directory_path . $unique_name;  
  
// 最終的に「upload/hogehoge.png」のような形になる
```

③④サーバの保存領域に移動 -> 表示 (file_upload.php)

- アップロード領域へファイルを移動.
- 権限の変更.
- で出力.

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

③④サーバの保存領域に移動 -> 表示(file_upload.php)

```
if (is_uploaded_file($temp_path)) {  
    // ↓ここでtmpファイルを移動する  
    if (move_uploaded_file($temp_path, $filename_to_save)) {  
        chmod($filename_to_save, 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  
}
```

ファイル送信の処理を実装しよう！

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！）

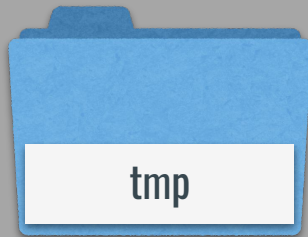
->「画像が`upload`フォルダに保存」されて「画面に表示」されていればOK

todoアプリに機能追加

画像や音声などのファイルをサーバに保存

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
2. tmp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



画像の保存場所のパスをDBに保存できるようにする

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

アップロードフォームの追加(todo_input.php)

```
// 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)

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


ファイル保存の流れ(create_file.php)

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！)
-- ここまで前項の処理と全く同じ --
4. DBに情報を作成
5. 一覧画面に画像を表示

準備:送信時にエラー等ないかどうか確認(create_file.php)

```
// 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（この後記述）  
    ...  
} 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');  
}
```

全く同じッ！！

①送信されたファイルの情報を取得(create_file.php)

```
// 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  
$uploaded_file_name = $_FILES['upfile']['name']; //ファイル名の取得  
$temp_path = $_FILES['upfile']['tmp_name']; //tmpフォルダの場所  
$directory_path = 'upload/'; //アップロード先フォルダ
```

全く同じッ！！

②ファイル名の準備(create_file.php)

```
// 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  
$extension = pathinfo($uploaded_file_name, PATHINFO_EXTENSION);  
$unique_name = date('YmdHis').md5(session_id()) . "." . $extension;  
$filename_to_save = $directory_path . $unique_name;  
  
// 最終的に「upload/hogehoge.png」のような形にな
```

全く同じッ！！

③④サーバの保存領域に移動 -> 表示(create_file.php)

- 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)

③サーバの保存領域に移動(create_file.php)

```
if (is_uploaded_file($temp_path)) {  
    if (move_uploaded_file($temp_path, $filename_to_save)) {  
        chmod($filename_to_save, 0644);           // 権限の変更  
        // 今回は権限を変更するところまで  
    } 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  
}
```

DBにデータ作成

```
// 他のデータと一緒にDBへ登録！
// 処理の流れはtodo_create.phpと同様

// INSERT文にimageカラムを追加！
// imageカラムには画像ファイルのパスが入る.
$sql = 'INSERT INTO
        todo_table(id, todo, deadline, image, created_at, updated_at)
        VALUES(NULL, :todo, :deadline, :image, sysdate(),sysdate())';
// ...省略 (todo_create.phpと同様)
$stmt->bindValue(':image', $filename_to_save, PDO::PARAM_STR);
// ...実行, エラー処理, etc...
```

⑤一覧画面に画像を表示(todo_read.php)

```
// 一覧画面で画像を表示
```

```
// ...
```

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

```
// ...
```


todoアプリのアップロード機能を追加しよう！

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

Ajax(えーじゃっくす)とは

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

扱う手法ッ！！

Ajax(えーじゃっくす)とは

メリット

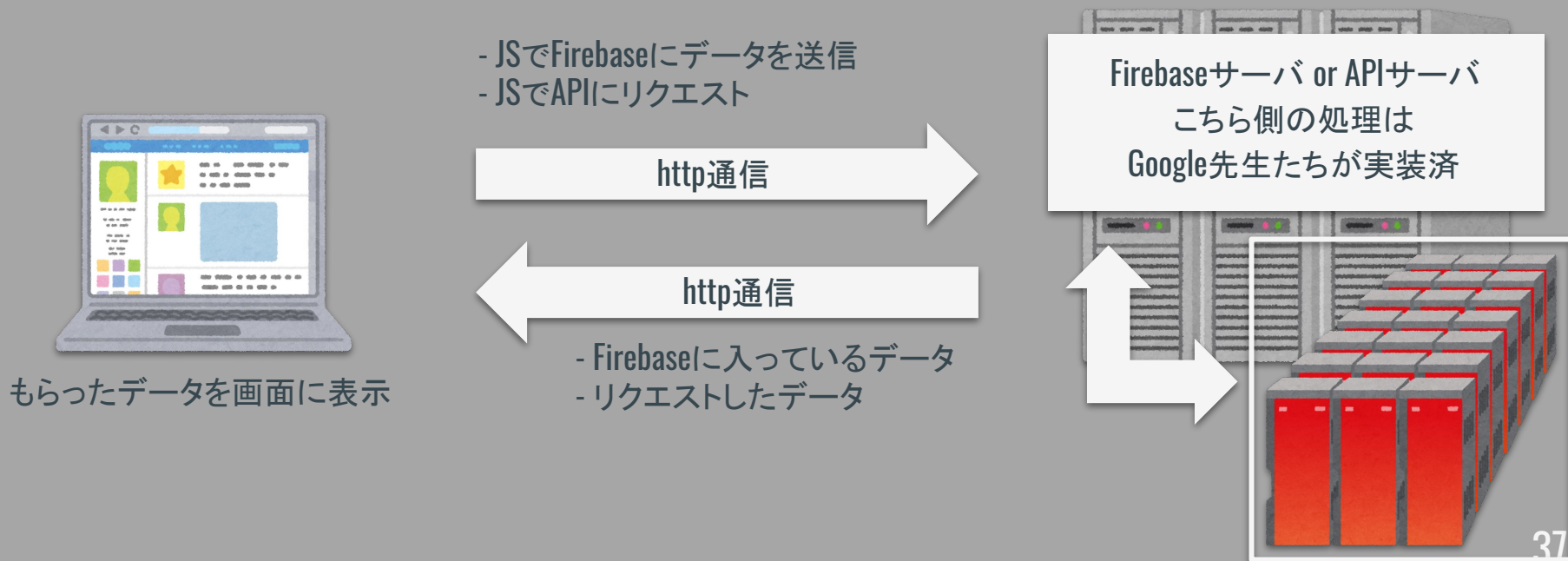
- データだけをやり取りするので速い&通信量が少ない！
- ファイル数が少なくできる&フロントとサーバの分業がしやすい！
- 通信時にリロードがない！ <- 無限スクロールなどで活用される

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.

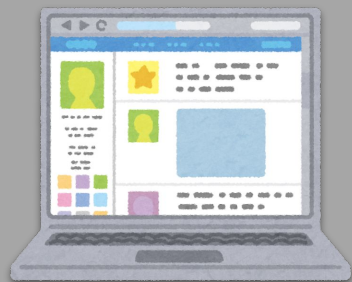
じつはFirebaseやAPIと似た動き. . . !

じつはサーバ(Firebase or API)と通信していた. . . !



じつはFirebaseと似た動き..！

サーバ側を自分で実装するのが今回！！！！



もらったJSONデータを
処理して画面に表示

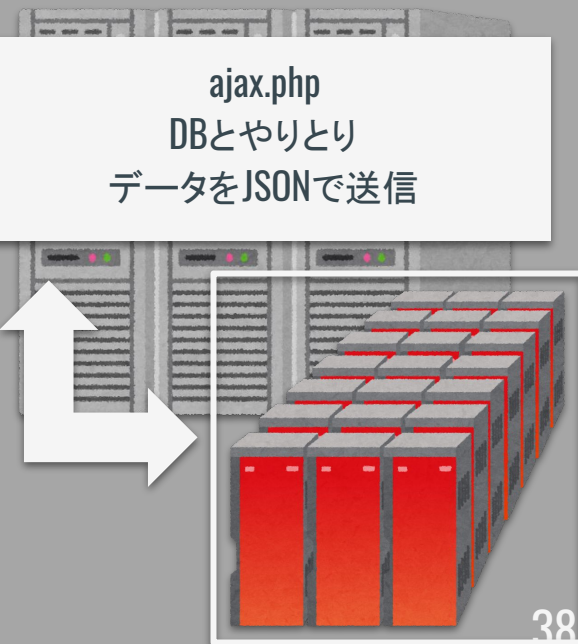
- JSでPHPにデータを送信

http通信

http通信

- DBに入っているデータ

ajax.php
DBとやりとり
データをJSONで送信



Ajax(えーじゃっくす)とは

JavaScriptでhttp通信するときの方法(下にいくほどオススメ)

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 / 慣れないと分かりづらい
axios	ReactとかVueでも使われていて使い勝手が良い

今回の目標

Ajaxを使ってリアルタイム検索を実装！！

- 検索ボックスになにか入力したら，該当するデータだけをDBから取り出して表示
- 検索ボタンではなく，入力した時点でリアルタイムに検索する.

※JavaScriptとPHPが入り乱れるので都度ファイル名を確認！！！！

リアルタイム検索の実装

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. <- やっていることは同じ
- 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.

「API」をPHPでつくる！！！！

リアルタイム検索の実装

処理の流れ

1. JavaScriptからPHPファイルにリクエスト(検索ワード)を送る. (JS)
2. DB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)
4. JavaScriptでデータを受け取る. (JS)← 今回はここまでつくろう！
5. (受け取ったデータをブラウザに表示)

JSの処理

①④JSでPHPにリクエスト送信 -> 結果受信 (ajax_search.html)

```
// 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 (e) {  
    console.log(e.target.value);           // inputの内容をリアルタイムに取得  
    const searchWord = e.target.value;  
    const requestUrl = 'ajax_get.php';     // リクエスト送信先のファイル  
    // ...続く  
});
```

①④JSでPHPにリクエスト送信 -> 結果受信 (ajax_search.html)

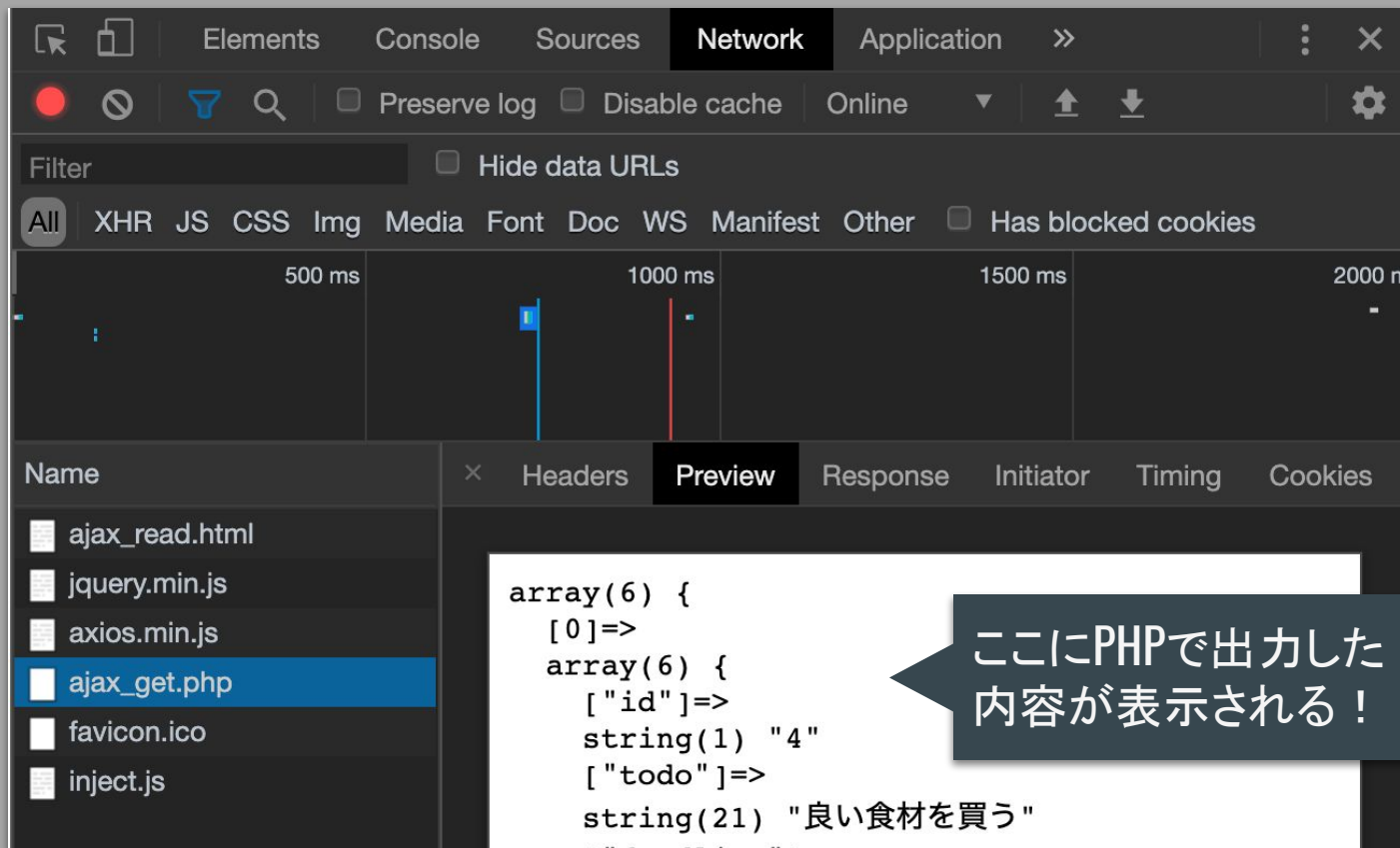
```
// phpへリクエストを送って結果を出力する処理
axios.get(`${requestUrl}?searchword=${searchWord}`) // リクエスト送信
  .then(function (response) {
    console.log(response); // responseにPHPから送られたデータが入る
    // 今回はconsoleでデータが出てくればOK.
    // できる人はここにブラウザに表示する処理を書こう！
  })
  .catch(function (error) {...})
  .finally(function () {...});
```

PHPの処理

②③DBからデータを取得してjsonで出力 (ajax_get.php)

```
// 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["searchword"]; // GETのデータ受け取り
$sql = "SELECT * FROM todo_table WHERE todo LIKE :search_word";
// 省略
$stmt->bindValue(':search_word', "%{$search_word}%",
PDO::PARAM_STR);
// 省略
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();
}
```

Ajaxでのエラー / var_dump()の確認にはNetworkタブを使え！！



The screenshot shows the Network tab in a web browser. The list of requests includes `ajax_read.html`, `jquery.min.js`, `axios.min.js`, `ajax_get.php` (highlighted), `favicon.ico`, and `inject.js`. The `ajax_get.php` request is selected, and the `Preview` tab is active. The response content is displayed as a PHP var_dump output:

```
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！

Ajaxを使ったリアルタイム検索を実装！

リアルタイム検索を実装しよう！

1. `axios.get()`でリクエストを送ろう！（`ajax_read.html`）
2. DBからデータを取得しよう！（`ajax_get.php`）
3. JSON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`）
5. （できる人はブラウザにデータを表示しよう！）

課題

卒制プロトタイプ

- とにかく「早く」動かそう！
- 動かすと「必要なもの」「不要なもの」がわかる！
- 動いたらデプロイも一旦やろう！（1年間は無料でデプロイし放題！）

締切は厳守！！

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！！