Here’s a curated list of **libraries and scripts** you’ll want to include in your raw PHP + Axios setup. These are lightweight, modular, and future-proofed for your workflow.

## 🧹 Final Cleanup Checklist for Raw PHP + Axios

### 🔧 Backend (PHP)

#### ✅ 1. **Modularize Your PHP Logic**

* Split logic into reusable files:
  + db.php – DB connection
  + auth.php – login/session checks
  + csrf.php – CSRF token generation/validation
  + utils.php – common functions (e.g. sanitization, logging)

#### ✅ 2. **Sanitize & Validate Inputs**

* Use filter\_input() or htmlspecialchars() for GET/POST
* Validate server-side even if frontend checks exist

#### ✅ 3. **Session Hygiene**

* Regenerate session ID on login: session\_regenerate\_id(true)
* Set secure cookie flags:
* session\_set\_cookie\_params([  
   'httponly' => true,  
   'secure' => true,  
   'samesite' => 'Strict'  
  ]);

#### ✅ 4. **CSRF Protection**

* Store token in session
* Validate X-CSRF-Token header
* Rotate token periodically

#### ✅ 5. **Error Handling**

* Use try/catch blocks for DB queries
* Return JSON responses with status codes:
* http\_response\_code(400);  
  echo json\_encode(['error' => 'Invalid input']);

### 🎨 Frontend (JS + Axios)

#### ✅ 6. **Centralize Axios Config**

Create a single Axios instance:

const api = axios.create({  
 baseURL: '/ajax/',  
 withCredentials: true,  
 headers: {  
 'X-CSRF-Token': document.querySelector('meta[name="csrf-token"]').content  
 }  
});

#### ✅ 7. **Modular JS Files**

Split logic by feature:

* auth.js – login/logout
* profile.js – user updates
* dashboard.js – data fetching

#### ✅ 8. **Graceful Error UI**

Show user-friendly messages:

api.post('update-profile.php', data)  
 .then(res => showSuccess(res.data))  
 .catch(err => showError(err.response?.data?.error || 'Something went wrong'));

### 🧠 Bonus: Future-Proofing Ideas

* Add a simple router (e.g. AltoRouter) for cleaner URLs
* Use .env for config (via vlucas/phpdotenv)
* Consider a lightweight templating engine (e.g. Plates or Twig)

## 📦 PHP Libraries

You can install these via Composer (no Laravel needed):

| Purpose | Library | Install Command | Notes |
| --- | --- | --- | --- |
| **CSRF Protection** | paragonie/anti-csrf | composer require paragonie/anti-csrf | Secure, session-based CSRF tokens |
| **Environment Config** | vlucas/phpdotenv | composer require vlucas/phpdotenv | Load .env variables (DB creds, etc.) |
| **Routing (Optional)** | altorouter/altorouter | composer require altorouter/altorouter | Clean URL routing without a full framework |
| **Validation** | respect/validation | composer require respect/validation | Declarative input validation |
| **Password Hashing** | Native PHP | *Built-in* | Use password\_hash() and password\_verify() |

## 🧠 JavaScript Libraries & Scripts

You don’t need Node/npm — just include via CDN or static files.

### ✅ Core Scripts

<!-- Axios -->  
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>  
  
<!-- Optional: SweetAlert2 for alerts -->  
<script src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>  
  
<!-- Optional: Lodash for utility functions -->  
<script src="https://cdn.jsdelivr.net/npm/lodash@4.17.21/lodash.min.js"></script>

### ✅ Your Custom JS Modules

<!-- Your modular scripts -->  
<script src="/js/axios.js"></script>  
<script src="/js/auth.js"></script>  
<script src="/js/profile.js"></script>  
<script src="/js/dashboard.js"></script>

## 🧰 Suggested Folder Structure

/public  
 index.php  
 /js  
 axios.js ← Axios instance setup  
 auth.js ← Login/logout logic  
 profile.js ← Profile update logic  
 dashboard.js ← Data fetch/display  
 /css  
 style.css  
 /ajax  
 login.php  
 update-profile.php  
 fetch-data.php  
/includes  
 db.php ← DB connection  
 auth.php ← Session & login checks  
 csrf.php ← CSRF token logic  
 utils.php ← Common helpers  
.env ← Config (DB creds, etc.)  
composer.json