🧠 Materi Praktek minggu 6: AJAX dengan PHP untuk Interaksi Dinamis

🤖 Live Search AJAX dengan PHP

💡 Studi Kasus:

Sistem pencarian data mahasiswa berdasarkan **nama atau NIM** tanpa reload halaman.

📁 Struktur Folder:

ajax-search/

├── index.php

├── search.php

├── db.php

└── style.css (opsional)

📦 1. Persiapan Database MySQL

🗂️ SQL: Buat Database dan Tabel Mahasiswa

```
CREATE DATABASE IF NOT EXISTS kampus;

USE kampus;

CREATE TABLE IF NOT EXISTS mahasiswa (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nim VARCHAR(15) NOT NULL,
    nama VARCHAR(100) NOT NULL,
    jurusan VARCHAR(100)
);

INSERT INTO mahasiswa (nim, nama, jurusan) VALUES
('23001', 'Budi Santoso', 'Teknik Informatika'),
```

```
('23002', 'Dewi Anggraini', 'Sistem Informasi'),
('23003', 'Adi Wijaya', 'Teknik Komputer'),
('23004', 'Putri Lestari', 'Manajemen'),
('23005', 'Rizky Hidayat', 'Sistem Informasi');
```

2. db.php – Koneksi Database

```
<?php
$host = "localhost";
$user = "root";
$pass = "";
$db   = "kampus";

$koneksi = new mysqli($host, $user, $pass, $db);

if ($koneksi->connect_error) {
   die("Koneksi gagal: " . $koneksi->connect_error);
}
?>
```

3. search.php – Proses AJAX ke MySQL

```
<?php
header('Content-Type: application/json');
include 'db.php';
$keyword = isset($_GET['keyword']) ? $koneksi->real_escape_string($_GET['keyword']) : '';
$sql = "SELECT nim, nama, jurusan FROM mahasiswa
        WHERE nim LIKE '%$keyword%' OR nama LIKE '%$keyword%'";
$result = $koneksi->query($sql);
$data = [];
while ($row = $result->fetch_assoc()) {
   $data[] = $row;
}
echo json_encode($data);
```

Penjelasan kode:

- **1. Mengatur Header Output**

```
<?php
header('Content-Type: application/json');
```

🧠 Penjelasan:

- Baris ini memberi tahu browser bahwa respons dari file ini adalah **JSON**, bukan HTML biasa.
- Ini penting agar fetch(...).then(res => res.json()) di frontend bisa memproses hasilnya dengan benar.

🔹 **2. Koneksi ke Database**

include 'db.php';

🧠 Penjelasan:

- Kita menyertakan file db.php yang biasanya berisi kode koneksi ke database (misalnya $koneksi = new mysqli(...)).
- Tanpa ini, kita tidak bisa query ke database.

🔹 **3. Mengambil Keyword dari URL**

$keyword = isset($_GET['keyword']) ? $koneksi->real_escape_string($_GET['keyword']) : '';

🧠 Penjelasan:

- Mengambil nilai keyword dari parameter URL (search.php?keyword=...).
- **$koneksi->real_escape_string()** mencegah SQL Injection dengan meloloskan karakter berbahaya dalam keyword.
- Jika tidak ada keyword, default-nya adalah string kosong ('').

🔹 **4. Query SQL ke Tabel Mahasiswa**

$sql = "SELECT nim, nama, jurusan FROM mahasiswa
WHERE nim LIKE '%$keyword%' OR nama LIKE '%$keyword%'";

🧠 Penjelasan:

- Query ini mencari mahasiswa berdasarkan **NIM** atau **Nama** yang mengandung keyword (menggunakan LIKE dengan wildcard %).
- %Dewi% akan cocok dengan "Dewi", "Dewi Ayu", "Ayu Dewi", dll.

🔹 **5. Eksekusi Query**

$result = $koneksi->query($sql);

$data = [];

🧠 Penjelasan:

- Jalankan query dengan $koneksi->query(...).
- Buat array kosong $data untuk menampung hasilnya.

🔹 **6. Menyimpan Hasil ke Array**

```
while ($row = $result->fetch_assoc()) {
   $data[] = $row;
}
```

🧠 Penjelasan:

- Gunakan fetch_assoc() untuk mengambil data sebagai array asosiatif (['nim' => ..., 'nama' => ..., ...]).
- Tambahkan setiap baris hasil ke dalam array $data.

🔹 **7. Mengirim Kembali sebagai JSON**

```
echo json_encode($data);
```

🧠 Penjelasan:

- Ubah array PHP $data menjadi **JSON string** dan kirim ke browser (frontend).
- Ini yang akan diterima dan diolah oleh JavaScript di index.html.

🔚 **Summary Alur Backend**

1. Terima keyword dari URL.
2. Lindungi dari karakter berbahaya.
3. Query ke database.
4. Ubah hasil query jadi array.
5. Encode jadi JSON.
6. Kirim ke browser.

📄 **4. index.php – Halaman Pencarian AJAX**

```
<!DOCTYPE html>
<html lang="en">
<head>
```

```
  <meta charset="UTF-8">
  <title>Live Search Mahasiswa</title>
  <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
  <style>
    #loading { display: none; font-weight: bold; }
  </style>
</head>
<body class="p-4">
  <div class="container">
    <h2 class="mb-4">Live Search Mahasiswa (AJAX + MySQL)</h2>
    <input type="text" id="search" class="form-control mb-3" placeholder="Ketik nama atau
NIM...">
    <div id="loading">Mencari...</div>
    <table class="table table-bordered table-striped">
      <thead>
        <tr>
          <th>NIM</th>
          <th>Nama</th>
          <th>Jurusan</th>
        </tr>
      </thead>
      <tbody id="result"></tbody>
    </table>
  </div>
  <script>
    const searchBox = document.getElementById("search");
    const result = document.getElementById("result");
    const loading = document.getElementById("loading");

    searchBox.addEventListener("keyup", function() {
      const keyword = searchBox.value.trim();
      if (keyword.length === 0) {
        result.innerHTML = "";
```

```
      return;
    }

    loading.style.display = "block";

    fetch("search.php?keyword=" + encodeURIComponent(keyword))
      .then(res => res.json())
      .then(data => {
        loading.style.display = "none";
        result.innerHTML = "";

        if (data.length === 0) {
          result.innerHTML = "<tr><td colspan='3' class='text-center'>Data tidak
ditemukan</td></tr>";
        } else {
          data.forEach(row => {
            result.innerHTML += `<tr>
              <td>${row.nim}</td>
              <td>${row.nama}</td>
              <td>${row.jurusan}</td>
            </tr>`;
          });
        }
      });
  });
 </script>
</body>
</html>
```

Penjelasan kode:

Bagian deteksi inputan user

- searchBox.addEventListener("keyup", function() {
  searchBox adalah elemen input HTML:

<input type="text" id="search" ... >

- addEventListener("keyup", ...):
  Kita menambahkan event listener ke elemen input ini, supaya setiap kali pengguna menekan tombol keyboard (dan melepaskannya) saat mengetik, fungsi di dalamnya akan dijalankan. Ini memungkinkan live search (pencarian secara langsung saat mengetik) tanpa harus klik tombol.
- const keyword = searchBox.value.trim();
  searchBox.value: mengambil teks yang diketik pengguna di input.
  .trim(): menghapus spasi di awal dan akhir input. Jadi kalau user nggak sengaja kasih spasi, nggak dianggap sebagai input kosong.
- if (keyword.length === 0) {
  result.innerHTML = "";
  return;
  }
  Mengecek apakah input kosong (tidak ada karakter setelah di-trim). Kalau kosong, kita bersihkan hasil pencarian (result.innerHTML = "") dan return, artinya menghentikan fungsi, tidak akan lanjut mengirim request AJAX ke search.php.
- Intinya bagian ini bertugas:
  - Mendeteksi setiap kali user mengetik sesuatu.
  - Mengambil kata kunci pencarian.
  - Jika input kosong, bersihkan hasil dan hentikan proses pencarian.

Bagian Kirim permintaan ke search.php

- fetch("search.php?keyword=" + encodeURIComponent(keyword))
  fetch() digunakan untuk melakukan permintaan HTTP secara asinkron (tanpa reload halaman). Kita mengakses file PHP bernama search.php, dan mengirimkan parameter keyword lewat URL.

  encodeURIComponent(keyword) digunakan untuk memastikan karakter khusus (spasi, simbol, dll.) tidak merusak format URL.

  Misal pengguna mengetik "Dewi", maka permintaan menjadi:

  search.php?keyword=Dewi

Bagian Ambil respons dan ubah jadi JSON

- .then(res => res.json())

  Setelah fetch berhasil, server akan mengembalikan respons. **res.json()** digunakan untuk mengubah hasil respons dari format JSON menjadi **JavaScript object** agar bisa diolah di sisi frontend. Ini cocok karena search.php mengembalikan echo json_encode(...) yaitu format JSON.

Bagian Tampilkan hasilnya di tabel

- .then(data => {

  loading.style.display = "none";

  result.innerHTML = "";

  Setelah data berhasil diambil, kita sembunyikan teks loading (Mencari...) dengan style.display = "none". Lalu kita kosongkan isi <tbody id="result"> agar data baru bisa ditampilkan (menghindari tumpukan data lama).

Bagian Kalau tidak ada data

- if (data.length === 0) {

  result.innerHTML = "<tr><td colspan='3' class='text-center'>Data tidak ditemukan</td></tr>";

  Kalau array data kosong (tidak ada mahasiswa yang cocok), maka kita tampilkan 1 baris tabel dengan pesan **"Data tidak ditemukan"**. colspan="3" agar pesan memenuhi 3 kolom (NIM, Nama, Jurusan).

Bagian Kalau ada data

- } else {

  data.forEach(row => {

  result.innerHTML += `<tr>

  <td>${row.nim}</td>

```
      <td>${row.nama}</td>
      <td>${row.jurusan}</td>
    </tr>`;
  });
}
```

Jika data ditemukan, kita loop (forEach) setiap barisnya dan tampilkan di tabel. Gunakan template string (${row.nama} dll.) untuk memasukkan nilai langsung ke dalam HTML.

🔹 **Kesimpulan: Proses AJAX Live Search 🔍**

1. User mengetik di input.
2. JavaScript menangkap input keyword.
3. AJAX request dikirim ke search.php.
4. PHP mengembalikan hasil pencarian dalam format JSON.
5. JavaScript menampilkan hasilnya di tabel secara real-time tanpa reload halaman.

✅ **Hasil Fitur:**

- Live search mahasiswa berdasarkan nama atau NIM
- Menampilkan hasil dalam tabel Bootstrap
- Menampilkan loader “Mencari...” saat pencarian berlangsung
- Data real-time dari MySQL

🎯 **Tugas Mahasiswa (Praktik Mandiri)**

Buat versi pengembangan dari live search:

- Menambahkan kolom "Jurusan"
- Gunakan efek loading atau spinner dari Bootstrap

- Optional: Tambahkan animasi hasil pencarian menggunakan fadeIn() jQuery atau transition CSS
- Simpan dalam github masing-masing dengan nama direktori tugas_ajax_xxxxx

---