MATERI UTS

Fungsi javascript:

```
function tambahkan(a, b) {
  return a + b;
}

var hasil = tambahkan(5, 3);
console.log("Hasil penambahan: " + hasil);
```

Conditional

```
var x = 7;
if (x < 5) {
  console.log("x kurang dari 5");
} else if (x === 5) {
  console.log("x sama dengan 5");
} else {
  console.log("x lebih besar dari 5");
}
```

DOM, atau Document Object Model, adalah representasi struktural dari dokumen HTML dan XML yang memungkinkan pemrogram untuk mengakses dan memanipulasi elemen-elemen pada halaman web atau dokumen XML melalui bahasa pemrograman seperti JavaScript. DOM adalah antarmuka pemrograman yang disediakan oleh browser web untuk memanipulasi elemen-elemen dalam halaman web.

Browser Object Model (BOM) adalah model objek yang menyediakan objek JavaScript untuk berinteraksi dengan browser web. Ini memungkinkan pengembang web untuk mengakses dan mengendalikan berbagai aspek dari browser seperti jendela, dokumen, lokasi, riwayat, pengaturan, dan lainnya melalui skrip JavaScript.

Berikut adalah beberapa komponen utama dalam Browser Object Model (BOM):

1. Window Object: Ini adalah objek utama dalam BOM. Ini mewakili jendela browser dan menyediakan metode dan properti untuk mengontrol dan berinteraksi dengan jendela. Anda dapat mengaksesnya dengan menggunakan `window` atau langsung menggunakan nama variabel, seperti `alert("Hello, world!");`, yang sebenarnya adalah `window.alert("Hello, world!");`.

2. Document Object: Ini adalah objek yang mewakili dokumen HTML dari halaman web saat ini. Anda dapat mengakses dan memanipulasi elemen HTML dalam dokumen dengan menggunakan objek `document`. Contohnya adalah mengubah isi elemen, menambahkan elemen baru, atau menghapus elemen.

3. Location Object: Ini digunakan untuk berinteraksi dengan informasi lokasi atau URL dari halaman web saat ini. Anda dapat mengakses alamat URL saat ini, mengarahkan halaman ke URL yang berbeda, atau mengakses parameter URL.

4. History Object: Ini memungkinkan Anda mengakses dan mengendalikan riwayat perambanan. Anda dapat melakukan navigasi maju, mundur, atau mengarahkan halaman ke URL tertentu dari riwayat.

5. Navigator Object: Ini berisi informasi tentang peramban yang digunakan oleh pengguna. Anda dapat menggunakan objek `navigator` untuk mendapatkan informasi tentang peramban, sistem operasi, perangkat yang digunakan, dan sebagainya.

6. Screen Object: Ini memberikan informasi tentang layar tempat peramban dijalankan. Anda dapat mengakses properti seperti resolusi layar dan mendeteksi berbagai properti tampilan seperti lebar dan tinggi layar.

7. Popup Alert, Prompt, dan Confirm: BOM juga menyediakan fungsi pop-up bawaan seperti `alert`, `prompt`, dan `confirm` yang digunakan untuk menampilkan pesan kepada pengguna.

BOM memberikan kontrol penuh kepada pengembang web untuk berinteraksi dengan berbagai aspek dari browser, yang memungkinkan pembuatan aplikasi web yang lebih kuat dan interaktif. Namun, penggunaan yang tidak benar atau berlebihan dari BOM juga dapat mengganggu pengalaman pengguna dan menciptakan potensi masalah keamanan, sehingga perlu digunakan dengan hati-hati.

Dalam JavaScript, ada beberapa metode yang dapat digunakan untuk menemukan elemen-elemen HTML di dalam dokumen. Berikut beberapa di antaranya:

1. getElementById(id)
   - Metode ini digunakan untuk menemukan elemen dengan ID tertentu. Ini adalah metode yang paling efisien karena ID harus unik dalam satu halaman.
   - Contoh:

```
var elem = document.getElementById("myElement");
```

2. getElementsByClassName(className)
   - Metode ini digunakan untuk mengambil semua elemen dengan kelas CSS tertentu dan mengembalikannya sebagai koleksi (NodeList).
   - Contoh:

```
var elements = document.getElementsByClassName("myClass");
```

3. getElementsByTagName(tagName)
   - Metode ini mengembalikan semua elemen dengan tag HTML tertentu sebagai koleksi (NodeList).
   - Contoh:

```
var elements = document.getElementsByTagName("div");
```

4. querySelector(selector)
   - Metode ini memungkinkan Anda untuk mencari elemen berdasarkan selektor CSS dan mengembalikan elemen pertama yang cocok.
   - Contoh:

```
var element = document.querySelector(".myClass");
```

5. querySelectorAll(selector)
   - Mirip dengan querySelector, tetapi mengembalikan semua elemen yang cocok dengan selektor CSS sebagai koleksi (NodeList).
   - Contoh:

```
var elements = document.querySelectorAll(".myClass");
```

6. getElementByName(name)
   - Metode ini mengambil elemen dengan atribut name, biasanya digunakan pada elemen-elemen dalam formulir HTML.
   - Contoh:

```
var element = document.getElementsByName("email")[0];
```

$_POST dan $_GET

$_POST dan $_GET adalah dua jenis variabel superglobal dalam PHP yang digunakan untuk mengambil data yang dikirim melalui metode POST dan GET dalam permintaan HTTP. Berikut adalah contoh penggunaan keduanya:

Contoh $_POST:

Misalkan Anda memiliki formulir HTML yang mengirimkan data menggunakan metode POST ke skrip PHP:

```
<form method="post" action="proses.php">
  <input type="text" name="nama" placeholder="Nama">
  <input type="email" name="email" placeholder="Email">
  <input type="submit" value="Kirim">
</form>
```

Sekarang, dalam skrip proses.php, Anda dapat mengakses data yang dikirimkan melalui $_POST:

```
$nama = $_POST['nama'];
$email = $_POST['email'];

echo "Nama: " . $nama;
echo "Email: " . $email;
```

Contoh $_GET:

Misalkan Anda memiliki tautan yang mengandung parameter GET:

```
<a href="profil.php?id=123&nama=John">Lihat Profil</a>
```

Dalam skrip profil.php, Anda dapat mengakses parameter GET menggunakan $_GET:

```
$id = $_GET['id'];
$nama = $_GET['nama'];

echo "ID: " . $id;
echo "Nama: " . $nama;
```

$_POST digunakan ketika Anda ingin mengirim data dalam permintaan HTTP yang tidak terlihat oleh pengguna (seperti ketika Anda mengirim kata sandi dalam formulir), sedangkan $_GET digunakan ketika Anda ingin mengirim data yang dapat dilihat oleh pengguna (seperti dalam tautan). Ingatlah bahwa data yang dikirim melalui $_GET muncul dalam URL, sedangkan data yang dikirim melalui $_POST tidak muncul dalam URL

Contoh validasi form

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $nama = $_POST["nama"];
    $email = $_POST["email"];
    $password = $_POST["password"];

    $errors = array();

    // Validasi nama
    if (empty($nama)) {
        $errors[] = "Nama harus diisi.";
    }

    // Validasi email
    if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
        $errors[] = "Email tidak valid.";
    }
```

```
    // Validasi password (minimal 6 karakter)
    if (strlen($password) < 6) {
        $errors[] = "Password harus memiliki minimal 6 karakter.";
    }

    // Jika tidak ada kesalahan validasi
    if (empty($errors)) {
        // Lakukan tindakan selanjutnya, seperti menyimpan data ke database
        echo "Pendaftaran berhasil!";
    } else {
        // Tampilkan pesan kesalahan
        foreach ($errors as $error) {
            echo $error . "<br>";
        }
    }
}
?>
```

Validasi form dengan js

```
<!DOCTYPE html>
<html>
<head>
    <title>Formulir Pendaftaran</title>
</head>
<body>
    <h2>Pendaftaran Pengguna Baru</h2>
    <form id="myForm">
        <label for="nama">Nama:</label>
        <input type="text" name="nama" id="nama" required>

        <label for="email">Email:</label>
        <input type="email" name="email" id="email" required>

        <label for="password">Password:</label>
        <input type="password" name="password" id="password" required>

        <input type="submit" value="Daftar">
    </form>

    <script>
        document.getElementById('myForm').addEventListener('submit',
function(event) {
```

```
        var nama = document.getElementById('nama').value;
        var email = document.getElementById('email').value;
        var password = document.getElementById('password').value;

        var errors = [];

        if (nama.trim() === '') {
            errors.push("Nama harus diisi.");
        }

        if (!isValidEmail(email)) {
            errors.push("Email tidak valid.");
        }

        if (password.length < 6) {
            errors.push("Password harus memiliki minimal 6 karakter.");
        }

        if (errors.length > 0) {
            event.preventDefault(); // Mencegah pengiriman formulir
            alert(errors.join("\n")); // Menampilkan pesan kesalahan
        }
    });

    function isValidEmail(email) {
        var emailPattern = /^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$/;
        return emailPattern.test(email);
    }
  </script>
</body>
</html>
```

Jawaban UTS 2021

1. Validasi Form dengan JS

```
<!DOCTYPE html>
<html>
 <head>
```

```
    <title>Soal Uts Nomor 1</title>
    <script>
      function validasi_form() {
        var nama = document.getElementById("nama").value;
        var email = document.getElementById("email").value;
        var bidang =
document.querySelector('input[name="bidang"]:checked');
        var subBidang = document.getElementById("sub_bidang").value;
        var password = document.getElementById("password").value;
        var konf_password =
document.getElementById("konf_password").value;

        // Validasi Nama
        if (nama.trim() === "") {
          alert("Nama tidak boleh kosong");
          return false;
        }

        // Validasi Email
        if (email.trim() === "") {
          alert("Email tidak boleh kosong");
          return false;
        }

        // Validasi Bidang
        if (!bidang) {
          alert("Pilih salah satu bidang");
          return false;
        }

        // Validasi Sub Bidang (Jika Bidang dipilih)
        if (bidang.value === "design" && subBidang === "none") {
          alert("Pilih sub bidang untuk Web Design");
          return false;
        }
        if (bidang.value === "program" && subBidang === "none") {
          alert("Pilih sub bidang untuk Web Programming");
          return false;
        }

        // Validasi Password
        if (password.length < 8) {
          alert("Password minimal 8 karakter");
```

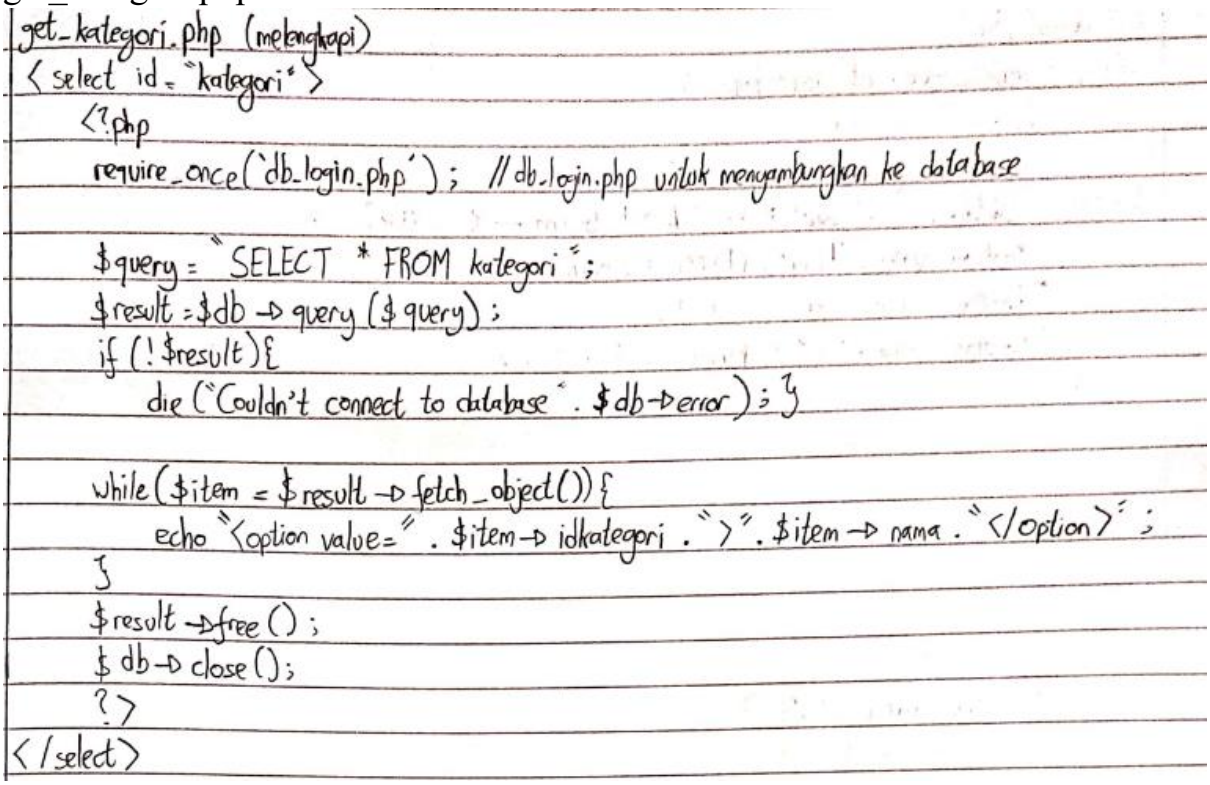
```
      return false;
    }

    // Validasi Konfirmasi Password
    if (konf_password !== password) {
      alert("Konfirmasi Password tidak sama dengan Password");
      return false;
    }

    return true;
  }

  function getSubbidang(bidang) {
    var subBidang = document.getElementById("sub_bidang");
    subBidang.innerHTML = "";

    if (bidang === "design") {
      var opsi = [
        "HTML",
        "CSS",
        "Javascript",
      ];
    } else if (bidang === "program") {
      var opsi = [
        "PHP",
        "AJAX",
        "Web Service",
      ];
    } else {
      var opsi = ["--Pilih Sub Bidang--"];
    }

    for (var i = 0; i < opsi.length; i++) {
      var option = document.createElement("option");
      option.value = opsi[i];
      option.text = opsi[i];
      subBidang.appendChild(option);
    }
  }
 </script>
</head>
<body>
 <form
```

```
    id="form_web_club"
    method="POST"
    action="proses_form.php"
    onsubmit="return validasi_form()"
  >
    <table>
     <tr>
      <td>Nama</td>
      <td><input type="text" id="nama" name="nama" /></td>
     </tr>

     <tr>
      <td>Email</td>
      <td><input type="email" id="email" name="email" /></td>
     </tr>
     <tr>
       <td>Bidang</td>
       <td>
        <input type="radio" name="bidang" value="design"
onchange="getSubbidang(this.value)" />Web Design
        <input type="radio" name="bidang" value="program"
onchange="getSubbidang(this.value)" />Web Programming
       </td>
      </tr>

     <tr>
      <td>Sub Bidang</td>
      <td>
       <select name="sub_bidang" id="sub_bidang">
        <option value="none">--Pilih Sub Bidang--</option>
        <!-- Tambahkan opsi sub bidang di sini -->
       </select>
      </td>
     </tr>
     <tr>
      <td>Password</td>
      <td><input type="password" id="password" name="password"
size="30" /></td>
     </tr>
     <tr>
      <td>Konfirmasi Password</td>
      <td><input type="password" id="konf_password"
name="konf_password" /></td>
```

```
      </tr>
      <tr>
        <td>Jadwal</td>
        <td>
          <input type="checkbox" name="jadwal[]" value="rabu" />Rabu
          <input type="checkbox" name="jadwal[]" value="kamis" />Kamis
          <input type="checkbox" name="jadwal[]" value="jumat" />Jumat
        </td>
      </tr>
      <tr>
        <td colspan="3"><br /><input type="submit" name="submit" /></td>
      </tr>
    </table>
  </form>
 </body>
</html>
```

2. Koneksi database dan nampilkan table

get_kategori.php



Ajax.js

```
// ajax.js
      function get_detail() {
         var xmlhttp = new XMLHttpRequest();

         // Mendefinisikan fungsi yang akan dipanggil saat permintaan
AJAX berhasil
         xmlhttp.onreadystatechange = function() {
            if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
               // Mengambil respon dari get_detail.php dan
memasukkannya ke dalam elemen "detail"
               document.getElementById("detail").innerHTML =
xmlhttp.responseText;
            }
         };

         // Mendapatkan nilai yang dipilih dari elemen "kategori"
         var selectedCategory =
document.getElementById("kategori").value;

         // Menjalankan permintaan AJAX
         xmlhttp.open("GET", "get_detail.php?kategori=" +
selectedCategory, true);
         xmlhttp.send();
      }
```

get_detail.php

```
<?php
   require_once("db_login.php");
   $db = new mysqli($db_host, $db_username, $db_password,
$db_database);
   if ($db->connect_errno) {
      die("Could not connect to the database: <br />" . $db-
>connect_error);
   }
   $id = $_GET['kategori'];

   $query = " SELECT * , subkategori, nama as namasub FROOM
produk INNER JOIN sub_kategori ON sub_kategori.idsub_kategori =
produk.idsub_kategori WHERE kategori.idkategori = ".$id;
   $result = $db->query($query);
```

```
    if(!$result){
        die("Could not query the database: <br />".$db->error);
    }
?>

<table>
    <tr>
        <th>Sub Kategori</th>
        <th>Nama</th>
        <th>Harga</th>
    </tr>
    <?php
        while($row = $result->fetch_object()){
            echo "<tr>";
            echo "<td>".$row->namasub."</td>";
            echo "<td>".$row->nama."</td>";
            echo "<td>".$row->harga."</td>";
            echo "</tr>";
        }
    ?>
</table>
<?php
    $result->free();
    $db->close();
?>
```

SOAP (Simple Object Access Protocol) dan REST (Representational State Transfer) adalah dua pendekatan yang berbeda untuk pembangunan layanan web. Berikut adalah perbedaan karakteristik dan elemen-elemen utama antara SOAP dan RESTful Web Service:

SOAP Web Service:
1. Protokol: SOAP adalah protokol yang terdefinisi dengan baik yang berarti ada panduan yang ketat untuk membuat dan mengakses layanan web dengan SOAP.
2. Pesan: Pesan SOAP menggunakan format XML yang sangat terstruktur. Ini mencakup elemen-elemen seperti `<Envelope>`, `<Header>`, dan `<Body>`.
3. Transport: SOAP dapat digunakan di berbagai protokol transportasi, seperti HTTP, SMTP, dan banyak lagi.

4. Pengkodean: Pesan SOAP harus dienkapsulasi dalam pesan HTTP atau dalam pesan protokol lainnya, sehingga lebih kompleks.
5. Metode: SOAP mendefinisikan empat jenis permintaan utama: Create, Read, Update, Delete (CRUD), yang sesuai dengan operasi database.
6. Kepamanan: SOAP menyediakan banyak langkah-langkah keamanan bawaan, seperti WS-Security.
7. Performa: SOAP cenderung lebih lambat dan memiliki ukuran pesan yang lebih besar karena format XML yang kuat.

RESTful Web Service:
1. Arsitektur: REST adalah arsitektur yang bersifat terbuka, yang berarti tidak ada panduan ketat yang harus diikuti. Ini didasarkan pada prinsip-prinsip seperti sumber daya, representasi, permintaan, tanggapan, dan aksi.
2. Pesan: Pesan RESTful menggunakan format yang lebih fleksibel, seperti XML, JSON, atau bahkan HTML. Ini tidak memiliki struktur pesan SOAP yang kaku.
3. Transport: RESTful web services umumnya berjalan di atas protokol HTTP, tetapi mereka bisa digunakan di berbagai protokol transportasi.
4. Pengkodean: Pesan RESTful tidak harus dienkapsulasi dalam pesan HTTP, yang membuatnya lebih sederhana.
5. Metode: RESTful web services tidak memiliki metode standar, tetapi mereka menggunakan metode HTTP standar seperti GET, POST, PUT, dan DELETE.
6. Keamanan: RESTful web services umumnya bergantung pada protokol keamanan tingkat transportasi seperti HTTPS. Keamanan per endpoint atau sumber daya biasanya diimplementasikan secara khusus.
7. Performa: RESTful web services cenderung lebih cepat dan memiliki ukuran pesan yang lebih kecil karena mereka menggunakan format yang lebih ringan, seperti JSON.

Dalam ringkasan, SOAP adalah protokol berorientasi tugas yang mendefinisikan prosedur yang ketat untuk berkomunikasi, sementara REST adalah pendekatan arsitektur yang lebih terbuka dan berbasis pada sumber daya dan prinsip-prinsip HTTP yang ada. Pemilihan antara keduanya bergantung pada kebutuhan proyek, dan terkadang keduanya digunakan dalam situasi yang berbeda dalam proyek yang sama.

Soal UTS 2022

1. Validasi Form

```
function validate_form() {
   var kelas = document.getElementById("kelas").value;
   var ekstrakur = document.getElementsByName("ekstrakur[]");
   var selectedEkstrakurCount = 0;
   for (var i = 0; i < ekstrakur.length; i++) {
      if (ekstrakur[i].checked) {
         selectedEkstrakurCount++;
      }
   }
   if (kelas === "10" && selectedEkstrakurCount < 1) {
      alert("Siswa kelas X minimal harus memilih satu ekstrakurikuler.");
      return false;
   } else if (kelas === "11" && selectedEkstrakurCount < 2) {
      alert("Siswa kelas XI minimal harus memilih dua ekstrakurikuler.");
      return false;
   } else if (kelas === "12" && selectedEkstrakurCount < 3) {
      alert("Siswa kelas XII minimal harus memilih tiga ekstrakurikuler.");
      return false;
   }
   return true;
}
```

2. AJAX dan php

File yang di-request(check_email.php)

```
<?php
   if ($_SERVER["REQUEST_METHOD"] == "POST") {
      $email = $_POST['email'];
      $query = "SELECT * FROM pengguna WHERE email = '$email'";
      $result = $db->query($query);
      if ($result->num_rows > 0) {

      $konf_email = "Email sudah digunakan";
      } else {
      $konf_email = "Email dapat digunakan";
      }
```

```
    }
?>
```

Ajax.js

```
function getXMLHttpRequest() {
  if (window.XMLHttpRequest) {
    return new XMLHttpRequest();
  } else {
    return new ActiveXObject("Microsoft.XMLHTTP");
  }
}
function check_email() {
  var email = $("#email").val();
  var inner = "konf_email";
  var url = "check_email.php?email=" + email;
  var xmlhttp = getXMLHttpRequest();
  xmlhttp.open("GET", url, true);
  xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
      document.getElementById(inner).innerHTML =
xmlhttp.responseText;
    }
    return false;
  };
  xmlhttp.send(null);
}
```

Email_input.php(file yang merequest)

```
<form id="form_email">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email"><br>
  <div id="konf_email"></div><br>

  <button type="button" id="btn_email">Check Email</button>
</form>
```

SQL INJECTION & PREPARED STATEMENT

SQL injection adalah jenis serangan keamanan yang digunakan oleh penyerang untuk memanipulasi perintah SQL yang dieksekusi oleh aplikasi. Serangan ini terjadi ketika aplikasi web atau perangkat lunak lainnya tidak memvalidasi atau

menyaring input pengguna dengan benar sebelum mengirimnya ke database. Penyerang dapat memasukkan kode SQL berbahaya ke dalam input yang dieksekusi oleh aplikasi, dan jika tidak ada perlindungan yang memadai, kode SQL ini akan dijalankan oleh database. Dampak dari serangan SQL injection bisa sangat merusak, termasuk kehilangan data, pengungkapan informasi sensitif, dan bahkan pengambilalihan sistem.

Prepared statement, di sisi lain, adalah salah satu teknik yang digunakan untuk mencegah serangan SQL injection. Ini adalah cara untuk menulis pernyataan SQL dengan parameter yang diikat ke nilai yang disediakan oleh pengguna, tanpa memasukkan nilai pengguna langsung ke dalam pernyataan SQL. Sebagai contoh, jika Anda ingin mengambil data dari database berdasarkan input pengguna, Anda dapat menggunakan prepared statement seperti ini dalam SQL:

```
SELECT * FROM users WHERE username = ?;
```

Tanda tanya (?) adalah placeholder untuk parameter. Kemudian, Anda mengikat nilai yang dimasukkan oleh pengguna ke parameter ini menggunakan metode atau fungsi yang disediakan oleh bahasa pemrograman atau framework yang Anda gunakan. Dengan menggunakan prepared statement, database akan memperlakukan nilai pengguna sebagai data, bukan sebagai kode SQL yang dapat dieksekusi. Ini secara efektif mencegah serangan SQL injection karena input pengguna tidak dianggap sebagai perintah SQL.

Berikut adalah contoh menggunakan prepared statement dalam bahasa pemrograman PHP:

```
<?php
$pdo = new PDO("mysql:host=hostname;dbname=database", "username", "password");
$username = $_POST['username'];
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
$stmt->execute([$username]);
$result = $stmt->fetch();
```

```
?>
```

Dalam contoh di atas, `$username` diikat ke placeholder `?` dalam prepared statement, dan kemudian pernyataan SQL dieksekusi dengan aman tanpa risiko serangan SQL injection. Prepared statement adalah praktik terbaik dalam menghindari serangan SQL injection dan menjaga keamanan aplikasi web atau perangkat lunak yang menggunakan database.

```php
// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname, lastname, email) VALUES (?, ?, ?)");
$stmt->bind_param("sss", $firstname, $lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
```

Cookies dan Session adalah dua metode yang digunakan dalam pengembangan web untuk menyimpan data di sisi klien (pada peramban pengguna) atau di sisi server (pada server web).

Cookies:

- Cookies adalah data kecil yang disimpan di peramban web pengguna.

- Ini dapat digunakan untuk menyimpan informasi seperti preferensi pengguna, riwayat sesi, dan data lain yang perlu dipertahankan antar kunjungan.

- Cookies memiliki batas kapasitas yang lebih kecil, biasanya sekitar 4KB per cookie.

- Cookies dapat disimpan selama waktu yang telah ditentukan atau selama sesi peramban.

- Dapat diakses dan dimodifikasi oleh klien (pengguna) dan server.

- Membuat dan mengakses cookie dengan JavaScript:

```
// Membuat cookie
document.cookie = "nama=John; expires=Thu, 18 Dec 2023 12:00:00 UTC; path=/";


// Mengakses cookie
var x = document.cookie;
```

Session:

- Session adalah mekanisme server untuk menyimpan data yang terkait dengan sesi pengguna.

- Data sesi disimpan di server dan diidentifikasi oleh token atau ID sesi yang diberikan ke peramban pengguna.

- Data sesi lebih aman karena disimpan di server, bukan di peramban pengguna.

- Tidak ada batasan kapasitas seperti pada cookies.

- Data sesi akan dihapus setelah sesi pengguna berakhir atau sesuai dengan konfigurasi server.

- Contoh kodingan PHP untuk mengatur dan mengakses data sesi:

```
// Memulai sesi
session_start();


// Menyimpan data dalam sesi
```

```
$_SESSION['user'] = 'John';


// Mengakses data sesi
$user = $_SESSION['user'];
```

Contoh Kodingan untuk Mengelola Cookies di PHP:

1. Membuat Cookie:

```
setcookie("username", "John", time() + 3600, "/");
```

2. Mengakses Cookie:

```
if (isset($_COOKIE["username"])) {
    $username = $_COOKIE["username"];
    echo "Welcome, " . $username;
} else {
    echo "Cookie 'username' tidak ada.";
}
```

3. Modifikasi Cookie:

```
if (isset($_COOKIE["username"])) {
    $username = $_COOKIE["username"];
    // Modifikasi nilai cookie
    setcookie("username", "Doe", time() + 3600, "/");
}
```

4. Menghapus Cookie:

```
if (isset($_COOKIE["username"])) {
```

```
    // Menghapus cookie dengan mengatur waktu kedaluwarsa ke masa lalu
    setcookie("username", "", time() - 3600, "/");
}
```

Catatan: Ingatlah untuk selalu memperlakukan data sensitif dengan hati-hati dan tidak menyimpan data rahasia seperti kata sandi dalam cookies karena ini dapat membahayakan keamanan. Penggunaan sesi lebih aman untuk data sensitif.

Di sini, saya akan memberikan contoh penggunaan session dalam PHP untuk akses, modifikasi, dan menghapusnya:

1. Akses Session:

```
// Mulai sesi
session_start();

// Menyimpan data dalam sesi
$_SESSION['user'] = 'John';

// Mengakses data sesi
if (isset($_SESSION['user'])) {
    $user = $_SESSION['user'];
    echo "Welcome, " . $user;
} else {
    echo "Data sesi tidak ditemukan.";
}
```

2. Modifikasi Session:

```
// Mulai sesi
```

```
session_start();

// Mengubah data sesi
if (isset($_SESSION['user'])) {
    $_SESSION['user'] = 'Jane'; // Modifikasi data sesi
    echo "Data sesi telah diubah.";
} else {
    echo "Data sesi tidak ditemukan.";
}
```

3. Hapus Session:

```
// Mulai sesi
session_start();

// Menghapus data sesi
if (isset($_SESSION['user'])) {
    unset($_SESSION['user']); // Hapus data sesi
    echo "Data sesi telah dihapus.";
} else {
    echo "Data sesi tidak ditemukan.";
}

// Menutup sesi
session_destroy();
```

Perhatikan bahwa dalam contoh ini, kita memulai sesi dengan `session_start()`. Kemudian kita dapat mengakses data sesi dengan mengakses `$_SESSION['key']`, mengubah data sesi dengan mengubah nilainya, dan

menghapus data sesi dengan menggunakan `unset()`. Sesudah selesai, sesi dapat dihentikan dengan `session_destroy()`.