# **Name: Raka Mahardika**

# **Email:** [**raka.mahardika@idstar.co.id**](mailto:raka.mahardika@idstar.co.id)

# **Learning Path: React JS**

# **Scenario**

1. Buatlah sebuah form input yang memungkinkan karyawan untuk memasukkan tugas baru.
2. Tampilkan daftar tugas yang sudah ditambahkan dengan menggunakan elemen HTML yang sesuai.
3. Berikan opsi untuk menandai tugas sebagai selesai atau belum selesai.
4. Tambahkan fitur penghapusan tugas dari daftar.
5. Tampilkan jumlah total tugas yang belum selesai.
6. Tampilkan pesan atau notifikasi jika karyawan telah menyelesaikan semua tugas.
7. Buatlah tombol untuk menghapus semua tugas sekaligus.

**Output**

| Name | Form Input New Task |
| --- | --- |
| Scenario | Buatlah sebuah form input yang memungkinkan karyawan untuk memasukkan tugas baru. |
| Output |  |

| Name | Show List Task |
| --- | --- |
| Scenario | Tampilkan daftar tugas yang sudah ditambahkan dengan menggunakan elemen HTML yang sesuai. |
| Output |  |

| Name | Selected Finished Task |
| --- | --- |
| Scenario | Berikan opsi untuk menandai tugas sebagai selesai atau belum selesai. |
| Output |  |

| Name | Remove Task |
| --- | --- |
| Scenario | 1. Tambahkan fitur penghapusan tugas dari daftar. 2. Tampilkan jumlah total tugas yang belum selesai. |
| Output |  |

| Name | Remove All Task and Show Notification |
| --- | --- |
| Scenario | 1. Tampilkan pesan atau notifikasi jika karyawan telah menyelesaikan semua tugas. 2. Buatlah tombol untuk menghapus semua tugas sekaligus. |
| Output |  |

# **Soal Essay**

1. Jelaskan perbedaan antara "bind()", "call()", dan "apply()" dalam konteks pengaturan nilai "this" dalam fungsi.
2. Soal: 2. Apa itu "memoization" dalam konteks pemrograman JavaScript? Jelaskan bagaimana memoization dapat meningkatkan performa fungsi.
3. Soal: 3. Jelaskan perbedaan antara "synchronous" dan "asynchronous" dalam konteks pemrograman JavaScript.
4. Soal: 4. Apa itu "generator" dalam JavaScript? Jelaskan bagaimana cara kerja generator.
5. Soal: 5. Jelaskan perbedaan antara "Promise" dan "async/await" dalam JavaScript.

**Output**

1. Jelaskan perbedaan antara "bind()", "call()", dan "apply()" dalam konteks pengaturan nilai "this" dalam fungsi.

- **bind()**: Metode bind() digunakan untuk membuat salinan fungsi dengan nilai "this" tertentu, tanpa menjalankan fungsi tersebut. Ini memungkinkan kita mengaitkan nilai "this" dengan fungsi dan menyimpannya untuk penggunaan di kemudian hari.

- **call()**: Metode call() digunakan untuk menjalankan fungsi dan secara eksplisit menetapkan nilai "this" bersama dengan argumen yang dipisahkan koma. Dengan menggunakan call(), kita dapat memanggil fungsi segera dengan nilai "this" yang diinginkan.

- **apply()**: Metode apply() serupa dengan call(), namun argumen disediakan dalam bentuk array. Ini berguna ketika jumlah argumen yang akan disertakan dalam fungsi tidak diketahui secara pasti atau dapat bervariasi.

1. Apa itu "memoization" dalam konteks pemrograman JavaScript? Jelaskan bagaimana memoization dapat meningkatkan performa fungsi.

- **Memoization**: Memoization adalah teknik pengoptimalan di pemrograman di mana hasil dari pemanggilan fungsi disimpan untuk argumen tertentu. Jika fungsi dipanggil lagi dengan argumen yang sama, hasilnya diambil dari cache, menghindari eksekusi fungsi yang sama berulang kali.

Memoization dapat meningkatkan performa fungsi dengan mengurangi jumlah kali fungsi dieksekusi untuk argumen yang sama. Dengan menyimpan hasil di cache, fungsi tidak perlu dijalankan ulang, menghemat waktu komputasi, terutama untuk operasi yang mahal secara komputasi.

1. Jelaskan perbedaan antara "synchronous" dan "asynchronous" dalam konteks pemrograman JavaScript.

- **Synchronous**: Operasi sinkron atau synchronous terjadi satu per satu, dijalankan dalam urutan yang ditentukan. Setiap operasi menunggu operasi sebelumnya selesai sebelum dimulainya.

- **Asynchronous**: Operasi asynchronous memungkinkan eksekusi beberapa operasi tanpa harus menunggu operasi sebelumnya selesai. Ini memberikan kemampuan untuk menjalankan tugas tanpa memblokir eksekusi program secara keseluruhan.

1. Apa itu "generator" dalam JavaScript? Jelaskan bagaimana cara kerja generator.

- **Generator**: Generator adalah fungsi khusus dalam JavaScript yang menghasilkan serangkaian nilai menggunakan pernyataan `yield`. Generator dapat dihentikan sementara dan dilanjutkan, memungkinkan eksekusi iteratif yang lincah.

Cara Kerja: Ketika fungsi generator dipanggil, ia mengembalikan objek generator yang dapat digunakan untuk mengontrol eksekusi fungsi. Setiap kali `yield` dipanggil, eksekusi dihentikan dan nilai dikembalikan. Kemudian, ketika fungsi generator dipanggil lagi, eksekusi dilanjutkan dari titik `yield` terakhir.

1. Jelaskan perbedaan antara "Promise" dan "async/await" dalam JavaScript.

- **Promise**: Promise adalah objek yang mewakili hasil sukses atau kegagalan dari operasi asinkron. Ini memungkinkan penanganan lebih mudah terhadap operasi yang membutuhkan waktu, seperti pengambilan data dari server.

- **async/await**: async dan await adalah fitur dalam JavaScript yang menyederhanakan penanganan operasi asinkron. Keyword `async` digunakan untuk membuat fungsi asynchronous, sementara `await` digunakan di dalamnya untuk menunggu hasil dari operasi asinkron, menjadikannya lebih mudah dibaca dan ditulis daripada menggunakan Promise secara langsung.