1. Jelaskan perbedaan antara "bind()", "call()", dan "apply()" dalam konteks pengaturan nilai "this" dalam fungsi.
   Jawaban:
   - bind(): Mengembalikan fungsi baru dengan "this" yang ditetapkan secara permanen. Contoh:
   ```
   const obj = { x: 42 };
   function getX() { return this.x; }
   const boundGetX = getX.bind(obj);
   console.log(boundGetX()); // 42
   ```

   - call(): Memanggil fungsi dengan "this" dan argumen yang ditentukan. Contoh:
   ```
   const obj = { x: 42 };
   function printX() { console.log(this.x); }
   printX.call(obj); // 42
   ```

   - apply(): Mirip dengan call(), tetapi menerima array sebagai argumen. Contoh:
   ```
   const obj = { x: 42 };
   function printX() { console.log(this.x); }
   printX.apply(obj); // 42
   ```

2. Apa itu "memoization" dalam konteks pemrograman JavaScript? Jelaskan bagaimana memoization dapat meningkatkan performa fungsi.
   Jawaban:
   Memoization adalah teknik caching hasil dari fungsi yang mahal secara komputasi untuk meningkatkan performa dengan menghindari perhitungan berulang untuk argumen yang sama. Contoh:
   ```
   function memoize(fn) {
     const cache = new Map();
     return function(...args) {
       const key = JSON.stringify(args);
       if (!cache.has(key)) {
         cache.set(key, fn(...args));
       }
       return cache.get(key);
     };
   }

   // Contoh penggunaan memoization
   function expensiveCalculation(num) {
     console.log('Menghitung...');
     return num * num;
   }

   const memoizedCalculation = memoize(expensiveCalculation);
   console.log(memoizedCalculation(5)); // Menghitung... kemudian 25
   console.log(memoizedCalculation(5)); // Langsung 25 tanpa menghitung ulang
   ```

3. Jelaskan perbedaan antara "synchronous" dan "asynchronous" dalam konteks pemrograman JavaScript.
   Jawaban:
   - Synchronous: Kode dieksekusi secara berurutan, satu per satu. Contoh:
   ```
   console.log('A');
   console.log('B');
   ```
   Output: A, B

   - Asynchronous: Kode dapat dieksekusi di luar urutan, seperti pada operasi non-blocking. Contoh:
   ```
   console.log('A');
   setTimeout(() => console.log('B'), 0);
   console.log('C');
   ```
   Output: A, C, B

4. Apa itu "generator" dalam JavaScript? Jelaskan bagaimana cara kerja generator.
   Jawaban:
   Generator adalah fungsi yang dapat berhenti sementara dan melanjutkan eksekusi. Digunakan `function*` dan `yield`. Contoh:
   ```
   function* generatorExample() {
     yield 1;
     yield 2;
     yield 3;
   }
   const gen = generatorExample();
   console.log(gen.next().value); // 1
   console.log(gen.next().value); // 2
   console.log(gen.next().value); // 3
   ```

5. Jelaskan perbedaan antara "Promise" dan "async/await" dalam JavaScript.
   Jawaban:
   - Promise: Objek yang merepresentasikan penyelesaian atau kegagalan operasi asinkron. Contoh:
   ```
   const promise = new Promise((resolve, reject) => {
     setTimeout(() => resolve('done'), 1000);
   });
   promise.then(result => console.log(result));
   ```

   - Async/await: Sintaks sugar untuk menangani Promise, membuat kode lebih bersih dan mudah dibaca. Contoh:
   ```
   async function asyncExample() {
     const result = await promise;
     console.log(result);
   }
   asyncExample();
   ```