1. **Jelaskan mengenai anonymous function di Javascript! Sertakan juga contoh kode program untuk mendukung penjelasan anda.**

Anonymous function di JavaScript adalah fungsi yang tidak memiliki nama (identifier) dan biasanya dibuat saat runtime. Fungsi ini dapat dideklarasikan dan segera digunakan tanpa harus memberinya nama terlebih dahulu.

Contoh anonymus function sebagai ekspresi fungsi:

```
const sayHello = function() {
        console.log("Hello, Welcome ");
};
sayHello();
```

2. **Berikan 5 contoh potongan kode program di Javascript yang memanfaatkan *callback*! Berikan juga penjelasan mengenai masing-masing kode program tersebut.**

Contoh 1 Callback dengan Fungsi Asinkron (setTimeout)

```
function sayHello(name, callback) {
  setTimeout(function() {
    console.log("Hello, " + name);
    callback();
  }, 1000);
}

function afterGreeting() {
  console.log("Greeting completed!");
}

sayHello("Gerald", afterGreeting);
```

Contoh 2 Callback untuk Mengambil Data dari Array

```
function getUserInfo(userId, callback) {
  const users = [
    { id: 1, name: "Gerald" },
    { id: 2, name: "Thomas" }
  ];

  const user = users.find(u => u.id === userId);
  callback(user);
}

getUserInfo(1, function(user) {
  console.log(user.name);
});
```

Contoh 3 Callback untuk Menangani Error

```
function divide(a, b, callback) {
  if (b === 0) {
    callback("Error: Division by zero");
    return;
  }
  callback(null, a / b);
}

divide(10, 2, function(error, result) {
  if (error) {
    console.log(error);
  } else {
    console.log(result);
  }
});

divide(10, 0, function(error, result) {
  if (error) {
    console.log(error);
  }
});
```

Contoh 4 Callback dengan Fungsi yang Memodifikasi Data

```
function modifyData(arr, callback) {
  const modifiedArr = arr.map(num => num * 2);
  callback(modifiedArr);
}

modifyData([1, 2, 3], function(result) {
  console.log(result);
});
```

Contoh 5 Callback untuk Menggabungkan Dua Fungsi

```
function add(a, b, callback) {
  callback(a + b);
}

function multiply(a, b, callback) {
  callback(a * b);
}

add(2, 3, function(sum) {
  console.log("Sum: " + sum);
  multiply(sum, 2, function(product) {
    console.log("Product: " + product);
  });
});
```

3. **Berikan sebuah contoh kasus dan solusi kode programnya yang membandingkan eksekusi menggunakan perintah yang synchronous, callback, promise, dan async/await. Kode program yang sama tetapi menggunakan 4 cara yang berbeda.**

Contoh Synchronous

```
function fetchData1() { return "Data 1"; }
function fetchData2() { return "Data 2"; }

function executeSynchronous() {
  const data1 = fetchData1();
  const data2 = fetchData2();
  console.log(data1 + " dan " + data2);
}

executeSynchronous();
```

Contoh Callback

```
function fetchData1(callback) { setTimeout(() => callback("Data 1"), 1000); }
function fetchData2(callback) { setTimeout(() => callback("Data 2"), 1000); }

function executeWithCallback() {
  fetchData1(data1 => {
    fetchData2(data2 => {
      console.log(data1 + " dan " + data2);
    });
  });
}

executeWithCallback();
```

Contoh Promise

```
function fetchData1() { return new Promise(resolve => setTimeout(() => resolve("Data 1"), 1000)); }

function fetchData2() { return new Promise(resolve => setTimeout(() => resolve("Data 2"), 1000)); }


function executeWithPromise() {
  fetchData1().then(data1 => {
    return fetchData2().then(data2 => {
      console.log(data1 + " dan " + data2);
    });
  });
}


executeWithPromise();
```

Contoh Async/Await

```
function fetchData1() { return new Promise(resolve => setTimeout(() => resolve("Data 1"), 1000)); }

function fetchData2() { return new Promise(resolve => setTimeout(() => resolve("Data 2"), 1000)); }


async function executeWithAsyncAwait() {
  const data1 = await fetchData1();
  const data2 = await fetchData2();
  console.log(data1 + " dan " + data2);
}


executeWithAsyncAwait();
```