Essential Javascript for React (Fundamentals)

Introduction

JavaScript adalah bahasa pemrograman yang digunakan dalam pengembangan website agar lebih dinamis dan interaktif. Sifat dari JavaScript adalah:

1. **Dinamis** — semuanya dapat berubah, misal kita dapat menambah dan menghapus properties pada object setelah dibuat. Kita juga dapat membuat object tanpa harus membuat object factory (contoh: class) terlebih dahulu.
2. **Dynamic Typing** — Variables dan object properties dapat selalu menyimpan nilai dari berbagai macam tipe.
3. **Functional dan object oriented** — JavaScript mendukung 2 paradigma bahasa pemrograman *functional programming* (first-class functions, closures, partial application via

`bind()`, built-in `map()` dan `reduce()`, untuk array, dll) dan *object-oriented programming* (mutable state, objects, inheritance, dll).

ES6 adalah kependekan dari EcmaScript 6, standarisasi kode Javascript bernama ECMAScript atau ES. ES6 merupakan versi JS yang ke 6 dan dipublikasikan di tahun 2015. Banyak JavaScript framework yang sudah menggunakan fitur ES6 salah satunya React, sehingga kita perlu memiliki pengetahuan dasar tentang fitur JavaScript ES6 yang biasa digunakan untuk mendukung pembuatan komponen pada React.

Preparation

Membuat 2 files, `index.html` dan `index.js`

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Pertemuan 2</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9"
      crossorigin="anonymous"
    />
  </head>
  <body>
    <h1>Javascript Fundamental</h1>
    <p id="demo"></p>

    <script src="index.js"></script>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm"
      crossorigin="anonymous"
    ></script>
  </body>
</html>
```

index.js

```
const demo = document.getElementById("demo");
```

Variabel

Variabel merupakan “named space in the memory” yang menyimpan value. Dengan kata lain, variabel berfungsi sebagai wadah untuk suatu nilai dalam suatu program. Nama variabel disebut dengan *identifier*. Berikut adalah aturan untuk *identifier*:

- I*dentifier* tidak dapat berupa kata kunci (atau reserved word yang memiliki makna spesifik dan tidak dapat digunakan ditempat lain, contoh: while, case, true, await, etc) .
- I*dentifier* dapat berisi huruf dan angka.
- I*dentifier* tidak boleh mengandung spasi dan karakter khusus, kecuali garis bawah (_) dan tanda dolar ($).
- Nama variabel tidak boleh dimulai dengan angka.

Type Syntax

Variabel harus dideklarasikan sebelum digunakan (hoisting). ES5 syntax menggunakan `var` keyword, sedangkan ES6 syntax mengenalkan variable declaration syntax seperti `const` dan `let`.

> 💡 JavaScript merupakan `un-type language` yang berarti variabel nya dapat menyimpan nilai dari berbagai tipe data. Tidak seperti kebanyakan bahasa pemrograman, kita tidak perlu menyebutkan tipe variabel saat melakukan deklarasi. Nilai dari variabel dapat berubah selama program dieksekusi dan JavaScript dapat mengurusnya secara otomatis. Fitur tersebut dinamakan `dynamic typing`.

Variable Scope

Variable scope adalah batasan dalam program di mana variabel tersebut didefinisikan. JavaScript mendefinisikan variable scope yaitu global dan local.

- Global Scope: Dapat diakses dari bagian manapun dalam kode JavaScript.
- Local Scope: Hanya dapat diakses dari dalam suatu blok fungsi di mana variabel tersebut dideklarasikan.
- Local Scope: Hanya dapat diakses dari dalam suatu blok fungsi di mana variabel tersebut dideklarasikan.

let vs var block scope

Block scope membatasi akses variabel ke dalam blok variabel yang dideklarasikan.

- var memberikan scope fungsi pada variabel.
- let membatasi akses variabel hanya pada blok terdekat.

```
const testLet = () => {
  let num = 100;
  {
    let num = 200;
  }
  return num;
}
document.getElementById("demo").innerHTML = testLet();

const testVar = () => {
  var num = 100;
  {
    var num = 200;
  }
  return num ;
}
document.getElementById("demo").innerHTML = testVar();
```

const variable scope

Deklarasi const membuat referensi *read-only* pada suatu nilai. Dengan menggunakan deklarasi const bukan berarti nilainya tidak dapat diubah atau *immutable*, tetapi *variable identifier*-nya yang tidak dapat dalokasikan ulang atau *reassigned*.

> 💡 **Nilai pada objek dan array adalah mutable**
> Meskipun *reassignment* tidak diizinkan untuk konstanta yang dideklarasikan dengan const, nilai-nilai dari objek dan array masih dapat dubah karena const digunakan untuk mendeklerasikan referensi konstan pada suatu nilai tertentu. Referensi tidak dapat diubah sehingga tidak dapat melakukan *reassignment* nilai baru (baik berupa array lain atau yang lainnya) ke konstanta tersebut. Tetapi kita dapat mengubah nilai dari array tersebut. Selain itu, ketika konten berupa objek, maka properti dalam objek dapat diubah.

Deklarasi const berada dalam *block scope*, seperti variabel yang didefinisikan dengan deklarasi let. Nilai dari sebuah deklarasi const tidak dapat berubah melalui *re-assignment*, dan tidak dapat dideklarasikan ulang.
Aturan-aturan berikut berlaku untuk variabel yang dideklarasikan menggunakan deklarasi const:

- Const tidak dapat di-assign kembali dengan suatu nilai.

- Const tidak dapat dideklarasikan ulang.
- Const membutuhkan inisialisasi saat deklarasi
- Nilai yang di-*assign* ke variabel konstan adalah *mutable* atau dapat diubah.

```
try {
  const PI = 3.141592653589793;
  PI = 3.14;
  document.getElementById("demo").innerHTML = PI;
}
catch (err) {
  document.getElementById("demo").innerHTML = err;
}
```

Pada praktiknya, `const` dan `let` lebih sering dan aman digunakan dibandingkan dengan `var` karena nilai dari kedua *variable scope* tersebut tidak mudah untuk dideklarasikan ulang atau diubah dengan nilai yang baru.

Keyword	Scope	Hoisting	Can Be Reassigned	Can Be Redeclared
`var`	Function scope	Yes	Yes	Yes
`let`	Block scope	No	Yes	No
`const`	Block scope	No	No	No

Tipe Data

Javascript memiliki 2 data type yaitu primitive dan references.

Tipe Data Primitive

Tipe data *primitive* adalah tipe data yang memiliki ukuran yang fix pada memori, contohnya: number hanya memiliki ukuran 8 byte pada memori, dan boolean hanya dapat direpresentasikan pada 1 bit saja. Number, boolean, null dan undefined adalah tipe data primitive.

```
// Deklarasi variable tipe data primitive
let name = 'Mita' //string
let age = 25 // number
let isApproved = true //boolean literal
let firstName // not defined -> undefined
```

```
let lastName = null // empty the variable, type of var : object
document.getElementById("demo").innerHTML = `<p>This Undefined variable is: ${firstName}</p>`

// Tipe data Primitive
var a = 3.14;  // Declare and initialize a variable
var b = a;     // Copy the variable's value to a new variable
a = 4;         // Modify the value of the original variable
document.getElementById("demo").innerHTML = b; // Displays 3.14; the copy has not changed
```

Tipe Data Reference

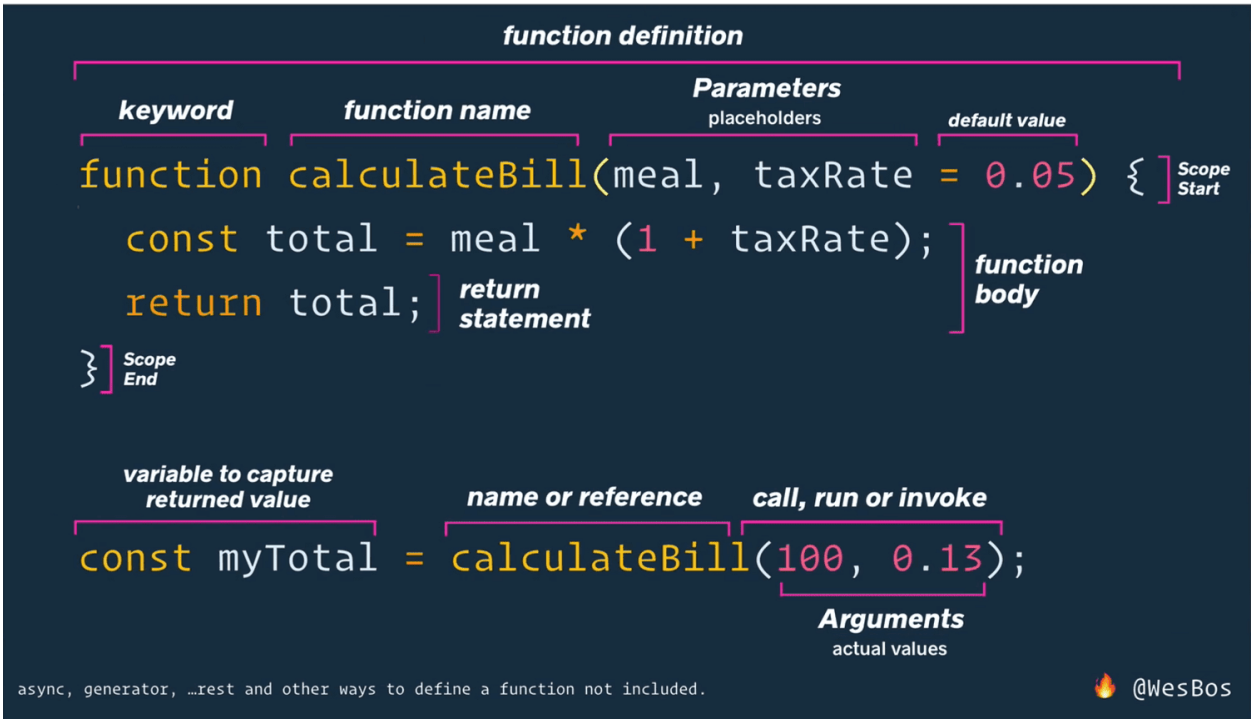
Sedangkan tipe data *reference* merupakan tipe data yang memiliki panjang yang tidak tetap. Karena tipe tersebut tidak memiliki ukuran yang tetap, value tidak dapat disimpan secara langsung pada memori 8 byte yang sudah dialokasikan untuk masing-masing variable sehingga menyimpan suatu *reference* dari nilai tersebut. Pada umumnya, reference berbentuk *pointer* atau alamat memori dan bukan data value. Tipe data reference memberikan informasi pada variable dimana lokasi valuenya. Object, array, function adalah tipe data references.

```
// Deklarasi Tipe data References
// Object, array and function -> copy by reference,
// semua variables & memory locations

// a. Object
let person = {
    name,
    age
}
console.log("Person :", person, "Name :", person.name, "Age :", person.age)

// Bracket notation
person['name'] = 'Marry'
person['age'] = 25
let selection = 'name'
console.log("Person :", person, "Name :", person['name'], "Age :", person['age'], person[selection])

// b. Array notation
let selectedColors = ['red', 'blue']
selectedColors.push('pink')
selectedColors.push(1)
console.log("Selected Colors:", selectedColors, selectedColors[0])

// c. Function
function greet(names) {
    console.log("hello " + names)
}

let names = 'John'
greet(names)
greet("Marry")
```

```
// Tipe data Reference
var a = [1,2,3];  // Initialize a variable to refer to an array
var b = a;        // Copy that reference into a new variable
a[0] = 99;        // Modify the array using the original reference
alert(b);         // Display the changed array [99,2,3] using the new reference
```

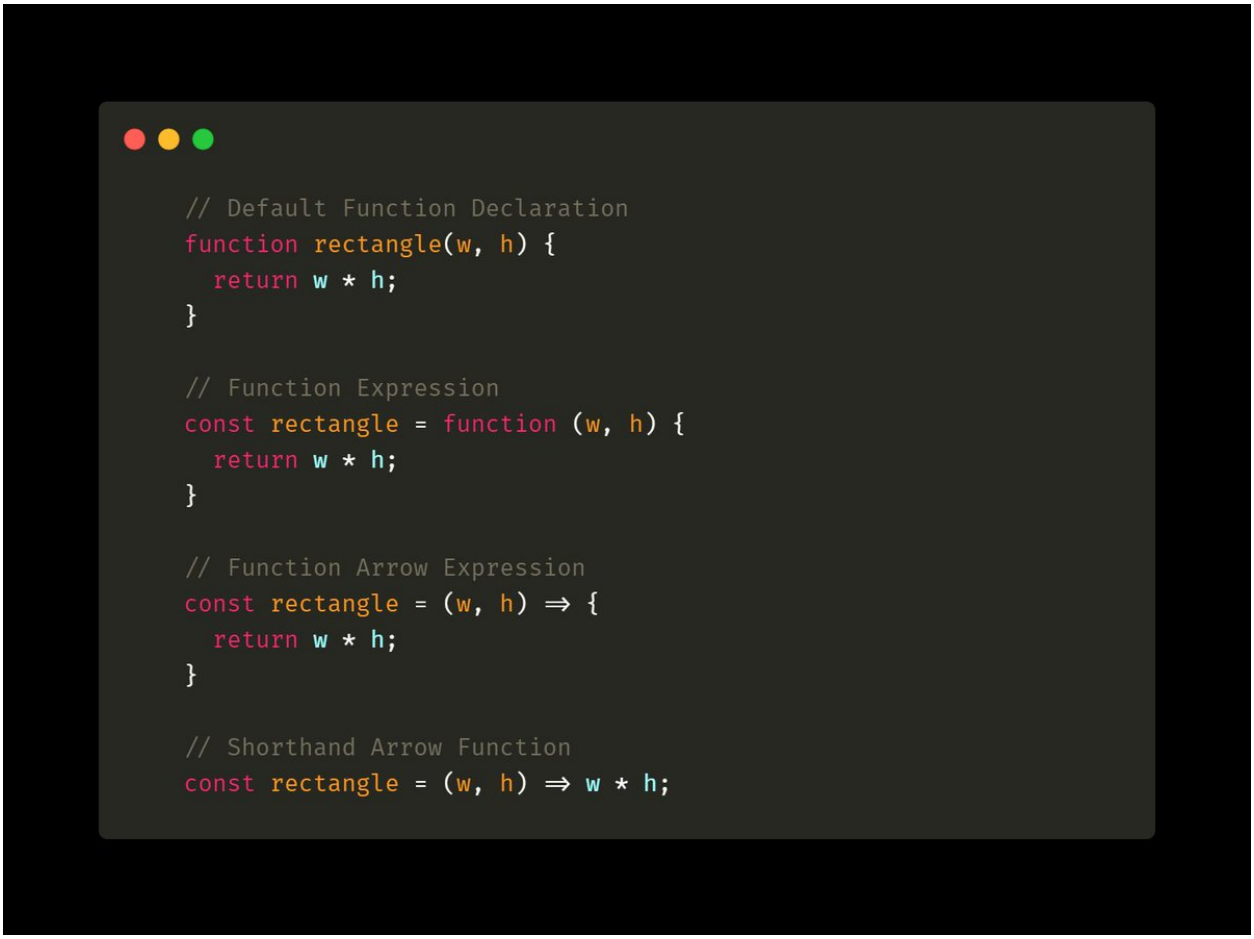
Fungsi

Fungsi adalah salah satu *building block* dalam JavaScript. Fungsi dalam JavaScript mirip dengan prosedur - kumpulan pernyataan yang melakukan tugas atau menghitung nilai. Prosedur dapat dianggap sebagai fungsi apabila dapat mengambil beberapa input dan mengembalikan output sehingga ada hubungan yang jelas antara keduanya. Untuk menggunakan fungsi, kita harus mendefinisikan dan memanggilnya pada suatu lingkup tertentu.



Function Declaration & Function Expression

Fungsi dibuat dan diisi variabel secara explisit seperti nilai lain manapun. Tak peduli bagaimana fungsi didefinisikan, fungsi dapat diperlakukan sebagai suatu nilai yang disimpan dalam variabel, dalam hal ini adalah `square`.



```
// Default Function Declaration
function rectangle(w, h) {
  return w * h;
}

// Function Expression
const rectangle = function (w, h) {
  return w * h;
}

// Function Arrow Expression
const rectangle = (w, h) => {
  return w * h;
}

// Shorthand Arrow Function
const rectangle = (w, h) => w * h;
```

Arrow functions

Keyword `this` digunakan di dalam sebuah fungsi dan mengandung nilai dari objek yang memanggil fungsi tersebut. Kita perlu menggunakan `this` untuk mengakses `method` dan `property` dari objek yang memanggil fungsi tersebut.

```
// ES5
var car = {
  name: "Honda",
  products: ["jazz", "civic", "hrv"],
  showProduct: function () {
    this.products.map(
      function (product) {
        console.log(`${this.name} has launched ${product}`);
      }.bind(this)
    );
  },
};

car.showProduct();
```

`bind` digunakan untuk membantu mengirimkan konteks `this` ke dalam fungsi. Karena bukan dimiliki oleh method atau fungsi itu sendiri, `this` selalu melakukan *binding* ke window object sehingga nilai default yang diberikan menunjuk pada window/global object yaitu `undefined`. Penulisan pada Arrow Function adalah cara yang lebih pendek untuk membuat fungsi `expression`.

```
// ES6
var car = {
  name: "Honda",
  products: ["jazz", "civic", "hrv"],
  showProduct: function () {
    this.products.map(() => {
      console.log(`${this.name} has launched ${product}`);
    });
  },
};

car.showProduct();
```

Template literals

Concatenation/string interpolation

Ekspresi dapat tambahkan dalam string template literal.

```
let firstName = "John";
let lastName = "Doe";

let text = `Welcome ${firstName}, ${lastName}!`;

document.getElementById("demo").innerHTML = text;
```

Multi-line strings

Menggunakan template literal, string pada JavaScript dapat menjangkau beberapa baris kode tanpa perlu penggabungan.

```
let text =

`The quick
brown fox
jumps over
the lazy dog`;
```

```
document.getElementById("demo").innerHTML = text;
```

Shorthand property

ES6 memperkenalkan notasi yang lebih pendek untuk menempatkan properti ke variabel nama yang sama.

```
const formatMessage = (name, id, avatar) => {
  return {
    name,
    id,
    avatar,
    timestamp: Date.now(),
    save () {
      //save message
    }
  }
}
const message = formatMessage("Belajar Programming", 1, "https://i.pravatar.cc/300");
document.getElementById("demo").innerHTML = message.name;
```

Destructuring (object matching)

Gunakan curly bracket untuk menulis properti suatu object ke variabel itu sendiri.

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

let { firstName, lastName: lname } = person;
document.getElementById("demo").innerHTML = `First name: ${firstName}, Last Name: ${lname}`;
```

Parameter Default

```
let func = (a, b = 2) => {
  return a + b
}

document.getElementById("demo").innerHTML = func(10) // return 12
document.getElementById("demo").innerHTML = func(10, 5) // return 12
```

Spread Operator

Penulisan spread dapat digunakan untuk memperluas elemen iterable seperti array menjadi suatu elemen.

```
// opsi 1
const q1 = ["Jan", "Feb", "Mar"];
const q2 = ["Apr", "May", "Jun"];
const q3 = ["Jul", "Aug", "Sep"];
const q4 = ["Oct", "Nov", "May"];

const year = [...q1, ...q2, ...q3, ...q4];

// opsi 2
const numbers = [23,55,21,87,56];
let maxValue = Math.max(...numbers);

document.getElementById("demo").innerHTML = year;
```

Penulisan spread dapat digunakan untuk fungsi argumen.

Modules - export/import

Modules dapat dibuat untuk mengekspor dan mengimpor kode antar file.

index.html

```
<script src="export.js"></script>
<script type="module" src="import.js"></script>
```

person.js

```
const name = "Jesse";
const age = 40;

export {name, age};
```

dengan default export

```
const message = () => {
const name = "Jesse";
const age = 40;
return name + ' is ' + age + 'years old.';
};

export default message;
```

import.js

```
<script type="module">
import { name, age } from "./person.js";

let text = "My name is " + name + ", I am " + age + ".";

document.getElementById("demo").innerHTML = text;
</script>
```

Referensi

- https://www.oreilly.com/library/view/javascript-the-definitive/0596000480/ch04s04.html#:~:text=The types can be divided,a fixed size in memory.
- https://github.com/randyviandaputra/programming-notes/blob/master/penulisan-es6-dan-pengenalan-fitur.md

Latihan

Javascript Fundamental

Pertemuan 2

Full Name: charles. Sex: His Age: 80 years.

Charles is a Programmer

Full Name: mary. Sex: Her Age: 23 years.

Full Name: jane. Sex: Her Age: 35 years.

Menampilkan *list of person* dari apa yang sudah dipelajari diatas.

Buat 3 file, index.html, person.js dan index.js. Fungsi person di export dan di import dalam index.js. Hasil pada index.js akan ditampilkan pada index.html

- person.js berisi module dengan objek Person yang memiliki struktur sebagai berikut:

```
// 1. set default value = fullName = "unknown", age = 25, isMale = false,
const Person = (fullName, age, isMale, avatar) => {
  // ternary operator, menggantikan if-else
  const info = () => {
  // 2. tampilkan full name, sex dan age
  };

  return {
    get getInfo() {
      // 3. kembalikan nilai info diatas
    },
    toString() {
      // 4. kembalikan nilai info diatas
    },
    addAge: // 5. kembalikan usia sekarang ditambah dengan tahun inputan user,
    setAge: // 6. fungsi yang mengeset usia yang baru,
    rename: // 7. fungsi yang mengeset nama yang baru,
    // 8. kembalikan semua nilai dengan shorthand property
  };
};
```

```
// 9. export objek person sebagai sebuah modul
```

- index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Pertemuan 2</title>
    <link
      href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
      rel="stylesheet"
      integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9"
      crossorigin="anonymous"
    />
  </head>
  <body>
    <h1 class="text-center">Javascript Fundamental</h1>
    <h2 class="text-center">Pertemuan 2</h2>
    <div id="charles">
      <img class="img" />
      <p class="info"></p>
      <p class="jobInfo"></p>
    </div>
    <div id="mary">
      <img class="img" />
      <p class="info"></p>
    </div>
    <div id="jane">
      <img class="img" />
      <p class="info"></p>
    </div>
    <script
      src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm"
      crossorigin="anonymous"
    ></script>
    // 1. definisikan index.js
    // 2. definisikan person.js

  </body>
</html>
```

- index.js
  - definisikan 3 person dengan ketentuan sebagai berikut:
    - Nama: Jane
      Usia: 20

Jenis kelamin: perempuan,
Avatar: https://picsum.photos/200

- Nama: Mary
  Usia: 23
  Jenis kelamin: perempuan,
  Avatar: https://picsum.photos/200?grayscale
- Nama : Charles
  Usia: 30
  Jenis kelamin laki-laki
  Avatar : https://picsum.photos/seed/picsum/200

```
// 1. import Person.js
// 2. Definisikan ketiga object person tersebut dengan ketentuan seperti diatas
const charles = ...
const mary = ...
const jane = ...
// 3. Tambah usia Jane 15 tahun
// 4. Set usia Charles menjadi 80 tahun

// 5. Tampilkan Informasi charles dengan querySelector
const charlesImg = document.querySelector("...");
const charlesInfo = document.querySelector("...");
// a. Set avatar charles pada selector image
charlesImg... = ...;
// b. Tampilkan info dengan Template literals
charlesInfo.innerHTML = ...;

// 6. Sama seperti Charles, tampilkan Informasi mary dan Jane dengan querySelector
const charlesJob = { job: "Programmer" };
// 7. Gabungkan objek charles dengan spread operator
const charlesJobInfo = { ... };

// 8. Definisikan fullName dan job milik Charles dengan object destructuring
const { ... } = charlesJobInfo;

// a. Tampilkan job info milik charles dengan query selector
const charlesJobInfoSelector = document.querySelector("...");
// b. Tampilkan info dengan Template literals
charlesJobInfoSelector.innerHTML = ...;
```