Materi 1: JavaScript, Node.js, `npm`, dan `pnpm`

Daftar isi

Sekilas tentang React

React (sebelumnya bernama "React.js") adalah library JavaScript open source untuk mempermudah kita dalam membangun user interface (UI) web. React memungkinkan kita membangun UI dari bagian-bagian individual yang disebut "component". Sebelum adanya Next.js (framework React), React umumnya digunakan untuk membangun *single-page application* (SPA), yaitu aplikasi satu halaman.

React pertama kali dirilis pada tahun 2013 oleh Meta (saat itu bernama "Facebook"). Selain React, ada beberapa alat (library maupun framework) yang digunakan untuk membangun UI:

- Angular (2010)
- Vue (2014)
- Svelte (2016)
- Solid (2021)
- Astro (2022)
- Qwik (2023)

Namun sampai saat ini React masih menjadi yang paling populer di antara alat-alat front-end lainnya.

Sebelum belajar React, kita harus memahami dulu tentang:

- Penggunaan code editor (Visual Studio Code)
- Dasar-dasar web development (HTML, CSS, dan JavaScript)
- Konsep-konsep JavaScript modern (ES6+)
- Node.js (opsional)

JavaScript modern

Berbagai cara menjalankan JavaScript:

1. Menggunakan file HTML (tag `<script>`).
2. Konsol JavaScript di browser, kita dapat membukanya dengan mengklik kanan halaman web dan memilih "Inspect" atau menekan `Ctrl+Shift+J`.
3. Menggunakan runtime JavaScript (Node.js, Deno, Bun, dll.).

Sebelum belajar React, kita harus terlebih dahulu memahami konsep-konsep dasar JavaScript, seperti variabel, tipe data, decision, loop, array, function, dll.

Selain itu, React dibangun menggunakan fitur JavaScript modern. Ada beberapa konsep JavaScript modern yang perlu kita ketahui sebelum belajar React.

JavaScript modern adalah sintaks-sintaks baru JavaScript yang sesuai standar ECMAScript (ES). ECMAScript diperbarui setiap tahun dengan nama versi ES1, ES2, ES3, dst.

Pada tahun 2015, dirilislah ES6 yang memperkenalkan peningkatan yang signifikan pada JavaScript. Mulai versi ES6 inilah JavaScript dianggap modern.

Setelah ES6, versi-versi berikutnya dinamai dengan nama tahun. ES2016 hingga versi terbaru saat ini (ES2023) telah menambahkan fitur-fitur tambahan walaupun tidak sebanyak ES6.

Function

Anonymous function

Anonymous function adalah function yang tidak memiliki nama. Biasanya function ini merupakan nilai dari variabel/konstanta atau sebagai argumen dari function lain.

Contoh sebagai nilai konstanta:

```
const welcome = function () {
  console.log("Selamat datang!");
};

// dipanggil menggunakan nama konstanta
welcome();
```

Contoh sebagai argumen pada pemanggilan function lain:

```
// memanggil fungsi1() dengan argumen berupa anonymous function
fungsi1(function () {
  console.log("Hello, world!");
});
```

Arrow function

Arrow function (fungsi panah) adalah bentuk alternatif yang lebih ringkas dari pernyataan traditional function, tetapi terbatas dan tidak dapat digunakan di semua situasi.

*Arrow function selalu berupa anonymous function.

Contoh traditional anonymous function:

```
function (name) {
  return "Selamat datang " + name;
}
```

Untuk mengubahnya menjadi arrow function, hilangkan kata `function` dan tambahkan `=>` di antara tutup kurung dan buka kurung kurawal.

```
(name) => {
  return "Selamat datang " + name;
};
```

Function di atas hanya memiliki 1 statement, sehingga kurung kurawalnya dapat dihilangkan. Dalam menghapus kurung kurawal, jika statement itu berupa pengembalian (return), maka kata `return` juga harus dihilangkan:

```
(name) => "Selamat datang " + name;
```

*Karena function di atas hanya memiliki 1 parameter, kurung parameternya dapat dihilangkan (tidak disarankan).

Untuk memberi nama pada arrow function di atas, simpan function itu ke sebuah variabel/konstanta.

```
const welcome = (name) => "Selamat datang " + name;
```

Untuk memanggil function di atas, kita dapat menggunakan nama variabel/konstantanya.

```
welcome("Anggi Permana");
// hasil return: Selamat datang Anggi Permana
```

Callback function

Callback function adalah function yang diteruskan ke function lain sebagai argumen, yang kemudian dipanggil di dalamnya.

Contoh:

```
function fungsi1(callback) {
  console.log("Halo");
  callback();
}

function fungsi2() {
  console.log("Hai");
}

fungsi1(fungsi2);

// output:
// Halo
// Hai
```

Rest parameter (`...`)

Sintaks rest parameter (parameter lainnya/sisa) memungkinkan suatu fungsi untuk menerima argumen dalam jumlah tak terbatas sebagai array.

Rest parameter dapat dibuat dengan menambahkan `...` di sebelum parameter terakhir.

Contoh:

```
function fungsi1(a, b, ...numbers) {
  console.log(a);
  console.log(b);
  console.log(numbers);
}
```

```
fungsi1(1, 2, 3, 4, 5);
// output:
// 1
// 2
// [ 3, 4, 5 ]
```

Object

Optional chaining (`.?`)

Operator optional chaining (`?.`) digunakan untuk mencegah pembacaan nilai properti jika objeknya tidak ada, sehingga tidak akan terjadi error.

```
const mahasiswa = {
  nama: "Putri Lestari",
  angkatan: {
    nama: "Integer",
  },
};

console.log(mahasiswa.programStudi.nama);
// output: error
// karena programStudi bernilai undefined
// program mencoba untuk mendapatkan properti 'nama' dari undefined

console.log(mahasiswa.programStudi?.nama);
// output: undefined
```

Array

Method-method array

Method adalah function dari sebuah objek atau class. Array memiliki banyak method, tapi ada 3 method yang penting untuk dipelajari sebelum belajar React, yaitu `map()`, `filter()`, `find()`, dan `reduce()`.

`forEach()`

Method `forEach()` digunakan untuk mengiterasi array. Argumen yang diperlukan adalah function dengan parameter yang mewakili elemen pada iterasi saat ini.

Contoh:

```
const numbers = [1, 4, 9, 16];
numbers.forEach((number) => console.log(number));

// output:
// 1
// 4
// 9
// 16
```

`map()`

Method `map()` mengembalikan array baru yang berisi kumpulan nilai baru untuk tiap-tiap elemennya.

Argumen yang diperlukan adalah function dengan parameter yang mewakili elemen pada iterasi saat ini, serta mengembalikan nilai baru untuk array baru yang dihasilkan nanti.

Contoh:

```
const numbers = [1, 4, 9, 16];
const result = numbers.map((number) => number * 2);
console.log(result);

// output: [ 2, 8, 18, 32 ]
```

`filter()`

Method `filter()` mengembalikan array baru yang berisi semua elemen yang memenuhi kondisi. Jika tidak ada nilai yang memenuhi kondisi maka akan mengembalikan array kosong.

Method `filter()` membutuhkan argumen berupa function yang mengembalikan kondisi untuk menguji tiap-tiap elemen pada array tersebut.

Contoh 1:

```
const numbers = [1, 2, 3, 4, 5, 6];
const result = numbers.filter((n) => n % 2 === 0);
console.log(result);

// output: [ 2, 4, 6 ]
```

Contoh 2:

```
const people = ["Dimas", "Fadli", "Wahyu", "Riyan", "Faisal"];
const result = people.filter((person) => person.length < 6);
console.log(result);

// output: [ 'Dimas', 'Fadli', 'Wahyu', 'Riyan' ]
```

`find()`

Method `find()` mengembalikan elemen array yang pertama memenuhi kondisi. Jika tidak ada nilai yang memenuhi kondisi, maka akan mengembalikan `undefined`.

```
const numbers = [5, 12, 8, 130, 44];
const found = numbers.find((number) => number > 10);
console.log(found);

// output: 12
```

`reduce()`

Method `reduce()` mengembalikan satu nilai dari hasil operasi dari elemen-elemen array.

```
const array1 = [1, 2, 3, 4];

// 0 + 1 + 2 + 3 + 4
const total = array1.reduce((a, b) => a + b, 0);

console.log(total);
// output: 10
```

* `0` adalah nilai awal

* `a` adalah nilai total sementara (dimulai dari nilai awal jika ditentukan, jika tidak maka indeks ke-0)

* `b` adalah nilai saat ini (dimulai dari indeks ke-0 jika nilai awal ditentukan, jika tidak maka indeks ke-1)

Destructuring assignment

Sintaks destructuring assignment adalah pernyataan untuk mengeluarkan nilai dari array atau properti dari objek untuk dimasukkan ke beberapa variabel.

Destructuring pada array

```
let a, b;
[a, b] = [10, 20];

console.log(a);
// output: 10

console.log(b);
// output: 20
```

Destructuring pada object

```
let mahasiswa = {
  nama: "Krise Rohalia",
  umur: 20,
};

let { nama, umur } = mahasiswa;

console.log(nama);
// output: Krise Rohalia

console.log(umur);
// output: 20
```

Rest property/element

Kita dapat mengakhiri pola destructuring dengan rest property `...rest`. Pola ini akan menyimpan semua properti object atau elemen array yang tersisa ke dalam object atau array baru.

```
const { a, ...others } = { a: 1, b: 2, c: 3 };
console.log(others); // { b: 2, c: 3 }
```

```
const [first, ...others2] = [1, 2, 3];
console.log(others2); // [2, 3]
```

Spread (`...`)

Sintaks spread (`...`) adalah pernyataan untuk mengeluarkan semua elemen array untuk dimasukkan ke array baru.

Contoh:

```
let numbers = [0, 1, 2];
let newNumber = [...numbers, 12];
console.log(newNumbers);

// output: [ 0, 1, 2, 12 ]
```

String

Template literal

Conditional (ternary) operator

Conditional operator atau ternary operator adalah operator untuk membuat keputusan dan memilih di antara dua expression berdasarkan suatu kondisi.

Operator ini sering digunakan sebagai alternatif dari pernyataan `if...else`.

```
function tagihan(anggota) {
  return anggota ? "Rp30.000" : "Rp150.000";
}

console.log(tagihan(true));
// output: Rp30.000

console.log(tagihan(false));
// output: Rp150.000

console.log(tagihan(null));
// output: Rp150.000
```

Binary logical operator

Binary logical operator (operator logika biner) adalah operator yang mengembalikan nilai boolean.

Binary logical operator ada 3, yaitu:

1. AND (`&&`)
2. OR (`||`)
3. Nullish coalescing (`??`)

AND (`&&`)

Jika kiri bernilai `false`, `null`, atau `undefined`, maka akan mengembalikan nilai itu sendiri (nilai kiri).

Jika kiri bernilai selain nilai-nilai di atas, maka akan mengembalikan nilai kanan.

Contoh:

```
let nama = false;
console.log(nama && "Agus");
// output: false
```

```
let nama = "Farhan";
console.log(nama && "Agus");
// output: Agus
```

OR (`||`)

Jika kiri bernilai `false`, `null`, atau `undefined`, maka akan mengembalikan nilai kanan.

Jika kiri bernilai selain nilai-nilai di atas, maka akan mengembalikan nilai itu sendiri (nilai kiri).

Contoh:

```
let nama = false;
console.log(nama || "Agus");
// output: Agus
```

```
let nama = "Farhan";
console.log(nama || "Agus");
// output: Farhan
```

Nullish coalescing (`??`)

Jika kiri bernilai `null` atau `undefined`, maka akan mengembalikan nilai kanan.

Jika kiri bernilai selain nilai-nilai di atas, maka akan mengembalikan nilai itu sendiri (nilai kiri).

Contoh:

```
let nama = null;
console.log(nama || "Agus");
// output: Agus
```

```
let nama = "Farhan";
console.log(nama || "Agus");
// output: Farhan
```

Module

Module adalah file berisi kode yang dapat digunakan kembali oleh module lain.

Konsep

- `import` digunakan untuk menyertakan dan menggunakan module yang disimpan di tempat lain, di sistem file lokal kita atau dari jarak jauh.
- `export` digunakan untuk menentukan bagian mana dari module kita yang dapat diakses oleh orang lain yang mengimpor module kita.

Module system

Node.js memiliki dua module system:

1. CommonJS module (cara asli)
2. ECMAScript module (cara baru, format standar resmi)

CommonJS module

Cara mengekspor:

```
exports.tambah = function (a, b) {
  return a + b;
};

exports.kali = function (a, b) {
  return a * b;
};
```

Cara mengimpor:

```
const { tambah, kali } = require("./aritmatika.js");

console.log(tambah(3, 4)); // 7
console.log(kali(5, 3)); // 15
```

ECMAScript module

Cara mengekspor:

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

export function kali(a, b) {
  return a * b;
}
```

Cara mengimpor:

```
import { tambah, kali } from "./aritmatika.js";

console.log(tambah(3, 4)); // 7
console.log(kali(5, 3)); // 15
```

Jenis-jenis module

Ada dua jenis module di Node.js:

1. core/built-in module (modul inti/bawaan)
2. custom module (modul kustom)

Core/built-in module

Core/built-in module (modul inti/bawaan) adalah module yang sudah tersedia di dalam Node.js.

Contoh core module:

- Module `fs`, menyediakan fungsi untuk membaca dan menulis file
- Module `http`, menyediakan fungsi untuk membuat server dan klien HTTP

Custom module

Custom module adalah module yang dibuat oleh developer.

Custom module terbagi menjadi dua macam:

1. User-defined module (modul buatan pengguna)
2. Third-party module (modul pihak-ketiga)

User-defined module

User-defined module (modul buatan pengguna) adalah module yang kita buat sendiri untuk mengatur dan menggunakan kembali kode kita sendiri.

Third-party module

Third-party module (modul pihak-ketiga) adalah module yang dibuat oleh komunitas dan diterbitkan di npm registry.

Asinkron

Asynchronous programming (pemrograman asinkron) adalah cara agar program dapat memulai tugas yang berpotensi berjalan lama dan masih dapat responsif terhadap peristiwa lain saat tugas itu berjalan, daripada harus menunggu sampai tugas itu selesai.

Untuk membuat tugas berjalan secara asinkron, saat ini ada 3 cara:

1. Callback
2. `Promise`

3. `async` dan `await`

Callback dulunya merupakan cara utama untuk menerapkan fungsi asinkron dalam JavaScript. Namun, kode berbasis callback dapat menjadi sulit dipahami jika callback itu sendiri harus memanggil fungsi yang menerima callback juga (nested callback).

Akhirnya pada tahun 2015 muncul standar baru JavaScript modern bernama ES6 yang memperkenalkan `Promise`. Kemudian pada tahun 2017 muncul ES2017 yang memperkenalkan `async` dan `await`.

Callback

Callback adalah fungsi yang dijadikan argumen saat memanggil fungsi lain kemudian dipanggil dari dalam fungsi lain tersebut.

Salah satu contoh umum adalah dengan menggunakan timer (`setTimeout()` dan `setInterval()`):

```
setTimeout(() => {
  // fungsi ini dipanggil setelah 2 detik
}, 2000);
```

```
setInterval(() => {
  // fungsi ini dipanggil setiap 2 detik
}, 2000);
```

`Promise`

`Promise` adalah sebuah class bawaan JavaScript.

`Promise` memiliki method `then()` dan `catch()` yang masing-masing membutuhkan 1 argumen berupa fungsi.

- `then()` digunakan untuk menentukan fungsi yang akan dipanggil jika berhasil.
- `catch()` digunakan untuk menentukan fungsi yang akan dipanggil jika gagal.

Constructor dalam class `Promise` membutuhkan 1 argumen berupa fungsi. Fungsi yang dijadikan argumen tersebut dapat memiliki 2 parameter.

- parameter pertama mewakili fungsi yang ditentukan dalam `then()`
- parameter kedua mewakili fungsi yang ditentukan dalam `catch()`

Contoh 1:

```
const promise1 = new Promise((berhasil, gagal) => {
  setTimeout(() => {
    berhasil("Alhamdulillah");
  }, 1000);
});

promise1.then((pesan) => {
  console.log(pesan);
});

// output setelah 1 detik:
// Alhamdulillah
```

Contoh 2:

```
const promise2 = new Promise((berhasil, gagal) => {
  setTimeout(() => {
    const sekarang = new Date();
    if (sekarang.getSeconds() % 2 === 0) {
      gagal("Innalillah");
    } else {
      berhasil("Alhamdulillah");
    }
  }, 1000);
});

promise2
  .then((pesan) => {
    console.log(`then() telah dipanggil dengan pesan: ${pesan}`);
  })
  .catch((pesan) => {
    console.log(`catch() telah dipanggil dengan pesan: ${pesan}`);
  });

// output jika detik genap:
// catch() telah dipanggil dengan pesan: Innalillah

// output jika detik ganjil:
// then() telah dipanggil dengan pesan: Alhamdulillah
```

Contoh 3 (fungsi bawaan):

```
fetch("http://next.pubpasim.org/api/students")
  .then((response) => response.json())
  .then((data) => console.log(data));
```

`async` dan `await`

Fungsi yang diawali `async` adalah fungsi asinkron. Saat dipanggil, fungsi ini akan mengembalikan objek `Promise`.

Untuk menunggu dan mendapatkan nilai yang diinginkan, kita perlu menambahkan `await` sebelum pemanggilannya.

Contoh:

```
async function getName() {
  return "Kurniawan";
}

const name = getName();
// name berisi objek Promise, bukan string "Kurniawan"

const name2 = await getName();
// name2 berisi string "Kurniawan"
```

Node.js

Node.js adalah runtime JavaScript yang dibuat di atas mesin JavaScript V8 Chrome.

Runtime adalah software yang memungkinkan program bahasa pemrograman tertentu bisa berinteraksi dengan sistem.

Berikut contoh-contoh runtime dari beberapa bahasa pemrograman.

Bahasa	Runtime
JavaScript, TypeScript	V8 (Node.js), Deno, Bun
Python	CPython, PyPy, Jython, IronPython
Go	Go Runtime
C#, F#, Visual Basic	.NET Runtime (CLR)
Java, Kotlin, Scala	Java Virtual Machine (JVM)
C, C++	C Runtime Library (CRT)
PHP	Zend Engine, Apache HTTP Server

*JVM sudah termasuk dalam Java Development Kit (JDK)

*Pengembangan PHP di Indonesia lebih umum menggunakan Apache HTTP Server (sudah termasuk dalam XAMPP) daripada Zend Engine.

Sebelum adanya Node.js, JavaScript hanya bisa berjalan di lingkungan browser, tidak dapat berinteraksi dengan sistem, termasuk file, basis data, dll.

Menginstal Node.js

1. Buka https://nodejs.org/en/.
2. Download yang versi LTS, LTS (long-term support) artinya memiliki dukungan jangka panjang, biasanya perusahaan menggunakan versi LTS.
3. Instal seperti biasa.
4. Untuk memastikan apakah Node.js telah terinstal, coba buka terminal, ketik `node` lalu enter, jika muncul `Welcome to Node.js` (dengan versinya) artinya Node.js telah terinstal.

Menjalankan program

```
> node nama-file.js
```

Atau tanpa ekstensi:

```
> node nama-file
```

Jika berada di dalam folder:

```
> node nama-folder/nama-file
```

Khusus untuk file bernama `index.js` kita bisa menggunakan lokasi foldernya saja.

```
> node nama-folder
```

Jika file `index.js` itu berada di folder saat ini maka menggunakan simbol titik (`.`).

```
> node .
```

`npm` dan `npm` registry

`npm` (Node Package Manager)

Package adalah kumpulan file yang digabungkan. Package Node.js berarti package yang berisi kumpulan kode JavaScript. Package Node.js dapat berupa module, library, framework, atau aplikasi.

Package manager adalah alat untuk mengelola package dan dependensi dari runtime (atau bahasa pemrograman) tertentu.

Berikut contoh-contoh package manager dari beberapa runtime:

Runtime	Package Manager
Python	`pip`, Anaconda
Node.js	`npm`, Yarn, `pnpm`
Go	`go get`
Rust	Cargo
.NET	NuGet
JVM	Maven, Gradle
PHP	Composer

`npm` (Node Package Manager) adalah package manager bawaan Node.js.

Menginisialisasi project

Untuk memulai project Node.js, dalam folder yang ingin kita jadikan project jalankan perintah:

```
> npm init
```

Perintah di atas akan meminta beberapa informasi project dan menyimpannya ke file `package.json`.

Jika kita ingin menghasilkan file `package.json` secara langsung dengan informasi default, tambahkan flag `--yes` atau `-y`:

```
> npm init -y
```

Menambahkan script

Properti `scripts` dalam file `package.json` digunakan untuk menjalankan beberapa hal dengan satu perintah.

Perintah untuk menjalankan script:

```
> npm run <nama-script>
```

Contoh di file `package.json`:

```
{
  "scripts": {
    "dev": "node . && node app.js && echo \"Hello, world!\""
  }
}
```

Lalu kita jalankan script `dev`:

```
> npm run dev
```

Perintah di atas akan menjalankan file `index.js` dan `app.js` serta mencetak "Hello, world!".

`npm` registry

`npm` registry (https://www.npmjs.com/) adalah basis data tempat developer dapat menerbitkan, membagikan, dan men-download package Node.js menggunakan package manager Node.js seperti npm, pnpm, Yarn, dll.

`pnpm`

`pnpm` adalah package manager Node.js baru yang lebih cepat dan efisien.

Keunggulan `pnpm` dibandingkan `npm` dan Yarn:

1. Lebih cepat dalam instalasi
2. Lebih hemat penggunaan ruang disk (hardisk maupun SSD)
3. Lebih hemat penggunaan bandwidth jaringan

Instalasi

Ada 3 cara menginstal pnpm di Windows:

1. Menggunakan PowerShell (direkomendasikan)
2. Menggunakan Corepack
3. Menggunakan `npm`

Menggunakan PowerShell (direkomendasikan)

Buka Windows PowerShell (atau Windows Terminal), lalu jalankan perintah:

```
> iwr https://get.pnpm.io/install.ps1 -useb | iex
```

Tunggu sampai proses download dan instalasi selesai.

Menggunakan Corepack

Buka terminal dengan shell apapun (PowerShell, Command Prompt, Git Bash, dll.), lalu jalankan perintah:

```
> corepack enable
```

Tunggu sampai proses instalasi selesai.

Menggunakan `npm`

Buka terminal dengan shell apapun, lalu jalankan perintah:

```
> npm install -g pnpm
```

Perbedaan perintah dengan `npm`

Perintah `npm`	Perintah `pnpm`
`npm install`	`pnpm install` atau `pnpm i`
`npm i <nama-package>`	`pnpm add <nama-package>`
`npm update`	`pnpm update` atau `pnpm up`
`npm uninstall <nama-package>`	`pnpm remove <nama-package>` atau `pnpm rm <nama-package>`
`npm run <nama-script>`	`pnpm <nama-script>`