By: Hilman Fikry

Frontend Development with React.js

React Introduction, Testing, Ecosystem, and More

Frontend Development

- Web development = membangun dan memelihara websites
- Jenis web development: frontend, backend, fullstack
- Frontend development = aspek yang berhadapan langsung dengan user
- Bahasa pada front-end cukup terstandardisasi:
  - HTML untuk markup
  - CSS untuk penampilan
  - JavaScript untuk scripting

React Itu Apa?

- Berdasarkan website React, React adalah:

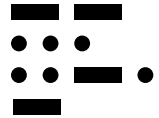
React

The library for web and native user interfaces

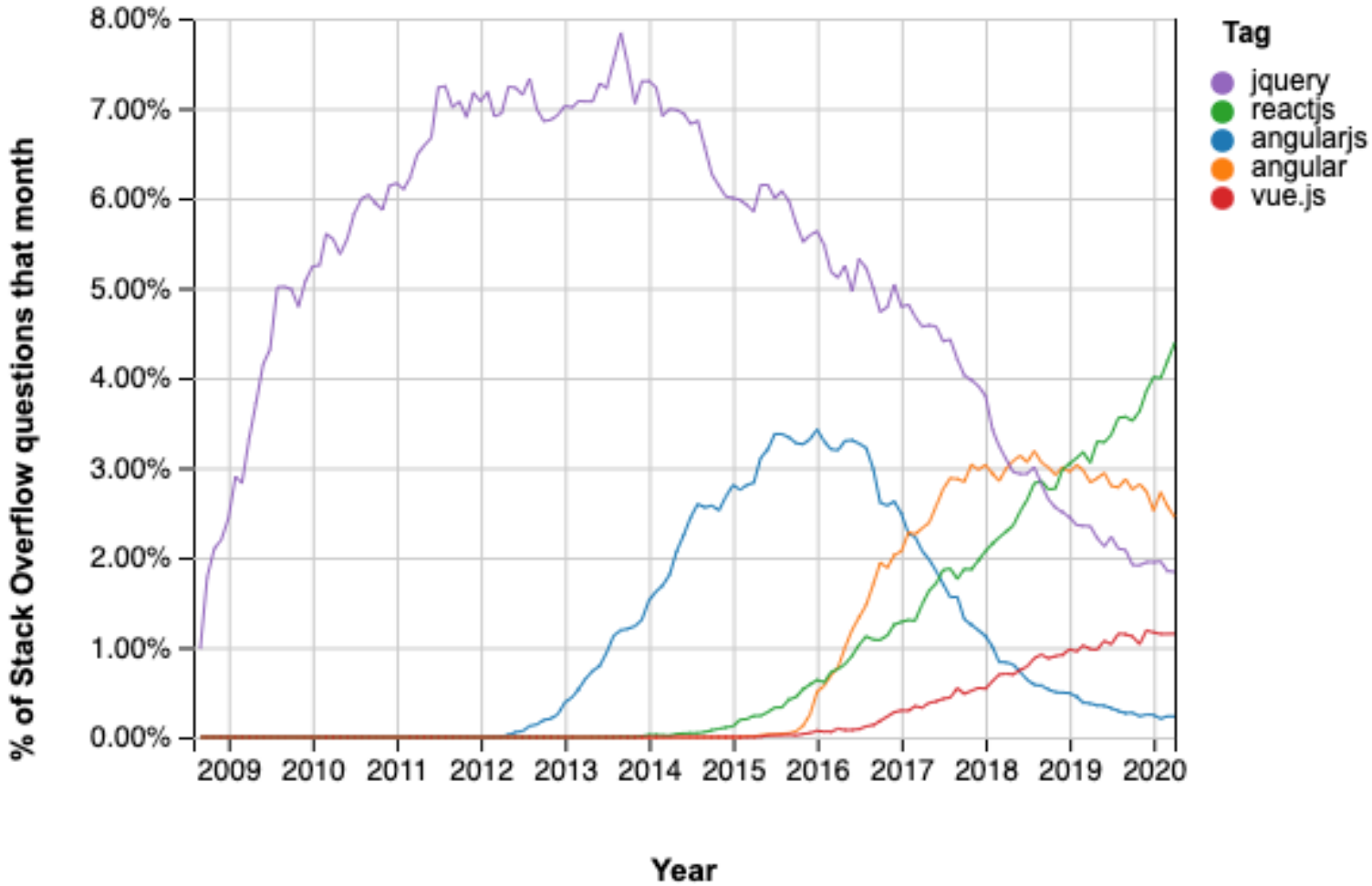
- Dalam programming, library adalah kumpulan kode yang sudah ditulis sebelumnya, yang dirancang untuk mempermudah development

Mengapa Menggunakan React?

- Saya bisa membuat websites dengan vanilla JavaScript, kenapa saya malah belajar React?
  - Komponen React bersifat reusable
  - Didukung dengan baik karena populer dan memiliki komunitas yang besar
  - Tidak opinionated
  - Lebih mudah, khususnya jika sudah memahami JavaScript



Life Cycle JavaScript Frameworks/Libraries

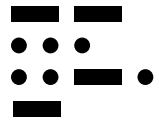


Setting Up React Environment

- Setting up React secara manual cukup sulit sehingga direkomendasikan menggunakan toolchain/framework, contohnya:
  - Vite
  - Gatsby
  - Next.js
- Untuk setup aplikasi React menggunakan Vite, jalankan:

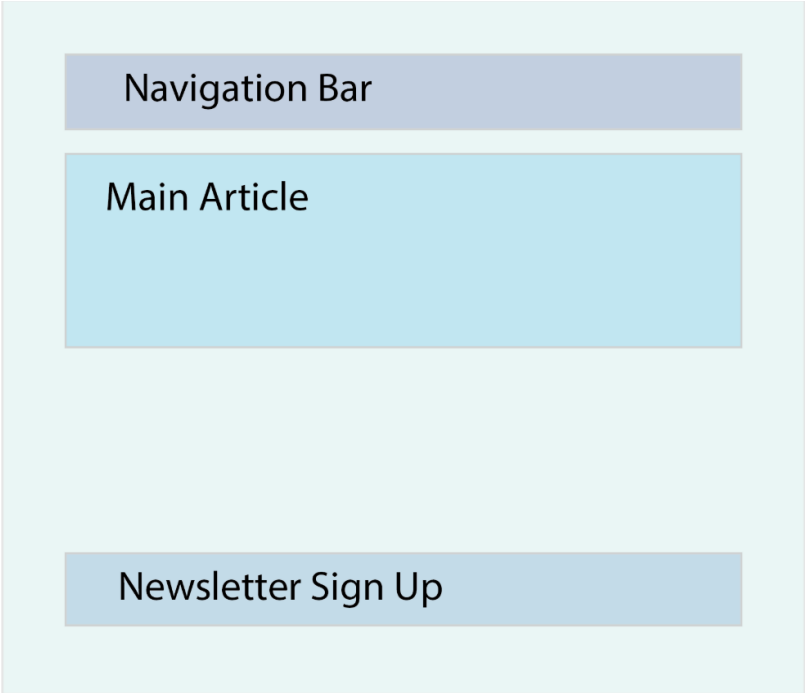
```
npm create vite@latest my-first-react-app -- --template react
```

- Untuk menggunakan React Developer Tools di Chrome, kita bisa install extensionnya: React Developer Tools
- Referensi Vite: Getting Started | Vite



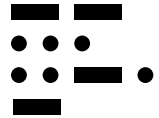
React Components

- React itu keren karena kita bisa membagi tampilan aplikasi (UI) menjadi bagian-bagian kecil yang berdiri sendiri dan bisa dipakai ulang. Bagian-bagian kecil ini disebut **components**. Contoh:



Website ini bisa dibagi menjadi beberapa components berikut:

- App
- Navbar
- MainArticle
- NewsletterForm



Apa Itu JSX?

- JSX adalah syntax extension untuk JavaScript yang memungkinkan kita menulis markup mirip HTML di dalam file JavaScript
- Pada dasarnya, JSX adalah syntactic sugar untuk fungsi `createElement` di React
- Berikut adalah aturan JSX:

Mengembalikan satu root element

✅ Benar:

```
function App() {
  // Could replace <></> with <div></div>
  return (
    <>
      <h1>Example h1</h1>
      <h2>Example h2</h2>
    </>
  );
}
```

❌ Salah:

```
function App() {
  return (
    <h1>Example h1</h1>
    <h2>Example h2</h2>
  );
}
```

Menutup semua tags

✅ Benar:

```
function App() {
  return (
    <>
      <input />
      <li></li>
    </>
  );
 }
```

❌ Salah:

```
function App() {
  return (
    <>
      <input>
      <li>
    </>
  );
}
```

Menggunakan camelCase untuk attribute

✅ Benar:

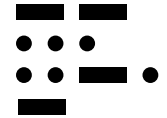
```
function App() {
  return (
  <div className="container">
    <svg>
      <circle cx="25" cy="75" r="20" stroke="green" strokeWidth="2" />
    </svg>
  </div>
  );
}
```

❌ Salah:

```
function App() {
  return (
    <div class="container">
      <svg>
        <circle cx="25" cy="75" r="20" stroke="green" stroke-width="2"
      </svg>
    </div>
  );
}
```

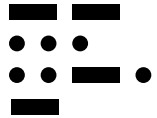
Referensi:

- Writing Markup with JSX
- JavaScript in JSX



Teknik Rendering

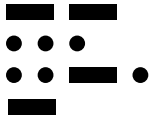
- Untuk menampilkan banyak components yang mirip dari sebuah kumpulan data, kita bisa menggunakan JavaScript array methods untuk memanipulasi array data
- Untuk menampilkan sesuatu yang berbeda tergantung pada kondisinya, kita bisa melakukan conditional rendering JSX dengan menggunakan JavaScript syntax seperti if statements, &&, dan operator ? :
- Referensi:
  - Rendering Lists – React
  - Conditional Rendering – React



Keys dalam React

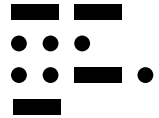
- React menemukan item tertentu dengan menggunakan key
- Selama keys tetap konsisten dan unik, React dapat mengelola DOM secara efektif dan efisien
- Keys diteruskan ke dalam component atau DOM element sebagai sebuah prop:

```
<Component key={keyValue} />
//or
<div key={keyValue} />
```



Mengirimkan Data antar-Components

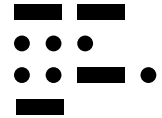
- Dalam React, data dikirim secara satu arah dari parent components ke child components melalui props
- Referensi: Passing Props to a Component – React



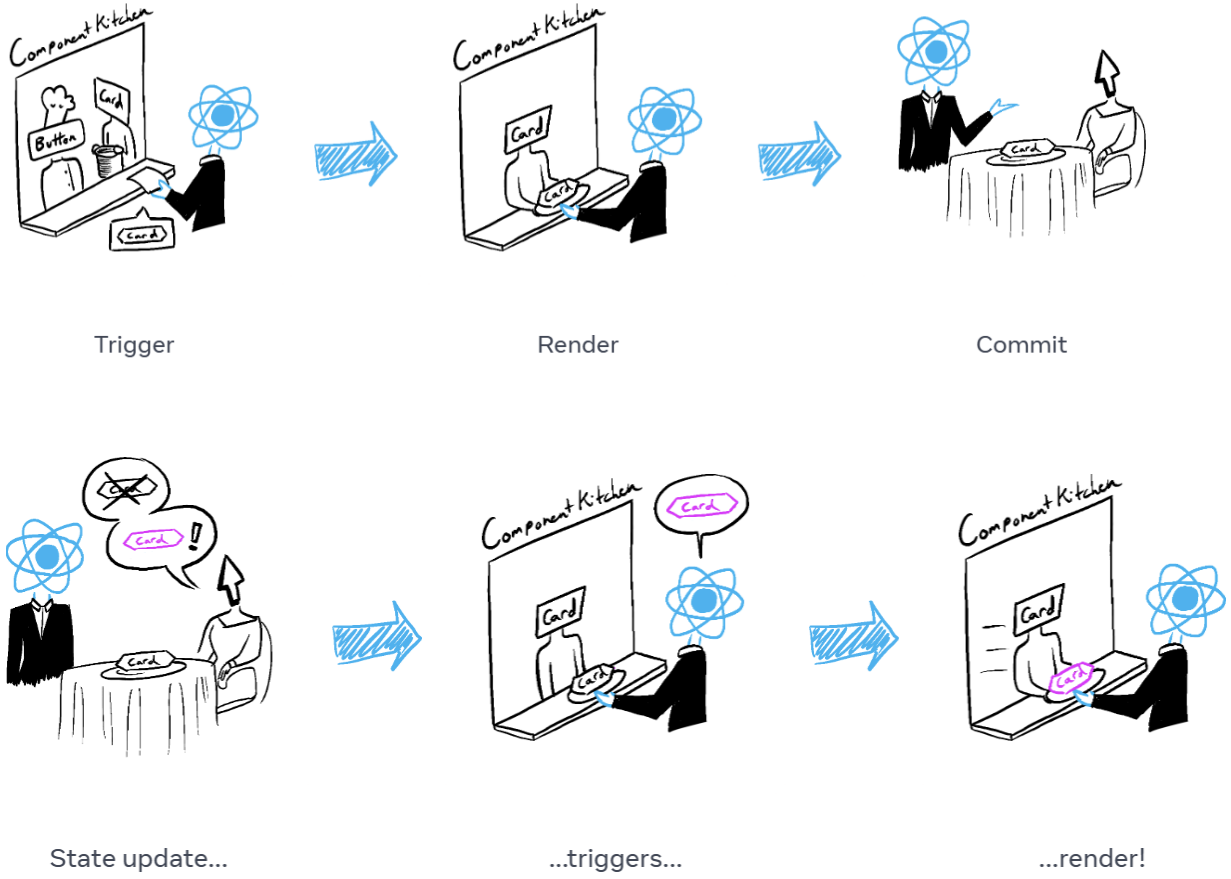
State dalam React

- State adalah memori component
- `useState` adalah built-in hook di React untuk mendefinisikan state dalam functional component.
- `useState` menerima nilai awal sebagai parameter dan mengembalikan sebuah array dengan dua elemen yang bisa kita destructure untuk mendapatkan:
  - Nilai state saat ini
  - Fungsi untuk memperbarui nilai state

```
1 | const [stateValue, setStateValue] = useState(initialValue);
```



Cara Kerja State dalam React



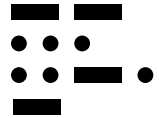

```
                                    React will provide the latest value
function App() {                                                    ignored
  const [backgroundColor, setBackgroundColor] = useState(COLORS[0]);

  const onButtonClick = (color) => () => {
    setBackgroundColor(color);
  };

recreated
from scratch
  return (
    <div
      className="App"
      style={{
        backgroundColor,
      }}
    >
      {COLORS.map((color) => (
        <button
          type="button"
          key={color}
          onClick={onButtonClick(color)}
          className={backgroundColor === color ? 'selected' : ''}
        >
          {color}
        </button>
      ))}
    </div>
  );
}
```

Apa Itu Hooks?

- Hooks adalah functions yang memungkinkan kita menggunakan fitur-fitur React
- Hooks memiliki aturan yang perlu kita patuhi:
  - Hooks hanya boleh dipanggil dari tingkat atas sebuah functional component
  - Hooks tidak boleh dipanggil dari dalam loops atau conditions
- Referensi:
  - State: A Component's Memory – React
  - Render and Commit – React
  - State as a Snapshot – React
  - Choosing the State Structure – React
  - Sharing State Between Components – React

Cara Menangani Side Effects

- Side-effect adalah interaksi component dengan hal-hal di luar dirinya, seperti:
  - mengambil data dari server
  - memanipulasi posisi elemen di halaman web, atau
  - mengirim data ke server
- Effects memungkinkan kita menjalankan kode untuk menyinkronkan component sesuai kebutuhan, baik saat rendering maupun ketika terjadi perubahan nilai reactive/state, bukan hanya pada suatu event tertentu
- Dalam React, kita menggunakan `useEffect` hook untuk menggunakan effects di dalam components



Struktur `useEffect`

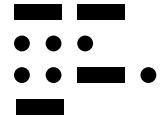
```
useEffect(
  () => {
    // execute side effect
    return () => {
      // cleanup function on unmounting or re-running effect
    }
  },
  // optional dependency array
  [/* 0 or more entries */]
)
```

Dependency array:

```
useEffect(() => {
  // This runs after every render
});

useEffect(() => {
  // This runs only on mount (when the component appears)
}, []);

useEffect(() => {
  // This runs on mount *and also* if either a or b have changed
}, [a, b]);
```

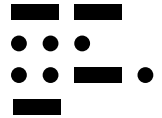
Referensi:

- Lifecycle of Reactive Effects – React
- You Might Not Need an Effect – React



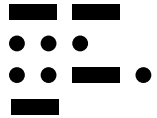
React Testing

- Karena kita menggunakan Vite, kita akan menggunakan Vitest sebagai test runner karena Vitest terintegrasi dengan baik bersama Vite
- Kita akan menambahkan lebih banyak kemampuan pada tests kita dengan menggunakan React Testing Library (RTL)
- Cara set up RTL dalam Vitest dengan menggunakan Vite: Vitest with React Testing Library
- `@testing-library` packages:
  - `@testing-library/react`: menampilkan components dalam pengujian
  - `@testing-library/jest-dom`: menulis asersi yang lebih ekspresif
  - `@testing-library/user-event`: membuat tes lebih realistis karena meniru cara pengguna berinteraksi
- Referensi:
  - Cheatsheet | Testing Library
  - Testing React.js Apps



Client-Side Routing

- Client-side routing adalah jenis routing di mana JavaScript mengambil alih tugas untuk menangani routes dalam application
- Client-side routing membantu membangun single-page applications (SPAs) tanpa perlu melakukan refresh saat pengguna melakukan navigasi
- React Router adalah standard routing library untuk React applications
- Referensi: React Router Official Documentation



Fetching Data dalam React

```
const image = document.querySelector("img");
fetch("https://picsum.photos/v2/list")
  .then((response) => response.json())
  .then((response) => {
    image.src = response[0].download_url;
  })
  .catch((error) => console.error(error));
```

```
import { useEffect, useState } from "react";

const Image = () => {
  const [imageURL, setImageURL] = useState(null);

  useEffect(() => {
    fetch("https://picsum.photos/v2/list", {
      headers: {
        "User-Agent": "the-odin-project"
      }
    })
    .then((response) => response.json())
    .then((response) => setImageURL(response[0].download_url))
    .catch((error) => console.error(error));
  }, []);

  return (
    imageURL && (
      <>
        <h1>An image</h1>
        <img src={imageURL} alt={"placeholder text"} />
      </>
    )
  );
};

export default Image;
```

Referensi: [Modern API data-fetching methods in React - LogRocket Blog]

Memberi Styling pada React Applications

- CSS utility frameworks adalah pilihan populer untuk memberi styling pada React applications. Framework ini menyediakan sekumpulan pre-defined classes yang bisa langsung digunakan di HTML atau JSX. Contoh: Tailwind CSS
- Kita juga bisa menggunakan component libraries, yaitu kumpulan components yang sudah siap pakai. Contoh: Material UI, Radix UI, Chakra UI, shadcn/ui, lucide react