ultimatics
turn knowledge into performance

TURN
KNOWLEDGE
INTO PERFORMANCE

SPECIALIZATION TRACK
FRONT-END : REACT

Moch. Ilham Anugrah

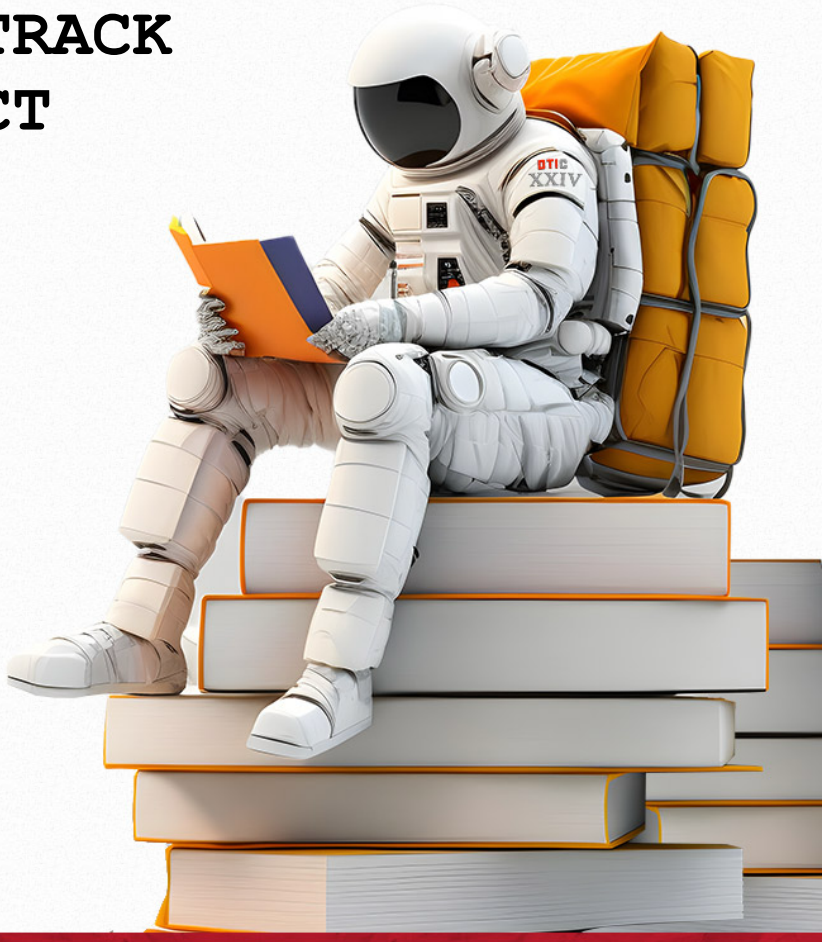

Table of Contents

DOM Manipulation

Pada bab ini, kita akan mulai membangun proyek dengan menggunakan JavaScript dan metode DOM untuk menambahkan tag **h1** ke dalam proyek Anda.

Buka *code editor* Anda dan buat sebuah file baru bernama **index.html**. Di dalam file HTML tersebut, tambahkan kode berikut:

```
<html>
  <body>
    <div></div>
  </body>
</html>
```

Kemudian berikan atribut **id** pada elemen `div` tersebut agar dapat Anda targetkan nanti:

```
<html>
  <body>
    <div id="app"></div>
  </body>
</html>
```

Untuk menuliskan JavaScript di dalam file HTML, tambahkan tag `<script>`:

```
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript"></script>
  </body>
</html>
```

Sekarang, di dalam tag `<script>`, Anda dapat menggunakan metode DOM `getElementById()` untuk memilih elemen `<div>` berdasarkan id-nya:

```
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript">
      const app = document.getElementById('app');
    </script>
```

```
  </body>
</html>
```

Anda dapat melanjutkan dengan menggunakan metode DOM untuk membuat elemen `<h1>` baru:

```
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript">
      // Pilih elemen div dengan id 'app'
      const app = document.getElementById('app');

      // Buat elemen H1 baru
      const header = document.createElement('h1');

      // Buat text node untuk elemen H1
      const text = 'Mari belajar bersama Multimatics.';
      const headerContent = document.createTextNode(text);

      // Tempelkan teks ke dalam elemen H1
      header.appendChild(headerContent);

      // Tempatkan elemen H1 ke dalam div
      app.appendChild(header);
    </script>
  </body>
</html>
```

Untuk memastikan semuanya berjalan dengan baik, buka file HTML Anda menggunakan browser pilihan. Anda seharusnya melihat sebuah tag **h1** yang berisi tulisan:
"Mari belajar bersama Multimatics."

Memulai React

Untuk menggunakan React di proyek baru Kita, kita gunakan dua skrip React dari situs eksternal bernama **unpkg.com**:

- **react** adalah pustaka inti React.
- **react-dom** menyediakan metode khusus DOM yang memungkinkan Anda menggunakan React dengan DOM.

index.html

```
<html>
  <body>
    <div id="root"></div>
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>

    <script type="text/javascript">
      const container = document.getElementById('root');
    </script>
  </body>
</html>
```

Untuk menyiapkan aplikasi react pada halaman kita, kita butuh method ReactDOM.createRoot() yang diambil dari package react-dom yang akan digunakan sebagai wadah untuk aplikasi.

Setelah itu, kita dapat memanggil root.render() untuk menampilkan elemen atau komponen React ke dalam halaman.

index.html

```
<html>
  <body>
    <div id="root"></div>
```

```
    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>

    <script type="text/javascript">
      const container = document.getElementById('root');
      const root = ReactDOM.createRoot(container);

      root.render(<h1> Mari belajar React bersama Multimatics.</h1>);
    </script>
  </body>
</html>
```

Jika kita mencoba menjalankan kode tersebut di browser, kita akan mendapatkan error:

```
Uncaught SyntaxError: expected expression, got '<'
```

Ini karena `<h1>...</h1>` **bukan JavaScript valid**—kode tersebut adalah **JSX**.

Apa itu JSX?

JSX adalah ekstensi sintaks JavaScript yang memungkinkan Anda menuliskan UI dengan sintaks mirip HTML. Keunggulan JSX adalah Anda tidak perlu mempelajari simbol atau aturan baru di luar HTML dan JavaScript, selain mengikuti tiga aturan dasar JSX.
Namun browser **tidak bisa memahami JSX secara langsung**, sehingga Anda memerlukan kompiler JavaScript seperti **Babel** untuk mengubah kode JSX menjadi JavaScript biasa.

Menambahkan Babel

Setelah tadi kita kena error karena <h1>...</h1> itu JSX dan bukan JavaScript murni, sekarang kita perlu “penerjemah” yang bisa mengubah JSX menjadi JavaScript yang dimengerti browser. Di sini lah Babel dipakai.

index.html lengkap dengan Babel

```
<html>
  <body>
    <div id="root"></div>

    <script src="https://unpkg.com/react@18/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@18/umd/react-
dom.development.js"></script>

    <!-- Babel Script -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

    <script type="text/jsx">
            const container = document.getElementById('root');
      const root = ReactDOM.createRoot(container);

      root.render(<h1> Mari belajar React bersama Multimatics.</h1>);
    </script>
  </body>
</html>
```

Untuk memastikan semuanya berfungsi, buka file HTML Anda di browser.

Membuat Komponen

Di React, komponen sebenarnya adalah sebuah **fungsi**.

Di dalam tag `<script>` kita, buat sebuah fungsi baru bernama `header`:

```
<script type="text/jsx">
  const root = document.getElementById('root');

  function Header() {}

  const appRoot = ReactDOM.createRoot(root);
  appRoot.render(<h1>Belajar React bersama Multimatics</h1>);
</script>
```

Sebuah komponen adalah fungsi yang mengembalikan elemen UI.

Di dalam return, kita bisa menuliskan JSX:

```
<script type="text/jsx">
  const root = document.getElementById('root');

  function Header() {
    return <h1>Belajar React bersama Multimatics</h1>;
  }

  const appRoot = ReactDOM.createRoot(root);
  appRoot.render(<h1>Belajar React bersama Multimatics</h1>);
</script>
```

Untuk menampilkan komponen Header, kita bisa memanggilnya di dalam root.render():

```
<script type="text/jsx">
  const root = document.getElementById('root');

  function Header() {
    return <h1>Belajar React bersama Multimatics</h1>;
  }

  const appRoot = ReactDOM.createRoot(root);
  appRoot.render(Header);
</script>
```

Tapi, tunggu dulu…

Kode di atas tetap akan error kalau dijalankan.

Supaya komponen React bisa bekerja, ada dua aturan penting:

1. **Nama komponen harus diawali huruf kapital**

   React membedakan komponen berdasarkan huruf kapital:

```
function Header() {
  return <h1>Belajar React bersama Multimatics</h1>;
}

const appRoot = ReactDOM.createRoot(root);
appRoot.render(Header);
```

2. **Komponen dipanggil seperti tag HTML**

   Gunakan format <Header />:

```
function Header() {
  return <h1>Belajar React bersama Multimatics</h1>;
}

const appRoot = ReactDOM.createRoot(root);

// memanggil komponen React seperti tag HTML
appRoot.render(<Header />);
```

Sekarang kalau kita buka lagi di browser, komponen React tersebut akan tampil dengan benar.

Komponen Bersarang

Dalam aplikasi nyata, biasanya kita membutuhkan lebih dari satu komponen.

Di React, kita bisa **menyusun komponen di dalam komponen lain**, sama seperti kita menyusun elemen HTML.

Mari kita buat komponen baru bernama **HomePage**:

```
const container = document.getElementById('root');
const root = ReactDOM.createRoot(container);

function Header() {
       return <h1>Belajar React bersama Multimatics</h1>;
}

function HomePage() {
       return (<div></div>);
}

root.render(<HomePage />);
```

Menyisipkan Komponen di Dalam Komponen Lain

Sekarang, kita sarangkan (nest) komponen **Header** ke dalam komponen **HomePage**.

Caranya sama seperti menyisipkan tag HTML:

```
function Header() {
  return <h1>Belajar React bersama Multimatics</h1>;
}


function HomePage() {
  return (
    <div>
      <Header />
    </div>
  );
}


root.render(<HomePage />);
```