React Introduction

Prerequisites

1. Familiar dengan HTML & CSS.
2. Pengetahuan dasar tentang JavaScript dan programming.
3. Pengetahuan basic tentang DOM.
4. Familiar dengan ES6 syntax dan fiturnya.
5. js dan npm installed globally.
6. Memiliki tools yang sudah terinstall di komputer masing-masing seperti VSCode.

Goals

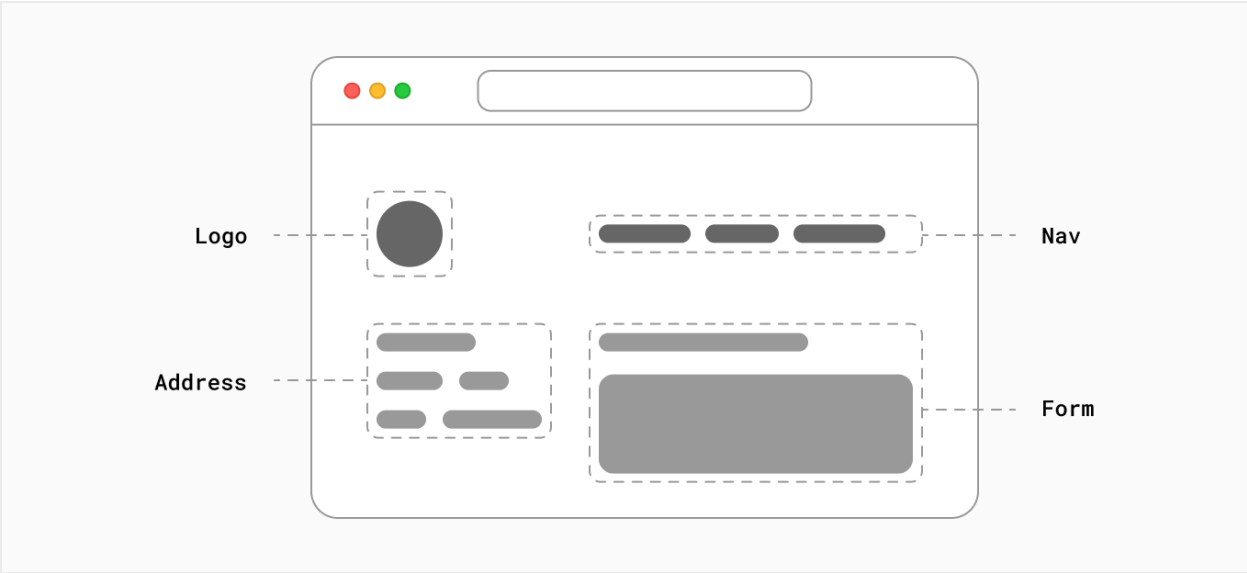
- Belajar konsep react dan istilah-istilah terkait, contohnya: Babel, Webpack, JSX, components, props, state, dan React lifecycle.
- Membangun aplikasi React sederhana sesuai dengan konsep diatas.

Scope

1. Belajar pemograman web disisi Front End. Tidak akan ada pengolahan data disisi server (database) atau Back End.
2. Menggunakan free public API untuk project yang digunakan

Apa itu React?

React adalah JS library untuk membangun user interface yang interaktif. Interface disini adalah elemen yang dilihat dan berinteraksi dengan user di layar.



React bukan framework tapi library **-** berarti menyediakan fungsi yang dapat membangun UI tetapi diserahkan kembali kepada developernya, apakah fungsi yang disediakan akan digunakan atau tidak. Komunitas developer saat ini banyak mengembangkan sebuah framework yang menyediakan fitur untuk kebanyakan aplikasi (tanpa instalasi third party secara mandiri) seperti routing, data fetching, dan *generating* HTML. Contoh framework pada React: NextJS, Remix (full stack React framework), Gatsby (CMS-backed website), Expo (react native). React digunakan untuk membangun user interface pada *front end* dan merupakan *view* layer dari aplikasi MVC.

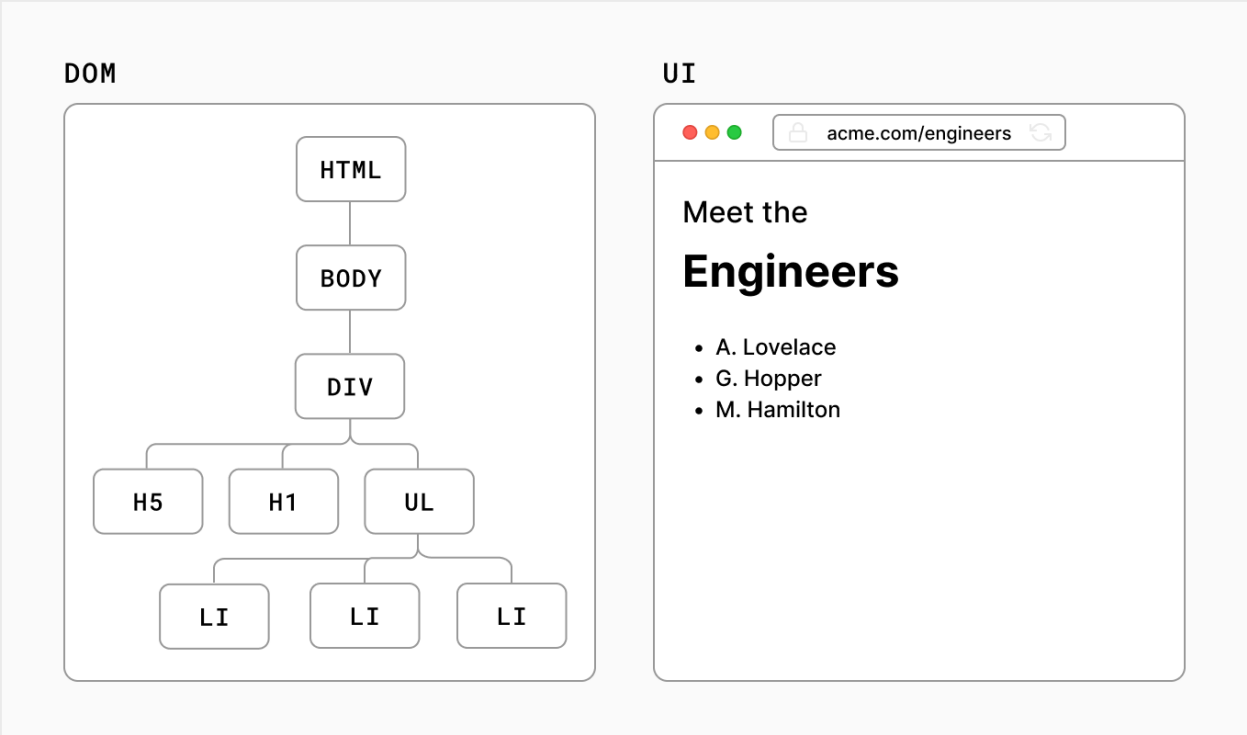
Keuntungan React

1. Arsitektur berbasis komponen — Reusable & composition memudahkan untuk code reusability, efisien dalam testing dan debugging

2. Virtual DOM: memungkinkan update UI tanpa *re-render* keseluruhan page (hanya DOM yang berubah saja) sehingga hasilnya akan lebih cepat dalam melakukan loading page.
3. JSX : React.js mengenalkan syntax extension sehingga kita dapat menulis HTML-like code didalam file JavaScript.
4. Data Binding: React mengenalkan uniderectional data flow yang berarti data flow pada satu arah (parent → children) tidak sebaliknya. Sehingga memudahkan debugging, menyederhanakan manajemen data, dan mengurangi resiko inkonsistensi data atau bug
5. Scalability: karena React merupakan library, maka penggunaannya dapat disesuaikan dengan kebutuhan, baik dalam aplikasi berskala besar atau kecil. React juga mendukung multi platform.

Rendering UI

Untuk memahami bagaimana React bekerja, perlu pemahaman dasar tentang bagaimana browser mengartikan kode untuk membuat antarmuka pengguna (UI) yang interaktif. Ketika pengguna mengunjungi sebuah halaman web, server mengembalikan berkas HTML ke browser seperti gambar berikut, kemudian browser membaca HTML tersebut dan membangun *Document Object Model* (DOM).



Apa itu DOM?

DOM adalah programming interface untuk dokumen web. DOM merepresentasikan objek dari elemen-elemen HTML, bertindak sebagai jembatan antara kode dan antarmuka pengguna, serta memiliki struktur mirip pohon yang diibaratkan seperti *hubungan induk dan anak*.

Kita dapat mengolah DOM dengan bahasa pemrograman JavaScript untuk melakukan *listen* pada *user event* dan memanipulasi DOM dengan memilih, menambahkan, memperbarui, dan menghapus elemen-elemen tertentu dalam antarmuka pengguna. Manipulasi DOM memungkinkan kita tidak hanya menargetkan elemen-elemen spesifik, tetapi juga mengubah *style* dan *content*-nya.

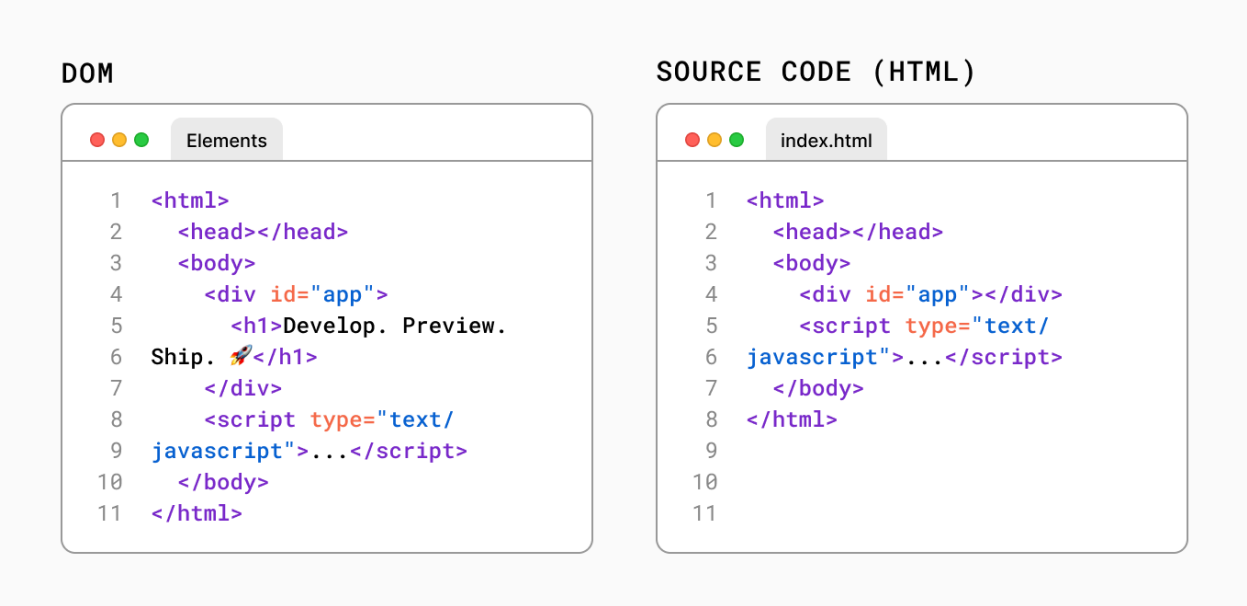
```
const paragraphs = document.querySelectorAll("p");
// paragraphs[0] is the first <p> element
// paragraphs[1] is the second <p> element, etc.
alert(paragraphs[0].nodeName);
```

Update UI JavaScript dan DOM Method

Perhatikan code berikut:

```
<!-- index.html -->
<html>
  <body>
    <div id="app"></div>
    <script type="text/javascript">
      // Select the div element with 'app' id
      const app = document.getElementById("app");

      // Create a new H1 element
      const header = document.createElement("h1");

      // Create a new text node for the H1 element
      const headerContent = document.createTextNode(
        "Develop. Preview. Ship. 🚀"
      );

      // Append the text to the H1 element
      header.appendChild(headerContent);

      // Place the H1 element inside the div
      app.appendChild(header);
    </script>
  </body>
</html>
```

Pada potongan kode tersebut, elemen h1 dibuat dengan memanipulasi DOM dengan menambahkan suatu node pada elemen induknya (app). Jika kita memperhatikan DOM element didalam browser developer tools, halaman page kita sangat berbeda dibandingkan dengan source code.



Hal tersebut terjadi karena HTML merepresentasikan *initial page content,* sedangakan DOM merepresntasikan *updated page content* yang sudah diupdate oleh JavaScript. Memperbarui DOM dengan *plain* JavaScript sangat *powerful* tetapi *verbose.*

Seiring dengan pertumbuhan tim atau apps, maka code yang ditulis akan menjadi lebih *complex*. Dengan memanipulasi DOM secara langsung, akan jauh lebih efektif karena kita dapat mendeskripsikan apa yang ingin kita tampilkan dan membiarkan komputer mencari tahu bagaimana cara memperbarui DOM secara otomatis.

Imperative vs Declarative Programming

Imperative berarti kita dapat menulis langkah yang dilakukan untuk memperbaiki antarmuka pengguna. Namun, dilain sisi, pendekatan deklaratif lebih sering disukai karena dapat mempercepat proses *development*. Menulis metode-metode DOM sangat membantu developer dalam mendeklarasikan apa yang ingin mereka tampilkan (dalam hal ini, tag h1 dengan beberapa teks).

Dengan kata lain, pemrograman imperatif diibaratkan memberikan instruksi langkah demi langkah kepada seorang koki tentang cara membuat pizza. Pemrograman deklaratif seperti memesan pizza tanpa perlu khawatir tentang langkah-langkah yang diperlukan untuk membuat pizza. 🍕

Salah satu *declarative library* yang populer yang membantu para pengembang membangun antarmuka pengguna adalah **React**. Sebagai seorang *developer*, kita

dapat memberi tahu React apa yang terjadi pada antarmuka pengguna. React akan mencari tahu langkah-langkah untuk memperbarui DOM.

Getting Started with React

Untuk menggunakan React dalam project (tanpa melakukan instalasi pada node package management), kita dapat memuat dua *script* React dari situs web eksternal bernama unpkg.com:

- **react** adalah javascript library.
- **react-dom** library yang menyediakan metode spesifik yang memungkinkan kita menggunakan React dengan DOM (complimentary library to React)

```
<!-- index.html -->
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <script type="text/javascript">
      const app = document.getElementById("app");
    </script>
  </body>
</html>
```

Daripada memanipulasi DOM secara langsung dengan *plain* JavaScript, kita dapat menggunakan metode `ReactDOM.render()` dari library `react-dom` untuk memberi tahu React untuk merender judul `<h1>` kita di dalam elemen `#app` kita.

```
<!-- index.html -->
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <script type="text/javascript">
      const app = document.getElementById("app");
      ReactDOM.render(<h1>Develop. Preview. Ship. 🚀</h1>, app);
    </script>
```

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

Namun, jika kita mencoba menjalankan kode ini di browser, kita akan mendapatkan syntax error:

Error terjadi karena `<h1>...</h1>` bukanlah syntax Javascript yang valid. Potongan kode ini adalah JSX.

Apa itu JSX?

JSX adalah syntax extension pada JavaScript untuk mendeskripsikan UI dengan *HTML-like* syntax. Kelebihan dari JSX adalah kita tidak perlu belajar simbol atau sintaks yang baru selain dari HTML dan JavaScript. Dalam praktiknya, kita harus mengikuti 3 aturan penulisan kode JSX sebagai berikut:

1. Mengembalikan single root element

   Untuk mengembalikan beberapa elemen dari sebuah komponen, ***wrap* dengan *single parent tag*.** Sebagai contoh, kita dapat menggunakan tag `<div>`:

```
<div>
  <h1>Hedy Lamarr's Todos</h1>
  <img src="https://i.imgur.com/yXOvdOSs.jpg" alt="Hedy Lamarr" class="photo" />
  <ul>
    ...
  </ul>
</div>
```

atau menggunakan tag kosong seperti berikut:

```
<>
  <h1>Hedy Lamarr's Todos</h1>
  <img
    src="https://i.imgur.com/yXOvdOSs.jpg"
    alt="Hedy Lamarr"
    class="photo"
  >
```

```
  <ul>
    ...
  </ul>
</>
```

Tag kosong diatas disebut dengan *fragment.* **Fragments** digunakan untuk mengelompokan sesuatu tanpa meninggalkan jejak pada pohon HMTL di browser.

💡 Mengapa *multiple tag JSX* harus dibungkus dalam suatu *parent tag*?
JSX terlihat seperti HTML, tetapi sebenarnya proses dibelakangnya diubah menjadi objek JavaScript biasa. Kita tidak bisa mengembalikan dua objek dari sebuah fungsi tanpa membungkusnya ke dalam sebuah array. Sama halnya dengan JSX, kita juga tidak bisa mengembalikan dua tag JSX tanpa membungkusnya ke dalam tag lain atau pada sebuah Fragment.

2. Tutup semua tags yang didefinisikan

   JSX mengharuskan tag untuk ditutup secara eksplisit: tag yang ditutup sendiri (*self-closing tag*) seperti `<img>` harus ditulis sebagai `<img />`, dan tag yang dibungkus seperti `<li> oranges` harus ditulis sebagai `<li>oranges</li>`. Contoh:

   ```
   <>
     <img
       src="https://i.imgur.com/yXOvdOSs.jpg"
       alt="Hedy Lamarr"
       class="photo"
      />
     <ul>
       <li>Invent new traffic lights</li>
       <li>Rehearse a movie scene</li>
       <li>Improve the spectrum technology</li>
     </ul>
   </>
   ```

3. Kebanyakan kode JSX ditulis dengan camelCase

   Contoh: `stroke-width` menjadi `strokeWidth`, class menjadi className, dll

```
<img
  src="https://i.imgur.com/yXOvdOSs.jpg"
  alt="Hedy Lamarr"
  className="photo"
/>
```

Perlu diingat bahwa browser tidak memahami JSX secara otomatis, sehingga kita perlu kompiler JavaScript seperti **Babel** agar dapat mengubah kode JSX Anda menjadi JavaScript biasa. Keuntungan dari kompiler tersebut adalah mengatasi masalah yang berkaitan dengan *browser compatibility*.

Menambahkan Babel pada project

Untuk menambahkan babel pada projek, copy dan paste the script berikut pada `index.html` file:

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

Sebagai tambahan, kita harus menginformasikan pada Babel kode apa yang ingin ditransformasi dengan mengubah tipe script ke `type=text/jsx`.

```
<html>
  <body>
    <div id="app"></div>
    <script src="https://unpkg.com/react@17/umd/react.development.js"></script>
    <script src="https://unpkg.com/react-dom@17/umd/react-dom.development.js"></script>
    <!-- Babel Script -->
    <script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
    <script type="text/jsx">
      const app = document.getElementById('app');
      ReactDOM.render(<h1>Develop. Preview. Ship. 🚀</h1>, app);
    </script>
  </body>
</html>
```

Kembali ke browser dan cek apakah kode yang baru saja kita tambahkan berjalan.

dengan kode **declarative** pada React, kita cukup menulis:

```
<script type="text/jsx">
  const app = document.getElementById("app")
  ReactDOM.render(<h1>Develop. Preview. Ship. 🚀</h1>, app)
</script>
```

dibandingkan dengan kode **imperative** JavaScript yang kita tulis pada bagian sebelumnya:

```
<script type="text/javascript">
  const app = document.getElementById('app');
  const header = document.createElement('h1');
  const headerContent = document.createTextNode('Develop. Preview. Ship. 🚀');
  header.appendChild(headerContent);
  app.appendChild(header);
</script>
```

Dari contoh tersebut dapat disimpulkan bahwa keuntungan menggunakan React adalah memangkas banyak kode yang berulang.

Referensi

- https://react.dev/learn/writing-markup-with-jsx#the-rules-of-jsx
- https://nextjs.org/learn/foundations/from-javascript-to-react
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
- https://www.w3schools.com/
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction