# Modul Panduan Praktikum Pemrograman Web

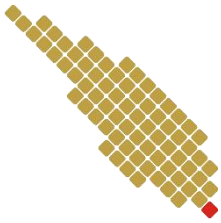
**PERTEMUAN**

**3**

## Preprocessors & Responsive Design


**Asisten Praktikum**

- Aminudin Fadila
- Defangga Aby Vonega
- Markus Togi Fedrian Rivaldi Sinaga


**TEKNIK INFORMATIKA**

JURUSAN TEKNOLOGI PRODUKSI INDUSTRI DAN INFORMASI

INSTITUT TEKNOLOGI SUMATERA

**2022**

## Dasar Teori

### 1. Block Element Modifier

Block Element Modifier (BEM) merupakan aturan penulisan atau penamaan *class* yang mengikuti suatu pola terstruktur.

> BEM bukan suatu *framework* ataupun *library*, melainkan sebuah metodologi. BEM dirancang untuk membantu dalam memodulasi struktur dengan memecah komponen menjadi blok yang berisi elemen dan dapat dimodifikasi. Dalam memberi nama untuk mengidentifikasi element HTML hanya menggunakan *class*, tidak menggunakan ID. Setiap elemen hanya boleh bergantung pada komponen blok atau *class* induk, meskipun blok itu merupakan bagian dari blok lain.

```
<nav class="main-nav">
  <ul class="main-nav__list">
    <li class="main-nav__item--modifier">
      <a href="#" class="main-nav__link">Home</a>
    </li>
    <li class="main-nav__item">
      <a href="#" class="main-nav__link">About</a>
    </li>
    <li class="main-nav__item">
      <a href="#" class="main-nav__link">Contact</a>
    </li>
  </ul>
</nav>
```

BEM mencakup tiga komponen penyusun, sesuai dengan namanya sendiri :

- **Block**

  Merupakan class atau komponen induk dari element yang didefinisikan.
  Contoh:
  `<nav class="main-nav">`

- **Element**

  Bagian dari komponen induk. Element yang berada di dalam **block** dengan nama class yang mengikutsertakan nama *class* dari element induk dan menggunakan dua garis bawah yang menghubungkan kedua class.
  Contoh :

```
<ul class="main-nav__list">
```

- **Modifier**

  Bertujuan untuk memodifikasi atau mengubah gaya tampilan **block** atau **element** dengan menambahkan dua tanda hubung di belakang class.
  Contoh :
  ```
  <li class="main-nav__item--modifier">
  ```

Dengan BEM kita tidak perlu menggunakan beberapa *class* CSS dalam satu deklarasi (meminimalisir penggunaan *combinator*). Penulisan CSS selector dari elemen-elemen diatas :

```
.main-nav{}
.main-nav__list {}
.main-nav__item {}
.main-nav__link{}
```

Contoh lain dari penggunaan BEM:

```
<button class="btn btn--primary">Edit</button>
<button class="btn btn--danger">Delete</button>

<style>
  .btn {
    display: inline-block;
  }
  .btn--primary {
    color: blue;
  }
  .btn--danger {
    color: red;
  }
</style>
```

BEM dapat dikatakan sebagai pendekatan ***Component First*** dalam penulisan *class* CSS pada elemen HTML. Tujuan dari penggunaan ini adalah untuk membuat penamaan *class* CSS menjadi lebih terstruktur dan bermakna (*semantic*), sehingga memudahkan pengembang ketika bekerja dalam tim, dan memudahkan untuk tujuan pemeliharaan di kemudian hari.

## 2. CSS Preprocessor

Jika dicermati lebih lanjut, tetap ada penulisan berulang dimana nama ***block*** selector selalu ditulis kembali, dalam hal ini CSS Preprocessor dapat menjadi solusi. CSS Preprocessor merupakan program yang digunakan untuk menghasilkan (*generating*) CSS menggunakan *preprocessed file* dan sintaks unik tertentu. Salah satu contoh CSS Preprocessor yang banyak digunakan adalah Sass. CSS Preprocessor seperti Sass membantu kita untuk melakukan hal-hal yang tidak tersedia di CSS murni (*pure CSS*), seperti :

- **Variables**

  Variabel merupakan cara untuk menyimpan informasi yang ingin kita gunakan kembali di seluruh stylesheet kita. Kita dapat menyimpan hal-hal seperti warna, jenis font, atau nilai CSS apa pun yang menurut kita ingin digunakan kembali. Sass menggunakan **$** simbol untuk membuat sesuatu menjadi variabel.

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

- **Nesting**

  Kita sudah memperhatikan bahwa HTML memiliki hirarki bersarang dan visual yang jelas, sementara di sisi lain, CSS tidak demikian.

  Sass memungkinkan kita untuk menyarangkan CSS *selector* dengan cara mengikuti hirarki visual yang sama seperti pada HTML. Ketahuilah bahwa aturan yang terlalu bersarang akan menghasilkan CSS yang *over-qualified*, yang umumnya sulit untuk dipelihara (*maintain*) dan umumnya dianggap sebagai *bad practice*.

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Untuk kasus seperti pada Dasar Teori BEM sebelumnya, kita bahkan dapat menggunakan bantuan **&** untuk mempersingkat pekerjaan kita, seperti yang dapat kita lihat pada contoh berikut :

**HTML**

```
<div class="posts">
  <ul class="list">
    <li class="list__item">
      <h2 class="list__title"></h2>
        <p class="list__description"></p>
    </li>
  </ul>
</div>
```

**SCSS**

```
.posts {
  &__list {
    &__item {

    }
    &__title {

    }
    &__description {

    }
  }
}
```

- **Partials / Modules**

  Kita dapat membuat file Sass terpisah yang berisi aturan-aturan yang dapat kita gunakan di dalam file Sass lain. Selain ini dapat memudahkan pemeliharaan, ini juga berguna ketika kita memiliki aturan yang hanya ingin kita gunakan di saat-saat tertentu (tidak selalu ingin kita gunakan). Untuk membuat file *partial*, kita dapat menggunakan garis bawah dalam penamaan file tersebut seperti `_partial.scss`, sehingga Sass tidak akan melakukan *generate* CSS dari file tersebut.

  Kemudian kita dapat menggunakan file *partial* tadi sebagai *module* dengan kata kunci **@use** di dalam file Sass yang kita inginkan. Dengan begitu kita dapat merujuk ke variabel, mixin, dan fungsi di dalam file partial yang saat ini sudah dianggap sebagai *module*.

```
// _base.scss
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

```
// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}

.inverse {
  background-color: #333;
  color: white;
}
```

- **Mixins**

  Mixin bekerja seperti “kelas” dalam PBO, atau lebih tepat seperti perpaduan “fungsi” dan “*struct*” dalam Struktur Data (mis. C++). Mixin memungkinkan kita

membuat grup deklarasi CSS yang ingin kita gunakan kembali di seluruh dokumen kita. Ini membantu menjaga Sass kita tetap **DRY** (***D**on't **R**epeat **Y**ourself*). Kita bahkan dapat mengirimkan nilai untuk membuat mixin kita menjadi lebih fleksibel. Berikut adalah contoh untuk penggunaan mixin untuk theme.

```scss
@mixin theme($theme: DarkGray) {
  background: $theme;
  box-shadow: 0 0 1px rgba($theme, .25);
  color: #fff;
}

.info {
  @include theme;
}
.alert {
  @include theme($theme: DarkRed);
}
.success {
  @include theme($theme: DarkGreen);
}
```

```css
.info {
  background: DarkGray;
  box-shadow: 0 0 1px rgba(169, 169, 169, 0.25);
  color: #fff;
}

.alert {
  background: DarkRed;
  box-shadow: 0 0 1px rgba(139, 0, 0, 0.25);
  color: #fff;
}

.success {
  background: DarkGreen;
  box-shadow: 0 0 1px rgba(0, 100, 0, 0.25);
  color: #fff;
}
```

Untuk membuat mixin kita dapat menggunakan kata kunci **@mixin** dan memberikannya nama. Kita juga dapat menggunakan variabel setelah penamaannya sehingga variabel ini akan berfungsi layaknya parameter dalam sebuah fungsi, dimana kita dapat memberikannya nilai. Setelah mixin-nya didefinisikan, kita dapat menggunakannya dengan kata kunci **@include**.

- **Extends / Inheritance**

  Menggunakan **@extend** membuat kita dapat berbagai sekumpulan aturan antar beberapa selector.

**SCSS**

```scss
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

.success {
  @extend %message-shared;
  border-color: green;
}

.error {
  @extend %message-shared;
  border-color: red;
}

.warning {
  @extend %message-shared;
  border-color: yellow;
}
```

**CSS**

```css
/* This CSS will print because %message-shared is extended. */
.message, .success, .error, .warning {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

- **Operator**

  Melakukan operasi matematika di CSS dapat menjadi sangat membantu. Sass memiliki beberapa operator matematika standar seperti **+**, **-**, *****, **math.div()**, dan **%**.

```scss
@use "sass:math";

.container {
  display: flex;
}

article[role="main"] {
  width: math.div(600px, 960px) * 100%;
}

aside[role="complementary"] {
  width: math.div(300px, 960px) * 100%;
  margin-left: auto;
}
```

```css
.container {
  display: flex;
}

article[role="main"] {
  width: 62.5%;
}

aside[role="complementary"] {
  width: 31.25%;
  margin-left: auto;
}
```

## 3. Responsive Web Design

Desain Web Responsif adalah tentang menggunakan HTML dan CSS untuk secara otomatis mengubah ukuran, menyembunyikan, mengecilkan, atau memperbesar elemen dalam sebuah situs web, agar terlihat baik di sebanyak mungkin jenis perangkat (desktop, tablet, dan ponsel). Terdapat beberapa teknik untuk membuat sebuah halaman web yang responsif, seperti :

- **Mengatur Viewport**

```html
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

  Meta tag dengan atribut **name="viewport"** memberi petunjuk kepada browser tentang pengaturan dimensi dan penskalaan halaman.

- **Ukuran Teks Responsif**

  Masing-masing dimensi layar *device* mempengaruhi ukuran text mana yang tepat untuk digunakan, maka dari itu harus diperhatikan juga atribut *font-size* dari teks pada halaman web kita. Untuk mencapainya, teks dapat dengan menggunakan satuan relatif (**em**, **rem**, dsb.), dapat dengan menyesuaikan ukuran *font* untuk masing-masing *breakpoint* ukuran layar perangkat, atau dengan menggunakan kombinasi dari keduanya.

- **Ukuran Gambar Responsif**

  Sama seperti teks, gambar juga menjadi elemen yang penting untuk dibuat responsif, hal ini dapat ditangani baik dengan mengganti gambar di masing-masing *breakpoint* ukuran layar perangkat, atau membuatnya responsive melalui *height* dan *width* yg menggunakan satuan relatif (**%**, **vh**, **vw**, dsb.).

- **Media Query**

Terkadang kita perlu membuat perubahan yang lebih luas secara signifikan pada tata letak elemen dalam halaman web kita untuk mendukung responsivitas pada ukuran layar tertentu, daripada yang dimungkinkan oleh teknik yang ditunjukkan di atas. Di sinilah *media query* mengambil peran.

***Media query*** adalah filter sederhana yang dapat diterapkan ke CSS. Filter ini memudahkan kita untuk mengubah gaya berdasarkan jenis dan fitur perangkat yang me-*render* konten dalam halaman web kita, misalnya lebar layar, tinggi layar, orientasi layar, kemampuan untuk mengarahkan kursor, dan apakah perangkat mendukung teknologi layar sentuh. Berikut adalah contoh penggunaan *media query* (**CONTOH HEADER**):

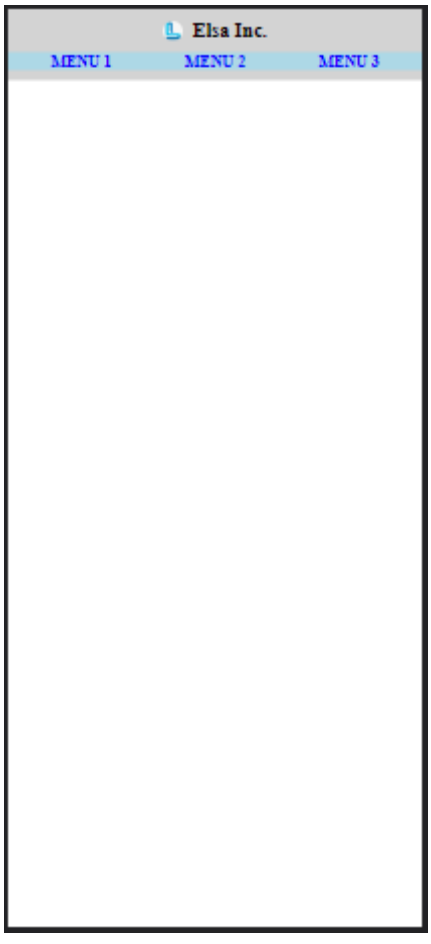
**HTML**

```html
<header>
    <div class="logo">
        <div class="logo-img">
            <img src="L.jpg" alt="Logo">
        </div>
        <p class="logo-text">Elsa Inc.</p>
    </div>
    <div class="menu">
        <p>Menu 1</p>
        <p>Menu 2</p>
        <p>Menu 3</p>
    </div>
</header>
```

**CSS**

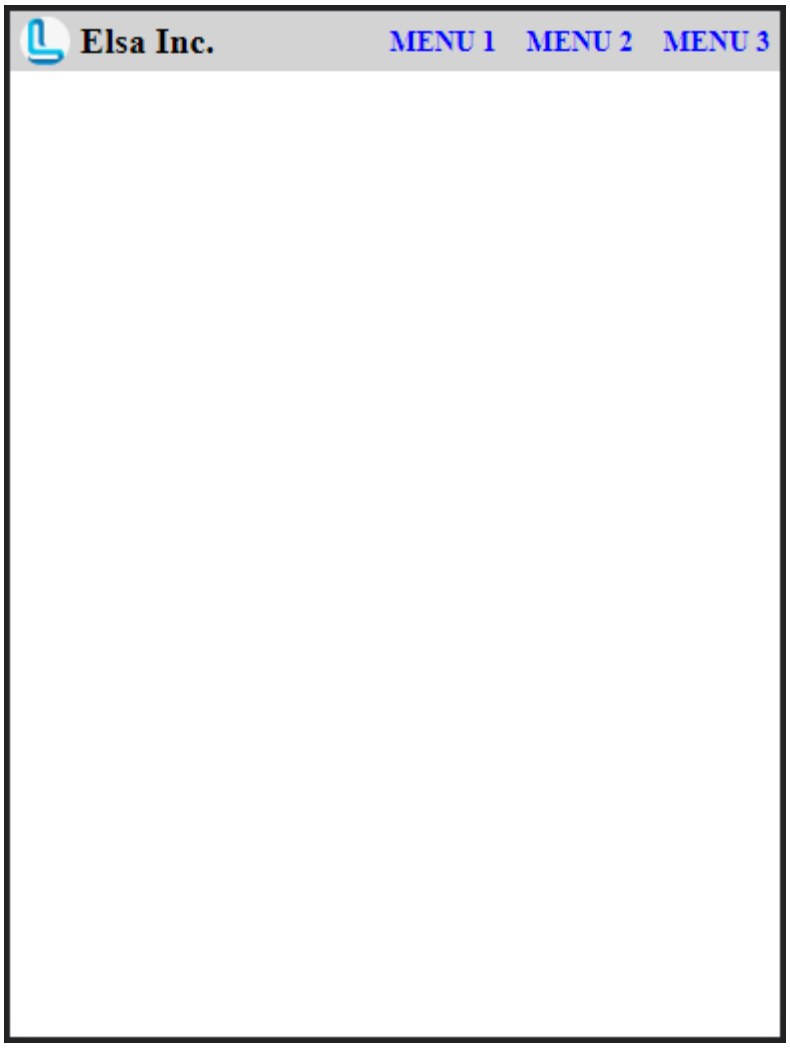
**Media Query untuk ukuran layar Generic Mobile**

```css
/* GENERIC MOBILE */
@media (min-width: 360px) {
    .logo-img {
        width: 20px;
    }
    .menu{
        font-size: 1rem;
    }
    .logo-text {
        font-size: 1.25rem;
    }
    .menu {
        gap: 30px;
    }
}
```



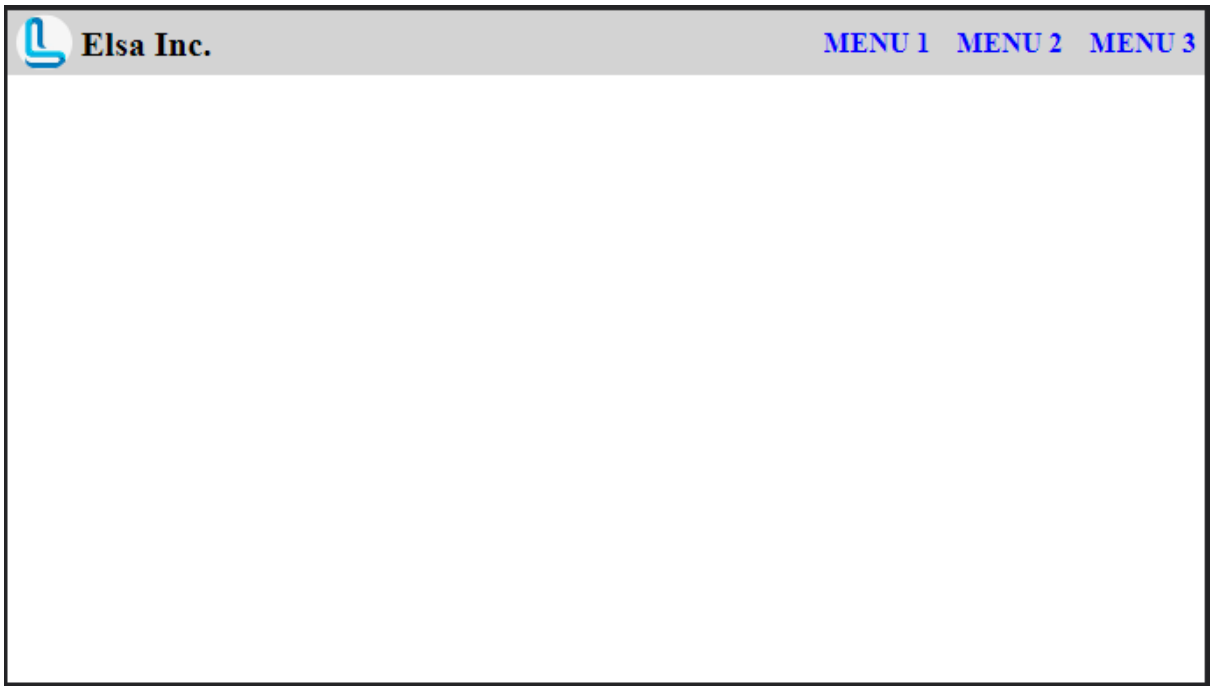
**Media Query untuk ukuran layar Generic Tablet**

```css
/* GENERIC TABLET */
@media (min-width: 768px) {
    .logo {
        margin: 0;
    }
    .logo-img {
        width: 50px;
    }
    .menu{
        font-size: 1.75rem;
        background-color: initial;
    }
    .logo-text {
        font-size: 2.25rem;
    }
    header {
        display: flex;
        padding: 5px 10px;
        justify-content: space-between;
        align-items: center;
    }
}
```



**Media Query untuk ukuran layar Generic Desktop**

```
/* GENERIC DESKTOP */
@media (min-width: 1366px) {
    .logo-img {
        width: 64px;
    }
    .menu{
        font-size: 2rem;
    }
    .logo-text {
        font-size: 2.5rem;
    }
}
```



## Latihan Percobaan

Coba buat Header halaman website seperti pada **CONTOH HEADER**

## Tugas

Modifikasi hasil Tugas Pertemuan 2 agar memiliki tampilan yang responsif. Tantang diri kalian untuk membuat tampilan yang sebaik mungkin!

| **Parameter Penilaian** | **Bobot Penilaian** |
|---|---|
| Kelengkapan | 50% |
| Kesesuaian penggunaan CSS selector dan fungsinya | 20% |
| Keindahan dan estetika ☺ | 20% |
| Penggunaan "*best practices*" menurut standar W3C (dapat melalui https://www.freeformatter.com/html-validator.html) | 10% |

## Daftar Pilihan Referensi

- mdn web docs
- W3Schools

## Latihan dan Sertifikasi Gratis

- freeCodeCamp - Responsive Web Design