

Modul Panduan Praktikum Pemrograman Web

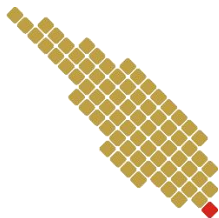
PERTEMUAN

2

CSS

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. CSS

CSS (*Cascading Style Sheets*) merupakan bahasa *stylesheet* yang digunakan untuk mendeskripsikan penyajian dokumen yang ditulis menggunakan bahasa markup HTML atau XML (termasuk SVG, MathML, dan XHTML). CSS mendeskripsikan bagaimana elemen harus di-*render* di layar, di kertas, atau media lainnya.

CSS merupakan salah satu dari sekian bahasa utama *open web*, yang diatur berdasarkan spesifikasi W3C. CSS digunakan untuk menerapkan “*style*” dan mengatur tata letak pada halaman web — seperti mengubah *font*, warna, ukuran, dan *spacing* dari konten halaman web, hingga menambahkan animasi atau fitur dekorasi lainnya.

HTML tanpa CSS

```
<head>
  <title>Document</title>
</head>
<body>
  <h1>Tanpa CSS</h1>
</body>
```

Tanpa CSS

HTML dengan CSS

```
<head>
  <title>Document</title>
</head>
<body>
  <h1
    style="
      font-family:Arial, Helvetica, sans-serif;
      color:green;
      background-color: lightgreen;
      padding: 10px;
      width: fit-content;
    ">
    Dengan CSS
  </h1>
</body>
```

Dengan CSS

Untuk penggunaan CSS (*styling*), dapat dilakukan dengan 3 cara, yaitu :

- **Inline**

Menggunakan atribut ***style***, untuk menerapkan *rule* ke elemen spesifik tertentu.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
</head>
<body>
  <h1 style="
    font-family: Arial;
    color: green;
  ">
    Heading 1
  </h1>
</body>
</html>
```

- **Internal**

Menggunakan **`<style>`** pada bagian **`<head>`** untuk menerapkan *rules* ke sebuah dokumen HTML.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
  <style>
    h1 {
      font-family: Arial;
      color: green;
    }
  </style>
</head>
<body>
  <h1>Heading 1</h1>
</body>
</html>
```

- **External**

Menggunakan file *stylesheet* (file CSS) eksternal, untuk menerapkan *rules* ke satu atau lebih dokumen HTML, melalui tag meta **`<link>`** pada bagian **`<head>`** dokumen HTML yang ingin menggunakannya.

CSS

```
h1 {
  font-family: Arial;
  color: green;
}
```

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Document</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Heading 1</h1>
</body>
</html>
```

Ketiga contoh kode diatas bila dijalankan di browser akan memiliki hasil yang sama sebagai berikut :

Heading 1

2. Cascade, Specificity, and Inheritance

- **Cascade**, secara sederhana dapat diartikan sebagai fitur dari CSS yang memperhatikan urutan dari *styling rules*, dimana, ketika terdapat 2 atau lebih *rules* yang sama diterapkan, dan semuanya memiliki *specificity* yang setara, maka yang terakhir kali didefinisikan akan digunakan (***last style wins***).
- **Specificity**, merupakan algoritma yang digunakan browser untuk menentukan *rule* mana yang akan diterapkan. Jika terdapat lebih dari satu blok *rule* yang mengatur properti yang sama, namun dengan nilai yang berbeda, dan ditujukan pada elemen yang sama, *specificity* berperan dalam menentukan properti mana yang akan diterapkan. *Specificity* akan dijelaskan lebih lanjut pada bagian CSS Selector.
- **Inheritance**, merupakan kondisi dimana sebagian properti dari *styling rule* yang diterapkan pada suatu elemen, akan diwarisi oleh elemen anak (*child elements*) di dalamnya – dan sebagian tidak. Untuk mengetahui properti *styling rule* apa saja yang diwarisi elemen anak ketika diterapkan ke elemen induknya, dapat dilihat [di sini](#).

Untuk memahami lebih lanjut bagaimana ketiga konsep ini bekerja secara bersamaan dapat dibaca di [Cascade and Inheritance](#).

3. CSS Selector

Dalam CSS, *selector* digunakan untuk merujuk/menargetkan elemen HTML di halaman web yang ingin kita terapkan *styling rule*. Terdapat beberapa kategori *selector* dalam CSS, yaitu :

A. Basic Selectors

Basic Selectors terbagi lagi menjadi :

- **Universal Selector**
Sintaks : `* ns|* *|*`
Contoh : `*` merujuk pada **seluruh** elemen pada dokumen HTML
- **Type Selector**
Sintaks : `elementname`
Contoh : `div` merujuk ke **seluruh** elemen `<div>` pada dokumen HTML
- **Class Selector**
Sintaks : `.classname`
Contoh : `.red` merujuk ke **seluruh** elemen pada dokumen HTML yang memiliki nilai `red` pada atribut **class**-nya.
- **ID Selector**
Sintaks : `#idname`
Contoh : `#yellow` merujuk ke elemen pada dokumen HTML yang memiliki nilai `yellow` pada atribut **id**-nya.
- **Attribute Selector**
Sintaks : `[attr]` `[attr=value]` `[attr~=value]` dsb.

Contoh : `[type=text]` merujuk ke **seluruh** elemen pada dokumen HTML yang memiliki atribut `type` dengan nilai `text` pada.

B. Grouping Selectors

Grouping Selectors (`()`) digunakan untuk mengelompokkan seluruh elemen yang pada mereka ingin diterapkan *styling rule* sekaligus.

Contoh : `div, span` merujuk ke **seluruh** elemen `<div>` dan elemen `` pada dokumen HTML.

C. Combinators

Combinators terbagi lagi menjadi :

- **Descendant Combinator**

Descendant Combinators ((spasi)) digunakan untuk memilih elemen (setelah spasi) yang merupakan keturunan dari elemen elemen sebelumnya (sebelum spasi).

Contoh : `div span` merujuk ke **seluruh** elemen `` yang berada di dalam elemen `<div>` pada dokumen HTML.

- **Child Combinator**

Child Combinators (`>`) digunakan untuk memilih elemen (setelah tanda lebih dari) yang merupakan keturunan/anak langsung dari elemen elemen sebelumnya (sebelum tanda lebih dari).

Contoh : `div > span` merujuk ke **seluruh** elemen `` yang merupakan elemen keturunan/anak langsung dari elemen `<div>` pada dokumen HTML.

- **General Sibling Combinator**

General Sibling Combinator (`~`) digunakan untuk memilih elemen (setelah tanda negasi) yang mengikuti elemen sebelumnya (sebelum tanda negasi) meski tidak secara langsung, selama keduanya berada di dalam elemen induk yang sama.

Contoh : `div ~ span` merujuk ke **seluruh** elemen `` yang didefinisikan setelah elemen `<div>` baik langsung maupun tidak langsung, selama keduanya berada dalam elemen induk yang sama.

- **Adjacent Sibling Combinator**

Adjacent Sibling Combinator (`+`) digunakan untuk memilih elemen (setelah tanda tambah) yang mengikuti elemen sebelumnya (sebelum tanda tambah) secara langsung dan keduanya berada di dalam elemen induk yang sama.

Contoh : `div + span` merujuk ke elemen `` **pertama** yang didefinisikan tepat setelah elemen `<div>`, dimana keduanya berada dalam elemen induk yang sama.

D. Pseudo-classes

CSS pseudo-class (diawali `:`) merupakan kata kunci yang ditambahkan ke sebuah *selector* yang menetapkan kondisi istimewa dari elemen yang dipilih.

Contoh : `button:hover` dapat digunakan untuk menentukan perubahan properti *styling rule* pada elemen `button` ketika *pointer* pengguna melaluinya.

Dokumentasi lengkap terkait CSS pseudo-class dapat dilihat di [Pseudo-Classes](#).

E. Pseudo-elements

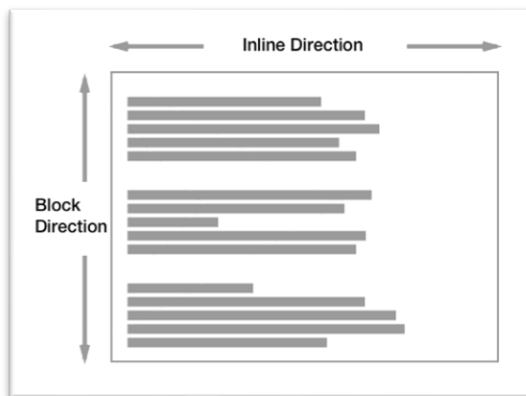
CSS pseudo-elements (diawali `::`) merupakan kata kunci yang ditambahkan ke sebuah *selector* yang memungkinkan kita untuk menerapkan *styling rule* ke bagian spesifik dari elemen yang dipilih.

Contoh : `p::first-line` dapat digunakan untuk memodifikasi baris pertama dari sebuah elemen `<p>`.

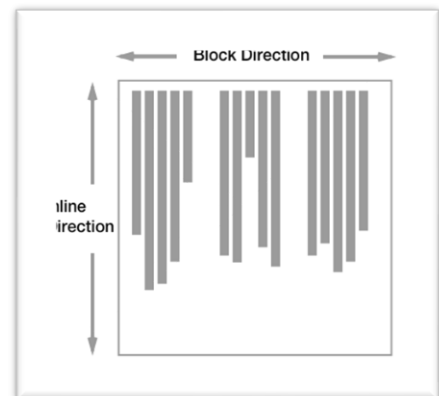
Dokumentasi lengkap terkait CSS pseudo-elements dapat dilihat di [Pseudo-Elements](#).

4. Box Model

Box Model merupakan konsep dimana seluruh elemen memiliki kotaknya masing-masing disekelilingnya, dimana setiap kotak ini memiliki **inner display type** dan **outer display type**. Terdapat dua tipe utama dalam CSS, yaitu **Block Box** dan **Inline Box**. Kedua tipe ini menunjukkan bagaimana sifat dari sebuah kotak dalam hal *flow* dari halaman, dan terkait hubungan antar kotak dalam halaman.



Arah *page flow* CSS box pada mode penulisan horizontal



Arah *page flow* CSS box pada mode penulisan vertikal

4.1. Block Box

Dalam **block formatting context**, kotak-kotak elemen disusun berurut secara vertikal dari atas ke bawah. Jarak vertikal antar elemen ini diatur oleh *margin*. Penting diingat bahwa untuk dua elemen yang saling bersebelahan, margin keduanya akan saling menimpa.

4.2. Inline Box

Dalam **inline formatting context**, kotak-kotak elemen disusun berurut secara horizontal dari sisi awal sebuah kotak yang menampung mereka. Jarak secara horizontal yang disebabkan oleh, *margin*, *padding*, dan *border*, pada kotak-kotak ini selalu diperhatikan. Sementara untuk secara vertikal, dapat disesuaikan dengan beberapa cara tambahan seperti : melakukan perataan (*alignment*) pada nilai *top* dan *bottom*, atau pada *baseline* teks yang mereka tampung.

4.3. *Outer Display Type*

- **Block**

- elemen ditampilkan di baris baru
- *width* dan *height* diperhatikan
- *padding*, *margin*, dan *border* akan mengakibatkan elemen lain didorong menjauh secara vertikal maupun horizontal
- kotak elemen memenuhi ruang dari penampungnya secara horizontal
- contoh : *heading* dan *paragraph*

- **Inline**

- elemen tidak ditampilkan di baris baru
- *width* dan *height* tidak diperhatikan
- *padding*, *margin*, dan *border* vertikal akan diterapkan namun tidak akan mengakibatkan elemen lain didorong menjauh
- *padding*, *margin*, dan *border* horizontal akan diterapkan dan akan mengakibatkan elemen lain didorong menjauh
- contoh : *anchor* dan *span*

4.4. *Inner Display Type*

Inner display type pada CSS bertugas untuk menentukan bagaimana elemen-elemen didalamnya disusun. Secara default dan bila tidak diberikan instruksi lainnya, elemen di dalam sebuah kotak (yang merupakan elemen juga), disusun secara normal flow, dan berlaku sebagai block box atau inline box.

Lebih lanjut, terdapat pilihan untuk mengubah inner display type sebuah elemen, seperti flex, dimana suatu elemen akan tetap menggunakan block box sebagai outer display type-nya, namun inner display type-nya akan diubah menjadi flex. Pada kasus ini, seluruh elemen yang merupakan elemen-anak-langsung dari elemen yang diberikan display bernilai flex, akan menjadi flex items, dan akan berlaku menyesuaikan dengan spesifikasi flexbox induknya.

4.5. *Alternative CSS Box Model*

Pada *Alternative CSS Box Model*, ukuran *width* dan *height* yang didefinisikan merupakan ukuran sebenarnya dari kotak yang ditampilkan di layar, sehingga area dalam kotak yang digunakan sebagai penampung konten merupakan hasil selisih dari *width* dengan *padding* dan *border*. Untuk menggunakannya, dapat dilakukan dengan menambahkan **box-sizing: border-box**, pada elemen yang kita inginkan.

Selengkapnya tentang CSS Box Model dapat dilihat [di sini](#).

5. **CSS Values and Units**

Pada CSS terdapat beberapa nilai (*value*) dan satuan (*unit*) yang baku. Nilai yang dimaksud berupa nilai valid yang dapat digunakan untuk properti tertentu, seperti warna dan ukuran. Beberapa nilai ini ada yang memiliki satuan, ada yang berupa *string* atau *identifier*, serta beberapa pula dapat berupa hasil dari suatu fungsi.

5.1. Warna

Pada warna di CSS, nilai yang dapat diterima bisa berupa :

- **Kata kunci warna**, seperti **red**, **green**, **blue**, dan beberapa kata kunci warna bawaan CSS lainnya.
- **RGB dan RGBA**, seperti **rgb(255,255,255)**, **rgb(0,0,0)**, dimana terdapat 16.777.216 kemungkinan warna. Kita juga dapat menggunakan parameter keempat yaitu nilai **Alpha** yang mengatur opasitas dari warna.
- **HSL dan HSLA**, seperti **hsl(360, 100%, 100%)** dan **hsl(0, 0%, 0%)**. Sama seperti pada RGB dan RGBA, kita juga dapat menggunakan parameter keempat yaitu nilai **Alpha** yang mengatur opasitas dari warna.
- **Hexadecimal RGB**, seperti **#ffffff**, **#000000**, dan sama seperti pada RGB, terdapat 16.777.216 kemungkinan warna. Pada hexadecimal, kita dapat menggunakan **Alpha** untuk mengatur opasitas dengan menambahkan kombinasi bilangan heksadesimal keempat, daftar lengkapnya dapat dilihat [di sini](#).

5.2. Ukuran dan Dimensi

Ukuran pada CSS dapat berupa bilangan bulat (*integer*), bilangan desimal (*number*), persentase (*percentage*), dan hasil dari sebuah fungsi. Untuk ukuran terdapat dua jenis satuan (*unit*) yang dapat kita pilih, yaitu :

- **Satuan Mutlak**, seperti **px**, **cm**, **in**, **pt**, dsb.
- **Satuan Relatif**, seperti **em**, **rem**, **vh**, **vw**, dsb.

Daftar lengkap dari satuan ukuran yang dapat digunakan di CSS dapat dilihat [di sini](#).

Dimensi lainnya yang umum digunakan pada CSS adalah **sudut** (contohnya **45deg**) dan waktu (contohnya **300ms** dan **2.5s**).

5.3. Image

Tipe nilai ini digunakan ketika suatu gambar yang dimasukkan berupa nilai yang valid, dapat berupa gambar secara harfiah yang dirujuk menggunakan **url()**, atau sebuah gradasi. Nilai ini umumnya digunakan pada **background-image**.

5.4. Position

Nilai ini digunakan untuk menyesuaikan posisi sebuah elemen secara dua dimensi (sumbu X dan sumbu Y), nilai yang tersedia adalah **top**, **right**, **bottom**, **left**, dan **center**, atau dapat pula diberi nilai berupa **ukuran** (*length*).

5.5. String dan Identifier

Nilai-nilai berupa kata kunci warna seperti red, green, blue, lebih sering dianggap sebagai *identifier*, pada CSS, nilai yang berupa *identifier* tidak perlu diberi tanda kutip. Namun terdapat pula beberapa kasus, dimana kita menggunakan *string* sebagai nilai pada CSS, berbeda dengan *identifier*, pada CSS *string* menggunakan tanda kutip dalam penggunaannya.

5.6. Fungsi

Pada CSS, kita juga dapat menggunakan hasil dari sebuah fungsi sebagai nilai. Selain `rgb()`, `hsl()`, dan `url()`, contoh yang paling sering digunakan adalah `calc()`, umumnya fungsi ini digunakan untuk memberikan nilai ukuran.

Selengkapnya tentang CSS Values and Units dapat dilihat [di sini](#).

6. Text Styling

Pada CSS terdapat beberapa properti yang dapat digunakan untuk menerapkan *styling rule* pada teks, beberapa contoh yang paling banyak digunakan adalah :

- **Font Family**, untuk menentukan jenis font pada teks, menggunakan properti `font-family`.
- **Font Size**, untuk menentukan ukuran font pada teks, menggunakan properti `font-size`, menggunakan nilai *identifier* seperti **large**, **small**, **medium**, atau menggunakan nilai ukuran dengan satuan mutlak seperti **px** dan **pt**, atau dapat pula dengan menggunakan nilai ukuran dengan satuan relatif seperti **em** dan **rem**.
- **Font Weight**, untuk menentukan ketebalan font pada teks, menggunakan properti `font-weight`, dengan pilihan nilai **100**, **200**, **300**, ..., **900**, atau nilai **bold**, **bolder**, **normal**, atau **lighter**.
- **Font Style**, untuk memberikan *style* seperti **bold**, **italic**, atau **oblique** pada teks, menggunakan properti `font-style`.
- **Font Color**, untuk menentukan warna font pada teks, menggunakan properti `color`, dengan nilai warna seperti yang sudah dijelaskan sebelumnya pada CSS Values and Units.
- **Text Align**, untuk menentukan perataan teks seperti **center**, **left**, dan **right**, menggunakan properti `text-align`.
- **Text Transform**, untuk menentukan penggunaan huruf kapital pada teks seperti **uppercase** untuk mengubah seluruh teks menjadi huruf kapital, atau **capitalize** untuk mengubah huruf pertama dalam setiap kata dalam teks menjadi huruf kapital, menggunakan properti `text-transform`.
- **Text Decoration**, untuk mengatur dekorasi berupa garis pada teks, menggunakan properti `text-decoration`.

Selengkapnya tentang Text Styling dapat dilihat [di sini](#).

7. Layouting

Layouting pada CSS bertujuan untuk mengatur bagaimana hubungan antara kotak (elemen) dengan *viewport* dan antar kotak tersebut satu sama lain, dalam hal penyusunan. Dalam layouting pada CSS, berikut adalah konsep-konsep terpenting yang perlu dipahami :

7.1. Normal Flow

Normal flow merupakan cara default browser dalam menata elemen-elemen HTML ketika kita tidak mendefinisikan aturan lain apapun.

7.2. Flexbox

Flexbox (singkatan dari *Flexible Box Layout*), membantu kita mengontrol tata letak elemen HTML kita dalam satu arah baik itu sebagai *row* (secara horizontal), maupun sebagai *column* (secara vertikal). Kita dapat menggunakan metode tata letak ini dengan menggunakan properti **display: flex**. Anak-elemen-langsung dari elemen yang diberikan properti ini secara otomatis akan menjadi **flex item(s)**.

Beberapa properti penting yang banyak digunakan dalam tata letak flexbox antara lain :

- **flex-direction**, menentukan arah aliran tata letak elemen di dalam flexbox.
- **align-items**, menentukan perataan (*alignment*) tata letak *flex items* dalam sumbu tegak lurus arah *flex-direction*.
- **justify-content**, menentukan perataan (*alignment*) tata letak *flex items* dalam sumbu sejajar arah *flex-direction*.
- **gap**, mengatur jarak antar *flex items* dalam flexbox.
- **flex-wrap**, mengatur bagaimana *flex items* dibungkus di dalam flexbox.

7.3. Grids

Grid membantu kita mengontrol tata letak elemen-elemen HTML dalam dua arah sebagai *row* (secara horizontal) dan sebagai *column* (secara vertikal). Kita dapat menggunakan metode tata letak ini dengan menggunakan properti **display: grid**.

Beberapa properti penting yang banyak digunakan dalam tata letak grid antara lain :

- **grid-auto-flow**, menentukan arah penempatan *grid items* secara otomatis dalam *grid container*.
- **grid-template-columns**, menentukan jumlah dan ukuran “*track*” kolom dalam *grid container*.
- **grid-template-rows**, menentukan jumlah dan ukuran “*track*” baris dalam *grid container*.
- **gap**, mengatur jarak antar *track* dalam *grid container*.
- **grid-column-start**, mengatur posisi *grid item* tertentu pada kolom dalam *grid container*.
- **grid-row-start**, mengatur posisi *grid item* tertentu pada baris dalam *grid container*.

7.4. Floats

Elemen yang diberikan properti *float*, akan keluar dari aliran (*flow*) normal dokumen, dan akan “menempel” pada sisi kiri/kanan (tergantung pada apa

yang digunakan) dari elemen induknya, kemudian elemen lain yang muncul setelahnya, akan membungkus elemen tersebut di sekelilingnya untuk mengisi ruang yang tersedia.

7.5. Positioning

Pada CSS terdapat beberapa jenis positioning, antara lain :

- **Static Positioning**

Pemosisian *static* merupakan nilai default yang didapatkan oleh seluruh elemen bila kita tidak mendefinisikan pemosisian lain padanya.

- **Relative Positioning**

Pemosisian *relative* merupakan nilai yang dengan menggunakannya kita menjadi dapat membuat pemosisian sebuah elemen menjadi lebih mudah untuk dikustomisasi, seperti menggunakan properti **top**, **right**, **bottom**, dan **left**, namun tetap berada dalam *normal flow*. Sehingga nilai **top**, **right**, **bottom**, dan **left**, yang diberikan akan relatif terhadap posisi awalnya.

- **Absolute Positioning**

Pemosisian *absolute* merupakan nilai yang tidak jauh berbeda dengan nilai *relative* sebelumnya, dimana perbedaannya hanya terletak pada elemen tersebut tidak lagi berada dalam *normal flow*, melainkan terisolasi dalam *flow*-nya sendiri. Pada pemosisian ini, nilai properti **top**, **right**, **bottom**, dan **left**, yang diberikan akan relatif pada elemen induknya.

- **Fixed Positioning**

Pemosisian *fixed* juga tidak jauh berbeda dengan pemosisian *absolute*, perbedaannya hanyalah, jika pada pemosisian *absolute* nilai properti **top**, **right**, **bottom**, dan **left**, yang diberikan akan relatif terhadap elemen induk-nya, pada pemosisian *fixed* nilai properti **top**, **right**, **bottom**, dan **left**, yang diberikan relatif terhadap elemen porsi layar yang tersedia, sehingga tidak akan bergerak meskipun kita melakukan *scroll* pada halaman.

- **Sticky Positioning**

Pemosisian *sticky* dapat dikatakan sebagai perpaduan antara pemosisian *relative* dengan pemosisian *fixed*, dimana dia akan menempati posisi relatif terhadap dirinya sendiri, hingga dia bergerak (diakibatkan melakukan *scroll*) sampai pada posisi tetap tertentu.

Selengkapnya tentang CSS Layouting dapat dilihat [di sini](#).

Latihan Percobaan




Hai, saya Elsa Asle

Seorang Viking menakutkan

TENTANG SAYA PENGALAMAN KERJA HUBUNGI SAYA

tentang saya



orem ipsum dolor sit amet consectetur
adipiscing elit. Ratione recusandae ducimus
placeat! Fugiat similique minus labore
doloremque esse, non impedit dolorum vel natus
quidem dolor! Earum, amet atque eveniet nihil
reiciendis nesciunt fugiat voluptate illum animi nulla vel quam
quo nobis optio quaerat alias accusantium asperiores
laudantium sed reprehenderit sunt?



Lorem ipsum ipsum lorem

pengalaman kerja

#	POSISI	PERUSAHAAN	WAKTU	
			MULAI	SELESAI
1	Lorem Ipsum Dolor	Dicta Modi Maiores	1928	1945
2	Id Provident Excepturi	Veniam Dolores Adipisci	1945	1973
3	Repellendus Debitis Iste	Adipiscing Elit Accusamus	1973	1982
4	Corporis Quidem Iure	Sit Amet Consectetur	1982	2022
5	Tenetur Nisi Architecto	Lorem Ipsum Dolor	2022	2200

hubungi saya

Nama

Sudah nonton video saya?

☐ Sudah ☐ Belum

Apa coba?

☐ Ini ☐ Itu ☐ Anu

KIRIM

Tugas

Modifikasi hasil Tugas Pertemuan 1 agar sesuai dengan tampilan yang kalian inginkan.

Parameter Penilaian	Bobot Penilaian
Kelengkapan	50%
Kesesuaian penggunaan CSS selector dan fungsinya	20%
Keindahan dan estetika 😊	20%
Penggunaan “ <i>best practices</i> ” menurut standar W3C (dapat melalui https://www.freeformatter.com/html-validator.html)	10%

Daftar Pilihan Referensi

- [mdn web docs](#)
- [W3Schools](#)
- [Web Programming Unpas \(YouTube\)](#)