**Pengenalan ke Styling**

Pada sub-modul pengenalan sudah disinggung seperti apa peran CSS pada website. Website nampak begitu membosankan dan mengerikan tanpa adanya CSS. *Cascading Style Sheet* atau biasa disingkat CSS merupakan W3C standar yang digunakan untuk mengatur visualisasi  berkas yang ditulis pada HTML. Pada sub-modul ini kita akan belajar penggunaan dasar dari CSS. Tepatnya mulai dari pembuatan berkas CSS, pengenalan struktur sintaksnya, hingga menerapkan dasar *styling* seperti memberikan warna pada sebuah teks.

CSS *syntax* berbeda dengan HTML, begitu pula dengan JavaScript. CSS bukanlah sebuah bahasa pemrograman karena di dalamnya tidak terdapat logika, tidak dapat sintaks pengondisian, tidak adanya proses iterasi, dsb. CSS hanya sebuah *declarative language* yang digunakan untuk mendeklarasikan suatu nilai yang nantinya digunakan untuk mengatur seperti apa sebuah elemen HTML ditampilkan pada browser.

## Keuntungan Menggunakan CSS

Dengan menerapkan CSS, tampilan website kita akan lebih menarik. Berikut beberapa keuntungan yang didapatkan ketika kita menerapkan CSS.

* Dapat mengontrol dan menerapkan layout secara presisi. Dengan menggunakan CSS kita bisa membuat sebuah website seperti dokumen cetak dengan desain yang menarik dan presisi.
* Menghindari pekerjaan yang berulang-ulang dalam menerapkan styling. Kita dapat menetapkan styling pada beberapa berkas HTML hanya dengan menggunakan satu berkas CSS.
* Didukung banyak browser. Seluruh browser saat ini minimal sudah mendukung CSS versi 2. Untuk browser yang populer seperti Chrome dan Firefox sudah mendukung CSS versi 3.

Sudah jelas, bukan? Tiada alasan kita tidak menerapkan CSS. Terkecuali dukungan pada browser yang terkadang berbeda setiap vendornya (contohnya [vendor prefixes](https://developer.mozilla.org/en-US/docs/Glossary/Vendor_Prefix)), tapi hal tersebut tidak menjadi masalah ketika kita tahu di mana letak perbedaan dan cara penerapannya.

## Bagaimana Style Sheet Bekerja?

Sebuah style sheet bekerja melalui tahapan berikut:

1. Dimulai dari sebuah dokumen yang telah ditandai dengan tag elemen HTML.
2. Menuliskan aturan styling untuk menentukan bagaimana elemen HTML akan ditampilkan.
3. Melampirkan aturan styling yang sudah dibuat pada dokumen HTML. Ketika browser memuat dokumen, tampilan elemen yang ditampilkan akan menyesuaikan dengan aturan styling yang sudah ditetapkan.

Mari, kita bahas satu persatu tahapannya.

### Memberikan Markup pada Dokumen

Pada sub-modul sebelumnya kita sudah mengenal banyak bagaimana cara untuk me-markup atau memberikan tag sebuah konten dalam HTML. Mungkin sebagian dari kita bingung, mengapa kita perlu mengelompokkan sebuah konten tapi tiada perubahan signifikan pada tampilan browser? Contohnya seperti elemen <div> yang tidak akan mengubah apapun sebelum kita menerapkan sebuah styling pada elemennya. Sebenarnya terdapat hubungan antara struktur dokumen dengan sebuah style sheet. Kita akan memahami ini seiring kita belajar hingga pembahasan lebih lanjut.

### Menuliskan Aturan Styling

Sebuah style sheet dibuat terdiri dari satu atau lebih aturan styling (biasa disebut dengan rules atau rule-sets) yang mendeskripsikan bagaimana sebuah elemen atau sebuah kelompok elemen ditampilkan dalam jendela browser.

Langkah awal belajar CSS adalah dengan memahami sebuah bagian rule. Berikut ini dua contoh rules yang dituliskan dalam sebuah CSS. Rule yang pertama menetapkan sebuah warna hijau pada elemen <h1> dan rule yang kedua menetapkan ukuran font dan tipe font pada sebuah elemen paragraf.

1. h1 { color: green; }
3. p {
4. font-size: small;
5. font-family: sans-serif;
6. }

Dalam penggunaan CSS, terdapat dua bagian dalam sebuah rule. Yang pertama adalah identitas elemen atau elemen yang akan menerapkan rule (singkatnya kita akan sebut selector) dan yang kedua adalah deklarasi atau instruksi yang akan diterapkan pada sebuah selector.

#### Selector

Pada contoh di atas, h1 dan p digunakan sebagai selector. Selector ini dipanggil melalui tipe elemennya, dan ini merupakan teknik dasar dari pemanggilan selector. Properti dan nilainya yang terdapat pada declaration/declaration block akan diterapkan pada seluruh elemen <h1> dan <p> yang ada pada dokumen HTML. Pada sub-modul selanjutnya kita akan mengetahui berbagai cara lainnya untuk menetapkan selector dengan lebih canggih lagi.

#### Declarations

Bagian deklarasi terdiri dari pasangan properti dengan nilainya. Kita bisa menetapkan lebih dari satu deklarasi pada satu rule, contohnya seperti pada selector p di atas. Kita menetapkan lebih dari satu deklarasi pada declaration block. Setiap deklarasinya harus diakhiri dengan semicolon (;) sebagai tanda diakhirinya sebuah deklarasi.

Karena CSS tidak memperhatikan spasi (sama seperti bahasa pemrograman pada umumnya), sebaiknya penulisan deklarasi selalu diawali dengan baris baru supaya mudah dibaca dan dipahami.

1. p {
2. font-size: small;
3. font-family: sans-serif;
4. /\* deklarasikan nilai properti lainnya pada baris baru \*/
5. }

Tapi ingat, walaupun CSS tidak memperhatikan spasi, untuk satuan nilai seperti px, em, rem, dan lainnya harus dituliskan tanpa spasi pada nilainya. Contohnya:

1. h1 {
2. margin: 2em;
3. }

Jika kita menambahkan spasi di antara satuan dan nilai, maka deklarasi tersebut tidak akan berfungsi.

1. h1 {
2. margin: 2 em; /\* Deklarasi tidak akan berfungsi karena terdapat spasi antara nilai dan satuan \*/
3. }

#### CSS Comments

CSS comments digunakan untuk menjelaskan kode dan dapat membantu ketika Anda mengedit kode sumber di lain waktu. Apa yang tertulis akan diabaikan oleh browser dan tidak ditampilkan di browser, sehingga dapat menjadi opsi baik untuk memberi catatan atau informasi dokumentasi pada kode.

Seperti contoh yang beberapa kali telah Anda lihat diatas, CSS comments ditempatkan pada elemen <style> dengan cara penulisan dimulai dengan **/\*** dan diakhiri **\*/** seperti contoh berikut:

1. p {
2. font-size: small;
3. font-family: sans-serif;
4. /\* memilih sans-serif sesuai hasil diskusi klien pada 2 Jan 2021\*/
5. }

## Melampirkan Styling pada Dokumen HTML.

Setelah kita menuliskan rules, maka tahapan selanjutnya adalah melampirkan atau menerapkan aturan tersebut pada berkas HTML. Sebenarnya, terdapat tiga cara untuk menerapkan styling pada elemen HTML.

### External Style Sheet

External Style Sheet merupakan berkas terpisah yang di dalamnya hanya terdapat sebuah rules. Berkas ini harus berekstensi **.css**, dan berkas ini nantinya dihubungkan pada dokumen HTML. Cara ini merupakan yang paling powerful dalam menerapkan styling. Karena dengan cara ini, satu berkas styling (.css) dapat digunakan oleh banyak berkas HTML.

Untuk menyambungkan berkas .css dengan dokumen HTML, kita dapat menggunakan elemen <link> pada <head> berkas HTML. Contohnya:

1. <head>
2. <title>Document Title</title>
3. <link rel="stylesheet" href="style.css">
4. </head>

Pada elemen <link> tersebut, kita tetapkan berkas CSS yang digunakan dengan menggunakan atribut href dan beri nilai “stylesheet” pada atribut rel sebagai relationship (hubungan) antara berkas style.css dengan dokumen HTML.

Pada contoh di atas kita tahu bahwa berkas css yang digunakan merupakan berkas lokal (berkas yang berada pada komputer/server kita sendiri). Nilai atribut href juga dapat berupa berkas .css yang tersedia melalui sebuah URL.

Contohnya, banyak pengembang menggunakan bootstraps.min.css untuk membantu penyusunan layout website-nya. Kita bisa menggunakannya pada berkas HTML dengan langsung menuliskan URL untuk berkas tersebut.

1. <head>
2. <title>Document Title</title>
3. <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
4. </head>

min.css merupakan penamaan format berkas .css yang sudah di-minify atau sudah diminimalkan dengan menghilangkan white space yang tidak digunakan

### Embedded Style Sheet

Embedded Style Sheet merupakan kumpulan rules yang dituliskan dalam berkas HTML dengan menggunakan elemen <style>. Dengan begitu rules yang dituliskan hanya dapat dicakup oleh satu berkas HTML. Penulisan rules harus dituliskan dalam elemen <style> dan ditempatkan di dalam <head> dari berkas HTML.

1. <head>
2. <title>Document Title</title>
3. <style>
4. /\*
5. \* Rules styling dituliskan di sini
6. \*/
7. </style>
8. </head>

### Inline Style

Inline Style merupakan styling yang diterapkan pada elemen HTML dengan menggunakan atribut style. Contohnya seperti berikut:

1. <h1 style="color: green">Kota Bandung</h1>

Untuk menambahkan styling properties lainnya (multiple properties), kita tuliskan dengan menggunakan semicolon (;) sebagai pemisah antar styling properties-nya.

1. <h1 style="color: green; margin-top: 2em">Kota Bandung</h1>

Inline styles hanya diterapkan pada elemen di mana atribut style diterapkan. Teknik ini seharusnya dihindari terkecuali benar-benar diperlukan untuk menggantikan sebuah styling yang ditetapkan pada Embedded Style Sheet atau External Style Sheet.

### Commenting in Style

Pada contoh kode di atas, kita sudah melihat sebuah teks pada rule yang dimulai dari /\* dan diakhiri dengan \*/. Teks tersebut merupakan commenting line di dalam sebuah CSS. Sama seperti commenting pada HTML dan bahasa pemrograman lainnya, Teks yang dijadikan sebuah komentar tidak akan diproses oleh browser (sebagai compiler) atau lebih tepatnya “diabaikan”.

Berikut adalah contoh memberikan komentar pada CSS:

1. /\* ini merupakan komentar satu baris \*/
3. /\* CSS juga mendukung komentar lebih dari satu baris,
4. seluruh teks yang berada pada pembuka dan penutup sebuah komentar
5. akan diabaikan oleh browser sebagai compiler.
6. \*/

## Menerapkan Styling Dasar pada Halaman Profil

Sejauh ini kita sudah mengetahui bagaimana menuliskan sebuah rule dan menggunakannya pada berkas HTML. Sekarang kita coba terapkan beberapa styling dasar pada halaman profil yang sudah kita buat dari beberapa sub-modul sebelumnya.

Latihan kali ini akan mencakup tahapan seperti pembuatan berkas CSS, menuliskan rule pada berkas CSS, hingga menghubungkan berkas CSS dengan HTML.

### Membuat Berkas CSS

Silakan buka kembali root folder proyek yang telah kita buat dan buka folder assets.

Jika kita sudah selesai mengikuti latihan pada beberpa sub-modul sebelumnya, maka di dalam folder assets akan terdapat folder image. Sekarang, kita tambahkan folder baru dan beri nama styles.

Folder styles ini akan kita gunakan untuk menyimpan berkas CSS. Maka silakan buka folder tersebut dan buat berkas baru dan beri nama style.css.

## CSS Conception

Sebelum membahas lebih detail mengenai styling, terdapat beberapa konsepsi dalam CSS yang harus kita pahami.

### Inheritance

Styling HTML bersifat inheritance yang artinya dapat mewarisi properti style “tertentu” pada elemen yang ada di dalamnya. Contohnya pada rules yang kita tuliskan untuk elemen <body> akan diterapkan pada seluruh elemen yang ada di dalam elemennya (body). Contoh lainnya, pada rules yang diterapkan pada elemen <footer> dengan properti color yang bernilai white, akan diterapkan pada seluruh elemen yang ada di dalam <footer>. Hal ini menjadi alasan mengapa memahami struktur dokumen itu penting.

### Group Selector

Jika kita menerapkan rule yang sama pada beberapa selector yang berbeda, di CSS kita dapat menggabungkan selector tersebut sehingga dapat meminimalisir penulisan kode yang berulang.

Contohnya pada styling yang sudah kita buat sebelumnya, terdapat rule yang serupa antara h2 dan h3.

1. h2 {
2. color: #00a2c6
3. }
5. h3 {
6. color: #00a2c6
7. }

Jika terdapat kasus seperti ini, kita dapat menuliskan dua selector sekaligus dalam satu struktur rule. Gunakan tanda koma (,) untuk memisahkan tiap selector-nya. Silakan kita buka kembali berkas style.css dan gabungkan rule untuk elemen <h2> dan <h3> menjadi seperti ini:

1. h2, h3 {
2. color: #00a2c6
3. }

### Rule Order

Sesuai dengan namanya, cascading artinya “mengalir.” Demikian halnya dengan alur kerja CSS dalam membaca kode yang mengalir dari atas ke bawah. Karena itu kita harus memperhatikan urutan dalam penulisan rules, terutama saat terjadi sebuah konflik.

Konflik dapat terjadi karena kita dapat menerapkan beberapa styling pada satu dokumen HTML. Contohnya, apa yang seharusnya ditampilkan pada browser ketika eksternal css mengharuskan elemen <p> menampilkan warna merah, tetapi pada embedded css <p> harus menampilkan warna biru? Kembali pada alur kerja CSS yang membaca dari atas ke bawah, sehingga warna yang akan diterapkan adalah warna yang paling akhir didefinisikan.

Untuk lebih jelasnya, kita bisa lihat contoh berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Berkas</title>
5. <style>
6. p {
7. color: red;
8. }
10. p {
11. color: blue;
12. }
13. </style>
14. </head>
15. <body>
16. <p>Sesuai dengan namanya cascading yang artinya <q>mengalir</q>, alur kerja CSS dalam membaca kode pun seperti itu.
17. Mengalir dari atas ke bawah sehingga kita harus memperhatikan urutan dalam penulisan rules <i>styling</i></p>
18. </body>
19. </html>

Maka pada browser elemen <p> menerapkan warna biru pada tulisannya.

Tetapi kita bisa membuat sebuah property styling agar dianggap penting untuk diterapkan oleh browser dan tidak memperhatikan urutannya. Kita bisa menambahkan keyword !important diakhir nilai propertinya.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. p {
7. color: red !important;
8. }
10. p {
11. color: blue;
12. }
13. </style>
14. </head>
15. <body>
16. <p>Sesuai dengan namanya cascading yang artinya <q>mengalir</q>, alur kerja CSS dalam membaca kode pun seperti itu. Mengalir dari atas ke bawah sehingga kita harus memperhatikan urutan dalam penulisan rules <i>styling</i></p>
17. </body>
18. </html>

Sehingga warna yang diterapkan pada teks paragraf adalah merah.

Gunakan !important ketika memang benar-benar dibutuhkan saja. Sebaiknya kita pahami aturan urutan pada CSS dengan baik sehingga meminimalisir penggunaan tanda tersebut.

Berikut catatan yang sudah kita pelajari sejauh ini tentang styling:

* Rule : Sebuah aturan styling yang harus diterapkan pada elemen HTML. Dalam sebuah rule terdapat selector dan deklarasi properti styling.
* Selector : Sebuah bagian dari rule, yang mengidentifikasi target elemen untuk menetapkan sebuah rule.
* Declaration : Sebuah bagian dari rule, yang terdiri dari pasangan properti dan nilainya.
* External Style Sheet : Berkas terpisah yang di dalamnya hanya terdapat sebuah rules yang akan digunakan pada website.
* Embedded Style Sheet : Kumpulan rules yang dituliskan dalam berkas HTML dengan menggunakan elemen <style>.
* Inline Style : styling yang diterapkan pada elemen HTML dengan menggunakan atribut style.

## Styling lebih Dalam

Pada sub-modul sebelumnya kita sudah mengenal apa itu CSS. Lalu, bagaimana ia bekerja, cara penulisan sintaksnya, hingga mengetahui konsep yang ada seperti inheritance, grouping selector, dan rule order.

Pada sub-modul ini kita akan berkenalan lebih dalam lagi dengan CSS, kita akan mengenal berbagai macam tipe selector, formatting text, bermain dengan warna, mengenal apa itu box model, positioning, hingga penyusunan layout pada website menggunakan teknik floating.

Untuk menerapkan materi yang kita pelajari pada sub-modul ini juga terdapat beberapa latihan dalam menerapkan styling pada halaman profil, tentunya pada akhir sub-modul ini kita dapat membuat halaman profil tampak seperti ini:

**More about Selector**

Ada banyak jenis *selector* untuk menargetkan aturan ke elemen tertentu dalam dokumen HTML. Pada sub-modul sebelumnya, kita sudah mengetahui salah satu cara dasar dalam menggunakan *selector*. Sekarang, mari kita bahas lebih detail mengenai macam macam *selector* yang ada.

**Selector Basic**

Pada sub-modul sebelumnya kita sudah mengetahui apa itu selector, dan menggunakannya. Yang kita gunakan tersebut merupakan *selector basic*. Sebenarnya terdapat beberapa macam dari *selector basic*, yakni:

* Type Selector
* Class Selector
* ID Selector
* Attribute Selector
* Universal Selector

Mari kita bahas satu persatu tipe *selector basic* yang ada.

**Type Selector**

*Type Selector* menggunakan nama elemen sebagai target untuk diterapkannya *rule*. Dengan kata lain, ketika menggunakan *selector* ini tentu rule akan diterapkan pada seluruh elemen target yang ada pada dokumen HTML. Contohnya sebagai berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. /\* Semua elemen span \*/
7. span {
8. color: red;
9. }
10. </style>
11. </head>
12. <body>
13. <span>Ini merupakan sebuah teks yang berada pada elemen span. Seharusnya elemen ini ditampilkan dengan warna teks merah.</span>
14. <p>Ini merupakan sebuah teks yang berada pada elemen paragraf, teks ini tidak seharusnya tidak akan terpengaruh oleh rule.</p>
15. <span>Ini merupakan sebuah teks yang berada pada elemen span lainnya. Seharusnya elemen ini ditampilkan dengan warna teks merah juga.</span>
16. </body>
17. </html>

Jika berkas tersebut dibuka pada browser, maka teks yang berada pada setiap elemen <span> akan berwarna merah.

**Class Selector**

Class selector menetapkan target elemen berdasarkan nilai dari atribut class yang diterapkan pada elemennya. Untuk penulisan *selector*, awali dengan tanda titik (.) kemudian lanjutkan dengan nama *class*-nya. Contoh:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. .red {
7. color: red;
8. }
10. .skyblue-bg {
11. background-color: skyblue;
12. }
14. .fancy {
15. font-weight: bold;
16. text-shadow: 4px 4px 3px #77f;
17. }
18. </style>
19. </head>
20. <body>
21. <p class="red">Paragraf dengan teks berwarna merah</p>
22. <p class="skyblue-bg">Paragraf dengan background berwarna biru langit</p>
23. <p class="fancy">Paragraf dengan gaya fancy</p>
24. <p>Paragraf yang menampilkan teks dengan warna standar tanpa menerapkan styling</p>
25. </body>
26. </html>

Jika berkas tersebut dibuka pada browser, maka akan nampak seperti ini:

Class bersifat *shareable*, sehingga dapat diterapkan pada banyak elemen dengan tipe elemen yang berbeda-beda. Misalkan sebuah class red dapat diterapkan pada elemen paragraf dan juga *heading* untuk menampilkan teks berwarna merah.

1. <h1 class="red">Heading dengan teks berwarna merah</h1>
2. <p class="red">Paragraf dengan teks berwarna merah</p>

Tidak hanya itu, sebuah elemen juga mungkin memiliki banyak nilai *class*, sehingga kita dapat menerapkan lebih dari satu rule atau gabungan rule pada elemen target. Untuk menggunakannya, pada atribut class kita cukup tuliskan nama kelasnya dan pisahkan tiap nilai kelasnya dengan spasi.

1. <h1 class="red skyblue-bg">Heading dengan teks berwarna merah dan background biru langit</h1>
2. <p class="red fancy">Paragraf dengan teks berwarna merah dan bergaya fancy</p>

Kita juga bisa menargetkan elemen secara spesifik yang memiliki sebuah *class*. Bagaimana jika kita ingin menerapkan gaya yang berbeda ketika menggunakan kelas bergaya keren alias *fancy* pada sebuah paragraf? untuk melakukannya, pada selector kita tuliskan nama elemen target diikuti dengan titik (.) kemudian nama kelasnya. Perhatikan contoh berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. .fancy {
7. font-weight: bold;
8. text-shadow: 4px 4px 3px #77f;
9. }
11. p.fancy {
12. text-shadow: 4px 4px 3px #00c8eb;
13. }
14. </style>
15. </head>
16. <body>
17. <h1 class="fancy">Heading dengan teks bergaya fancy</h1>
18. <p class="fancy">Paragraf dengan teks bergaya fancy</p>
19. </body>
20. </html>

Jika berkas tersebut dibuka pada browser, maka gaya fancy yang diterapkan pada elemen heading dan elemen paragraf akan berbeda.

### ID Selector

ID selector menetapkan target elemen berdasarkan nilai dari atribut id yang diterapkan pada elemennya. Sama seperti class, atribut id dapat diterapkan pada seluruh elemen HTML, dan kebanyakan atribut ini digunakan untuk memberikan sebuah arti pada generic element seperti <div> dan <span>. Namun atribut id ini tidak bersifat shareable, yang artinya nilai id ini harus unik dan digunakan pada satu elemen saja.

Untuk menetapkan selector dengan menggunakan id, kita gunakan tanda octothorpe (#) atau lebih familiar disebut dengan hash. Berikut contohnya:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. #special {
7. background-color: skyblue;
8. }
9. </style>
10. </head>
11. <body>
12. <div id="special">
13. <p>Ini merupakan konten di dalam sebuah div yang diberi id special.</p>
14. </div>
15. <div>
16. <p>Ini merupakan konten di dalam sebuah div biasa.</p>
17. </div>
18. </body>
19. </html>

Jika kode di atas dibuka pada browser, maka tampilan akan tampak seperti ini:

Yang harus kita perhatikan kembali adalah id ini bersifat unik. Jika kita ingin menerapkan sebuah rule pada banyak elemen, sebaiknya gunakan atribut class, bukan dengan id.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. #special {
7. background-color: skyblue;
8. }
9. </style>
10. </head>
11. <body>
12. <div id="special">
13. <p>Ini merupakan konten di dalam sebuah div yang diberi id special.</p>
14. </div>
15. <div id="special"> <!-- ini merupakan contoh yang salah dalam penerapan id -->
16. <p>Ini merupakan konten di dalam sebuah div biasa.</p>
17. </div>
18. </body>
19. </html>

### Attribute Selector

Attribute selector merupakan cara menetapkan target elemen berdasarkan sebuah atribut yang digunakan atau bahkan bisa lebih spesifik dengan nilainya. Contohnya sebagai berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. /\* <a> element yang menerapkan href attribut \*/
7. a[href] {
8. color: blue;
9. }
11. /\* <a> element yang menerapkan nilai pada href dengan awalan "#" \*/
12. a[href^="#"] {
13. background-color: gold;
14. }
16. /\* <a> element yang menerapkan nilai pada href yang mengandung teks "example" \*/
17. a[href\*="example"] {
18. background-color: silver;
19. }
21. /\* <a> element yang menerapkan nilai pada href yang mengandung teks "insensitive" tidak mementingkan huruf kapital\*/
22. a[href\*="insensitive" i] {
23. color: cyan;
24. }
26. /\* <a> element yang menerapkan nilai pada href dengan akhiran ".org" \*/
27. a[href$=".org"] {
28. color: red;
29. }
30. </style>
31. </head>
32. <body>
33. <ul>
34. <ul>
35. <li><a href="#internal">Internal link</a></li>
36. <li><a href="http://example.com">Example link</a></li>
37. <li><a href="#InSensitive">Insensitive internal link</a></li>
38. <li><a href="http://example.org">Example org link</a></li>
39. </ul>
40. </ul>
41. </body>
42. </html>

Jika kode di atas dibuka pada browser, maka tampilan akan tampak seperti ini:

Dari kode di atas terlihat banyak sekali kondisi yang dapat diterapkan pada atribut selector. Supaya lebih mudah untuk dipahami, mari kita rangkum dalam sebuah tabel berikut.

|  |  |
| --- | --- |
| **Syntax** | **Description** |
| [attr] | Menargetkan elemen yang menerapkan atribut attr. |
| [attr=value] | Menargetkan elemen yang menerapkan atribut attr dengan nilai value. |
| [attr~=value] | Menargetkan elemen yang menerapkan atribut attr dengan salah satu nilainya adalah value. |
| [attr^=value] | Menargetkan elemen yang menerapkan atribut attr yang nilainya diawali dengan nilai value. |
| [attr$=value] | Menargetkan elemen yang menerapkan atribut attr yang nilainya diakhiri dengan value. |
| [attr\*=value] | Menargetkan elemen yang menerapkan atribut attr yang nilainya mengadung value. |

### Universal Selector

Universal selector digunakan untuk diterapkan pada seluruh elemen. Tetapi selector ini juga bisa secara spesifik menargetkan sebuah elemen dengan menggabungkan bersama selector yang lainnya. Berikut ini contohnya:

1. <!DOCTYPE html>
2. <html lang="id">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. /\* Menargetkan seluruh tipe elemen \*/
7. \* {
8. color: green;
9. }
11. /\* Menargetkan seluruh tipe elemen yang mengandung nilai "en" pada atribut lang \*/
12. \*[lang^=en] {
13. font-style: italic;
14. }
16. /\* Menargetkan seluruh tipe elemen yang memiliki nilai "warning" pada atribut class \*/
17. \*.warning {
18. color: red;
19. }
21. /\* Menargetkan seluruh tipe elemen yang memiliki nilai "content" pada atribut id \*/
22. \*#content {
23. border: 1px solid blue;
24. padding: 20px;
25. }
26. </style>
27. </head>
28. <body>
29. <p>
30. Ini merupakan paragraf biasa atau tidak menerapkan atribut apapun. Maka teks pada paragraf ini akan berwarna
31. </p>
32. <p lang="en-us">
33. This is an English paragraph contains en-us value of lang attribute, so this text will be green and
34. italic.
35. </p>
37. <h1>Ini merupakan sebuah header biasa</h1>
38. <h1 lang="en-us">This is an English Header</h1>
40. <p class="warning">
41. Perhatikan paragraf ini! Paragraf ini merupakan paragraf yang memiliki kelas bernilai warning,
42. sehingga teks dari paragraf ini akan berwarna merah
43. </p>
45. <div id="content">
46. <p>
47. Ini merupakan sebuah teks di dalam sebuah div yang memiliki id bernilai "content", seharusnya paragraf ini
48. dibungkus dalam border yang memiliki padding 20px
49. </p>
50. </div>
51. </body>
52. </html>

Hasilnya:

## Pseudo Selector

Selain beberapa selector yang sudah kita pelajari, CSS masih memiliki dua selector lagi yang dapat kita manfaatkan membantu menyeleksi elemen dalam menerapkan sebuah rule, yakni Pseudo-class dan Pseudo-element.

Sebelum menjelaskan lebih detail tentang kedua selector tersebut, perlu kita ketahui pseudo selector berbeda dari selector yang sudah dibahas sebelumnya, selector ini menargetkan elemen pada bagian yang “tidak terlihat” seperti sifat pada elemen, sehingga untuk menetapkannya, kita tidak bisa menggunakan selector biasa. Salah satu contoh yang paling sering kita terapkan adalah :hover, Pseudo Selector tersebut kita gunakan untuk menetapkan rule ketika cursor diarahkan ke target elemen.

1. a:hover {
2. color: blue;
3. }

Contoh kode di atas merupakan salah satu pseudo-class selector dimana elemen anchor akan menerapkan warna biru ketika kursor diarahkan pada elemen tersebut.

Mari kita bahas lebih detail mengenai pseudo-class selector dan pseudo-element selector.

### Pseudo-class Selector

Pseudo-class merupakan sebuah class “semu” yang sebenarnya ada pada tiap elemen HTML. Pada contoh sebelumnya kita sudah mengetahui salah satu pseudo-class selector, dengan menggunakan selector ini kita dapat memilih elemen berdasarkan class yang tidak nampak pada dokumen. Kita bisa menetapkan rule hanya ketika sebuah tautan telah dikunjungi (:visited) atau ketika sebuah elemen diarahkan dengan kursor (:hover).

Untuk menggunakan pseudo-class kita gunakan tanda titik dua (:) pada basic selector kemudian diikuti dengan pseudo-class nya. Berikut contoh penerapan beberapa pseudo-class pada sebuah elemen anchor:

1. /\* rule akan diterapkan pada sebuah tautan yang belum pernah dikunjungi \*/
2. a:link {
3. color: red;
4. }
6. /\* rule akan diterapkan pada sebuah tautan yang sudah pernah dikunjungi \*/
7. a:visited {
8. color: green;
9. }
11. /\* rule akan diterapkan pada sebuah tautan ketika diarahkan dengan kursor \*/
12. a:hover {
13. color: pink;
14. }
16. /\* rule akan diterapkan pada sebuah tautan ketika ditekan \*/
17. a:active {
18. color:orange;
19. }

Maka hasilnya akan seperti ini:

Tidak hanya yang dicontohkan di atas, ada banyak sekali class semu yang menjadi standar pseudo-classes dalam CSS. Kita pun bisa melihat indeks standar yang ada pada pseudo-class pada tautan resmi Mozilla berikut: <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>

### Pseudo-elemen Selector

Sama seperti pseudo-class, pseudo-elemen merupakan sebuah elemen “semu” yang sebenarnya ada tetapi tidak tampak secara tertulis pada berkas HTML. Selector ini biasa digunakan ketika kita ingin menambahkan sebuah konten tepat sebelum dan setelah sebuah elemen paragraf. Alhasil kita tidak perlu menuliskan struktur elemen tersebut pada berkas HTML. Cukup manfaatkan pseudo-elemen ::before dan ::after kemudian tuliskan konten tersebut cukup pada CSS. Contohnya kita ingin menambahkan tanda kutip sebelum dan sesudah elemen blockquote. Seperti ini cara menuliskannya:

1. blockquote::before,
2. blockquote::after {
3. content: '"';
4. font-size: 24px;
5. font-style: italic;
6. font-weight: bold;
7. }

Dengan menuliskan rule tersebut, elemen <blockquote> memiliki tanda kutip pada awal dan akhir elemennya.

1. <blockquote>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aut eius error explicabo ipsum molestiae necessitatibus nesciunt possimus reprehenderit sed voluptates. Aliquam aspernatur autem est nobis officia praesentium quas recusandae rem.</blockquote>

Tampilan yang dihasilkan:

Untuk menggunakan pseudo-elemen kita gunakan dua buah tanda titik dua (::) kemudian diikuti dengan pseudo-elemen nya. Sebenarnya kita bisa menggunakan satu buah tanda tanda titik dua (:) namun kita perlu membedakannya dengan pseudo-class. Karena itulah baiknya ketika menggunakan pseudo-elemen, gunakanlah dua buah titik dua (::).

Pseudo-element tidak hanya ::before dan ::after. Dengan pseudo-elemen kita juga dapat menentukan rule pada awal karakter konten elemen.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Judul Dokumen</title>
5. <style>
6. /\* Rule styling akan diterapkan pada karakter pertama disebuah paragraf \*/
7. p::first-letter {
8. font-size: 32px;
9. font-weight: bold;
10. color: saddlebrown;
11. }
12. </style>
13. </head>
14. <body>
15. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Architecto deserunt, dicta laudantium quae quam sequi
16. soluta vitae! A, architecto beatae, consequuntur et eveniet fuga laudantium molestias praesentium temporibus, unde
17. velit.</p>
18. </body>
19. </html>

Tampilan yang dihasilkan:

Untuk mengetahui lebih lengkap mengenai apa saja pseudo-element yang dapat kita manfaatkan, kita bisa melihat dokumentasi resmi yang diberikan Mozilla pada tautan berikut:<https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements>

# Formatting Text (Text and Font Styling)

Setelah kita tahu bagaimana menerapkan CSS pada berkas HTML, mengetahui struktur penulisan CSS, dan menggunakan selector sesuai dengan kebutuhan, apa berikutnya? Kita akan fokus mempelajari secara detail bagaimana menerapkan styling pada sebuah teks.

Perlu kita ketahui, CSS memisahkan styling untuk font dan teks. Kita harus tahu kapan seharusnya menetapkan styling pada font atau menetapkannya pada teks. Mengapa kedua hal tersebut dipisah? Padahal teks dan font merupakan satu kesatuan, di mana setiap teks pasti menggunakan font untuk menampilkannya, bukan? Sebabnya, CSS sendiri mempunyai properti untuk kedua hal tersebut secara masing-masing

Jika kita ingin menetapkan styling pada tampilan teks itu sendiri, gunakanlah properti font. Contohnya properti font dapat mengatur tipe font, ukuran, ketebalan, dan lainnya. Sedangkan properti teks digunakan untuk mendukung hal lainnya dalam menampilkan teks seperti, menetapkan text alignment, dekorasi, spacing, dan masih banyak lagi. Karena dua hal ini dipisah, mari kita bahas satu persatu.

## Font Styling

Ketika kita membuat sebuah dokumen teks, termasuk dokumen cetak, langkah awal kita biasanya adalah menentukan jenis font yang akan digunakan. Pada pengembangan website pun demikian. Dalam CSS, font ditentukan dengan menggunakan beberapa paket properti font. Kita bisa atur tipe font, ukuran, ketebalan, dan gaya. Berikut ini merupakan properti font yang akan kita pelajari antara lain:

* font-family : Menetapkan jenis font yang akan diterapkan pada target.
* font-size : Menentukan ukuran pada teks.
* font-weight : Menentukan ketebalan pada teks.
* font-style : Menetapkan styling yang diterapkan pada teks.
* font-variant : Menentukan teks untuk menggunakan gaya small caps (huruf kapital kecil).
* font : Shorthand dari properti font yang ada.

Mari kita bahas properti tersebut satu persatu secara mendetail.

### font-family

Pada sub-modul pengenalan CSS kita sudah mencoba menggunakan font properties ini untuk mengubah standar font yang ditampilkan pada browser dengan menggunakan font-family pada elemen <body>.

1. body {
2. font-family: sans-serif;
3. }

Sehingga menghasilkan tampilan:

Pada rule tersebut kita mengubah standar font yang digunakan browser dengan font ‘sans-serif’. Sebenarnya untuk nilai dari properti ini dapat lebih dari satu (dikenal sebagai font stack). Tujuannya adalah sebagai fallback jika terjadi kegagalan dalam menggunakan font yang kita gunakan.

Untuk menuliskan lebih dari satu nilai font, berikut aturan yang harus kita perhatikan:

* Seluruh nilai font yang bukan merupakan generic font families harus dituliskan secara kapital. Contohnya “Arial” bukan dituliskan “arial”.
* Gunakan tanda koma (,) untuk memisahkan antar nilai font yang digunakan.
* Selalu tanda kutip (“) untuk membungkus nilai font yang memiliki spasi pada namanya (Contohnya “Open Sans”).

Mungkin kita bertanya-tanya mengapa perlu memberikan lebih dari satu nilai pada font-family? Hal tersebut penting karena tidak semua browser mendukung semua jenis font. Memberikan lebih dari satu nilai font dapat menawarkan alternatif tampilan font pada browser. Terutama jika font utama yang diterapkan tidak didukung oleh browser yang digunakan.

Bagaimana urutan prioritasnya? Mulai dari jenis font yang pertama dituliskan. Jika font pertama didukung oleh browser, maka browser akan menggunakannya. Namun jika tidak, maka browser akan memeriksa nilai font yang kedua dan menggunakannya (jika didukung), demikian dan seterusnya.

Pastikan untuk menggunakan [generic font families](https://developer.mozilla.org/en-US/docs/Web/CSS/font-family#values) pada akhir nilai properti font-family, karena nilai ini dipastikan didukung oleh seluruh browser saat ini. Lantas apa saja nilai dari generic font families ini? Berikut nilai-nilai generic font families yang dapat kita gunakan untuk fallback mechanism:

* serif : jenis font yang memiliki runcing pada garis akhir karakternya. Times New Roman merupakan salah satu jenis serif font.
* sans-serif : jenis font yang tidak meruncing pada garis akhir karakternya. Contohnya “Open Sans”, “Fira Sans” dan lainnya.
* monospace : jenis font yang memiliki nilai lebar tiap karakternya sama. Consolas merupakan salah satu jenisnya.
* cursive: jenis font yang tampak seperti handwriting atau hasil tulisan tangan.
* fantasy : jenis font yang merepresentasikan karakteristik yang menyenangkan.
* system-ui : jika menerapkan nilai ini maka font yang diterapkan akan sama seperti font yang digunakan pada sistem operasi kita.
* math : jenis font yang digunakan untuk penulisan rumus-rumus matematika.
* emoji : jenis font yang digunakan untuk menampilkan emoji.
* fangsong : jenis font yang menampilkan gaya penulisan Chinese.

Dalam memilih jenis font terdapat istilah yang dinamakan **web safe font.** Web safe font adalah jenis font yang umumnya sudah pasti tersedia di sebagian besar komputer. Sehingga dapat dipastikan bahwa website akan terlihat sebagaimana mestinya pada browser. Berikut merupakan beberapa contoh font yang masuk ke kategori ini.

* Arial (sans-serif)
* Verdana (sans-serif)
* Helvetica (sans-serif)
* Tahoma (sans-serif)
* Trebuchet MS (sans-serif)
* Times New Roman (serif)
* Georgia (serif)
* Garamond (serif)
* Courier New (monospace)
* Brush Script MT (cursive)

### font-size

Mengubah nilai font pada sebuah dokumen adalah hal yang sangat wajar terjadi, begitu pula pada website. Untuk menetapkan ukuran font, kita perlu menerapkan properti font-size. Kita bisa menetapkan nilai dari properti ini dengan menuliskan langsung nilai dan satuannya. Contohnya seperti ini:

1. h1 {
2. font-size: 1.5em;
3. }

Pastikan bahwa saat menuliskan nilai dan satuannya, tidak ada jarak (spasi).

1. /\* Rule tidak akan diterapkan pada target karena kesalahan penulisan nilai properti \*/
2. h1 {
3. font-size: 1.5 em;
4. }

Satuan dalam menetapkan ukuran font terdapat dua jenis. Yang pertama relative, yakni satuan yang nilainya tergantung pada sesuatu hal, contohnya ukuran dari viewport, induk elemen ataupun ukuran teks standar. Dan yang kedua adalah absolute, yakni satuan yang nilainya telah ditentukan atau digunakan dalam dunia nyata.

Berikut merupakan nilai satuan yang dapat kita manfaatkan dalam menetapkan ukuran font beserta fungsinya:

**Relative unit**

|  |  |  |
| --- | --- | --- |
| **Satuan** | **Relative to** | **Fungsi** |
| em | Font size | Satuan relatif terhadap ukuran font yang sedang digunakan pada elemen (contohnya, 2em berarti 2 kali lebih besar dari ukuran font seharusnya). |
| ex | Font height | Satuan relatif terhadap tinggi font saat ini, satuan ini sangat jarang sekali digunakan |
| rem | Font size | Mirip seperti em, tetapi rem merupakan satuan relatif terhadap ukuran font dari root element. |
| ch | Font width | Satuan relatif terhadap lebar dari karakter “0” nol. |
| vw | Viewport width | Satuan relatif terhadap 1% lebar viewport. Contoh 1vw = 1% dari lebar viewport. Satuan ini tidak didukung pada browser IE8 ke bawah. |
| vh | Viewport height | Satuan relatif terhadap 1% tinggi viewport. Contoh 1vh = 1% dari tinggi viewport. Satuan ini tidak didukung pada browser IE8 ke bawah. |

**Absolute unit**

|  |  |
| --- | --- |
| **Satuan** | **Fungsi** |
| px | Menetapkan nilai font berdasarkan ukuran pixel |
| pt | Menetapkan nilai font berdasarkan points (1/72 inch di CSS2.1) |
| pc | Menetapkan nilai font berdasarkan picas (1 pica = 12 point) |
| mm | Menetapkan nilai font berdasarkan millimeters |
| cm | Menetapkan nilai font berdasarkan centimeters |
| in | Menetapkan nilai font berdasarkan inches |

Selain dengan menetapkan nilai dan satuannya secara langsung, untuk mengatur ukuran font kita juga bisa gunakan nilai persentase.

1. body {
2. font-size: 16px;
3. }
4. h1 {
5. font-size: 150%; /\* 150% dari 16 = 24px \*/
6. }

Pada contoh ini ukuran font dari elemen <h1> seharusnya memiliki ukuran 16px karena mewarisi dari induk elemennya (body). Namun, di bawahnya terdapat rule yang menargetkan secara spesifik untuk elemen <h1> untuk menerapkan ukuran font sebesar 150% dari ukuran induknya sehingga elemen <h1> akan nampak 50% lebih besar dari elemen lain yang ada di dalam <body>.

Dan yang terakhir kita juga bisa menentukan ukuran font dengan menuliskan kata kunci secara spesifik yang tersedia pada CSS. Kata kunci tersebut adalah: xx-small, x-small, small, medium, large, x-large, dan xx-large.

Kata kunci tersebut tidak ada kaitannya dengan pengukuran tertentu (bukan ukuran yang absolute) tetapi nilainya diubah secara konsisten satu sama lain.

Pada gambar di atas kita bisa lihat bahwa standarnya browser menampilkan teks dengan nilai medium. Properti font-size dapat diaplikasikan ke seluruh elemen yang ada di HTML dan nilainya dapat diturunkan pada elemen turunannya.

### font-weight

Setelah mengenal font families dan font size, selanjutnya ada juga font-weight yang digunakan untuk mengatur ketebalan dari font yang ditampilkan. Nilai dari properti ini dapat ditentukan dengan menggunakan numeric values (100 sampai 900) atau dengan menggunakan descriptive terms (normal, bold, bolder, dan lighter).

Properti font-weight dapat diaplikasikan ke seluruh elemen yang ada di HTML dan nilainya dapat diturunkan pada elemen turunannya.

### font-style

Properti selanjutnya adalah font-style. Properti ini digunakan untuk menentukan postur dari teks yang ditampilkan, apakah bentuknya vertikal (normal) atau miring (italic dan oblique).

Italic dan oblique keduanya menampilkan teks yang miring. Perbedaanya adalah italic menerapkan tipe miring (italic font version) dari suatu font sedangkan oblique adalah font normal yang hanya dibuat miring.

Properti font-style dapat diaplikasikan ke seluruh elemen yang ada di HTML dan nilainya dapat diturunkan pada elemen turunannya.

### font-variant

Kita yang terbiasa dengan aplikasi document editor (rich text) seperti Microsoft Word, tentu tahu atau sudah mencoba fitur small caps. Fitur ini dapat membuat teks menjadi kapital tetapi dituliskan secara kecil dan merapat, seperti ini:

Untuk membuat tulisan seperti gambar di atas, kita bisa memanfaatkan properti font-variant dengan nilai small-caps pada propertinya.

Properti font-variant dapat diaplikasikan ke seluruh elemen yang ada di HTML dan nilainya dapat diturunkan pada elemen turunannya.

### Shorthand

Menspesifikasikan masing-masing nilai properti font akan menghasilkan banyak sekali kode repetitif. Dengan begitu CSS memberikan suatu “jalan pintas” untuk menuliskan properti-properti tersebut ke dalam satu properti yaitu font.

Dengan menggunakan properti font kita dapat menuliskan beberapa properti hanya dalam satu properti pada satu rule.

Nilai dari properti font merupakan nilai dari seluruh properti dari font yang sudah kita bahas. Tiap nilai dipisahkan menggunakan spasi. Pada properti ini urutan nilai merupakan hal yang penting, sehingga jangan sampai salah urutan dalam menuliskannya. Kesalahan penulisan atau urutan menyebabkan seluruh struktur rule menjadi tidak valid.

Walaupun begitu kita bisa tidak menuliskan seluruh nilai properti yang ada, tetapi nilai dari properti font-size dan font-family wajib ada ketika menggunakan properti ini. Berikut contohnya penulisan minimal ketika kita menggunakan properti font:

1. p {
2. font: 1em sans-serif;
3. }

## Text Styling

Pembahasan sebelumnya kita terfokus pada formating bentuk karakter yang ditampilkan dengan menggunakan beberapa properti font yang ada. Sekarang kita akan mempelajari bagaimana seorang developer bisa memberikan formatting pada keseluruhan teks yang ada pada baris paragraf, seperti menetapkan indent, jarak antar baris, kata dan huruf, dan sebagainya. Maka dari itu, mari kita bahas satu persatu.

### Line Height

Properti line-height digunakan untuk mengatur jarak minimal dari garis dasar ke garis dasar dalam menampilkannya teks pada halaman. Jika kita terbiasa dengan software document editor (rich text) seperti Microsoft Word, properti ini mirip dengan fungsi line and paragraph spacing.

Pada penjelasan diatas disebutkan “minimal”, karena jika terdapat sebuah karakter yang tinggi atau besar dalam sebuah baris, maka tinggi dari baris pun akan menyesuaikan agar jarak tetap mengakomodirnya.

Berikut merupakan contoh tiga cara berbeda dalam menerapkan tinggi baris dua kali lebih tinggi dari ukuran font:

1. p {
2. line-height: 2;
3. }
4. p {
5. line-height: 200%;
6. }
7. p {
8. line-height: 2em;
9. }

Cara pertama merupakan cara yang paling mudah digunakan, karena kita dapat menentukan nilai hanya dengan satu angka, di mana angka tersebut nantinya dikalikan dengan nilai font-size sebelum diterapkan pada nilai properti line-height. Contohnya, ukuran font standar pada paragraf adalah 16 pixel. Kita definisikan properti line-height dengan nilai 2, maka nilai properti line-height seharusnya adalah **16 pixel \* 2 = 32 pixel (dua kali lebih besar dari ukuran font).**

Properti line-height dapat diaplikasikan ke seluruh elemen yang ada pada HTML dan nilainya dapat diturunkan pada elemen turunannya.

### Text Indent

Dalam membuat sebuah dokumen tidak jarang kita membutuhkan jarak/lekuk di awal paragraf. Hal tersebut dapat dilakukan pada website dengan menerapkan properti text-indent.  Kita dapat menentukan nilai properti ini melalui perhitungan panjang dalam px, em, dan in atau bisa menggunakan nilai persentase (%). Nilai persentase dihitung berdasarkan lebar dari induk elemen. Berikut merupakan contoh penggunaannya:

1. p#first {
2. text-indent: 2em;
3. }
5. p#second {
6. text-indent: 25%;
7. }
9. p#third {
10. text-indent: -35px;
11. }

Jika elemen menerapkan rule tersebut, akan tampak seperti ini:

Pada contoh ke-tiga kita bisa melihat bahwa pada nilai properti ini dapat diberikan nilai negatif. Jika kita menggunakannya, maka baris pertama pada paragraf akan keluar dari batas elemen yang menampungnya (biasa disebut hanging indent).

Perlu diingat kembali, properti ini hanya berpengaruh pada awal baris paragraf. Jika kita ingin menetapkannya untuk seluruh baris kita dapat gunakan margin atau padding, keduanya akan dibahas pada materi box model.

### Text Alignment

Kita bisa mengatur text alignment (perataan teks) pada website seperti kita melakukannya pada aplikasi Microsoft Word dengan menggunakan properti text-align. Untuk asalnya, properti ini bernilai left atau biasa kita sebut rata kiri. Namun jika kita ingin mengubah perataan teks ini berada di posisi lainnya, maka kita dapat mengubah value-nya menjadi value yang sesuai dengan keinginan kita.

Berikut ini nilai yang dapat digunakan pada properti text-align:

|  |  |
| --- | --- |
| **Nilai Properti** | **Fungsi** |
| text-align: **left** | Membuat perataan teks pada ujung kiri |
| text-align: **right** | Membuat perataan teks pada ujung kanan |
| text-align: **center** | Membuat perataan teks secara menengah |
| text-align: **justify** | Membuat perataan teks yang setara pada ujung kiri dan kanannya |

Bagaimana? Pasti kita sudah familiar dengan nilai-nilai tersebut? Berikut contoh penggunaan dari properti text-align:

1. p#first {
2. text-align: left;
3. }
5. p#second {
6. text-align: right;
7. }
9. p#third {
10. text-align: center;
11. }
13. p#fourth {
14. text-align: justify;
15. }

Jika elemen menerapkan rule tersebut, akan tampak seperti ini:

### Text Decoration

Properti ini paling populer digunakan ketika kita ingin membuat garis bawah atau underline pada teks. Tapi tak hanya itu, ada beberapa nilai lain yang dapat kita gunakan untuk properti ini. Detailnya sebagai berikut:

|  |  |
| --- | --- |
| **Nilai properti** | **Fungsi** |
| text-decoration: **underline** | Memberikan garis bawah (underline) pada teks |
| text-decoration: **overline** | Memberikan garis atas (overline) pada teks |
| text-decoration: **line-through** | Memberikan efek tulisan dicoret (strikethrough) |
| text-decoration: **none** | Menghilangkan dekorasi teks yang ada pada elemen |

Dengan menggunakan text-decoration, kita dapat menghilangkan efek underline pada link yang ditampilkan dengan memberikan nilai text-decoration: none pada elemen <a>.

### Text Transform

Pasti kita pernah mencoba fitur pada document editor yang dapat mengubah kapitalisasi pada teks, bukan? Fitur ini sangat membantu di kala kita ingin mengubah kapitalisasi tanpa harus menuliskan kembali teksnya. Pada CSS juga terdapat fitur serupa, yaitu dengan menggunakan properti text-transform. Ketika kita menerapkan properti text-transform pada elemen teks, maka kapitalisasi akan berubah ketika halaman di-render tanpa harus mengubahnya pada dokumen HTML.

Properti ini dapat berisikan nilai sebagai berikut:

|  |  |
| --- | --- |
| **Nilai Properti** | **Fungsi** |
| text-transform: **none** | Teks yang ditampilkan sama seperti yang dituliskan |
| text-transform: **capitalize** | Membuat huruf pertama besar pada tiap katanya |
| text-transform: **lowercase** | Membuat seluruh teks menggunakan huruf kecil |
| Text-transform: **uppercase** | Membuat seluruh teks menggunakan huruf besar |

Seperti ini contoh tampilan ketika menerapkan properti dan nilai di atas

### Word and Letter Spacing

Properti selanjutnya yang bisa kita gunakan untuk formating teks adalah letter-spacing dan word-spacing. Seperti namanya, properti ini digunakan untuk mengatur spasi atau jarak pada teks. Properti letter-spacing digunakan untuk mengatur jarak antar huruf, sedangkan word-spacing digunakan untuk mengatur jarak antar kata. Berikut contohnya:

1. p#letter {
2. letter-spacing: 4px;
3. }
4. p#word {
5. word-spacing: 1.5em;
6. }

Berikut contoh tampilan jika menerapkan rule di atas:

Kedua elemen ini dapat diaplikasikan ke seluruh elemen yang ada pada HTML dan nilainya dapat diturunkan pada elemen turunannya.

### Text Shadow

Memberikan bayangan pada teks telah menjadi hal yang umum digunakan meskipun tidak memiliki dukungan di semua browser. Pada CSS untuk membuat bayangan pada teks (atau biasa disebut drop shadow) kita dapat gunakan properti text-shadow.

Nilai dari properti ini cukup rumit karena membutuhkan tiga buah nilai dan satu buah nilai warna sehingga membutuhkan empat nilai dalam satu properti untuk menentukan bayangannya.

* Nilai pertama : menunjukkan seberapa jauh ke kiri atau kanan (horizontal) bayangan harus ditampakkan.
* Nilai kedua : menunjukkan jarak ke atas atau ke bawah (vertical) bayangan harus ditampakkan.
* Nilai Ketiga (opsional) : menentukan tingkat keburaman yang harus diterapkan pada bayangan.
* Nilai Keempat :  menentukan warna yang digunakan pada bayangan.

Berikut ini contoh penggunaan dari properti drop shadow:

1. p#one {
2. text-shadow: 1px 1px 0px #000000;
3. background-color: #eeeeee;
4. color: #666666;
5. }
7. p#two {
8. text-shadow: 1px 1px 3px #666666;
9. background-color: #dddddd;
10. color: #666666;
11. }
13. p#three {
14. text-shadow: 2px 2px 7px #111111;
15. background-color: #cccccc;
16. color: #ffffff;
17. }
19. p#four {
20. text-shadow: -1px -1px #666666;
21. background-color: #cccccc;
22. color: #bbbbbb;
23. }

Berikut contoh tampilan jika menerapkan rule di atas:

## Menerapkan Font Styling pada Halaman Profil

Sejauh ini kita sudah mengenal banyak sekali properti yang dapat mengatur formatting text, kita juga sudah mengetahui bahwa CSS memisahkan properti styling untuk font dan text. Nah, sekarang saatnya kita terapkan beberapa properti yang sudah kita pelajari pada halaman profil yang sudah kita buat.

Pada latihan kali ini juga, kita akan belajar bagaimana menerapkan font yang bersumber dari Google Font, sehingga kita bisa menampilkan tipe font selain hanya generic font families.

Sudah siap? Lanjutkan ke materi berikutnya, ya!

### Menggunakan Font dari Google Font

Kita sudah mengetahui bagaimana cara mengubah font pada website dan juga mengetahui tipe font standar yang tersedia pada browser. Berbicara tentang tipe font, ketika kita menjelajah beberapa website yang ada di internet tidak jarang kita menemukan website yang menggunakan font yang unik atau font yang tidak dimiliki pada komputer kita. Lantas bagaimana cara font tersebut dapat ditampilkan?

Untuk hal ini, CSS memiliki fitur yang digunakan untuk memasukkan font external ke dalam CSS, yakni menggunakan @font-face. Berikut contoh penulisan @font-face pada CSS:

1. @font-face {
2. font-family: "Dicoding Font";
3. src: url('FILE-FONT.TTF');
4. }

Kode tersebut berfungsi untuk mendefinisikan custom font, dengan nama “Dicoding Font”. Ketika mendefinisikan custom font, kita juga membutuhkan berkas font, bisa dilihat berkas font pada contoh adalah ‘FILE-FONT.TTF”.

Setelah mendefinisikan font-face, maka font-family dengan nama “Dicoding Font” siap diterapkan pada elemen HTML.

1. p {
2. font-family: "Dicoding Font";
3. }

Pada kelas ini kita tidak akan membahas secara detail mengenai @font-face. Jika ingin mempelajarinya, Anda bisa mengunjungi dokumentasi Mozilla pada tautan berikut: <https://developer.mozilla.org/en-US/docs/Web/CSS/@font-face>. Mengapa kita tak mempelajari hal ini secara detail? Karena sekarang terdapat cara yang lebih mudah dan sering digunakan oleh web developer, yakni menggunakan font dari Google Font.

Google Font, layanan Google untuk memudahkan kita mendesain font pada website. Google Font banyak menyediakan tipe font gratis yang dapat kita manfaatkan. Saat materi ini ditulis, Google Font memiliki jenis font sekitar 1.364.

Perlu jadi catatan bahwa Google Font hanya dapat digunakan ketika komputer dalam keadaan online. Jika kita sedang mengembangkan website dalam keadaan offline, sebaiknya gunakan pendekatan @font-face.

Untuk menggunakan font yang disediakan Google Font, langkah awal yaitu mengunjungi website google font pada tautan berikut: <https://fonts.google.com/>.

Saat ini kita akan mencari font dengan nama “Quicksand**.**” Lakukan pencarian font tersebut pada search bar yang tersedia pada website tersebut.

Jika sudah menemukan font yang dicari, maka klik pada font pilihan kita dan akan diarahkan ke halaman berikutnya.

Tekan tombol “+ Select this style” untuk memilih font dengan ukuran yang diinginkan. Di sini kita contohkan dengan memilih font ukuran Regular 400. Ketika menekan tombol tersebut, Google Font akan menyimpannya sebagai selected families (pilih tombol pada pojok kanan atas untuk melihat selected families).

Kita juga bisa memilih ketebalan fontnya. Yuk, pilih juga versi tebalnya (Bold 700) dan masukan kembali ke dalam collection.

Setelah selesai memilih font yang dicari, maka tampilan dari selected families akan tampak seperti ini:

Selanjutnya klik pada tab embed dan pilih tulisan @import. Salin kode @import beserta URL-nya.

* [**Teks yang disalin**](https://www.dicoding.com/academies/123/tutorials/4229?from=4226#tab1-code1)

1. @import url('https://fonts.googleapis.com/css?family=Quicksand:400,700&display=swap');

Kemudian buka kembali berkas **style.css** pada project kita dan sisipkan kode tersebut pada baris paling awal. Sehingga keseluruhan kode pada berkas style.css seperti ini:

1. @import url('https://fonts.googleapis.com/css?family=Quicksand:400,700&display=swap');

4. body {
5. font-family: sans-serif;
6. }

9. h2, h3 {
10. color: #00a2c6
11. }

14. footer {
15. padding: 20px;
16. color: white;
17. background-color: #00a2c6
18. }

Untuk menerapkan font-nya, kita ubah nilai properti font-family pada body dengan menambahkan nilai **‘**Quicksand**’**sebelum generic font sans-serif.

1. body {
2. font-family: 'Quicksand', sans-serif;
3. }

Dengan begitu tampilan font pada halaman website akan menggunakan Quicksand.

Bagaimana? Mudah kan? Sebenarnya google font menggunakan teknik yang sama, yakni menggunakan @font-face. Jika kita coba buka URL yang ada pada script @import tadi pada browser, hasilnya adalah  sebuah kode CSS di mana terdapat banyak @font-face dituliskan disana.

Dengan fungsi @import pada CSS, seolah-olah kode tersebut dituliskan pada berkas style.css yang kita buat.

### Menetapkan ukuran, ketebalan, dan alignment teks pada Halaman Profil

Setelah menentukan tipe font baru dalam menampilkan teks, selanjutnya kita percantik lagi teksnya dengan mengatur ukuran, ketebalan, dan alignment pada teks. Jika Anda sudah menutup berkas **style.css** pada text editor, silakan buka kembali dan ikuti langkah - langkah berikut.

Langkah pertama kita atur ukuran font menjadi 18px pada teks yang berada di elemen <a> yang merupakan turunan dari elemen <nav>. Silakan tambahkan rule dibawah ini.

1. nav a {
2. font-size: 18px;
3. }

Kemudian kita atur juga ketebalan teksnya menjadi **400** dan hilangkan garis bawah pada elemen tersebut sehingga block rule-nya menjadi seperti ini:

1. nav a {
2. font-size: 18px;
3. font-weight: 400;
4. text-decoration: none;
5. }

Sehingga jika kita preview pada browser, maka ukuran teks pada navigasi akan tampak lebih besar dan tidak memiliki garis bawah.

Selain itu, kita juga bisa memberikan efek tebal pada teks ketika salah satu item navigasi disorot menggunakan kursor. Tambahkan rule baru dengan memanfaatkan pseudo-class :hover pada elemen <a>.

1. nav a:hover {
2. font-weight: bold;
3. }

Sehingga tampilan menjadi seperti ini:

Selanjutnya teks padaheader aside akan kita buat menjadi rata tengah dengan menggunakan properti text-align. Sebelum itu, untuk memudahkan menetapkan selector, berikan attribute class pada elemen <article> yang berada di dalam elemen <aside> dengan nilai profile.

1. <aside>
2. <article class="**profile**">
3. <header>
4. .....
5. </header>
6. <section>
7. .....
8. </section>
9. </article>
10. </aside>

Setelah itu, kita dapat menentukan selector dengan mudah untuk mengatur <header> pada elemen <article> tersebut:

1. .profile header {
2. text-align: center;
3. }

Maka teks pada header aside akan menjadi rata tengah.

Terakhir, kita buat juga teks pada footer menjadi rata tengah dan tebal dengan menambahkan properti text-align: center; dan font-weight: bold; pada footer selector:

1. footer {
2. padding: 20px;
3. color: white;
4. background-color: #00a2c6;
5. text-align: center;
6. font-weight: bold;
7. }

Sehingga tampilan footer akan tampak seperti ini:

# Color

Warna memberikan kesan hidup pada sebuah website, Jika kita kembali pada tahun 1993 saat website hanya menampilkan background abu dengan teks hitam, tentu membosankan sekali, bukan?

[](https://www.dicoding.com/academies/123/tutorials/4233?from=4230)  
Yahoo Search Engine pada tahun 1994, Sumber: <https://www.webdesignmuseum.org/>

Pada saat itu memang belum tersedia sebuah browser yang dapat mengontrol warna. Sampai akhirnya Netscape Navigator hadir dan memungkinkan hal tersebut, walaupun belum sempurna layaknya saat ini. Syukurlah pada saat ini kita bisa menggunakan properti - properti yang ada pada CSS untuk mengatur warna pada teks dan background. Terlebih, seluruh browser saat ini sudah mendukung styling menggunakan CSS sehingga kita tidak perlu lagi memikirkan kompatibilitasnya.

Pembahasan kali ini kita akan fokus mengenai warna pada CSS. Anda akan dikenalkan pada properti yang digunakan untuk mengatur warna teks dan background. Tapi sebelum itu, mari kita pelajari dahulu bagaimana cara menetapkan sebuah warna pada CSS.

## Menetapkan Nilai Warna

Setiap warna pada layar komputer kita terdiri dari campuran warna merah, hijau, dan biru. Layar komputer dibuat dari ribuan kotak kecil yang biasa disebut pixel. Ketika layar monitor tidak menyala maka pixel pun tidak akan menyala. Ketika layar monitor menyala, tiap pixel dapat menghasilkan warna yang berbeda sehingga dapat menampilkan gambar.

Di dalam pixel terdapat tiga buah lampu kecil berwarna merah, hijau, dan biru. Ribuan warna dapat dihasilkan dari kombinasi tiga warna tersebut.

Ketika ingin menetapkan warna, kita tidak perlu menghafal kombinasi dari ketiga warna tersebut. Banyak pemilih warna yang dapat membantu kita dalam menetapkan warna yang diinginkan. Contohnya seperti color picker yang ada pada Photoshop, GIMP atau Gravit Designer.

Kita bisa melihat nilai RGB melalui color picker. Lantas bagaimana jika kita tidak memiliki aplikasi design seperti yang telah disebutkan tadi? Tenang, sekarang color picker banyak tersedia online, bahkan ketika kita menuliskan “color picker” pada Google Search, tools tersebut tersedia pada hasil pencariannya.

Untuk menetapkan nilai warna pada CSS kita dapat menggunakan Numeric Value dan Predefined Color Name. Apa sih maksud kedua hal tersebut? Mari kita bahas satu persatu yah.

### Numeric Value

Cara yang paling banyak digunakan dalam menetapkan warna adalah  dengan menggunakan numeric value. Mengapa demikian? Dengan menggunakannya kita dapat menentukan warna yang kita inginkan dengan tepat. Terlebih dengan bantuan color picker maka kita tak perlu repot memahami dan mencari nilai kode warna secara manual.

Berikut contoh penulisan nilai warna pada CSS dengan numeric value:

1. /\* Warna hijau menggunakan format RGB \*/
2. color: rgb(78,231,23);
4. /\* Warna hijau Menggunakan format HEX \*/
5. color: #4ee717;
7. /\* Warna hijau menggunakan format HSL (hanya di CSS3)\*/
8. color: hsl(104, 82%, 50%);

Kode tersebut merupakan nilai warna hijau yang dituliskan dalam beberapa format. Dari kode tersebut kita tahu tidak hanya format RGB yang dapat digunakan dalam menetapkan warna. Selain RGB ada juga format Hex dan HSL (Hue, Saturation, Light).

### Predefined Color Name

Menentukan warna bisa juga dilakukan selain dengan menggunakan format angka. Kita bisa menggunakan sebuah kata seperti blue, yellow, red dan lainnya

1. color: black;
2. color: white;
3. color: blue;

Dengan menggunakan kata tentu kita dapat lebih mudah dan cepat dalam menentukan warna, tetapi mungkin hanya warna dasar yang sudah kita hafal. Faktanya, varian warna ada banyak jumlahnya! Hampir seluruh browser saat ini mendukung 140 nama warna, wow! Kita dapat melihat apa saja warna yang tersedia pada tautan berikut: <https://www.w3schools.com/colors/colors_names.asp>.

### Opacity

CSS3 mengenalkan properti opacity yang dapat menspesifikasikan nilai transparan pada elemen HTML. Nilai dari properti opacity merupakan angka desimal antara 0.0 hingga 1.0, jika kita menetapkan nilainya 0.5 itu berarti 50% transparan.

1. div#opacity {
2. background-color: red;
3. opacity: 0.5;
4. }

Jika elemen div menerapkan rule tersebut, maka akan tampak seperti ini:

Selain pada elemen, menetapkan opacity juga dapat kita lakukan pada warna. Untuk melakukannya bisa menggunakan format **rgba,** dan **hsla**, di mana format ini merupakan format rgb ditambahkan nilai alpha. Selain itu, dapat juga dilakukan menggunakan format **hex** dengan memberikan dua digit dibelakang kode hexadecimal.

1. p {
2. color: rgba(78,231,23, 0.5);
3. color: hsla(104, 82%, 50%, 0.5);
4. color: #4ee717ee; /\*ee merupakan digit tambahan untuk menerapkan opacity\*/
5. }

## Text Color (Foreground Color)

Pada sub-modul Pendahuluan Styling kita sudah mencoba bagaimana mudahnya mengganti warna pada teks yang ditampilkan. Sejujurnya tidak ada lagi yang perlu kita bahas disini. Kita dapat dengan mudahnya mengganti warna teks dengan properti color.

Nilai dari properti color dapat berupa predefined color name atau sebuah numeric value dengan menggunakan RGB, RGBA, HEX, HSL, ataupun HSLA. Berikut sebagai contoh, seluruh elemen <p> akan diberi warna abu-abu (gray) dengan menggunakan beberapa cara yang ada:

1. p { color: gray; }
2. p { color: #666666; }
3. p { color: #666; }
4. p { color: rgb(102,102,102); }

**Catatan:** Terdapat dua cara dalam penetapan nilai color menggunakan format **hexadecimal**, yaitu dengan menyebutkan digit hexadecimal secara lengkap (6 digit) dan menyebutkan tiga digit (paling sedikit) hexadecimal. Pada contoh diatas, nilai #666666 dan #666 memiliki hasil yang sama. Artinya setiap digit dari #666 akan melakukan duplikasi dari digit pertama ke digit kedua, digit ketiga ke digit keempat dan digit kelima ke digit keenam, sesuai urutan pendefinisiannya.

Properti color dapat diaplikasikan ke seluruh elemen yang ada pada HTML dan nilainya dapat diturunkan pada elemen turunannya. Jadi kita bisa mengubah warna teks pada seluruh dokumen HTML dengan menerapkan properti color pada elemen <body>, seperti ini:

1. body {
2. color: steelblue;
3. }

Sehingga tampak seperti ini:

## Background Color

CSS memperlakukan setiap elemen HTML seperti berada pada sebuah kotak (kita akan mempelajari ini pada pembahasan box model). Properti background-color dapat mengatur warna latar belakang dari kotak tersebut.

Sama seperti text color, kita dapat menspesifikasikan warna yang dipilih dengan numeric values atau dengan predefined color name. Properti background-color akan bernilai transparan jika tidak kita tetapkan (default).

Kebanyakan browser menetapkan nilai putih sebagai standar untuk nilai background-color, tetapi nilai standar tersebut dapat pengguna ubah menggunakan pemilihan mode cerah atau mode gelap pada pengaturan browser-nya. Maka untuk memastikan website kita memiliki tampilan background color putih, kita dapat terapkan nilai background-color: white; pada elemen <body>.

Biasanya kita juga menerapkan padding saat menetapkan background-color guna memberi jarak antara konten dan tepi kotak elemen.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. body {
6. font-family: sans-serif;
7. }
9. div#content {
10. background-color: rgb(200, 200, 200);
11. padding: 20px 15px;
12. }
14. #content > h2 {
15. background-color: #4c5450;
16. padding: 15px;
17. color: white;
18. }
20. #content > h3 {
21. background-color: #60d0a8;
22. padding: 15px;
23. color: white;
24. }
26. #content > p {
27. background-color: hsl(56, 62%, 85%);
28. padding: 20px;
29. }
30. </style>
31. </head>
32. <body>
33. <div id="content">
34. <h2>Lorem Ipsum</h2>
35. <h3>Dolor sit amet lorem ipsum</h3>
36. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Natus officiis perspiciatis quidem ratione? Distinctio eos ex expedita iusto necessitatibus velit, veritatis. Aliquid, debitis dignissimos in iusto magnam nulla sed tempora.</p>
37. </div>
38. </body>
39. </html>

Jika berkas di atas dijalankan pada browser, maka hasilnya akan tampak seperti ini:

# Box Model

Pada pembahasan background-color disinggung bahwa setiap elemen ditampilkan seperti sebuah boxes atau kotak. Box model merupakan salah satu konsep dari CSS. Setiap elemen yang dibuat pada HTML akan menciptakan sebuah kotak untuk menampung kontennya.  Layaknya bentuk kotak pada umumnya, pasti terdapat nilai seperti width (lebar), height (tinggi), padding (tebal lapisan), border (garis tepi), dan juga jarak. Pada CSS nilai - nilai tersebut dapat kita atur.

Kita mungkin sudah sedikit paham bagaimana konsep box model ini bekerja dengan mengikuti latihan sebelumnya. Di sini kita akan membahas lebih lanjut mengenai properti apa saja yang dapat memanipulasi boxes tersebut. Tapi sebelum itu, kita pahami lebih tentang konsep box atau kotak pada elemen HTML agar kita dapat gambaran lebih tentang hal tersebut.

**Box Element**

Website dicoding.com dengan menerapkan rule \* {border: 1px solid red}

Seperti yang kita lihat pada gambar di atas, setiap elemen pada HTML baik itu block-level atau inline elemen, akan menghasilkan sebuah kotak elemen. Perhatikan ilustrasi berikut agar lebih jelasnya:

* *Content* : Merupakan sebuah inti dari kotak elemen yang merupakan konten dari elemen itu sendiri
* *Padding* : Area yang menjadi jarak antara border elemen dengan konten yang ditampilkan. Ketika kita menerapkan background-color maka warna tersebut diterapkan pada area padding. Penerapan ukuran padding pada elemen merupakan opsional.
* *Border* : Merupakan garis terluar yang mengelilingi konten dan padding. Penerapan border pada elemen merupakan opsional.
* *Margin* : Berbeda dengan padding, margin merupakan area jarak diluar dari border. Pada ilustrasi diatas, margin ditunjukkan dengan warna merah muda namun pada aslinya margin selalu transparan dan penerapan margin pada elemen merupakan opsional.
* *Edge of element* : merupakan batas dari suatu elemen. Di dalamnya terdapat margin, border, padding dan konten. Pada ilustrasi di atas, batas elemen ditunjukkan dengan garis putus tetapi pada aslinya batas elemen selalu tidak terlihat.

## Box Dimensions

Secara standar sebuah box yang dihasilkan tiap elemen selalu cukup untuk menampung konten. Tetapi kita dapat mengatur nilai dimensi dari box tersebut dengan properti width dan height.

Cara yang paling banyak digunakan dalam menentukan dimensi kotak adalah dengan menggunakan pixel, persentase, atau ems. Secara tradisional, pixel merupakan cara yang paling populer karena kita dapat merancang dan mengontrol ukuran secara akurat.

Berbeda ketika kita menggunakan persentase, ukuran kotak akan relative atau menyesuaikan dari ukuran lain, seperti ukuran jendela browser atau ukuran induk yang menaunginya. Sedangkan ketika menggunakan ems, nilai dimensi kotak akan menyesuaikan berdasarkan ukuran teks yang ditampilkan pada konten elemen tersebut.

Pada saat ini banyak developer mulai merancang menggunakan persentase dan ems untuk menetapkan ukuran box agar dapat menyesuaikan dengan berbagai macam ukuran layar.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .box {
6. height: 300px;
7. width: 300px;
8. background-color: #11C5C6;
9. }
10. p {
11. height: 75%;
12. width: 75%;
13. background-color: #FBDD1C;
14. }
15. </style>
16. </head>
17. <body>
18. <div class="box">
19. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Natus officiis perspiciatis quidem ratione? Distinctio eos ex expedita iusto necessitatibus velit, veritatis. Aliquid, debitis dignissimos in iusto magnam nulla sed tempora.</p>
20. </div>
21. </body>
22. </html>

Berkas di atas jika kita buka melalui browser akan tampak seperti ini:

Pada contoh di atas kita dapat melihat elemen <div> memiliki dimensi elemen dengan lebar **300px** dan tinggi **300px**. Di dalamnya terdapat elemen <p> yang memiliki ukuran elemen 75% dari lebar dan tinggi elemen induknya. Dengan begitu berarti elemen <p> memiliki ukuran **225px** untuk panjang dan lebarnya.

### Limiting Dimension

Beberapa website yang ada sekarang menampilkan layout yang dapat melebar dan menyempit mengikuti ukuran layar pengguna. Pada prinsip tampilan tersebut mungkin kita memerlukan sebuah limitasi ukuran yang harus ditampilkan agar konten selalu dapat ditampilkan secara proporsional. Untuk melakukannya kita manfaatkan properti min-width dan max-width.

* min-width : merupakan properti yang digunakan untuk menetapkan nilai lebar minimal yang harus dimiliki elemen.
* max-width : merupakan properti yang digunakan untuk menetapkan nilai lebar maksimal yang harus dimiliki elemen.

Keduanya merupakan properti yang sangat membantu untuk memastikan konten halaman dapat terbaca oleh pengguna (terutama ketika pengguna menggunakan ponsel). Misalnya, kita dapat menggunakan properti max-width untuk memastikan bahwa baris teks yang muncul tidak terlalu lebar.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .content {
6. max-width: 800px;
7. height: 400px;
8. margin: 0 auto;
9. background-color: deeppink;
10. }
12. p {
13. font-size: 1.5em;
14. font-weight: bold;
15. }
16. </style>
17. </head>
18. <body>
19. <div class="content">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Cupiditate eius explicabo fuga iusto magni minus odit praesentium, quasi quisquam quos repellat suscipit tempora tenetur? Assumenda cum laborum officiis quos ratione.</p>
21. </div>
22. </body>
23. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti ini:

Dengan cara yang sama, mungkin kita juga perlu membatasi ukuran panjang. Kita bisa gunakan min-height dan max-height.

### Overflowing Content

Dimensi box yang dihasilkan elemen selalu cukup untuk menampung konten tetapi hal ini tidak berlaku jika kita tetapkan secara manual panjang dan lebarnya. Tak jarang terjadi overflow ketika kita menerapkan ukuran pada elemen tetapi konten di dalamnya begitu banyak. Contohnya seperti berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. div {
6. height: 200px;
7. width: 200px;
8. background-color: lightgreen;
9. }
10. </style>
11. </head>
12. <body>
13. <div>
14. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ab accusantium amet corporis cupiditate dignissimos error expedita explicabo facere facilis, ipsam itaque libero modi officiis optio perspiciatis provident quam sequi voluptatem.</p>
15. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. At aut corporis facere harum inventore iure, quisquam recusandae reprehenderit veritatis voluptatum. Ducimus exercitationem illo nam sapiente suscipit, tenetur ullam ut voluptas!</p>
16. </div>
17. </body>
18. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti ini:

Untuk menangani kasus seperti ini kita bisa gunakan properti overflow. Properti ini dapat bernilai:

* visible : Merupakan nilai default pada properti ini. Konten yang tidak tertampung (overflow) akan tetap ditampilkan seperti pada standarnya.
* hidden : Jika terjadi overflow, maka konten yang tidak tertampung akan disembunyikan.
* scroll : Memunculkan scroll bar pada pinggir elemen, sehingga konten yang tidak tertampung akan ditampilkan dengan scroll bar. Jika menggunakan nilai ini, scroll bar akan tetap muncul walaupun konten tidak terjadi overflow.
* auto : Sama seperti scroll, bedanya jika tidak terjadi overflow, maka nilai visible yang akan diterapkan.

### Box-sizing

Pada CSS2 ukuran lebar dan panjang elemen mengacu pada konten elemen (content-box). Itu berarti ukuran elemen seluruhnya merupakan nilai panjang dan lebar yang kita spesifikasikan ditambah dengan nilai padding dan border yang diterapkan pada elemen. Hal tersebut membuat sebagian developer menjadi sulit menetapkan ukuran dimensi.

Setelah CSS3 kita dapat memilih tipe pengukuran lain dalam menentukan dimensi elemen. Dengan menggunakan properti box-sizing kita dapat menentukannya berdasarkan border box, di mana ukuran elemen sudah termasuk content, padding dan border. Dengan metode ini, hasil elemen yang ditampilkan (termasuk padding dan border) akan memiliki dimensi yang sama persis seperti yang kita tentukan.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. div {
6. height: 200px;
7. width: 200px;
8. background-color: lightgreen;
9. border: 10px solid cornflowerblue;
10. padding: 20px;
11. }
13. .content {
14. box-sizing: content-box;
15. }
17. .box {
18. box-sizing: border-box;
19. }
20. </style>
21. </head>
22. <body>
23. <div class="content">
24. </div>
25. <p>Elemen menerapkan <code>box-sizing: content-box;</code> Ukuran box secara keseluruhan akan menjadi 260px lebar, 260px tinggi; 260 = 200 + 20 + 20 + 10 + 10</p>
26. <br>
27. <div class="box">
28. </div>
29. <p>Elemen menerapkan <code>box-sizing: border-box;</code> Ukuran box akan tetap 200px lebar, 200px tinggi meskipun menerapkan padding dan border</p>
30. </body>
31. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti ini:

## Border

Border merupakan sebuah garis yang mengelilingi area konten dan padding (opsional). Kita bisa mengatur tipe, ketebalan, serta warna garis yang ditampilkan sesuai dengan yang kita inginkan. Kita juga bisa mengatur dalam menampilkan sebagian atau keseluruhan garis pada elemen. Mari kita eksplorasi apa saja properties yang dapat mengatur border.

### Border Width

Properti border-width digunakan untuk mengatur ketebalan dari border. Nilai dari properti ini dapat berupa pixel atau menggunakan predefined names value seperti thin, medium, dan thick. Kita tidak bisa menggunakan nilai persentase (%) pada properti ini.

Kita dapat mengatur ukuran garis secara individual dengan menggunakan empat properti terpisah seperti ini:

1. .box {
2. border-top-width: 2px;
3. border-right-width: 1px;
4. border-bottom-width: 1px;
5. border-left-width: 2px;
6. }

Tetapi kita juga dapat menetapkan nilai keempatnya sekaligus dalam satu properti seperti ini

1. .box {
2. border-width: 2px 1px 1px 2px; /\*top right bottom left\*/
3. }

Properti border-width dapat ditentukan dengan menggunakan satu, dua, tiga, atau empat nilai. Berikut penjelasannya:

* Ketika satu nilai ditentukan, maka nilai berlaku untuk empat sisi.
* Ketika dua nilai ditentukan, nilai pertama berlaku untuk sisi atas dan bawah, nilai kedua untuk sisi kiri dan kanan.
* Ketika tiga nilai ditentukan, nilai pertama berlaku untuk sisi atas, nilai yang kedua untuk sisi kiri dan kanan, nilai ketiga untuk sisi bawah.
* Ketika empat nilai ditentukan, nilai pertama berlaku untuk sisi atas, nilai yang kedua untuk sisi kanan, nilai yang ketiga untuk sisi bawah, dan nilai yang keempat untuk sisi kiri. Urutan tersebut berdasarkan arah jarum jam (clockwise).

### Border Style

Kita bisa menetapkan tipe border dengan menggunakan properti border-style. Berikut nilai - nilai yang dapat digunakan pada properti ini:

|  |  |  |
| --- | --- | --- |
| **Nilai Properti** | **Penjelasan** | **Gambar** |
| solid | Tipe garis padat (tidak terputus - putus) |  |
| dotted | Garis yang dibentuk dari serangkaian titik-titik (jika ketebalan garis 2px, maka titik-titik akan berukuran 2px dan memiliki jarak 2px antar titiknya). |  |
| dashed | Garis yang dibentuk dari serangkaian garis pendek. |  |
| double | Garis yang dibentuk dari dua buah garis padat. |  |
| groove | Tipe garis yang berbentuk seperti frame |  |
| hidden | Digunakan untuk menyembunyikan garis pada elemen. |  |

Kita juga bisa menetapkan tipe garis secara individual pada sisi elemen dengan menggunakan empat properti terpisah. Contohnya seperti ini:

1. .box {
2. border-top-style: solid;
3. border-right-style: dotted;
4. border-bottom-style: groove;
5. border-left-style: double;
7. border-width: 4px;
8. border-color: red;
9. width: 200px;
10. height: 50px;
11. }

Jika kita menerapkan rule seperti di atas, maka hasilnya akan tampak seperti ini:

### Border Color

Properti terakhir adalah border-color. Properti ini digunakan untuk menentukan warna pada garis dengan menggunakan nilai RGB, Hex, atau nama warna pada CSS.

1. /\* menggunakan rgb format \*/
2. border-color: rgb(80, 138, 212);
4. /\* menggunakan format hex \*/
5. border-color: #4ee717;
7. /\* menggunakan nama warna \*/
8. border-color: red;

Sama seperti properti border yang lain, kita dapat menentukan warna pada individual sisi pada elemen dengan menggunakan properti yang terpisah.

1. border-top-color: #919191;
2. border-right-color: #111111;
3. border-bottom-color: #4ee717;
4. border-left-color: #00c8eb;

Tetapi kita juga dapat menetapkan nilai keempatnya sekaligus dalam satu properti seperti ini:

1. border-color: #919191 #111111 #4ee717 #00c8eb;

Hasilnya seperti ini:

### Shorthand

Untuk menerapkan border pada elemen kita harus mendefinisikan seluruh properti border yang ada. Dimulai dari menetapkan ketebalan (border-width), tipe (border-style), dan warna (border-color). Jika kita lupa menetapkan salah satu properti tersebut, maka garis tidak akan nampak pada elemen.

Dengan begitu tentu untuk menetapkan border pada elemen, kita perlu menuliskan properti yang banyak bukan? Ya memang, tetapi CSS menyediakan jalan pintas (shorthand) untuk membuat border dengan satu properti saja. Properti border memiliki tiga buah nilai yang digunakan untuk menentukan ketebalan, tipe dan warna pada border. Berikut contoh penerapannya:

1. .box {
2. border: 4px dashed #00a2c6;
3. width: 200px;
4. }

Hasilnya seperti ini:

Perlu kita perhatikan bahwa penulisan urutan harus benar. Nilai pertama digunakan untuk ketebalan, yang kedua untuk tipe, dan yang ketiga untuk warna garis.

## Centering Content

Bagaimana meletakkan sebuah kotak tepat pada tengah sebuah halaman atau di dalam elemen induknya? Aturlah margin kanan dan kiri dengan nilai auto. Lebih lanjut, kita juga harus menentukan lebar dari kotak tersebut (menggunakan properti width). Jika tidak, kotak akan mengambil lebar penuh pada halaman atau induk elemen.

Setelah kita menentukan lebar kotak dan mengatur margin kiri dan kanan menjadi auto, maka secara otomatis browser akan memberi jarak yang sama di setiap sisi horizontal kotak, sehingga membuat kotak berada di tengah halaman.

Berikut contohnya:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Margin</title>
6. <style>
7. .center {
8. margin: 0 auto;
9. }
10. .box {
11. width: 50%;
12. border: 4px solid darkblue;
13. padding: 20px;
14. margin-bottom: 20px;
15. }
16. </style>
17. </head>
18. <body>
19. <div class="box">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aspernatur autem commodi dignissimos dolores ea, eaque,
21. earum esse harum illo in incidunt molestias nam non qui recusandae sunt ullam veniam vero!</p>
22. </div>
23. <div class="box center">
24. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Commodi ea, id. Aliquid consectetur dolorum
25. exercitationem ipsam, necessitatibus nostrum pariatur sunt! Accusantium architecto at dolorem itaque quisquam
26. quod soluta sunt voluptatum.</p>
27. </div>
28. </body>
29. </html>

Jika berkas tersebut kita buka pada browser, maka akan tampak seperti berikut:

## Centering Content

Bagaimana meletakkan sebuah kotak tepat pada tengah sebuah halaman atau di dalam elemen induknya? Aturlah margin kanan dan kiri dengan nilai auto. Lebih lanjut, kita juga harus menentukan lebar dari kotak tersebut (menggunakan properti width). Jika tidak, kotak akan mengambil lebar penuh pada halaman atau induk elemen.

Setelah kita menentukan lebar kotak dan mengatur margin kiri dan kanan menjadi auto, maka secara otomatis browser akan memberi jarak yang sama di setiap sisi horizontal kotak, sehingga membuat kotak berada di tengah halaman.

Berikut contohnya:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Margin</title>
6. <style>
7. .center {
8. margin: 0 auto;
9. }
10. .box {
11. width: 50%;
12. border: 4px solid darkblue;
13. padding: 20px;
14. margin-bottom: 20px;
15. }
16. </style>
17. </head>
18. <body>
19. <div class="box">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aspernatur autem commodi dignissimos dolores ea, eaque,
21. earum esse harum illo in incidunt molestias nam non qui recusandae sunt ullam veniam vero!</p>
22. </div>
23. <div class="box center">
24. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Commodi ea, id. Aliquid consectetur dolorum
25. exercitationem ipsam, necessitatibus nostrum pariatur sunt! Accusantium architecto at dolorem itaque quisquam
26. quod soluta sunt voluptatum.</p>
27. </div>
28. </body>
29. </html>

Jika berkas tersebut kita buka pada browser, maka akan tampak seperti berikut:

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <meta charset="UTF-8">
5. <title>Display</title>
6. <style>
7. li {
8. display: inline;
10. margin-left: 5px;
11. }
12. </style>
13. </head>
14. <body>
15. <ul>
16. <li>Home</li>
17. <li>Product</li>
18. <li>About</li>
19. <li>Contact</li>
20. </ul>
21. </body>
22. </html>

## Box Shadow

Pada materi formatting text kita sudah belajar cara menambahkan drop shadow pada teks. Pada CSS3 kita juga dapat menambahkan drop shadow di sekitar kotak elemen (tidak termasuk margin) dengan menggunakan properti box-shadow.

Nilai dan cara kerja dari properti box-shadow mirip seperti text-shadow, yaitu nilainya menentukan jarak vertikal dan horizontal, tingkat keburaman, dan warna pada bayangan. Pada box shadow, kita juga dapat menentukan tingkat sebaran (spread) bayangan. Semakin besar  nilai, bayangan yang nampak akan semakin luas. Berikut contoh penerapan box shadow pada CSS:

1. box-shadow: 6px 6px 5px 10px #666666;

Jika rules tersebut diterapkan pada box, maka akan tampak seperti ini:

Berikut penjelasan tiap-tiap nilai dari propertinya:

* Nilai pertama : menunjukkan seberapa jauh ke kiri atau kanan (horizontal) bayangan harus ditampakkan.
* Nilai kedua : menunjukkan jarak ke atas atau ke bawah (vertical) bayangan harus ditampakkan.
* Nilai Ketiga (opsional) : menentukan tingkat keburaman yang harus diterapkan pada bayangan.
* Nilai Keempat (opsional) : menentukan tingkat sebaran (spread) bayangan. Semakin besar nilai yang ditentukan, bayangan yang nampak pun semakin luas.
* Nilai Kelima : menentukan warna yang digunakan pada bayangan.

## Box Shadow

Pada materi formatting text kita sudah belajar cara menambahkan drop shadow pada teks. Pada CSS3 kita juga dapat menambahkan drop shadow di sekitar kotak elemen (tidak termasuk margin) dengan menggunakan properti box-shadow.

Nilai dan cara kerja dari properti box-shadow mirip seperti text-shadow, yaitu nilainya menentukan jarak vertikal dan horizontal, tingkat keburaman, dan warna pada bayangan. Pada box shadow, kita juga dapat menentukan tingkat sebaran (spread) bayangan. Semakin besar  nilai, bayangan yang nampak akan semakin luas. Berikut contoh penerapan box shadow pada CSS:

1. box-shadow: 6px 6px 5px 10px #666666;

Jika rules tersebut diterapkan pada box, maka akan tampak seperti ini:

Berikut penjelasan tiap-tiap nilai dari propertinya:

* Nilai pertama : menunjukkan seberapa jauh ke kiri atau kanan (horizontal) bayangan harus ditampakkan.
* Nilai kedua : menunjukkan jarak ke atas atau ke bawah (vertical) bayangan harus ditampakkan.
* Nilai Ketiga (opsional) : menentukan tingkat keburaman yang harus diterapkan pada bayangan.
* Nilai Keempat (opsional) : menentukan tingkat sebaran (spread) bayangan. Semakin besar nilai yang ditentukan, bayangan yang nampak pun semakin luas.
* Nilai Kelima : menentukan warna yang digunakan pada bayangan.

## Rounded Corner

CSS3 memperkenalkan kemampuan untuk membuat rounded corner atau sudut bundar pada box dengan menggunakan properti border-radius. Nilai dari properti ini merupakan tingkat lengkungan border dalam pixel.

1. .rounded {
2. border-radius: 10px;
3. }

Jika diterapkan, pinggiran border akan tampak seperti ini:

Kita bisa menetapkan nilai pada individu siku kotak dengan menggunakan properti yang terpisah, seperti ini:

1. .rounded {
2. border-top-right-radius: 5px;
3. border-bottom-right-radius: 10px;
4. border-bottom-left-radius: 5px;
5. border-top-left-radius: 10px;
6. }

Atau kita dapat menggunakan shorthand sehingga dapat menetapkan keempat nilainya dalam satu properti.

1. .rounded {
2. border-radius: 10px 5px 10px 5px;
3. }

Maka tiap sisi dari border memiliki ketajaman atau ketumpulan yang berbeda seperti gambar di bawah ini.

## Menerapkan Box Model pada Halaman Profil

Pada latihan sebelumnya, kita sudah menerapkan font yang didapatkan dari Google Font dengan properti font-family dan mengubah warna font dengan menggunakan properti color.  Nah setelah mempelajari komponen box seperti padding, border, dan margin saatnya kita terapkan pada proyek yang sudah kita buat sebelumnya. Mulai dari menetapkan ukuran gambar, membuat efek gaya kartu, dan membuat jumbotron.

Hingga pada akhir latihan ini kita dapat mempercantik tampilan website profil tampak seperti ini:

Sudah siap? Silakan buka kembali proyeknya dengan text editor yang Anda miliki.

### Menetapkan ukuran pada gambar

Sekarang saatnya kita mulai terapkan apa yang baru kita pelajari. Langkah pertama mari kita ubah dimensi dari gambar yang kita tampilkan. Pada berkas ***style.css***, buat rule baru untuk selector **.featured-images** dengan properti dan nilai sebagai berikut:

1. .featured-image {
2. width: 100%;
3. max-height: 300px;
4. }

Kemudian terapkan rule tersebut pada seluruh elemen gambar kecuali gambar logo kota Bandung yang terdapat di dalam elemen <aside>. Untuk menerapkannya kita tuliskan class atribut dengan nilai featured-image.

1. <img src="assets/image/history.jpg" class="featured-image" alt="sejarah">
3. ...
5. <img src="assets/image/geografis.jpg" class="featured-image" alt="geografis">
7. ...
9. <img src="assets/image/farm-house.jpg" class="featured-image" alt="farm house">
11. ...
13. <img src="assets/image/bosscha.jpg" class="featured-image" alt="bosscha">

Setelah menerapkannya maka tampilan gambar akan menjadi seperti ini:

Gambar yang ditampilkan jadi memenuhi lebar jendela browser. Namun karena kita menetapkan juga properti max-height, gambar yang ditampilkan menjadi tidak proporsional. Solusinya kita tambahkan properti object-fit: cover; dan object-position: center; pada .featured-image selector.

1. .featured-image {
2. width: 100%;
3. max-height: 300px;
4. object-fit: cover;
5. object-position: center;
6. }

Kedua properti tersebut membuat gambar yang ditampilkan dapat menyesuaikan lebar dan panjang dengan mempertahankan kualitas rasio aslinya. Tetapi untuk menyesuaikannya terdapat area gambar yang terpotong atau tidak ditampilkan, tetapi tampilan gambar akan tetap proporsional.

Selanjutnya kita atur juga ukuran gambar logo kota Bandung pada elemen <aside>. Agar logo tampak tidak terlalu besar kita atur properti width dengan nilai **200px**. Silakan buat rules baru dengan selector .profile img, kemudian tuliskan properti width dengan nilainya.

1. .profile img {
2. width: 200px;
3. }

Sehingga sekarang logo akan nampak lebih kecil.

### Menerapkan card style pada elemen <article>

Selanjutnya kita atur tampilan pada tiap elemen <article>, buat elemen tersebut bergaya card effect dengan menerapkan properti box-shadow dan border-radius sehingga hasilnya akan tampak seperti ini:

Pertama kita buat rule baru dengan selector .card, kemudian terapkan properti dan nilainya sebagai berikut:

1. .card {
2. box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
3. border-radius: 5px;
4. }

Selanjutnya terapkan rule tersebut pada seluruh elemen <article> yang ada. Untuk menerapkannya, tambahkan atribut class dengan nilai card.

1. <article id="sejarah" class="card">
3. ...
5. <article id="geografis" class="card">
7. ...
9. <article id="wisata" class="card">
10. ...
12. <aside>
13. <article class="profile card">
14. ....

Sekarang tiap sisi elemen <article> akan tampak dikelilingi oleh drop shadow dan memiliki ujung yang bulat pada tiap sisinya.

Jika dirasa drop shadow tak terlalu jelas, kita bisa mengubah nilainya. Agar mudah dalam menentukan nilai drop shadow yang kita inginkan, kita bisa gunakan tools yang tersedia secara online pada tautan berikut: <https://www.cssmatic.com/box-shadow>.

Selanjutnya kita perbaiki tepian konten pada card karena tampak terlalu sempit dari sisi cardnya. Alhasil, konten agak sulit dibaca. Kita bisa mengaturnya dengan properti margin dan padding. Silakan tambahkan properti dan nilai berikut pada rule .card:

1. .card {
2. box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
3. border-radius: 5px;
4. padding: 20px;
5. margin-top: 20px;
6. }

Sekarang konten di dalam card lebih proporsional dan mudah dibaca, bukan?

### Menerapkan Jumbotron dan Navigasi pada header

Selanjutnya kita akan mengatur header dengan membuat efek jumbotron dan memperbaiki tampilan navigasi menjadi nampak seperti ini:

Sebagai langkah awal, yuk kita buat rule baru dengan selector .jumbotron lalu terapkan properti dan nilainya seperti ini:

1. .jumbotron {
2. padding: 60px;
3. background-color: #00c8eb;
4. text-align: center;
5. font-size: 20px;
6. color: white;
7. }

Lalu pada berkas HTML, letakkan elemen <h1> dan <p> yang terdapat pada <header> dengan menggunakan <div> yang diberikan atribut class jumbotron. Sehingga struktur HTML pada elemen <header> tampak seperti ini:

1. <header>
2. <div class="jumbotron">
3. <h1>Bandung</h1>
4. <p>Kota metropolitan terbesar di Provinsi Jawa Barat, sekaligus menjadi ibu kota provinsi tersebut.</p>
5. </div>
6. <nav>
7. <ul>
8. <li><a href="#sejarah">Sejarah</a></li>
9. <li><a href="#geografis">Geografis</a></li>
10. <li><a href="#wisata">Wisata</a></li>
11. </ul>
12. </nav>
13. </header>

Sehingga sekarang header akan nampak dengan efek jumbotron.

Lanjut, kita akan memperbaiki tampilan navigasi agar terlihat cocok dengan header yang ditampilkan. Seperti biasa, buatlah rule baru dengan selector nav li. Selector tersebut akan mengatur elemen <li> yang menjadi turunan dari elemen <nav>, berikan properti dan nilai berikut pada selector tersebut:

1. nav li {
2. display: inline;
3. list-style-type: none;
4. margin-right: 20px;
5. }

Nilai inline pada properti display akan membuat elemen <li> menerapkan sifat inline, sehingga elemen akan ditampilkan tanpa membuat baris baru. Dan properti list-style-type dengan nilai none akan menghilangkan tanda titik yang dihasilkan tiap item pada list. Dengan begitu navigasi akan nampak seperti ini:

Selanjutnya beri background-color dan ubah warna teks pada navigasi agar nampak sesuai dengan header. Tuliskan rule baru dengan selector nav kemudian atur background-color dan padding dengan nilai berikut:

1. nav {
2. background-color: #00a2c6;
3. padding: 5px;
4. }

Kemudian untuk mengatur warna teks pada navigasi, kita tambahkan properti color pada selector nav a yang sudah kita buat pada latihan sebelumnya.

1. nav a {
2. font-size: 18px;
3. font-weight: 400;
4. text-decoration: none;
5. color: white;
6. }

Dengan begitu navigasi akan terlihat sesuai dengan header yang ditampilkan.

Pada langkah ini, website yang kita bangun sudah mulai tampak menarik bukan?

### Menyesuaikan Padding dan Margin pada body dan elemen <main>

Terakhir kita akan menyesuaikan margin dan padding yang ditampilkan pada body dan elemen <main>. Sehingga akan tampak seperti ini:

Pertama kita hilangkan nilai margin dan padding standar browser dengan menuliskan properti margin dan padding yang diberi nilai **0** pada body selector.

1. body {
2. font-family: 'Quicksand', sans-serif;
3. margin: 0;
4. padding: 0;
5. }

Dengan begitu layout yang ditampilkan akan memenuhi seluruh lebar dan tinggi dari jendela browser.

Tetapi konten yang terdapat di dalam elemen <main> perlu sedikit jarak agar konten yang ditampilkan sedikit nyaman untuk dilihat. Kita dapat atur jarak tersebut dengan menambahkan padding pada selector main. Silakan tuliskan rule baru dengan selector main dan beri nilai **20px** pada properti padding.

1. main {
2. padding: 20px;
3. }

Nah sekarang website pun jadi lebih nyaman dilihat.

Pada latihan selanjutnya kita akan memindahkan elemen aside sehingga berada di samping elemen content. Untuk melakukannya kita harus mengenal dulu apa itu positioning dan floating. Sampai jumpa di materi selanjutnya ya!

# Positioning

Kita sudah mengetahui cara mengubah posisi dari sebuah elemen dengan menggunakan margin. Namun ketika melakukannya, posisi elemen lain di sekitarnya akan terpengaruh. Lantas bagaimana jika kita ingin memindahkan tanpa mengganggu posisi elemen lainnya? Solusinya kita perlu mengubah positioning schema dengan menggunakan properti position dalam mengontrol letak elemen tersebut. Berikut nilai dari properti position yang ada pada CSS:

* Normal Flow / Static Flow : Merupakan default behaviour yang dimiliki elemen, di mana setiap elemen block akan ditampilkan dalam baris baru ketika dibuat. Sehingga setiap elemen block selalu muncul di bawah dari elemen block sebelumnya. Bahkan jika masih terdapat ruang kosong pada samping elemennya, mereka tidak akan nampak bersebelahan.
* Relative Positioning : Membuat elemen bisa berpindah posisi ke atas, kanan, bawah, maupun kiri dari posisi semula atau posisi seharusnya elemen tersebut berada. Perpindahan posisi ini tidak akan berpengaruh terhadap posisi elemen di sekitarnya karena ketika menggunakan relative positioning, elemen tersebut akan dipindahkan dari normal flow.
* Absolute Positioning : Sama seperti relative, elemen akan dipindahkan keluar dari normal flow sehingga kita dapat memindahkan posisi elemen ke atas, kanan, bawah, maupun kiri secara leluasa tanpa mengganggu elemen di sekitarnya. Namun posisinya relatif pada jendela browser dan induk elemen selama induk elemen juga berada di luar dari normal flow.
* Fixed Positioning : Merupakan absolute position namun posisinya selalu relatif pada jendela browser. Bahkan ketika pengguna scrolling pun, posisinya di layar akan tetap tak berubah.

Sebelum kita membahas satu persatu skema tersebut, mungkin kita perlu memahami lebih detail lagi apa sebenarnya normal flow atau biasa disebut “static flow” itu, dan mengapa untuk memindahkan posisi elemen kita perlu mengeluarkannya dari static flow.

Sebenarnya CSS memiliki dua buah *flow* yang bisa digunakan untuk menampilkan elemen, yakni *static* dan *non-static*. Agar mudah memahami perbedaan antar keduanya, kita gambarkan sebuah halaman website dengan tampilan tiga dimensi.

Bayangkan kita memiliki tiga buah elemen div berukuran 200px x 200px yang masing-masing memiliki warna yang berbeda.

1. .box {
2. width: 200px;
3. height: 200px;
4. }
6. .first {
7. background-color: #60d0a8;
8. }
10. .second {
11. background-color: #6495ed;
12. }
14. .third {
15. background-color: #ffa500;
16. }

Karena kita tidak mengatur properti position dari ketiga elemen tersebut, maka tiap elemen akan ditampilkan dengan static flow seperti ini:

Ketika kita ingin mengubah letak kotak biru (kotak kedua) dengan menggunakan margin-top: 20px; tentu akan berpengaruh pada posisi elemen di bawahnya.

1. .second {
2. background-color: #6495ed;
3. margin-top: 20px;
4. }

Pada ilustrasi di atas kita bisa lihat bahwa kotak yang berwarna oranye ikut bergeser ke bawah. Berbeda ketika kita menerapkan properti position yang dapat membuat elemen keluar dari static flow. Contohnya kita menerapkan properti position dengan nilai relatif.

1. .second {
2. background-color: #6495ed;
3. position: relative;
4. }

Pada tampilan browser mungkin tidak terdapat perbedaan apapun setelah menerapkan nilai relative pada atribut position. Namun sebenarnya elemen yang menerapkannya akan diangkat dari luar *static flow* seperti yang ditampilkan pada ilustrasi 3D. Sehingga elemen tersebut dapat leluasa berpindah posisi tanpa mempengaruhi elemen yang berada pada static flow.

Untuk mengubah posisi elemen yang berada di *non-static flow*, kita dapat menggunakan properti top, right, bottom, maupun left.

1. .second {
2. background-color: #6495ed;
3. position: relative;
4. top: 30px;
5. left: 10px;
6. }

Perlu kita ingat ya, bahwa properti top, left, right, dan bottom pada CSS hanya akan berpengaruh pada elemen yang menerapkan *non-static flow*(elemen yang menerapkan nilai relative, absolute, dan fixed pada properti position).

## Normal Flow

Dalam flow normal, setiap elemen block ditempatkan di bawah elemen sebelumnya. Karena ini merupakan cara standar browser memperlakukan elemen HTML, kita tidak perlu menetapkan nilai properti position ketika ingin membuat perilaku seperti ini tetapi secara sintaks perilaku ini ditetapkan dengan nilai static.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <title>Normal Flow</title>
5. <style>
6. h1 {
7. background-color: #aaaaaa;
8. padding: 30px;
9. }
10. p {
11. width: 450px;
12. }
13. </style>
14. </head>
15. <body>
16. <h1>Lorem ipsum dolor sit amet.</h1>
17. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium iusto non optio vel voluptas? Assumenda
18. blanditiis consequatur doloribus porro quasi similique vero voluptatibus. Adipisci, tenetur.</p>
19. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus commodi consectetur dolore eligendi, esse fuga
20. hic laboriosam nobis nostrum, nulla omnis optio pariatur repellendus reprehenderit saepe, sit soluta vel vero!</p>
21. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut corporis deleniti doloremque et excepturi
22. expedita labore libero, placeat, quaerat rerum ullam vitae voluptate. Accusantium et explicabo qui tenetur unde.
23. Amet aperiam doloremque enim, expedita id illo minima porro quod.</p>
24. </body>
25. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti ini:

Pada contoh di atas kita tidak menetapkan nilai lebar pada elemen heading, tapi bisa kita lihat bahwa elemen heading tersebut mencakup seluruh lebar ruang yang ada pada jendela browser.

Sedangkan elemen paragraf kita atur lebarnya dengan nilai 450px sehingga elemen tersebut tidak mencakup seluruh lebar ruang yang ada. Elemen paragraf selanjutnya tetap ditampilkan di bawahnya meskipun masih terdapat ruang yang cukup disampingnya.

Hal tersebut merupakan contoh dari gambaran bagaimana flow normal bekerja.

## Relative Positioning

Seperti yang kita ketahui sebelumnya, dengan menetapkan relative pada properti position, kita dapat melakukan perpindahan posisi elemen ke atas, kanan, bawah, maupun kiri. Perpindahan posisi yang dilakukan tidak akan berpengaruh terhadap posisi elemen di sekitarnya karena dengan  relative positioning, elemen tersebut akan dipindahkan dari normal flow. Untuk menggunakan relative positioning kita tetapkan nilai relative pada properti position.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. h1 {
6. background-color: #aaaaaa;
7. padding: 30px;
8. }
10. p {
11. width: 450px;
12. }
14. .relative {
15. position: relative;
16. top: 10px;
17. left: 100px;
18. }
19. </style>
20. </head>
21. <body>
22. <h1>Lorem ipsum dolor sit amet.</h1>
23. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium iusto non optio vel voluptas? Assumenda
24. blanditiis consequatur doloribus porro quasi similique vero voluptatibus. Adipisci, tenetur.</p>
25. <p class="relative">Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus commodi consectetur dolore eligendi, esse fuga hic laboriosam nobis nostrum, nulla omnis optio pariatur repellendus reprehenderit saepe, sit soluta vel vero!</p>
26. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut corporis deleniti doloremque et excepturi expedita labore libero, placeat, quaerat rerum ullam vitae voluptate. Accusantium et explicabo qui tenetur unde. Amet aperiam doloremque enim, expedita id illo minima porro quod.</p>
27. </body>
28. </html>

Berkas di atas jika kita buka melalui browser akan tampak seperti:

Pada contoh di atas, setelah menetapkan nilai position pada paragraf kedua, kita mengubah posisinya menjadi 10 pixel lebih bawah dan **100px** lebih ke kanan dari posisi yang seharusnya. Perlu kita ingat bahwa perpindahan posisi mengacu pada posisi elemen seharusnya berada.

## Absolute Positioning

Ketika properti position diberikan nilai absolute, akan berperilaku sama dengan relative. Elemen akan dikeluarkan dari normal flow sehingga jika elemen dipindahkan posisinya tidak akan berpengaruh pada elemen lain di sekitarnya.

Namun yang membedakannya adalah elemen ini benar-benar tidak dianggap ada oleh elemen pada normal flow. Akibatnya, lokasi awal elemen yang diberikan nilai absolute akan ditempati oleh elemen di bawahnya.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. h1 {
6. width: 250px;
7. }
9. p {
10. width: 450px;
11. }
13. .absolute {
14. position: absolute;
15. top: 0;
16. left: 500px;
17. }
18. </style>
19. </head>
20. <body>
21. <h1 class="absolute">Lorem ipsum dolor sit amet.</h1>
22. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium iusto non optio vel voluptas? Assumenda
23. blanditiis consequatur doloribus porro quasi similique vero voluptatibus. Adipisci, tenetur.</p>
24. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus commodi consectetur dolore eligendi, esse fuga
25. hic laboriosam nobis nostrum, nulla omnis optio pariatur repellendus reprehenderit saepe, sit soluta vel vero!</p>
26. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut corporis deleniti doloremque et excepturi
27. expedita labore libero, placeat, quaerat rerum ullam vitae voluptate. Accusantium et explicabo qui tenetur unde.
28. Amet aperiam doloremque enim, expedita id illo minima porro quod.</p>
29. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus aliquid asperiores blanditiis cumque eos ex
30. expedita facilis laborum maiores necessitatibus neque, officiis pariatur porro temporibus unde veritatis vitae?
31. Beatae dolores error ex fugiat molestias neque placeat quod ratione repellat veritatis! A culpa cumque, delectus
32. distinctio ducimus earum et ex quas quia quis tempore temporibus veritatis voluptates. Delectus doloremque mollitia
33. nobis?</p>
34. </body>
35. </html>

Jika berkas di atas dibuka pada browser, maka akan tampak seperti ini:

Pada contoh di atas kita bisa lihat bahwa ketika elemen heading menerapkan properti position dengan nilai absolute maka elemen di bawahnya (paragraf) akan menempati posisi yang sebelumnya ditempati oleh elemen heading tersebut.

Dan perpindahan posisi ketika menggunakan absolute akan relatif pada jendela browser. Kita bisa lihat bahwa elemen heading bergeser 500px dari kiri. Tapi selain relatif terhadap jendela browser, nilai ini juga dapat relatif pada elemen yang menaunginya (elemen induk) selama elemen tersebut juga berada di luar normal flow. Namun jika tidak, elemen akan menghiraukan elemen induknya tersebut. Contohnya:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .parent {
6. width: 500px;
7. margin: 0 auto;
8. border: 4px solid black;
9. padding: 20px;
10. }
11. h1 {
12. width: 250px;
13. }
14. .absolute {
15. position: absolute;
16. top: 0;
17. left: 50px;
18. }
19. </style>
20. </head>
21. <body>
22. <div class="parent">
23. <h1 class="absolute">Lorem ipsum dolor sit amet.</h1>
24. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium iusto non optio vel voluptas? Assumenda blanditiis consequatur doloribus porro quasi similique vero voluptatibus. Adipisci, tenetur.</p>
25. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus commodi consectetur dolore eligendi, esse fuga hic laboriosam nobis nostrum, nulla omnis optio pariatur repellendus reprehenderit saepe, sit soluta vel vero!</p>
26. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut corporis deleniti doloremque et excepturi expedita labore libero, placeat, quaerat rerum ullam vitae voluptate. Accusantium et explicabo qui tenetur unde. Amet aperiam doloremque enim, expedita id illo minima porro quod.</p>
27. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus aliquid asperiores blanditiis cumque eos ex expedita facilis laborum maiores necessitatibus neque, officiis pariatur porro temporibus unde veritatis vitae? Beatae dolores error ex fugiat molestias neque placeat quod ratione repellat veritatis! A culpa cumque, delectus distinctio ducimus earum et ex quas quia quis tempore temporibus veritatis voluptates. Delectus doloremque mollitianobis?</p>
28. </div>
29. </body>
30. </html>

Jika kita buka berkas di atas pada browser maka akan tampak seperti ini:

Pada contoh kode di atas kita bisa lihat pada .parent kita tidak menetapkan properti position, sehingga elemen yang menerapkannya berada pada flow normal. Sehingga elemen heading akan menghiraukan elemen induknya.

Berbeda ketika .parent menerapkan properti position dengan nilai yang dapat mengeluarkannya dari normal flow, contohnya nilai relative.

1. position: relative;

Dengan begitu posisi elemen heading akan relatif terhadap .parent.

## Fixed Positioning

Fixed positioning merupakan absolute position namun posisinya selalu relatif pada jendela browser (meskipun diletakan di dalam induk elemen diluar dari flow normal). Bahkan ketika pengguna melakukan scrolling posisinya akan tetap nampak pada posisinya di layar.

1. <!DOCTYPE html>
2. <html lang="en">
3. <head>
4. <style>
5. body {
6. min-height: 200vh;
7. }
9. h3 {
10. padding: 10px;
11. background-color: #AAAAAA;
12. margin: 0;
13. width: 100%;
14. }
16. .fixed {
17. position: fixed;
18. top: 0;
19. left: 0;
20. }
21. </style>
22. </head>
23. <body>
24. <h3 class="fixed">Lorem ipsum dolor sit amet.</h3>
25. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium iusto non optio vel voluptas? Assumenda
26. blanditiis consequatur doloribus porro quasi similique vero voluptatibus. Adipisci, tenetur.</p>
27. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus commodi consectetur dolore eligendi, esse fuga
28. hic laboriosam nobis nostrum, nulla omnis optio pariatur repellendus reprehenderit saepe, sit soluta vel vero!</p>
29. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Adipisci aut corporis deleniti doloremque et excepturi
30. expedita labore libero, placeat, quaerat rerum ullam vitae voluptate. Accusantium et explicabo qui tenetur unde. Amet
31. aperiam doloremque enim, expedita id illo minima porro quod.</p>
32. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusamus aliquid asperiores blanditiis cumque eos ex
33. expedita facilis laborum maiores necessitatibus neque, officiis pariatur porro temporibus unde veritatis vitae? Beatae
34. dolores error ex fugiat molestias neque placeat quod ratione repellat veritatis! A culpa cumque, delectus distinctio
35. ducimus earum et ex quas quia quis tempore temporibus veritatis voluptates. Delectus doloremque mollitia nobis?</p>
36. </body>
37. </html>

Berkas di atas jika kita buka pada browser, tampilannya akan tampak seperti berikut.

Pada contoh di atas, elemen heading diposisikan di sudut kiri atas jendela browser dengan properti:

1. top: 0;
2. left: 0;

Ketika pengguna melakukan scrolling pada halaman, elemen heading tetap berada pada posisi yang sama dan paragraf yang ditampilkan nampak di belakang elemen heading tersebut.

**Floating**

Sederhananya properti float dapat membuat elemen berada pada sebelah kanan atau kiri. Saat diterapkan pada elemen inline, properti float juga memungkinkan elemen di sekitarnya mengelilingi elemen tersebut (wrap).

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. padding: 10px;
9. }
11. img {
12. float: left;
13. margin: 10px;
14. }
15. </style>
16. </head>
17. <body>
18. <div class="container">
19. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Amet asperiores commodi corporis doloribus eum ipsum obcaecati pariatur quam quo, voluptatum? A, assumenda atque delectus dolore nam natus neque quisquam repudiandae?</p>
21. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. At deserunt illum inventore numquam officia recusandae, sit! Ab asperiores delectus dolor, eius est explicabo impedit ipsa ipsum itaque mollitia neque nihil nisi officia praesentium sint! Accusantium libero obcaecati provident qui? Error!</p>
22. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aliquam debitis dolorem eaque eius eos error eum ex facilis fuga ipsa iste iusto mollitia nam natus necessitatibus praesentium quam qui quod, rerum sed sequi sint voluptates? Atque enim, ex facere fugiat ipsum mollitia quos sunt. At, autem commodi dolorum eos, est expedita, incidunt molestias nobis nostrum officia porro reiciendis saepe voluptatum.</p>
23. </div>
24. </body>
25. </html>

Berkas di atas, jika kita buka melalui browser akan tampak seperti gambar ini:

Selain dalam membuat *text wrapping*. Float juga merupakan salah satu teknik dalam membuat *layout* website. Teknik ini masih banyak digunakan oleh developer karena sangat mudah dipahami dan tergolong mudah untuk *multiple column* seperti gambar berikut

Hal ini dapat dilakukan dengan mudah menggunakan properti float, karena nilai dari properti ini adalah left dan right. Nilai tersebut menunjukan posisi elemen harus diletakkan. Contoh penerapannya seperti berikut ini:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. /\* digunakan untuk menghapus seluruh padding dan margin standar yang diberikan browser pada elemen \*/
7. margin: 0;
8. padding: 0;
10. /\* Menggunakan border-box dalam perhitungan dimensi box-nya \*/
11. box-sizing: border-box;
12. }
14. .container {
15. width: 800px;
16. height: 400px;
17. border: 1px solid black;
18. margin: 0 auto;
19. }
21. .left-content {
22. text-align: center;
23. line-height: 400px;
24. width: 30%;
25. height: 100%;
26. background-color: #00c7ed;
28. /\* digunakan untuk memindahkan posisi elemen ke sisi kiri container \*/
29. float: left;
30. }
32. .right-content {
33. text-align: center;
34. line-height: 400px;
35. width: 70%;
36. height: 100%;
37. background-color: #60d0a8;
39. /\* digunakan untuk memindahkan posisi elemen ke sisi kanan container \*/
40. float: right;
41. }
42. </style>
43. </head>
44. <body>
45. <div class="container">
46. <div class="left-content">
47. <h3>Left Content</h3>
48. </div>
49. <div class="right-content">
50. <h3>Right Content</h3>
51. </div>
52. </div>
53. </body>
54. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti berikut:

Dari contoh di atas, kita bisa melihat pada selector .left-content dan .right-content ditetapkan properti float dengan nilai left dan right, sehingga elemen tersebut dapat diposisikan secara bersebelahan.

Ketika menggunakan properti float jangan lupa menentukan ukuran lebar dari propertinya, karena nilai dari lebar menentukan seberapa banyak ruang yang dicakup oleh elemen. Pada contoh di atas kita menetapkan nilai 30% untuk konten kiri dan 70% untuk konten sebelah kanan.

Lantas jika nilainya hanya left dan right, bagaimana jika ada lebih dari dua kolom dalam menyusun layout? Hal tersebut sangat mudah dilakukan. Kita hanya perlu menetapkan nilai left pada ketiga kolomnya, lantas kolom akan tampil bersebelahan sesuai urutan penulisan elemen.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. /\* digunakan untuk menghapus seluruh padding dan margin standar yang diberikan browser pada elemen \*/
7. margin: 0;
8. padding: 0;
10. /\* Menggunakan border-box dalam perhitungan dimensi box-nya \*/
11. box-sizing: border-box;
12. }
14. .container {
15. width: 800px;
16. height: 400px;
17. border: 1px solid black;
18. margin: 0 auto;
19. }
21. .left-content {
22. text-align: center;
23. line-height: 400px;
24. width: 33.3%;
25. height: 100%;
26. background-color: #00c7ed;
27. float: left;
28. }
30. .center-content {
31. text-align: center;
32. line-height: 400px;
33. width: 33.3%;
34. height: 100%;
35. background-color: #d0b541;
36. float: left;
37. }
39. .right-content {
40. text-align: center;
41. line-height: 400px;
42. width: 33.3%;
43. height: 100%;
44. background-color: #60d0a8;
45. float: left;
46. }
47. </style>
48. </head>
49. <body>
50. <div class="container">
51. <div class="left-content">
52. <h3>Left Content</h3>
53. </div>
54. <div class="center-content">
55. <h3>Center Content</h3>
56. </div>
57. <div class="right-content">
58. <h3>Right Content</h3>
59. </div>
60. </div>
61. </body>
62. </html>

Maka akan menghasilkan tampilan seperti gambar berikut:

# Floating

Sederhananya properti float dapat membuat elemen berada pada sebelah kanan atau kiri. Saat diterapkan pada elemen inline, properti float juga memungkinkan elemen di sekitarnya mengelilingi elemen tersebut (wrap).

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. padding: 10px;
9. }
11. img {
12. float: left;
13. margin: 10px;
14. }
15. </style>
16. </head>
17. <body>
18. <div class="container">
19. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Amet asperiores commodi corporis doloribus eum ipsum obcaecati pariatur quam quo, voluptatum? A, assumenda atque delectus dolore nam natus neque quisquam repudiandae?</p>
21. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. At deserunt illum inventore numquam officia recusandae, sit! Ab asperiores delectus dolor, eius est explicabo impedit ipsa ipsum itaque mollitia neque nihil nisi officia praesentium sint! Accusantium libero obcaecati provident qui? Error!</p>
22. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aliquam debitis dolorem eaque eius eos error eum ex facilis fuga ipsa iste iusto mollitia nam natus necessitatibus praesentium quam qui quod, rerum sed sequi sint voluptates? Atque enim, ex facere fugiat ipsum mollitia quos sunt. At, autem commodi dolorum eos, est expedita, incidunt molestias nobis nostrum officia porro reiciendis saepe voluptatum.</p>
23. </div>
24. </body>
25. </html>

Berkas di atas, jika kita buka melalui browser akan tampak seperti gambar ini:

Selain dalam membuat text wrapping. Float juga merupakan salah satu teknik dalam membuat layout website. Teknik ini masih banyak digunakan oleh developer karena sangat mudah dipahami dan tergolong mudah untuk multiple column seperti gambar berikut

Hal ini dapat dilakukan dengan mudah menggunakan properti float, karena nilai dari properti ini adalah left dan right. Nilai tersebut menunjukan posisi elemen harus diletakkan. Contoh penerapannya seperti berikut ini:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. /\* digunakan untuk menghapus seluruh padding dan margin standar yang diberikan browser pada elemen \*/
7. margin: 0;
8. padding: 0;
10. /\* Menggunakan border-box dalam perhitungan dimensi box-nya \*/
11. box-sizing: border-box;
12. }
14. .container {
15. width: 800px;
16. height: 400px;
17. border: 1px solid black;
18. margin: 0 auto;
19. }
21. .left-content {
22. text-align: center;
23. line-height: 400px;
24. width: 30%;
25. height: 100%;
26. background-color: #00c7ed;
28. /\* digunakan untuk memindahkan posisi elemen ke sisi kiri container \*/
29. float: left;
30. }
32. .right-content {
33. text-align: center;
34. line-height: 400px;
35. width: 70%;
36. height: 100%;
37. background-color: #60d0a8;
39. /\* digunakan untuk memindahkan posisi elemen ke sisi kanan container \*/
40. float: right;
41. }
42. </style>
43. </head>
44. <body>
45. <div class="container">
46. <div class="left-content">
47. <h3>Left Content</h3>
48. </div>
49. <div class="right-content">
50. <h3>Right Content</h3>
51. </div>
52. </div>
53. </body>
54. </html>

Berkas di atas jika kita buka pada browser akan tampak seperti berikut:

Dari contoh di atas, kita bisa melihat pada selector .left-content dan .right-content ditetapkan properti float dengan nilai left dan right, sehingga elemen tersebut dapat diposisikan secara bersebelahan.

Ketika menggunakan properti float jangan lupa menentukan ukuran lebar dari propertinya, karena nilai dari lebar menentukan seberapa banyak ruang yang dicakup oleh elemen. Pada contoh di atas kita menetapkan nilai 30% untuk konten kiri dan 70% untuk konten sebelah kanan.

Lantas jika nilainya hanya left dan right, bagaimana jika ada lebih dari dua kolom dalam menyusun layout? Hal tersebut sangat mudah dilakukan. Kita hanya perlu menetapkan nilai left pada ketiga kolomnya, lantas kolom akan tampil bersebelahan sesuai urutan penulisan elemen.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. /\* digunakan untuk menghapus seluruh padding dan margin standar yang diberikan browser pada elemen \*/
7. margin: 0;
8. padding: 0;
10. /\* Menggunakan border-box dalam perhitungan dimensi box-nya \*/
11. box-sizing: border-box;
12. }
14. .container {
15. width: 800px;
16. height: 400px;
17. border: 1px solid black;
18. margin: 0 auto;
19. }
21. .left-content {
22. text-align: center;
23. line-height: 400px;
24. width: 33.3%;
25. height: 100%;
26. background-color: #00c7ed;
27. float: left;
28. }
30. .center-content {
31. text-align: center;
32. line-height: 400px;
33. width: 33.3%;
34. height: 100%;
35. background-color: #d0b541;
36. float: left;
37. }
39. .right-content {
40. text-align: center;
41. line-height: 400px;
42. width: 33.3%;
43. height: 100%;
44. background-color: #60d0a8;
45. float: left;
46. }
47. </style>
48. </head>
49. <body>
50. <div class="container">
51. <div class="left-content">
52. <h3>Left Content</h3>
53. </div>
54. <div class="center-content">
55. <h3>Center Content</h3>
56. </div>
57. <div class="right-content">
58. <h3>Right Content</h3>
59. </div>
60. </div>
61. </body>
62. </html>

Maka akan menghasilkan tampilan seperti gambar berikut:

## Permasalahan penerapan float

Properti float terlihat sangat mudah untuk digunakan, baik dalam text wrapping maupun dalam penyusunan layout. Tetapi bukan berarti ketika menggunakan properti ini, tiada efek samping atau masalah yang ditimbulkan. Apa itu?

Jika sebuah elemen induk hanya memiliki satu elemen dengan menerapkan properti float, ia akan memiliki nilai tinggi **0px**. Contohnya seperti berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. }
10. img {
11. float: left;
12. margin: 10px;
13. }
14. </style>
15. </head>
16. <body>
17. <div class="container">
18. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
19. </div>
20. </body>
21. </html>

Jika dijalankan pada browser akan menghasilkan tampilan seperti:

Pada contoh tersebut kita bisa melihat bahwa elemen gambar berada di dalam sebuah container div. Tetapi container tersebut tidak memiliki tinggi senilai gambar yang ditampilkan. Mengapa bisa demikian? Untuk lebih jelasnya kita coba tambahkan paragraf di dalam container tersebut tanpa menggunakan properti float.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. }
10. img {
11. float: left;
12. margin: 10px;
13. }
14. </style>
15. </head>
16. <body>
17. <div class="container">
18. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
19. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium adipisci architecto aspernatur dolorem
20. doloremque eum excepturi fuga hic, molestias obcaecati officia perferendis perspiciatis qui saepe ut velit
21. veniam vitae voluptatum.</p>
22. </div>
23. </body>
24. </html>

Hasilnya:

Hal ini menunjukkan bahwa elemen yang menerapkan float “tidak dianggap ada” oleh induk elemen yang menaunginya karena jika menggunakan float, elemen akan dikeluarkan dari normal flow.

Bagaimana Anda menanganinya?

Terdapat dua cara yang akan kita bahas, yakni dengan pertama, menggunakan properti clear dan kedua, menetapkan nilai overflow: auto pada container.

### Clear Property

Cara yang pertama adalah dengan menggunakan properti clear. Properti clear memang dibuat untuk menghilangkan sifat float, dengan demikian elemen yang menerapkan properti float akan kembali “dianggap ada”. Namun untuk menerapkan properti ini kita perlu membuat sebuah elemen kosong (biasanya menggunakan div tanpa konten) yang menerapkan properti clear.

Nilai dari properti clear merupakan nilai yang digunakan pada properti float karena properti clear akan menghapus sifat float sesuai dengan nilai yang ditetapkan. Contohnya seperti ini:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. }
10. img {
11. float: left;
12. margin: 10px;
13. }
14. .clear {
15. clear: left;
16. }
17. </style>
18. </head>
19. <body>
20. <div class="container">
21. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
22. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium adipisci architecto aspernatur dolorem
23. doloremque eum excepturi fuga hic, molestias obcaecati officia perferendis perspiciatis qui saepe ut velit
24. veniam vitae voluptatum.</p>
25. <div class="clear"></div>
26. </div>
27. </body>
28. </html>

Maka hasilnya akan seperti ini:

Jika kita mencoba mengubah clear menjadi clear: right;, maka akan kembali seperti hasil sebelum menerapkannya.

Lalu bagaimana jika terdapat dua elemen yang menerapkan nilai float berbeda (left dan right)?  Terdapat satu nilai lagi yang dapat menghilangkan kedua sifat float tersebut, yakni nilai both. Gunakanlah nilai tersebut jika kita ingin sepenuhnya menghilangkan sifat float pada elemen.

### overflow: auto;

Cara yang kedua adalah dengan menetapkan properti overflow: auto; pada elemen yang menampung elemen float. Sebagian developer lebih memilih menggunakan cara ini karena lebih praktis dan tidak perlu membuat elemen baru ketika menggunakannya.

Untuk menerapkannya sangatlah simpel. Cukup tambahkan properti overflow dengan nilai auto pada container.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. width: 800px;
7. border: 4px solid black;
8. overflow: auto;
9. }
11. img {
12. float: left;
13. margin: 10px;
14. }
15. </style>
16. </head>
17. <body>
18. <div class="container">
19. <img src="https://i.imgur.com/cs2BJzw.jpg" width="200px" alt="dicoding">
20. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Accusantium adipisci architecto aspernatur dolorem
21. doloremque eum excepturi fuga hic, molestias obcaecati officia perferendis perspiciatis qui saepe ut velit
22. veniam vitae voluptatum.</p>
23. </div>
24. </body>
25. </html>

Maka hasilnya akan sama seperti yang dilakukan pada cara pertama..

## Menerapkan Positioning dan Float pada Halaman Profil

Seperti yang sudah disinggung pada latihan sebelumnya, latihan kali ini kita akan memindahkan elemen aside agar berada di pinggir halaman halaman detail. Sehingga tampilan website akan terlihat seperti ini:

Namun tidak hanya sampai disitu, latihan kali ini akan sedikit lebih panjang, karena kita akan menerapkan responsibilitas tampilan pada halaman profil dengan menggunakan teknik media query. Seperti apa tekniknya? Mari kita ketahui bersama.

Yang pasti pada akhirnya kita dapat membuat halaman aplikasi tampak seperti berikut:

### Menetapkan float pada #content dan aside

Seperti yang sudah dipelajari, properti float berfungsi untuk memindahkan suatu elemen untuk berada sebelah di kiri atau di sebelah kanan, dengan menggunakan nilai right atau left. Ketika menggunakan properti tersebut kita perlu menetapkan lebar pada elemen yang akan diterapkan properti float sehingga kita tahu berapa banyak cakupan lebar yang disediakan untuk elemen tersebut.

Untuk langkah awal, mari kita buat rule baru dengan selector #content dan tuliskan nilai left pada properti float.

1. #content {
2. float: left;
3. }

Jangan lupa untuk menetapkan nilai lebar juga. Pada kasus kali ini kita gunakan nilai 75% untuk menetapkan lebar #content.

1. #content {
2. float: left;
3. width: 75%;
4. }

Ketika kita buka hasilnya pada browser, tampilan akan jadi berantakan seperti ini:

Tenang, ini normal. Dikarenakan kita belum menetapkan nilai float pada elemen <aside>. Elemen <aside> dan <footer> akan berpindah menempati ruang yang sebelumnya digunakan oleh #content (collapse). Hal tersebut telah kita pelajari pada materi sebelumnya bahwa elemen yang diberikan properti float akan dikeluarkan dari normal flow.

Langkah selanjutnya, kita terapkan juga properti float dengan nilai right pada elemen <aside>.

1. aside {
2. float: right;
3. }

Jangan lupa untuk menetapkan lebar pada elemennya. Karena sebelumnya elemen content sudah menerapkan 75% lebar dari parent, berarti elemen aside hanya dapat memiliki 25% ruang yang tersedia untuk ditempati. Silakan tetapkan nilai 25% pada properti width untuk elemen aside.

1. aside {
2. float: right;
3. width: 25%;
4. }

Jika kita coba membuka berkas HTML dengan apa yang sudah kita terapkan, maka tampilan akan jadi lebih mengerikan seperti ini:

Hal tersebut disebabkan nilai elemen <main> sekarang memiliki tinggi 0px. Di mana #content dan elemen <aside> dianggap tidak ada, sehingga elemen <footer> akan naik mengisi ruang yang kosong. Agar #content dan elemen <aside> dianggap ada oleh <main> (sebagai induk elemennya) kita perlu tetapkan nilai overflow: auto; pada main.

Silakan tambahkan properti overflow dengan nilai auto pada selector main yang sudah kita buat sebelumnya.

1. main {
2. padding: 20px;
3. overflow: auto;
4. }

Setelah menerapkan overflow: auto; pada main, tampilan website sudah lebih baik dan elemen footer sudah berada pada posisi yang seharusnya.

Namun kita perlu memberikan sedikit jarak antara #content dan aside agar tidak nampak berhimpitan. Tambahkanlah properti padding-left pada elemen <aside> agar konten yang ditampilkan lebih menjorok ke kiri.

Silakan tambahkan properti padding-left dengan nilai 20px pada selector aside.

1. aside {
2. float: right;
3. width: 25%;
4. padding-left: 20px;
5. }

Setelah menerapkannya, buka kembali hasilnya pada browser. Voila, ini hasilnya:

Ups, elemen <aside> sekarang berada di bawah #content karena dengan menambahkan padding pada <aside>, lebar elemen sudah bukan 25% lagi.

Untuk mengatasinya kita bisa ubah tipe pengukuran untuk dimensi kotaknya. Masih ingat kan materi tentang box-sizing? Dengan menerapkan nilai border-box pada properti box-sizing, kita dapat menentukan dimensi kotak dengan tepat, tanpa dipengaruhi oleh padding dan border.

Mari kita tetapkan pengukuran box-sizing untuk seluruh elemen HTML dengan menggunakan selector \* seperti berikut:

1. \* {
2. box-sizing: border-box;
3. }

Setelah menetapkan rule tersebut, maka tampilan website akan kembali normal.

### Menambahkan meta tag viewport

Setelah berhasil membuat layout dengan menggunakan float, apakah terpikir bagaimana tampilan yang dihasilkan pada mobile device? Tampilan website kita terlihat baik ketika dibuka menggunakan laptop atau desktop tetapi belum tentu akan sama pada mobile device.

Pada browser Google Chrome dan Mozilla Firefox terdapat fitur inspection. Fitur ini sangat bermanfaat bagi developer web dalam membantu pengembangan website. Jika ingin mempelajari lebih lanjut mengenai fitur ini, Anda bisa melihat detailnya pada dokumentasi yang tersedia pada tautan berikut:

* [Chrome page inspector (chrome dev tools)](https://developers.google.com/web/tools/chrome-devtools)
* [Mozilla Firefox page inspector](https://developer.mozilla.org/id/docs/Tools/Page_Inspector)

Yang jelas pada inspector tersebut terdapat fitur di mana kita dapat mensimulasikan halaman website dalam tampilan mobile device atau tablet. Untuk membuka fitur inspector kita bisa gunakan shortcut ctrl + shift + i pada browser, atau dengan melakukan klik kanan -> inspect / inspect page.

Fitur Inspection pada Google Chrome  
  
Fitur Inspection pada Mozilla Firefox

Kedua gambar di atas menunjukan fitur dari inspection pada Google Chrome dan Mozilla Firefox. Untuk mensimulasikan tampilan pada mobile device, kita dapat menekan icon yang dilingkari pada gambar di atas.

Dengan begitu, tampilannya akan muncul seperti berikut.

Di sini kita bisa memilih peranti apa yang akan digunakan untuk mensimulasikan halaman, untuk melihat tampilan dalam mobile devices, kita bisa memilih salah satu opsi handphone yang tersedia. Contohnya Pixel 2 XL.

Lalu setelah memilih tampilan pada mobile devices, apakah tampilan website sekarang sudah nampak baik? Tentu tidak, karena tampilan pada simulasi device di browser Google Chrome masih berantakan tampak seperti ini.

Untuk mengatasinya, kita perlu menetapkan tag viewport agar dapat responsif pada peranti mobile.

Viewport merupakan area yang dapat dilihat oleh pengguna kita pada halaman website. Ukuran viewport bervariasi berdasarkan device-nya. Ukuran viewport pada sebuah peranti mobile, lebih kecil dibandingkan dengan layar komputer.

Sebelum adanya tablet ataupun handphone, halaman website didesain hanya untuk ukuran layar komputer. Dengan begitu, banyak sekali website yang menerapkan tampilan dan ukuran yang static. Maka ketika halaman tersebut diakses melalui handphone atau tablet, ukuran akan terlalu besar untuk ditampilkan. Untuk mengatasi permasalahan tersebut kita perlu mengatur sebuah viewport pada halaman website kita.

Mengatur viewport dapat melalui sebuah tag <meta> yang disisipkan pada head berkas html. Silakan buka berkas index.html kemudian tambahkan script berikut di dalam <head>.

1. <meta name="viewport" content="width=device-width, initial-scale=1">

Sehingga sekarang tampilan pada mobile device menjadi seperti ini:

Tampilan pada mobile device setelah ditambahkan meta tag viewport

### Menetapkan rules pada Ukuran Layar Tertentu

Walaupun sudah menetapkan meta tag viewport, tampilan pada mobile device belum baik karena informasi yang ditampilkan pada elemen <aside> sulit terbaca.

Untuk mengatasi hal ini, CSS menyediakan sebuah fitur yang dapat kita gunakan untuk menentukan rule hanya pada ukuran viewport tertentu. Fitur tersebut dinamakan CSS Media Queries. Berikut contoh penulisannya:

1. /\* Rule yang dituliskan dalam block @media di bawah akan diterapkan pada device yang memiliki ukuran viewport di bawah 992px \*/
2. @media screen and (max-width: 992px) {
3. /\* ..... \*/
4. }
6. /\* Rule yang dituliskan dalam block @media di bawah akan diterapkan pada device yang memiliki ukuran viewport di bawah 600px \*/
7. @media screen and (max-width: 600px) {
8. /\* ..... \*/
9. }

Canggih, bukan? Setelah mengetahui fitur media queries, kita dapat memperbaiki halaman profil dengan memindahkan elemen <aside> di bawah dari elemen #content. Pemindahan ini hanya ada pada device yang memiliki ukuran layar kecil sehingga <aside> dapat menyesuaikan kembali lebarnya.

Untuk melakukannya, kita tetapkan elemen <aside> dan #content agar memiliki lebar 100% (memenuhi layar). CSS properti ini akan diterapkan jika ukuran lebar viewport kurang dari 1000px (maksimal 999px). Nilai ini didefinisikan pada media queries yang dapat disesuaikan sesuai kebutuhan. Silakan tuliskan kode berikut pada **style.css**:

1. @media screen and (max-width: 1000px) {
2. #content,
3. aside {
4. width: 100%;
5. padding: 0;
6. }
7. }

Dengan menerapkan media query, sekarang halaman profil dapat menyesuaikan tampilan ketika diakses oleh peranti mobile.

Selamat! Sampai dengan materi ini kita sudah dapat membuat sebuah website dengan tampilan yang menarik, Ingin paham lebih jauh tentang konsep responsive pada website? Anda bisa membaca artikel yang ada pada blog kami atau bisa membaca dokumentasi yang diberikan oleh Google mengenai ini.

* [Teknik Dasar Tampilan Responsif pada Website](https://www.dicoding.com/blog/dasar-tampilan-responsif-pada-website/)
* [Responsive Web Design Basics](https://developers.google.com/web/fundamentals/design-and-ux/responsive)

### Membuat Sticky Navigation

Ada satu fitur yang mungkin perlu diterapkan lagi pada halaman profil yang kita buat. Pada halaman profil sudah terdapat sebuah navigasi yang berfungsi untuk mengarahkan pengguna langsung pada informasi yang dipilih. Tetapi navigasi tersebut agak sulit diakses ketika pengguna melakukan scrolling pada halaman website. Pasalnya, navigasi tersebut selalu berada di bawah jumbotron website. Akibatnya, untuk mengakses navigasi yang dimaksud, pengguna perlu kembali melakukan scrolling ke atas hingga navigasi kembali terlihat.

Dari permasalahan di atas, sepertinya kita perlu meletakan navigasi yang menempel pada jendela browser, sehingga ketika user melakukan scrolling pada halaman website, navigasi tetap dapat dicangkup oleh pengguna untuk digunakan.

Terdapat satu nilai properti position yang belum dibahas pada pembahasan positioning, yaitu sticky positioning yang merupakan nilai kombinasi dari relative positioning dan fixed positioning.

Dengan menggunakan nilai sticky pada properti position, membuat navigasi dapat melekat pada jendela browser (layaknya fixed position) tetapi posisi awalnya tetap menempati posisi di mana ia seharusnya berada (layaknya relative position). Sehingga tampilan navigasi dapat memiliki sifat seperti ini:

Untuk menerapkannya cukup mudah, pertama kita tambahkan properti position dengan nilai sticky dan properti top dengan nilai 0 pada elemen <nav>.

1. nav {
2. background-color: #00a2c6;
3. padding: 5px;
4. position: sticky;
5. top: 0;
6. }

Nilai top diperlukan agar ketika pengguna melakukan scrolling, elemen navigasi tetap berada atau menyangkut di atas jendela browser.

Namun ini masih belum cukup, navigasi bar masih belum tertahan ketika pengguna melakukan scrolling halaman website. Bagaimana agar navigasi dapat menempel pada seluruh halaman? Pada latihan sebelumnya, kita telah membuat elemen <header> sebagai induk dari elemen <nav> dan elemen <div> yang memiliki atribut class dengan nilai jumbotron. Elemen <header> tersebut perlu diterapkan properti display dengan nilai inline. Silakan tuliskan rule berikut pada berkas **style.css**.

1. header {
2. display: inline;
3. }

Setelah menerapkannya, seharusnya kini navigasi pada halaman profil akan tampak menyangkut di atas jendela browser saat pengguna melakukan scrolling.

## Memperbaiki CSS

Terdapat banyak cara dan best practice dalam memperbaiki CSS yang telah Anda tulis. Salah satu cara yang mudah dan cepat dilakukan adalah dengan menggunakan tools untuk otomatis mencari kesalahan pada penulisan.

### Mencari Kesalahan CSS

Salah satu cara yang paling cepat untuk memastikan bahwa CSS yang telah Anda tulis itu valid dan dapat berjalan adalah menggunakan berbagai tools online yang tersedia secara gratis. Pengecekan cukup dengan cara copy-paste kode CSS, upload berkas CSS, atau menulis alamat situs yang ingin diperiksa.

Berikut beberapa opsi gratis yang tersedia.

* <https://jigsaw.w3.org/css-validator/>
* <https://codebeautify.org/cssvalidate>
* <http://csslint.net/>

### Kecilkan Ukuran CSS

Ukuran CSS sangat penting untuk dapat dihemat karena akan mempengaruhi seberapa cepat situs Anda dapat dimuat pada browser. Cara untuk menghemat ukuran berkas CSS adalah dengan menghapus spasi atau karakter yang tidak diperlukan dari kode. Berikut contoh spasi atau karakter yang dapat dihapus:

* New line characters (Baris baru)
* Whitespace characters (Spasi)
* Block delimiters (Pembatas block)
* CSS comments (Komentar CSS)

Perbaikan di atas dapat dilakukan secara manual namun Anda juga dapat menggunakan tools online yang dapat memperbaiki secara otomatis. Berikut beberapa opsi yang tersedia.

* <https://cssnano.co/playground/>
* <https://cssminifier.com/>
* <https://www.minifier.org/>

## Responsive Layout menggunakan Flexbox

Sampai tahap ini kita sudah memahami dua pilar dalam membentuk sebuah website, yakni HTML dan CSS, dengan membuat sebuah halaman website sederhana dalam menampilkan informasi mengenai kota Bandung.

Dalam pengembangannya kita sudah memahami perbedaan block element dan inline element. Kita juga sudah terapkan konsep semantic HTML sehingga kerangka HTML yang dibuat, sesuai dengan fungsi elemennya.

Lalu kita juga sudah memahami bagaimana cara kerja CSS dalam mendekorasi elemen HTML, menetapkan sebuah selector, membangun layout dengan menggunakan float, dan mengetahui prinsip pendekatan layout yang responsif dengan teknik media query.

Selanjutnya kita akan belajar mengenai pilar ketiga dalam pengembangan website, yaitu JavaScript. Dalam mempelajarinya kita akan coba menyelesaikan sebuah studi kasus, yakni membuat kalkulator berbasis website. Targetnya pada akhir kelas ini kita dapat membangun kalkulator sederhana yang tampak gambar di bawah ini:

Menarik bukan? Tentunya sebelum kita membahas JavaScript lebih jauh, kita akan belajar bagaimana cara membuat layout kalkulator.

Dalam proses pembuatanya, kita akan mencoba teknik baru dalam penyusunan layout, yakni menggunakan Flexbox. Teknik ini hadir pada CSS3 dan menjadi salah satu pendekatan populer dalam penyusunan layout yang responsif. Nantinya kalkulator yang kita kembangkan ini dapat digunakan dengan baik pada mobile device.

**Flexible Box Model**

Jadi apa sebenarnya itu flexbox atau *flexible* *box model*? Pada dasarnya flexbox merupakan mode layout yang mudah dan praktis untuk mengatur elemen dengan menggunakan sebuah container.

Sebelumnya kita sudah menggunakan teknik float dalam penyusunan layout. Float memang cukup mudah dipahami tetapi tak cukup praktis untuk digunakan. Pasalnya, kita memerlukan pengukuran dimensi yang tepat agar elemen berada di posisi yang diinginkan.

Dengan Flexbox kita dapat mengatur ukuran secara otomatis dan mampu beradaptasi dengan ukuran container-nya. Dengan kemampuan flexibelnya, tampilan layout yang disusun menggunakan flexbox menjadi mobile-friendly. Selain itu, flexbox dapat dikombinasikan dengan *media query*, sehingga ia dapat lebih optimal lagi digunakan pada peranti mobile.

Berikut beberapa konsep dari flexbox yang perlu kita ketahui:

* Dapat mengubah ukuran dimensi elemen dengan menyesuaikan ukuran yang cocok bagi ruang kosong yang ada pada container-nya.
* Flexbox is *directional agnostic*. ini berbeda dengan konsep **block** model di mana elemen selalu ditampilkan secara vertikal dengan membuat baris baru. Ini berbeda pula dengan konsep **inline**model di mana elemen selalu ditampilkan secara horizontal. Dengan flexbox kita dapat melakukan kedua hal tersebut dengan mudah.
* Dibuat untuk menyusun layout yang *mobile friendly*.

### Flex Container

Flex container merupakan sebuah elemen yang menampung beberapa flex item. Karena Parent-child merupakan relasi yang wajib ada dalam penerapan flexbox, maka setiap flex item harus merupakan anak / child dari flex container.

Secara default deretan flex-item pada container ditampilkan secara horizontal. Ketika menggunakan flex, kita tidak perlu mengatur dimensi dari tiap flex item secara manual untuk mengisi ruang kosong pada container. Sebelum ada flexbox, hal ini jadi kendala umum. Alih-alih, kita harus melakukan perhitungan yang tepat agar tak terjadi overflow pada layout yang ditampilkan.

Untuk membuat sebuah flex container kita gunakan properti **display** dengan nilai **flex.**Dengan demikian seluruh anak dari container tersebut akan menjadi flex item. Berikut contoh penerapan flexbox untuk membuat tampilan seperti gambar di atas:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. display: flex;
8. /\* Properti lainnya \*/
9. width: 800px;
10. height: 250px;
11. background-color: #11C5C6;
12. border: 2px solid black;
13. padding: 20px;
14. border-radius: 10px;
15. margin: 0 auto;
16. }
17. .box {
18. flex-grow: 1;
20. /\* properti lainnya \*/
21. background-color: #FBDD1C;
22. margin: 5px;
23. border: 2px solid black;
24. border-radius: 10px;
25. }
26. </style>
27. </head>
28. <body>
29. <div class="container">
30. <div class="box"></div>
31. <div class="box"></div>
32. <div class="box"></div>
33. </div>
34. </body>
35. </html>

Output dari kode di atas adalah:

### lex Grow

Bisa kita lihat seluruh box di dalam container dapat menyesuaikan ukurannya sesuai ruang kosong yang ada pada container. Pada box, kita tidak menetapkan nilai dimensi seperti width dan height. Kita cukup menggunakan satu properti yakni flex-grow: 1;.

Properti flex-grow ini digunakan untuk memberitahu berapa banyak ukuran yang harus ditetapkan oleh flex-item. **Nilai dari properti ini bukan nilai dari dimensi asli pada flex item, melainkan nilai yang relatif terhadap ruang kosong pada flex container.**

Jika kita menetapkan nilai **flex-grow** yang sama pada seluruh flex item, maka dimensi dari tiap flex item akan sama rata dan memenuhi ruang kosong yang ada pada container. Namun jika kita memberikan nilai yang berbeda dari salah satu item-nya, contohnya nilai yang lebih besar, maka flex item tersebut akan mencakup ukuran yang lebih besar. Flex item yang lain akan menyusut menyesuaikan agar tetap masuk pada ruang flex container. Contohnya seperti berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. display: flex;
8. /\* Properti lainnya \*/
9. width: 800px;
10. height: 250px;
11. background-color: #11C5C6;
12. border: 2px solid black;
13. padding: 20px;
14. border-radius: 10px;
15. margin: 0 auto;
16. }
17. .box {
18. /\* properti lainnya \*/
19. background-color: #FBDD1C;
20. margin: 5px;
21. border: 2px solid black;
22. border-radius: 10px;
23. }
25. .first {
26. flex-grow: 1;
27. }
28. .second {
29. flex-grow: 2;
30. }
32. .third {
33. flex-grow: 1;
34. }
35. </style>
36. </head>
37. <body>
38. <div class="container">
39. <div class="box first"></div>
40. <div class="box second"></div>
41. <div class="box third"></div>
42. </div>
43. </body>
44. </html>

Output dari kode di atas adalah:

Cara kerja flex-grow mirip seperti potongan kue. Ruang kosong pada elemen akan dibagi-bagi sesuai besaran nilainya. Contoh di atas memberi kita gambaran seperti sebuah kue dengan luas total 4, kemudian kue tersebut dipotong menjadi 3 potong. Potongan yang tengah mendapatkan 2 bagian dan potongan yang lainnya masing - masing mendapatkan 1 bagian. Maka potongan yang tengah akan lebih besar dari potongan yang lain.

Betapa fleksibelnya bukan? Yang harus kita ingat, ketika kita mengubah ukuran dari suatu flex item, maka flex item yang lain pun akan terpengaruh.

### Flex Direction

Seperti yang sudah kita ketahui sebelumnya, flexbox merupakan directional agnostic, di mana kita dapat mengubah arah munculnya flex-item yang berada di flex container. Secara default deretan flex-item ditampilkan secara **horizontal**, namun kita dapat mengubahnya dengan menetapkan properti **flex-direction** pada flex container-nya.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .container {
6. display: flex;
7. flex-direction: column;
9. /\* Properti lainnya \*/
10. width: 800px;
11. height: 250px;
12. background-color: #11C5C6;
13. border: 2px solid black;
14. padding: 20px;
15. border-radius: 10px;
16. margin: 0 auto;
17. }
18. .box {
19. /\* properti lainnya \*/
20. background-color: #FBDD1C;
21. margin: 5px;
22. border: 2px solid black;
23. border-radius: 10px;
24. }
26. .first {
27. flex-grow: 3;
28. }
29. .second {
30. flex-grow: 2;
31. }
32. .third {
33. flex-grow: 1;
34. }
35. </style>
36. </head>
37. <body>
38. <div class="container">
39. <div class="box first"></div>
40. <div class="box second"></div>
41. <div class="box third"></div>
42. </div>
43. </body>
44. </html>

Output dari kode di atas adalah:

Ada empat nilai yang bisa digunakan untuk properti **flex-direction**, antara lain:

* **row**: merupakan nilai default, di mana deretan flex-item pada container ditampilkan secara horizontal.
* **row-reverse**: memiliki sifat yang sama seperti row, namun urutan flex item-nya ditukar.
* **column**: Deretan flex-item pada container ditampilkan secara vertikal.
* **column-reverse**: memiliki sifat yang sama seperti column, namun urutan flex item-nya ditukar.

Dengan menggunakan properti ini, kita dapat membuat dua dimensional layout dengan menempatkan flex container di dalam flex container, dalam arti lain sebuah flex container dapat memiliki child berupa flex container lain.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. .flex-column {
6. display: flex;
7. flex-direction: column;
9. /\* properti lainnya \*/
10. width: 800px;
11. height: 300px;
12. padding: 20px;
13. margin: 0 auto;
14. border: 2px solid black;
15. border-radius: 10px;
16. background-color: #11C5C6;
17. }
19. .flex-row {
20. display: flex;
21. flex-direction: row;
22. flex-grow: 1;
24. /\* properti lainnya \*/
25. padding: 20px;
26. margin: 5px;
27. background-color: #FBDD1C;
28. border: 2px solid black;
29. border-radius: 10px;
30. }
32. .box {
33. flex-grow: 1;
35. /\* properti lainnya \*/
36. margin: 5px;
37. border: 2px solid black;
38. border-radius: 10px;
39. background-color: #fe899a;
40. }
41. </style>
42. </head>
43. <body>
44. <div class="flex-column">
45. <div class="flex-row">
46. <div class="box"></div>
47. <div class="box"></div>
48. <div class="box"></div>
49. </div>
50. <div class="flex-row">
51. <div class="box"></div>
52. <div class="box"></div>
53. <div class="box"></div>
54. </div>
55. </div>
56. </body>
57. </html>

Output dari kode di atas adalah:

Hal ini yang akan kita gunakan nanti sebagai dasar dalam membangun layout kalkulator.

### Flex Basis

Selain menggunakan flex-grow, untuk menentukan ukuran flex item, kita bisa gunakan properti flex-basis. Properti ini mirip seperti width dalam menentukan dimensi box. Kita bisa menggunakan nilai satuan tetap seperti px, pt, pc, cm dll. Selain itu, kita juga bisa menggunakan satuan persentase (%).

Properti flex-basis ini digunakan ketika kita ingin menetapkan ukuran awal pada sebuah flex-item. Alhasil, kita dapat mengatur ukuran dengan lebih leluasa. flex-basis biasa digunakan ketika kita menerapkan nested flex-container dan terdapat perbedaan jumlah child pada container-nya. Untuk lebih mudah menggambarkannya, perhatikan contoh berikut:

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. box-sizing: border-box;
7. }
8. .container {
9. width: 800px;
10. border: 2px solid black;
11. margin: 0 auto;
12. }
13. .flex-column {
14. display: flex;
15. flex-direction: column;
16. }
17. .flex-row {
18. display: flex;
19. flex-direction: row;
20. flex-grow: 1;
21. }
22. .box {
23. flex-grow: 1;
24. background-color: cornflowerblue;
25. border: 2px solid black;
26. padding: 40px;
27. text-align: center;
28. color: white;
29. font-size: 1.5em;
30. }
31. </style>
32. </head>
33. <body>
34. <div class="container">
35. <div class="flex-column">
36. <div class="flex-row">
37. <div class="box">1</div>
38. <div class="box">2</div>
39. <div class="box">3</div>
40. <div class="box">4</div>
41. </div>
42. <div class="flex-row">
43. <div class="box">5</div>
44. <div class="box">6</div>
45. <div class="box">7</div>
46. </div>
47. </div>
48. </div>
49. </body>
50. </html>

Jika dijalankan melalui browser, output dari kode di atas adalah seperti berikut.

Pada contoh di atas, kita memiliki satu buah flex-container dengan flex-direction: column dan di dalamnya terdapat dua buah flex-container dengan flex-direction: row. Kemudian, elemen div.flex-row yang pertama memiliki empat buah flex-item dan yang kedua memiliki tiga buah flex-item.

Standarnya ketika kita menggunakan nilai flex-grow: 1 pada seluruh flex-item yang ada, maka tampilan akan seperti gambar di atas. Namun, bagaimana jika kita ingin mengatur flex-item di dalam elemen div.flex-row yang kedua dengan tampilan berbeda seperti di bawah ini?

Jika kita menggunakan flex-grow, mungkin kita dapat memberikan nilai flex-grow: 2 pada box ke tujuh.

1. <!-- ..... -->
2. <div class="box">4</div>
3. </div>
4. <div class="flex-row">
5. <div class="box">5</div>
6. <div class="box">6</div>
7. **<div class="box double">7</div>**
8. </div>
9. </div>
10. <!-- ..... -->
11. <!-- ..... -->
12. .double {
13. flex-grow: 2;
14. }

Namun, jika dijalankan kembali, hasilnya akan tampak seperti berikut.

Hal di atas tidak sesuai dengan keinginan karena properti flex-grow tidak menjamin ukuran dari elemen yang menerapkan nilai 2 akan lebih besar 2 kali lipat dari nilai 1.

Sebabnya, flex-grow akan mencari nilai yang sesuai yang dapat dibagi pada flex-items. Agak terdengar aneh, bukan? Nah, temukan artikel yang cukup menarik yang membahas permasalahan ini pada tautan [berikut](https://css-tricks.com/flex-grow-is-weird/).

Sebenarnya bisa saja kita menggunakan flex-grow untuk mendapatkan hasil yang diinginkan namun kita harus mencari nilai yang pas secara manual. Ini tentunya akan memakan waktu lebih.

Pada kasus seperti ini, solusinya adalah menggunakan properti flex-basis. Dengan properti ini kita dapat asumsikan bahwa total ruang kosong pada flex-container adalah 100%. Jika dibagi rata terhadap empat buah flex-item, maka tiap itemnya harus memiliki nilai 25%.

1. <!doctype html>
2. <html lang="en">
3. <head>
4. <style>
5. \* {
6. box-sizing: border-box;
7. }
9. .container {
10. width: 800px;
11. border: 2px solid black;
12. margin: 0 auto;
13. }
15. .flex-column {
16. display: flex;
17. flex-direction: column;
18. }
20. .flex-row {
21. display: flex;
22. flex-direction: row;
23. flex-grow: 1;
24. }
26. .box {
27. flex-basis: 25%;
28. background-color: cornflowerblue;
29. border: 2px solid black;
30. padding: 40px;
31. text-align: center;
32. color: white;
33. font-size: 1.5em;
34. }
35. </style>
36. </head>
37. <body>
38. <div class="container">
39. <div class="flex-column">
40. <div class="flex-row">
41. <div class="box">1</div>
42. <div class="box">2</div>
43. <div class="box">3</div>
44. <div class="box">4</div>
45. </div>
46. <div class="flex-row">
47. <div class="box">5</div>
48. <div class="box">6</div>
49. <div class="box">7</div>
50. </div>
51. </div>
52. </div>
53. </body>
54. </html>

Jika kode di atas dijalankan, hasil yang ditampilkan adalah seperti berikut.

Dengan demikian, seluruh flex-item akan memiliki ukuran yang sama. Namun, pada flex-container baris kedua terdapat 25% ruang bebas (ruang sisa) karena jumlah flex-itemnya hanya tiga. Kita bisa menggunakan ruang kosong tersebut untuk membuat box tujuh lebih besar dua kali lipat dengan menggunakan nilai flex-basis: 50%**.**

1. <!-- ..... -->
2. <div class="box">4</div>
3. </div>
4. <div class="flex-row">
5. <div class="box">5</div>
6. <div class="box">6</div>
7. **<div class="box double">7</div>**
8. </div>
9. </div>
10. <!-- ..... -->
11. <!-- ..... -->
12. .double {
13. flex-basis: 50%;
14. }

## Membuat Layout Kalkulator

Setelah mengenal dasar penggunaan flexbox, mari sekarang kita terapkan apa yang sudah kita pelajari dengan membuat dasar layout kalkulator.

### Membuat struktur HTML kalkulator

Mari kita membuat sebuah project baru dengan membuat folder baru dan beri nama folder sesuai keinginan atau pada contoh kali ini kita beri nama “WebCalculator”. Lalu buka folder tersebut menggunakan teks editor yang Anda miliki dan buat berkas HTML baru dengan nama “index.html”.

Tampilan project yang dibuka menggunakan Visual Studio Code

Setelah itu, buka berkas index.html dan tuliskan struktur dasar HTML, mulai dari elemen <html>, <head>, dan <body>.

1. <!DOCTYPE html>
2. <html>
3. <head></head>
4. <body></body>
5. </html>

Jangan lupa tambahkan elemen <title> di dalam <head>.

1. <!DOCTYPE html>
2. <html>
4. <head>
5. <title>Web Calculator</title>
6. </head>
8. <body></body>
10. </html>

Sesuai yang sudah kita pelajari untuk menggunakan flexbox, kita membutuhkan sebuah container, jadi pada elemen <body> buatlah sebuah column container. Di dalamnya terdapat empat buah row container dan di dalam masing masing row container terdapat empat buah item-flex. Maka tuliskan kode berikut di dalam elemen **<body>**:

1. <div class="flex-container-column">
2. <div class="flex-container-row">
3. <div class="button"></div>
4. <div class="button"></div>
5. <div class="button"></div>
6. <div class="button"></div>
7. </div>
8. <div class="flex-container-row">
9. <div class="button"></div>
10. <div class="button"></div>
11. <div class="button"></div>
12. <div class="button"></div>
13. </div>
14. <div class="flex-container-row">
15. <div class="button"></div>
16. <div class="button"></div>
17. <div class="button"></div>
18. <div class="button"></div>
19. </div>
20. <div class="flex-container-row">
21. <div class="button"></div>
22. <div class="button"></div>
23. <div class="button"></div>
24. <div class="button"></div>
25. </div>
26. </div>

Pada kode di atas, elemen column container kita berikan nilai class “**flex-container-column**”, tiap elemen row container kita berikan nilai class “**flex-container-row**” dan tiap flex item pada row container kita berikan nilai class “**button**”.

Kemudian setiap elemen .button, di dalamnya kita berikan teks yang digunakan untuk melabelkan tiap-tiap tombolnya. Tuliskan teks angka 0 hingga 9 beserta operator-operator yang akan digunakan pada kalkulator dengan urutan seperti ini:

1. <div class="flex-container-column">
2. <div class="flex-container-row">
3. <div class="button">7</div>
4. <div class="button">8</div>
5. <div class="button">9</div>
6. <div class="button">+/-</div>
7. </div>
8. <div class="flex-container-row">
9. <div class="button">4</div>
10. <div class="button">5</div>
11. <div class="button">6</div>
12. <div class="button">-</div>
13. </div>
14. <div class="flex-container-row">
15. <div class="button">1</div>
16. <div class="button">2</div>
17. <div class="button">3</div>
18. <div class="button">+</div>
19. </div>
20. <div class="flex-container-row">
21. <div class="button">CE</div>
22. <div class="button">0</div>
23. <div class="button">=</div>
24. <div class="button"></div>
25. </div>
26. </div>

**Pada .flex-container-row terakhir, terdapat satu flex-item yang tidak diberikan nilai teks. Item tersebut memang tidak kita butuhkan sehingga kita bisa hapus elemen tersebut.**

Jika kita perhatikan dari beberapa tombol di atas, ada yang berupa tombol angka, operator dan fungsi. Untuk tombol operator seperti tambah (+) dan kurang (-) berikan class tambahan dengan nilai “operator”.

1. .....
2. <div class="button operator">-</div>
3. .....
4. <div class="button operator">+</div>

Kemudian untuk tombol fungsi seperti negative (+/-), clear (CE), dan equals (=) berikan class tambahan dengan nilai sesuai dengan fungsinya tersebut.

1. ....
2. <div class="button negative">+/-</div>
3. ....
4. <div class="button clear">CE</div>
5. ....
6. <div class="button equals">=</div>

Penamaan class yang spesifik seperti ini dibutuhkan untuk memudahkan kita mengambil nilai elemen nanti ketika menggunakan JavaScript, hal ini juga membantu kita untuk memudahkan penentuan selector untuk styling menggunakan CSS.

Sebuah kalkulator tentu tidak hanya terdiri dari tombol saja bukan? Tiap kalkulator pasti memiliki layar untuk menampilkan apa yang pengguna input dan menampilkan hasil kalkulasinya. Dengan begitu kita membutuhkan elemen lain yang berfungsi sebagai layar. Tambahkan elemen div baru tepat di bawah tag pembuka .flex-container-column dan beri atribut class dengan nilai “display”.

1. <div class="flex-container-column">
2. <div class="display"></div>
3. <div class="flex-container-row">
4. ...........

Kemudian di dalam elemen .display tersebut tambahkan elemen <h1></h1> sebagai teks yang berfungsi untuk menampilkan angka pada layar tersebut.

1. <div class="display">
2. <h1 id="displayNumber">0</h1>
3. </div>

Sejauh ini struktur kode yang ada pada berkas index.html akan tampak seperti ini:

1. <!DOCTYPE html>
2. <html>
4. <head>
5. <title>Web Calculator</title>
6. </head>
8. <body>
9. <div class="flex-container-column">
10. <div class="display">
11. <h1 id="displayNumber">0</h1>
12. </div>
13. <div class="flex-container-row">
14. <div class="button">7</div>
15. <div class="button">8</div>
16. <div class="button">9</div>
17. <div class="button negative">+/-</div>
18. </div>
19. <div class="flex-container-row">
20. <div class="button">4</div>
21. <div class="button">5</div>
22. <div class="button">6</div>
23. <div class="button operator">-</div>
24. </div>
25. <div class="flex-container-row">
26. <div class="button">1</div>
27. <div class="button">2</div>
28. <div class="button">3</div>
29. <div class="button operator">+</div>
30. </div>
31. <div class="flex-container-row">
32. <div class="button clear">CE</div>
33. <div class="button">0</div>
34. <div class="button equals">=</div>
35. </div>
36. </div>
37. </body>
39. </html>

Jika kita buka melalui browser, tampilan saat ini masih terlihat aneh.

Yups, hal tersebut wajar karena <div> tidak akan berpengaruh apa apa hingga kita memberikan styling pada elemen tersebut.

### Menerapkan Flexbox pada Halaman Kalkulator

Setelah kita selesai membuat struktur HTML, selanjutnya kita akan percantik elemen - elemen tersebut dengan styling.

Buatlah folder baru di dalam project **WebCalculator** dan berikan nama “assets”. Kemudian di dalam folder tersebut buatlah berkas css dan berikan nama “style.css”. Sehingga struktur project kita saat ini akan tampak seperti ini:

Struktur project WebCalculator

Sebelum memulai menuliskan styling pada CSS, jangan lupa untuk hubungkan berkas style.css sebagai stylesheet pada index.html. Tuliskan tag <link> berikut ini di dalam elemen <head>:

1. <link rel="stylesheet" href="assets/style.css">

Untuk memulai menuliskan styling, kita buka berkas style.css. Kita mulai dengan membuat seluruh elemen menerapkan nilai border-box pada properti box-sizing. Tuliskan rule berikut pada berkas style.css.

1. \* {
2. box-sizing: border-box;
3. }

Lalu kita tetapkan font-family pada seluruh teks yang ada di dalam body dengan menggunakan sans-serif.

1. body {
2. font-family: sans-serif;
3. }

Selanjutnya kita mulai terapkan flexbox pada .flex-container-column dengan memberikan nilai **flex** pada properti **display**. Jangan lupa tetapkan properti flex-direction: column dan properti pendukung lainnya. Sehingga styling .flex-container-column akan tampak seperti berikut:

1. .flex-container-column {
2. display: flex;
3. flex-direction: column;
5. /\* properti pendukung \*/
6. max-width: 800px;
7. margin: 0 auto;
8. text-align: right;
9. }

Terapkan juga properti display: flex pada .flex-container-row. Namun kita tidak harus menerapkan flex-direction pada elemen ini, karena nilai row sudah menjadi default pada flex-container. Sehingga penulisan styling untuk .flex-container-row cukup dituliskan seperti berikut:

1. .flex-container-row {
2. display: flex;
3. }

Mari kita lanjut atur styling untuk tombol kalkulatornya. Pada tiap .button kita akan beri nilai flex-basis sebesar 25% beserta properti pendukung lainnya sehingga tombol kalkulator akan memenuhi seluruh ruang kosong pada container. Tuliskan styling berikut untuk .button:

1. .button {
2. flex-basis: 25%;
4. /\* properti pendukung \*/
5. font-size: 1.5em;
6. text-align: center;
7. padding: 40px;
8. border: 1px solid black;
9. background: -webkit-linear-gradient(top, #d2d2d2, #ddd);
10. cursor: pointer;
11. }

Simpan berkas style.css dan coba buka index.html pada browser. Tampilan akan nampak seperti ini:

Sekarang sudah terlihat tampilan kalkulator pada umumnya, bukan? Tombol kalkulator akan memenuhi seluruh ruang kosong pada container terkecuali tombol yang terdapat pada flex-container-row terakhir karena jumlah button nya hanya tiga buah.

Agar kontainer terisi penuh, kita buat tombol equals (=) memiliki nilai flex-basis: 50% sehingga tombol tersebut memiliki besar 2 kali dibandingkan dengan tombol yang lainnya. Tambahkan nilai class “double” pada tombol tersebut.

1. <div class="button equals double">=</div>

Kemudian tetapkan nilai flex-basis: 50%; pada .double seperti berikut:

1. .double {
2. flex-basis: 50%;
3. }

Sehingga sekarang tampilan button pada browser akan tampak seperti ini:

Sejauh ini kode yang terdapat pada style.css dan index.html akan tampak seperti berikut:

1. \* {
2. box-sizing: border-box;
3. }
5. body {
6. font-family: sans-serif;
7. }
9. .flex-container-column {
10. display: flex;
12. /\* properti pendukung \*/
13. flex-direction: column;
14. max-width: 800px;
15. margin: 0 auto;
16. text-align: right;
17. }
19. .flex-container-row {
20. display: flex;
21. }
23. .button {
24. flex-basis: 25%;
26. /\* properti pendukung \*/
27. font-size: 1.5em;
28. text-align: center;
29. padding: 40px;
30. border: 1px solid black;
31. background: -webkit-linear-gradient(top, #d2d2d2, #ddd);
32. cursor: pointer;
33. }
35. .double {
36. flex-basis: 50%;
37. }
38. <!DOCTYPE html>
39. <html>
41. <head>
42. <title>Web Calculator</title>
43. <link rel="stylesheet" href="assets/style.css">
44. </head>
46. <body>
47. <div class="flex-container-column">
48. <div class="display">
49. <h1 id="displayNumber">0</h1>
50. </div>
51. <div class="flex-container-row">
52. <div class="button">7</div>
53. <div class="button">8</div>
54. <div class="button">9</div>
55. <div class="button negative">+/-</div>
56. </div>
57. <div class="flex-container-row">
58. <div class="button">4</div>
59. <div class="button">5</div>
60. <div class="button">6</div>
61. <div class="button operator">-</div>
62. </div>
63. <div class="flex-container-row">
64. <div class="button">1</div>
65. <div class="button">2</div>
66. <div class="button">3</div>
67. <div class="button operator">+</div>
68. </div>
69. <div class="flex-container-row">
70. <div class="button clear">CE</div>
71. <div class="button">0</div>
72. <div class="button equals double">=</div>
73. </div>
74. </div>
75. </body>
77. </html>

### Mempercantik tampilan kalkulator

Setelah selesai dengan penerapan flexbox untuk tombol kalkulator, selanjutnya kita akan mempercantik bagian lain pada kalkulator. Pada bagian kali ini sebenarnya Anda bisa bebas berekspresi dalam menerapkan styling. Anda bisa gunakan kemampuan seni Anda ditambah pengetahuan CSS yang sudah Anda kuasai.

Tapi jika Anda ingin membuat tampilan kalkulator tampak seperti yang sudah dicontohkan ini:

Anda bisa mengikuti langkah - langkah materi kali ini:

Yang pertama kita akan mempercantik display yang ada pada kalkulator dengan menambahkan background-color, padding, border, dan lain sebagainya. Untuk itu mari kita tetapkan styling pada .display dengan nilai - nilai properti sebagai berikut:

1. .display {
2. color: white;
3. width: 100%;
4. padding: 10px 20px;
5. background-color: #333333;
6. border: 1px solid black;
7. font-size: 2em;
8. }

Dengan begitu maka display akan tampak seperti ini:

Selanjutnya kita akan membuat tombol kalkulator yang merupakan sebuah operator, fungsi negative dan fungsi equals berwarna orange. Dengan begitu kita perlu menetapkan nilai background-color pada elemen yang memiliki class .operator, .negative, dan .equals.

1. .operator,
2. .negative,
3. .equals {
4. background: orange;
5. }

Maka sekarang tombol tersebut akan tampak berwarna orange.

Lalu kita akan membuat efek card pada calculator agar kalkulator tampak seperti berada pada sebuah kartu yang terdapat bayangan pada tepiannya. Hal ini sudah pernah kita lakukan pada latihan project halaman profile, masih ingat kan?

Buatlah rule styling baru dengan selector .card, dan beri nilai properti seperti di bawah ini:

1. .card {
2. box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
3. border-radius: 5px;
4. padding: 30px;
5. margin-top: 20px;
6. }

Kemudian terapkan dengan menambahkan class card pada flex-container column.

1. <div class="flex-container-column card">

Sehingga sekarang kalkulator akan tampak seolah - olah berada di dalam kartu.

Terakhir, kita juga bisa tambahkan sebuah efek lain pada tombol dengan memanfaatkan pseudo-class :hover. Kita buat efek teks menjadi lebih tebal ketika kursor diarahkan pada salah satu tombol. Tambahkan rule dengan selector .button:hover dengan nilai properties berikut.

1. .button:hover {
2. font-weight: bold;
3. }

Sekarang teks pada tombol akan menjadi tebal ketika terdapat kursor yang diarahkan.

Sejauh ini berikut kode yang ada pada index.html dan style.css:

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Web Calculator</title>
5. <link rel="stylesheet" href="assets/style.css">
6. </head>
7. <body>
8. <div class="flex-container-column card">
9. <div class="display">
10. <h1 id="displayNumber">0</h1>
11. </div>
12. <div class="flex-container-row">
13. <div class="button">7</div>
14. <div class="button">8</div>
15. <div class="button">9</div>
16. <div class="button negative">+/-</div>
17. </div>
18. <div class="flex-container-row">
19. <div class="button">4</div>
20. <div class="button">5</div>
21. <div class="button">6</div>
22. <div class="button operator">-</div>
23. </div>
24. <div class="flex-container-row">
25. <div class="button">1</div>
26. <div class="button">2</div>
27. <div class="button">3</div>
28. <div class="button operator">+</div>
29. </div>
30. <div class="flex-container-row">
31. <div class="button clear">CE</div>
32. <div class="button">0</div>
33. <div class="button equals double">=</div>
34. </div>
35. </div>
36. </body>
37. </html>
38. \* {
39. box-sizing: border-box;
40. }
42. body {
43. font-family: sans-serif;
44. }
46. .flex-container-column {
47. display: flex;
48. /\* properti pendukung \*/
49. flex-direction: column;
50. max-width: 800px;
51. margin: 0 auto;
52. text-align: right;
53. }
55. .flex-container-row {
56. display: flex;
57. }
59. .button {
60. flex-basis: 25%;
61. /\* properti pendukung \*/
62. font-size: 1.5em;
63. text-align: center;
64. padding: 40px;
65. border: 1px solid black;
66. background: -webkit-linear-gradient(top, #d2d2d2, #ddd);
67. cursor: pointer;
68. }
70. .double {
71. flex-basis: 50%;
72. }
74. .display {
75. color: white;
76. width: 100%;
77. padding: 10px 20px;
78. background-color: #333333;
79. border: 1px solid black;
80. font-size: 2em;
81. }
83. .operator,
84. .negative,
85. .equals {
86. background: orange;
87. }
89. .card {
90. box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
91. border-radius: 5px;
92. padding: 30px;
93. margin-top: 20px;
94. }
96. .button:hover {
97. font-weight: bold;
98. }

### Tampilan Responsif pada Mobile

Sejauh ini layout kalkulator yang kita buat sudah cukup baik, namun apakah sudah cukup baik juga ketika tampil pada device mobile? Untuk melihatnya, kita bisa coba dengan menggunakan Dev Tools pada browser kita.

Gambar di atas merupakan kalkulator yang kita buat di akses melalui device Google Pixel 2 XL. Tampilan kalkulator sudah responsif namun, teks tombol masih terlalu kecil tampilannya. Aspek aksesibilitas penggunanya, kurang.

Agar tampilan lebih optimal pada viewport mobile devices, pertama kita tambahkan meta tag viewport pada **index.html** agar tampilan dapat menyesuaikan dengan beberapa layar yang berbeda.

1. <head>
2. <title>Web Calculator</title>
3. <link rel="stylesheet" href="assets/style.css">
4. <meta name="viewport" content="width=device-width, initial-scale=1.0">
5. </head>

Dengan begitu ukuran teks dan box dimension akan disesuaikan pada mobile devices. Namun tampilan tombol akan overflow karena padding pada tiap tombol yang kita tentukan terlalu besar.

Pada situasi ini kita dapat menggunakan @media query untuk menetapkan nilai padding yang pas ketika di akses pada mobile device. Tuliskan rule berikut pada styles.css:

1. @media screen and (max-width: 513px) {
2. .button {
3. padding: 10px;
4. }
5. }

Sekarang tampilan tombol akan menerapkan nilai padding yang lebih kecil ketika diakses menggunakan ukuran layar dengan lebar di bawah 513px.