

Andre Pratama



CSS Uncover

Panduan Belajar CSS Untuk Pemula

DuniaIlkom

CSS Uncover

Panduan Belajar CSS untuk Pemula

Andre Pratama

Buku ini ditulis dan diterbitkan secara mandiri oleh DuniaIlkom (www.duniailkom.com)

23 Maret 2021

~ Update Log ~

- ✓ CSS Uncover 1.0 – September 2015
- ✓ CSS Uncover 2.0 – Agustus 2016
- ✓ CSS Uncover 3.0 – Maret 2021

Cover Photo by [Brett Sayles](#) on [Pexels](#)

© 2021 DuniaIlkom

Daftar Isi

Ucapan Terima kasih.....	14
Tentang Penulis.....	15
Lisensi.....	16
Kata Pengantar.....	19
Asumsi / Pengetahuan Dasar.....	20
Contoh Kode Program.....	21
1. Berkenalan Dengan CSS.....	22
1.1. Pengertian CSS.....	22
1.2. Fungsi CSS.....	23
1.3. Contoh Penggunaan CSS.....	24
2. Sejarah dan Perkembangan CSS.....	29
2.1. Sejarah CSS.....	29
Kebutuhan akan Style Sheet Language.....	29
Awal Lahirnya CSS: Milis www-style.....	29
Standar CSS 1.0.....	30
Dukungan Web Browser untuk CSS 1.0.....	31
Standar CSS 2.0.....	31
Standar CSS 2.1.....	32
Standar CSS 3.....	32
Standar CSS 4.....	33
2.2. Perbedaan CSS 1.0, CSS 2.1 dan CSS3.....	33
Fitur dalam CSS 1.0.....	33
Fitur dalam CSS 2.1.....	34
Fitur dalam CSS 3.....	34
Tahapan Status Spesifikasi W3C.....	34
Tahap Candidate Recommendation.....	35
Tahap Proposed Recommendation.....	35
Tahap Recommendation.....	35
2.3. Modul-modul dalam CSS3.....	36

3. Web Browser dan Text Editor.....	38
3.1. Memilih Web Browser.....	38
3.2. Mengenal Web Browser Layout Engine.....	39
3.3. Dukungan Web Browser untuk CSS3.....	40
3.4. Text Editor.....	42
Notepad++.....	42
Visual Studio Code.....	43
Bagaimana dengan Adobe Dreamweaver?.....	46
4. Aturan Dasar Penulisan CSS.....	48
4.1. Pengertian Selector, Declaration, Property dan Value CSS.....	48
Cara Penulisan Selector, Property dan Value.....	49
4.2. Case Sensitivity.....	49
4.3. Whitespace.....	51
4.4. Baris Komentar.....	52
4.5. Cara Menginput kode CSS.....	53
Inline Style CSS.....	53
Internal Style CSS.....	54
External Style CSS.....	56
Mana yang Sebaiknya Dipakai?.....	60
Menggabung Ketiga Metode Input CSS.....	60
4.6. CSS Media Type.....	61
5. CSS Selector.....	66
5.1. Pengertian Selector.....	66
Pentingnya Struktur HTML.....	66
5.2. Universal Selector.....	67
5.3. Element Selector.....	68
5.4. Class Selector.....	69
5.5. Id Selector.....	72
5.6. Attribute Selectors.....	74
5.7. Group Selector.....	81
5.8. Element Spesific Selector.....	83
5.9. Descendant Selector.....	84
5.10. Child Selector.....	88
5.11. Adjacent Selector.....	92
5.12. General Sibling Selector.....	95

5.13. Dynamic Pseudo Class Selector.....	97
Selector :link.....	98
Selector :visited.....	98
Selector :hover.....	99
Selector :active.....	100
Selector :target.....	102
Selector :focus.....	103
5.14. UI Element State Pseudo Class Selector.....	104
5.15. Structural Pseudo Class Selectors.....	106
Selector :first-child dan :last-child.....	108
Selector :first-of-type dan :last-of-type.....	109
Pseudo Selector :nth-child() dan :nth-last-child().....	111
Pseudo Selector :nth-of-type() dan :nth-last-of-type().....	112
Pseudo Selector :not().....	114
5.16. Pseudo Element Selector.....	116
Pseudo Element Selector ::first-line.....	116
Pseudo Element Selector ::first-letter.....	117
Pseudo Element Selector ::before dan ::after.....	118
6. Cascade, Inheritance dan Specificity.....	124
6.1. "Cascade" dari Cascading Style Sheet.....	124
6.2. Inheritance Style CSS.....	129
Nilai Property Inherit.....	134
6.3. Specificity Selector.....	135
6.4. Keyword Sakti: !important.....	139
6.5. Periksa dengan Developer Tools.....	141
6.6. Merencanakan Kode CSS.....	143
7. CSS Typography.....	145
7.1. Property font-family.....	145
Mengenal Generic Font.....	146
Web Safe Font.....	149
7.2. Property font-size.....	150
Satuan Length CSS.....	151
Nilai Absolut.....	151
Nilai Relatif.....	153
7.3. Property color.....	163

Satuan Color (Warna) CSS.....	164
7.4. Property font-weight.....	175
7.5. Property font-style.....	179
7.6. Property text-decoration.....	181
7.7. Property text-decoration CSS3.....	183
7.8. Property text-transform.....	185
7.9. Property font-variant.....	186
7.10. Property text-align.....	187
7.11. Property line-height.....	189
7.12. Property margin-bottom.....	191
7.13. Property letter-spacing dan word-spacing.....	193
7.14. Property text-indent.....	194
7.15. Font Shorthand Notation.....	197
7.16. Property text-shadow.....	200
7.17. Mengenal CSS Vendor Prefix.....	204
7.18. Font External (@font-face).....	206
Mengatasi Dukungan Format Font.....	209
Mengenal Berbagai Tipe Font.....	213
7.19. Menggunakan Google Font.....	218
8. CSS Box Model.....	225
8.1. Pengertian CSS Box Model.....	225
8.2. Property width dan height.....	226
Satuan Persen untuk width dan height.....	227
Satuan em untuk width dan height.....	229
Sifat Default Property width dan height.....	230
Block dan Inline element.....	232
8.3. Property overflow.....	234
8.4. Property padding.....	237
Padding Shorthand Notation (Penulisan Singkat Padding).....	239
8.5. Property border.....	242
Border Positioning.....	242
Jenis Border Style.....	246
8.6. Property border-radius CSS3.....	247
Border-Radius Longhand Notation.....	247
Border-Radius Shorthand Notation.....	250
8.7. Property margin.....	253

Margin Collapsing.....	256
Margin Auto.....	259
8.8. Content Width vs Element Width.....	261
8.9. CSS Reset.....	267
Haruskah Menggunakan CSS Reset?.....	270
8.10. Property box-sizing.....	271
8.11. Property outline.....	275
Outline untuk Accessibility.....	278
8.12. Property min-width dan max-width.....	279
8.13. Property min-height dan max-height.....	283
8.14. Length Value: vw dan vh.....	285
9. CSS Background.....	287
9.1. Property background-color.....	287
9.2. Property background-image.....	289
9.3. Property background-image CSS3.....	292
9.4. Property background-repeat.....	293
9.5. Property background-repeat CSS3.....	296
9.6. Property background-position.....	298
Bagaimana jika menggunakan nilai keyword?.....	300
Nilai persen untuk background-position.....	301
Satu, tiga atau empat nilai untuk background-position.....	303
9.7. Property background-position CSS3.....	304
9.8. Property background-size CSS3.....	305
9.9. Property background-attachment.....	310
9.10. Property background-origin dan background-clip CSS3.....	314
9.11. Property background-blend-mode CSS3.....	319
9.12. Property background (Shorthand Notation).....	322
10. CSS Positioning.....	326
10.1. Mengenal Normal Document Flow.....	326
10.2. Property position.....	332
Static Positioning.....	332
Relative Positioning.....	333
Absolute Positioning.....	336
Fixed Positioning.....	341
Sticky Positioning.....	343

10.3. Property display.....	345
Display none.....	346
Display inline.....	349
Display block.....	351
Display inline-block.....	352
10.4. Property z-index.....	354
10.5. Property float.....	358
Efek float pada Normal Document Flow.....	360
Mengatasi Container Collapse.....	364
10.6. Float Positioning.....	367
10.7. Property Clear.....	372
11. CSS3 Gradient.....	379
11.1. Membuat Gradient dengan Gambar.....	379
11.2. Property Gradient CSS3.....	382
11.3. Linear Gradient.....	382
Linear Gradient Color Stop.....	383
Linear Gradient Length.....	384
Linear Gradient Direction.....	386
Repeating Linear Gradients.....	389
11.4. Radial Gradient.....	391
Radial Gradient Position.....	393
Radial Gradient Extends.....	394
Radial Gradient Color Stop.....	396
Repeating Radial Gradients.....	397
11.5. CSS Gradient Generator.....	398
12. CSS3 Multiple Column.....	401
12.1. Property column-count.....	401
12.2. Property column-width.....	403
12.3. Property column-fill.....	403
12.4. Property column-gap.....	405
12.5. Property column-rule.....	405
12.6. Property column-span.....	406
13. CSS3 Border Image.....	410
13.1. CSS3 Border Image.....	410

13.2. Property border-image-source.....	411
13.3. Property border-image-slice.....	412
13.4. Property border-image-width.....	418
13.5. Property border-image-repeat.....	419
13.6. Property border-image-outset.....	422
13.7. Property border-image (shorthand).....	423
13.8. Border Image dari Gradient.....	424
13.9. CSS Border Image Generator.....	426
14. CSS3 Box Shadow.....	428
14.1. Property box-shadow.....	428
14.2. Box Shadow Offset.....	428
14.3. Box Shadow Blur-radius.....	429
14.4. Box Shadow Spread.....	430
14.5. Box Shadow Color.....	431
14.6. Box Shadow Inset.....	431
14.7. Multiple Box Shadow.....	432
14.8. CSS Box Shadow Generator.....	433
15. CSS3 2D Transform.....	435
15.1. Property transform.....	435
15.2. Transform Rotate.....	435
Transform Rotate dengan transform-origin.....	437
15.3. Transform Translate.....	439
15.4. Transform Scale.....	440
15.5. Transform Skew.....	442
15.6. Transform dengan Matriks.....	443
16. CSS3 Transitions dan CSS3 Animations.....	444
16.1. CSS3 Transitions.....	444
16.2. Property transition-property.....	446
16.3. Property transition-duration.....	447
16.4. Property transition-timing-function.....	450
16.5. Property transition-delay.....	452
16.6. Property transition (shorthand).....	453
16.7. Multiple Transition.....	455
16.8. CSS3 Animations.....	457

Animations Keyframe.....	457
16.9. Property animation-name.....	458
16.10. Property animation-duration.....	458
16.11. Property animation-timing-function.....	460
16.12. Property animation-delay.....	460
16.13. Property animation-iteration-count.....	461
16.14. Property animation-direction.....	461
16.15. Property animation-fill-mode.....	464
16.16. Property animation-play-state.....	466
16.17. Property animation shorthand.....	468
16.18. Multiple Animation.....	469
17. CSS3 Media Query.....	470
17.1. Mengenal Jenis-jenis Layout.....	470
Fixed Layout.....	471
Fluid Layout.....	471
Responsive Layout.....	472
17.2. CSS Media Type.....	473
17.3. Cara Penulisan Media Query.....	473
17.4. Media Feature: Width dan Height.....	475
17.5. Media Feature: Resolution.....	481
17.6. Media Feature: Device Width dan Device Height.....	482
Menguji Fitur Responsive di Developer Tools.....	483
17.7. Media Feature: Orientation.....	484
17.8. Membuat Layout Responsive Sederhana.....	485
18. CSS3 FlexBox.....	490
18.1. Flex Container.....	490
Jenis-jenis Flexbox Property.....	492
18.2. Property flex-direction.....	493
18.3. Property flex-wrap.....	496
18.4. Property flex-flow.....	499
18.5. Property justify-content.....	500
18.6. Property align-items.....	506
Membuat Efek Absolute Centering.....	512
18.7. Property align-content.....	513
18.8. Property flex-basis.....	517

18.9. Property flex-grow.....	521
18.10. Property flex-shrink.....	523
18.11. Property flex.....	525
18.12. Property align-self.....	528
18.13. Property order.....	530
18.14. Membuat Layout Responsive Sederhana.....	532
Layout Untuk Layar Smartphone.....	534
Layout Untuk Layar Tablet.....	536
Layout Untuk Layar Desktop.....	536
19. CSS3 Grid.....	539
19.1. Grid Container.....	539
Jenis-jenis Grid Property.....	541
19.2. Property grid-template-columns.....	542
Nilai Length auto dan fr.....	544
Mengulang Nilai dengan Fungsi repeat().....	545
Mengatur Lebar Kolom dengan Fungsi minmax().....	547
Membuat Jumlah Kolom Dinamis (auto-fill dan auto-fit).....	548
19.3. Property grid-template-rows.....	551
19.4. Property row-gap, column-gap dan gap.....	552
19.5. Property grid-column-start dan grid-column-end.....	553
19.6. Property grid-row-start dan grid-row-end.....	557
19.7. Property grid-column dan grid-row.....	558
19.8. Property grid-template-areas dan grid-area.....	562
19.9. Property justify-items.....	566
19.10. Property align-items.....	567
19.11. Property place-items.....	569
19.12. Property justify-self, align-self, dan place-self.....	570
19.13. Membuat Layout Responsive Sederhana.....	571
Layout Untuk Layar Smartphone.....	572
Layout Untuk Layar Tablet.....	573
Layout Untuk Layar Desktop.....	573
20. Materi CSS Lainnya.....	576
20.1. CSS Variable (custom property).....	576
Selector :root.....	579
20.2. Fungsi calc().....	580

20.3. Property Filter.....	581
Filter drop-shadow vs box-shadow.....	584
20.4. Property cursor.....	585
20.5. Property resize.....	586
20.6. Property list-style.....	587
Property list-style-type.....	588
Property list-style-image.....	590
Property list-style-position.....	591
Property list-style.....	592
20.7. Property scroll-behavior.....	593
20.8. Property border-collapse.....	594
Merapikan Tampilan Tabel.....	597
20.9. Property object-fit.....	598
20.10. Property clip-path.....	601
Nilai clip-path: circle().....	601
Nilai clip-path: ellipse().....	603
Nilai clip-path: inset().....	604
Nilai clip-path: polygon().....	605
Memakai clip-path Untuk Gambar.....	606
20.11. Property user-select.....	608
21. Font Awesome.....	610
21.1. Pengertian Font Awesome.....	610
21.2. Mendownload Font Awesome.....	610
21.3. Menghubungkan File Font Awesome.....	611
File Font Awesome di CDN.....	613
21.4. Cara Penggunaan Font Awesome.....	613
21.5. Manipulasi Icon Font Awesome.....	617
Memperbesar Ukuran Icon.....	618
Mengubah Warna Icon.....	619
Fixed Width Icon.....	620
Icon List.....	621
Rotating Icon.....	622
Spinning Icon.....	623
Stacked Icon.....	624
22. CSS Case Study.....	626

22.1. Membuat Menu Navigasi Vertikal.....	626
22.2. Membuat Menu Navigasi Horizontal.....	629
22.3. Membuat Menu Navigasi Dropdown.....	630
22.4. Membuat Header (Logo + Navigasi).....	633
22.5. Membuat Header Responsive (Hamburger Menu).....	636
22.6. Membuat Hero Image.....	641
22.7. Membuat Showcase.....	644
22.8. Membuat Product and Services.....	649
22.9. Membuat Blog Index.....	652
22.10. Membuat Footer.....	658
22.11. Final Project.....	662
Penutup: CSS Uncover.....	678
Daftar Pustaka.....	680

Ucapan Terima kasih

Dalam kesempatan ini saya ingin mengucapkan terima kasih kepada Allah SWT karena dengan karuniaNya saya masih diberi kesempatan dan kesehatan untuk bisa menulis (serta meng-update) buku kedua DuniaIlkom: **CSS Uncover**.

Selanjutnya kepada keluarga yang terus memberi motivasi dan dukungan tiada henti untuk terus mengembangkan DuniaIlkom.

Terakhir kepada rekan-rekan pembaca dan pengunjung setia DuniaIlkom. Terutama bagi yang telah memberikan donasi untuk membeli buku saya sebelumnya. Karena *feedback* dan dukungan rekan-rekan lah saya bisa lanjut menulis buku ini. Terima kasih :)

Padang Panjang, 2021

Penulis

Andre Pratama

www.duniaIlkom.com

Tentang Penulis



Andre Pratama

Andre memiliki background S1 Ilmu Komputer dari Universitas Sumatera Utara (angkatan 2005). Sempat terjun ke dunia kerja sebagai Assistant Manager di Bank Mandiri tahun 2010 - 2014.

Di akhir 2014, memutuskan untuk menjadi praktisi dan penulis buku programming. Saat ini full time mengelola web duniaIlkom yang sudah dirintis sejak tahun 2012. Harapannya, web duniaIlkom bisa menjadi sebagai salah satu media belajar programming dan ilmu komputer terbaik di Indonesia.

Andre berdomisili di kota Padang Panjang, Sumatera Barat. Jika ada pertanyaan, saran, kritik yang membangun bisa menghubungi duniaIlkom@gmail.com atau WA ke 083180285808.

Lisensi

Terima kasih untuk tidak memperbanyak / mengedarkan / mencopy eBook ini

Menulis sebuah buku hingga ratusan halaman butuh waktu yang tidak sebentar. Belum lagi saya harus berjuang mempelajari referensi yang kebanyakan dalam bahasa inggris. Ini saya lakukan agar pembaca bisa mendapatkan materi yang detail, update, dan berkualitas.

Saya menyadari kekurangan sebuah ebook adalah mudah dicopy-paste dan disebarluaskan. Tapi dengan eBook, harga buku bisa ditekan. Selain tidak perlu mencetak, eBook DuniaIlkom ini bisa di dapat dengan mudah dan murah, termasuk bagi teman-teman di daerah yang ongkos kirimnya lumayan mahal (jika berbentuk buku fisik).

Atas dasar itulah saya mohon kerjasamanya dari rekan-rekan semua untuk **tidak memperbanyak, menggandakan, atau mengupload ulang buku ini di forum, situs maupun media lain dalam bentuk apapun (termasuk tidak membuat video youtube dari materi buku).**

Saya juga berharap rekan-rekan tidak memposting materi apapun yang ada di dalam buku ini. Jika ingin sebagai bahan artikel untuk postingan blog/situs, silahkan ambil materi yang ada di website duniaIlkom (jangan yang dari buku).

Apabila rekan-rekan memperoleh buku ini **bukan** dari DuniaIlkom, saya mohon bantuan donasinya untuk membeli versi asli. Donasi pembelian buku ini adalah sumber mata pencarian saya untuk menafkahi keluarga. Lisensi atau hak guna buku ini hanya untuk 1 orang, yakni yang telah membeli langsung ke duniaIlkom@gmail.com.

Dengan kualitas yang ditawarkan, harga buku ini cukup terjangkau. Buku ini saya buat dengan waktu yang tidak sebentar, hingga berbulan-bulan, kadang sampai tengah malam. Bantuan donasi dari rekan-rekan yang membeli buku secara resmi sangat saya hargai, selain mendapat ilmu yang berkah, ini juga bisa menjadi penyemangat saya untuk terus berkarya dan menghadirkan ebook-ebook programming berkualitas lainnya.

Untuk yang membeli dari DuniaIlkom, saya ucapan banyak terimakasih :)

Anda diperbolehkan untuk:

- ✓ Mencetak eBook ini untuk keperluan pribadi dan dibaca sendiri.
- ✓ Mencopy eBook ini ke laptop/smartphone/tablet milik sendiri.
- ✓ Membuat ringkasan buku untuk digunakan sebagai bahan ajar (bukan keseluruhan isi buku).

Anda tidak dibolehkan untuk:

- ✗ Mencetak eBook ini untuk dibaca oleh orang lain, walaupun gratis.
- ✗ Mencopy eBook ini untuk dijual ulang, maupun dibagikan kepada orang lain dengan gratis.
- ✗ Membeli buku ini untuk dibaca bersama-sama (lisensi buku ini hanya untuk 1 orang).
- ✗ Mengambil sebagian atau seluruh isi buku untuk di publish ke blog, situs, artikel, dan media lain dalam bentuk apapun.
- ✗ Menjadikan materi buku sebagai bahan video YouTube / media public lain.
- ✗ Membagikan eBook ini kepada murid/siswa/mahasiswa (jika digunakan untuk bahan pengajaran).

Setiap pelanggaran dari lisensi ini akan dituntut sesuai undang-undang yang berlaku di Republik Indonesia, terutama **Pasal 12 UU No. 19 Tahun 2002** tentang **Hak Cipta**.

Penjelasan lebih lanjut bisa ke: [Apakah Mengunduh E-book Termasuk Perbuatan Illegal?](#)

Khusus untuk pembaca muslim bisa ke: [Hukum Memakai Barang Bajakan](#). Mari kita jaga agar ilmu yang di dapat berkah dan bermanfaat, bukan dari sumber yang haram.

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan :

Pencipta

Nama : Andre Pratama

Alamat :

Kewarganegaraan :

Indonesia

Pemegang Hak Cipta

Nama : Andre Pratama

Alamat :

Kewarganegaraan :

Indonesia

Jenis Ciptaan :

e-Book

Judul Ciptaan :

CSS Uncover

Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia

: 5 September 2015, di Padang Panjang

Jangka waktu pelindungan

: Berlaku selama hidup Pencipta dan terus berlangsung selama 70 (tujuh puluh) tahun setelah Pencipta meninggal dunia, terhitung mulai tanggal 1 Januari tahun berikutnya.

Nomor pencatatan :

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.

Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL

Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001



Kata Pengantar

Buku **CSS Uncover** ditujukan bagi rekan-rekan yang ingin mempelajari CSS mulai dari dasar hingga perkembangan terbaru CSS3.

Dalam dunia web programming, CSS adalah bahasa kode yang dipakai untuk mendesain tampilan web. Mulai dari mengubah warna teks, mengatur ukuran font, membuat efek bayangan hingga membuat animasi, semuanya bisa dilakukan dengan CSS.

Versi pertama buku **CSS Uncover** saya rilis di tahun 2015. Tidak terasa sudah 6 tahun berlalu dan di tahun 2021 ini saatnya mengupdate kembali materi yang ada.

Cukup banyak perubahan pada revisi kali ini, terutama tambahan bab tentang **CSS Flexbox** dan **CSS Grid**. Keduanya mengubah teknik perancangan layout web modern yang sebelumnya menggunakan property float.

Selain kedua bab tersebut, tambahan materi lain dipecah ke beberapa bab yang sudah ada, termasuk perubahan mini project di akhir buku.

Dibandingkan dengan HTML, CSS memang sedikit lebih kompleks. Namun saya berusaha menyusun materi yang mudah dipahami disertai banyak contoh kode program. Besar harapan semoga buku **CSS Uncover** ini bisa memandu rekan-rekan untuk menguasai teknik mendesain web modern. Sampai jumpa di bab terakhir :)

Asumsi / Pengetahuan Dasar

Sesuai dengan judul buku, **CSS Uncover** fokus membahas bahasa CSS. Untuk dapat memahami materi yang ada, saya berasumsi rekan-rekan sudah paham sedikit banyak tentang HTML, cara kerja web browser, serta cara menjalankan file HTML.

Idealnya, buku CSS Uncover dibaca sebagai lanjutan dari buku **HTML Uncover**. Namun tidak masalah jika materi HTML dipelajari dari sumber lain. Di website duniaIlkom juga tersedia beberapa tutorial dasar HTML.

Jika belum pernah mempelajari CSS, saya sangat sarankan membaca buku ini secara berurutan dari bab 1 hingga akhir karena semua materi saling terhubung satu sama lain.

Tips Menghapal Kode CSS

Ketika mempelajari buku ini, anda mungkin akan bertanya "bagaimana cara saya menghapal kode CSS ini?"

Tidak perlu dihapal! Cukup pahami cara penggunaannya saja. Jadikan buku **CSS Uncover** sebagai referensi jika nanti mulai terjun ke pembuatan web yang sebenarnya.

Tidak ada programmer yang hapal seluruh kode CSS. Tapi jika sudah paham cara penggunaannya, akan sangat mudah memahami kembali dengan melihatnya sekilas.

Sebagai contoh analogi, saya yakin sebagian besar dari kita pernah belajar **geometri**. Masih ingatkah dengan rumus menghitung keliling lingkaran?

Jika anda seperti saya (yang sudah puluhan tahun tamat SD), mustahil masih hapal rumus ini. Tapi jika ada yang tanya, tinggal googling sebentar dan rumusnya adalah $2\pi r$. Saya tidak perlu membaca lebih jauh apa itu π dan r karena sudah paham konsep dasarnya.

Begitu juga dengan programming, kode-kode program jumlahnya sangat banyak. CSS baru satu bidang, belum lagi PHP, JavaScript, MySQL, serta materi lanjutan seperti Bootstrap, Laravel, Code Igniter, React, dan Vue.

Jadi, silahkan nikmati "belajar coding". Tidak usah khawatir harus menghapal setiap kode. Saya pun masih sering membuka buku ini jika lupa sesuatu.

Contoh Kode Program

Seluruh contoh kode program yang ada di buku **CSS Uncover** bisa di download dari folder sharing Google Drive yang saya kirim pada saat pembelian: **belajar_css.zip**.

Sebagaimana yang sudah kita pelajari di HTML, halaman web lengkap diawali dengan kode HTML seperti `<!DOCTYPE html>` serta tag-tag HTML lain seperti `<head>`, `<title>` dan `<body>`. Agar menghemat tempat, pada buku ini kode pembuka HTML tersebut tidak selalu saya tulis. Di dalam file `belajar_html.zip` kode-kode ini akan dilengkapi kembali.

Penomoran baris (*line numbering*) pada contoh kode program berguna untuk memudahkan pembahasan. Jika ingin men-copy kode ini langsung dari eBook **pdf**, gunakan kombinasi tombol **ALT + tahan tombol mouse selama proses seleksi** agar *line numbering* tidak ikut di-copy. Namun ini hanya bisa dilakukan dari aplikasi pdf reader tertentu seperti **Adobe Acrobat Reader**.

Cara yang lebih disarankan adalah dengan mengetik ulang seluruh kode program yang ada supaya lebih cepat paham sekaligus bisa menghafal fungsi dari setiap kode. Jika setelah diketik ternyata tidak jalan, besar kemungkinan ada penulisan yang salah.

Di dalam programming, 1 saja karakter yang kurang apakah itu berupa titik, tanda koma, atau tanda " > ", kode program tidak akan jalan sempurna. Jika ini yang terjadi, coba samakan dengan file yang ada di dalam folder **belajar_css.zip**.

1. Berkenalan Dengan CSS

CSS adalah dunianya **web design**. Jika anda ingin mempelajari cara mendesain web, CSS mutlak harus dikuasai. Dalam bab pertama buku **CSS Uncover** ini kita akan membahas pengertian CSS, fungsi CSS, serta melihat sekilas contoh penggunaan CSS dalam proses pembuatan web.

1.1. Pengertian CSS

CSS adalah singkatan dari **Cascading Style Sheet**. Kata style dari kepanjangan ini menegaskan bahwa CSS berfungsi untuk mengatur gaya atau display tampilan (style) dari halaman web.

Namun CSS juga tidak bisa berdiri sendiri, ia perlu HTML yang menjadi komponen dasar dari web tersebut. Sebagaimana yang mungkin sudah anda ketahui, kode program untuk halaman depan web modern tersusun dari 3 komponen dasar: **HTML** untuk membuat struktur, **CSS** untuk tampilan, dan **JavaScript** untuk interaksi.

Jika halaman web diibaratkan sebuah bangunan, CSS adalah tampilan luar seperti warna dinding atau warna atap. Dengan CSS, kita bisa menukar warna dinding bangunan tanpa perlu mengubah struktur dasarnya (yang dibuat dari HTML).

Begitu pula dengan halaman web. Menggunakan CSS, kita bisa mengubah tampilan website tanpa perlu menyentuh kode HTML. Apabila saat ini warna website mayoritas merah, minggu depan bisa diubah menjadi biru hanya dengan menukar beberapa baris kode CSS saja.

Berikut pengertian CSS yang dikutip dari [wikipedia](#)¹:

"Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML".

Terjemahan bebasnya:

"Cascading Style Sheets (CSS) adalah bahasa style sheet yang digunakan untuk mengatur tampilan dari sebuah dokumen yang ditulis dengan bahasa markup seperti HTML".

Terdapat 2 istilah penting dari pengertian ini: **bahasa style sheet** (*style sheet language*) dan **bahasa markup** (*markup language*).

Style sheet language menjelaskan bahwa CSS adalah format bahasa khusus yang terdiri dari kumpulan kode untuk mengatur tampilan sebuah dokumen. Sebagaimana yang akan kita lihat

¹ <https://en.wikipedia.org/wiki/CSS>

dari sejarah CSS dalam bab selanjutnya, pada awal perkembangan web terdapat berbagai variasi *style sheet language*, dimana salah satunya adalah **CSS**.

Markup language merujuk kepada dokumen yang dibuat menggunakan tanda (*mark*). Salah satu contoh dari markup language adalah **HTML** (Hypertext Markup Language). Artinya CSS tidak hanya ditujukan untuk HTML saja, tapi juga bisa dipakai oleh bahasa markup lain seperti **XML** (Extensible Markup Language) dan **SVG** (Scalable Vector Graphics).

Kata **Cascade** dari kepanjangan **CSS** juga perlu kita bahas. Dalam kamus bahasa Inggris, cascade berarti "air terjun kecil, riam, jeram, mengalir/berpancaran ke bawah". Sehingga dapat diartikan bahwa *cascade* adalah "sesuatu yang mengalir dari atas ke bawah".

Di dalam CSS, style atau aturan tampilan yang dibuat bisa saja saling menimpa satu sama lain tergantung dari posisinya dan ke-spesifikasi kode CSS tersebut (nanti akan kita bahas dengan detail dalam bab khusus).

Sebagai contoh, jika pada baris pertama kode CSS terdapat perintah untuk mengubah warna paragraf menjadi biru, di baris kedua warna ini bisa ditimpakan dengan perintah yang sama. Perintah kedua akan "menang" karena ia berada di baris yang lebih bawah. Inilah salah satu penerapan dari konsep *cascading* di dalam CSS.

Sebagai kesimpulan, secara sederhana CSS bisa didefinisikan sebagai "*format bahasa khusus yang digunakan untuk mengatur tampilan dari halaman web*".

Sama seperti HTML, CSS bukanlah bahasa pemrograman komputer, tapi bahasa struktur (*structural language*). CSS terdiri dari kode-kode sederhana tanpa fitur pemrograman seperti perulangan, logika if, dll. Oleh karena itu CSS relatif lebih mudah untuk dipelajari.

1.2. Fungsi CSS

Seperti yang telah kita singgung sebelumnya, CSS berfungsi untuk mengatur tampilan (*style*) dari sebuah dokumen. Khusus dalam buku ini dokumen yang dimaksud adalah HTML.

Dengan CSS, kita bisa mengatur hampir seluruh tampilan HTML. Mulai dari warna teks, gambar background, besar font, posisi judul, hingga layout. Ditambah dengan update dari CSS3 yang membawa banyak fitur lanjutan seperti *color gradient*, *transitions*, dan *animations*.

Keuntungan lain dari CSS, ia bisa digunakan oleh banyak dokumen HTML sekaligus. Sebagai contoh, untuk website yang terdiri dari ribuan halaman, kode CSS yang diperlukan bisa ditempatkan pada 1 file saja. Dengan mengubah beberapa baris kode CSS pada file ini, tampilan seluruh website akan ikut berubah.

CSS juga memiliki fitur '*media type*' untuk mendeteksi tipe perangkat yang digunakan ketika mengakses halaman web, apakah itu dari layar komputer/smartphone, printer, screen reader,

dll. Dengan fitur ini, kita bisa membuat style yang berbeda-beda untuk setiap perangkat. Misalnya jika halaman akan di print, gambar background bisa dihapus agar menghemat tinta.

Pada CSS3, fitur media type disempurnakan lagi dengan 'media query'. Menggunakan media query, kita bisa membuat halaman web yang dapat menyesuaikan diri dengan ukuran layar. Tipe website seperti ini dikenal dengan istilah **Responsive Web Design (RWD)**, atau cukup disingkat dengan **Web Responsive**.

Selain hal di atas, masih banyak fitur canggih CSS lain yang tidak bisa didapat jika menggunakan HTML saja. Dalam buku ini kita akan membahas sebagian besar diantaranya, mulai dari dasar penulisan CSS, cara penggunaan selector dan property, tipe-tipe selector, hingga cara membuat web responsive.

Khusus untuk web responsive, Google telah memutuskan untuk memberi nilai tambah bagi website 'mobile friendly'. Dengan demikian, mau tidak mau kita harus menguasai cara membuat responsive web design mulai saat ini.

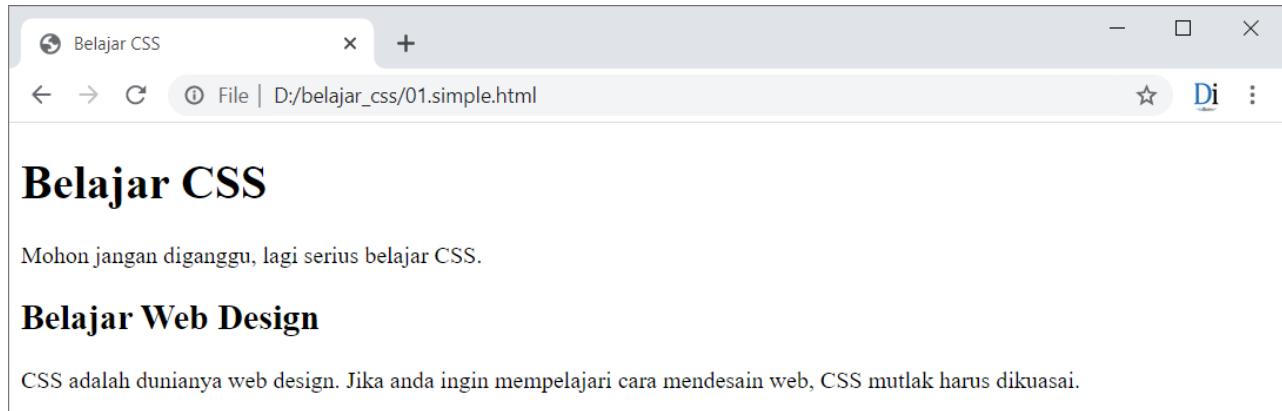
1.3. Contoh Penggunaan CSS

Agar perkenalan kita dengan CSS semakin lengkap, saya ingin memperlihatkan sekilas contoh penggunaan CSS di dalam dokumen HTML. Anda tidak perlu memahami kode CSS yang digunakan karena nanti akan kita bahas dengan lebih detail.

Perhatikan kode HTML berikut:

```
01.simple.html
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <h1>Belajar CSS</h1>
9      <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
10     <h2>Belajar Web Design</h2>
11     <p>CSS adalah dunianya web design.
12         Jika anda ingin mempelajari cara mendesain web,
13         CSS mutlak harus dikuasai.</p>
14 </body>
15 </html>
```

Kode di atas hanya file HTML sederhana yang terdiri dari tag `<h1>`, `<h2>` dan 2 buah tag `<p>`. Bagi yang sudah memahami HTML, tentunya tidak asing dengan fungsi dari setiap tag tersebut. Berikut tampilannya di dalam web browser Google Chrome:

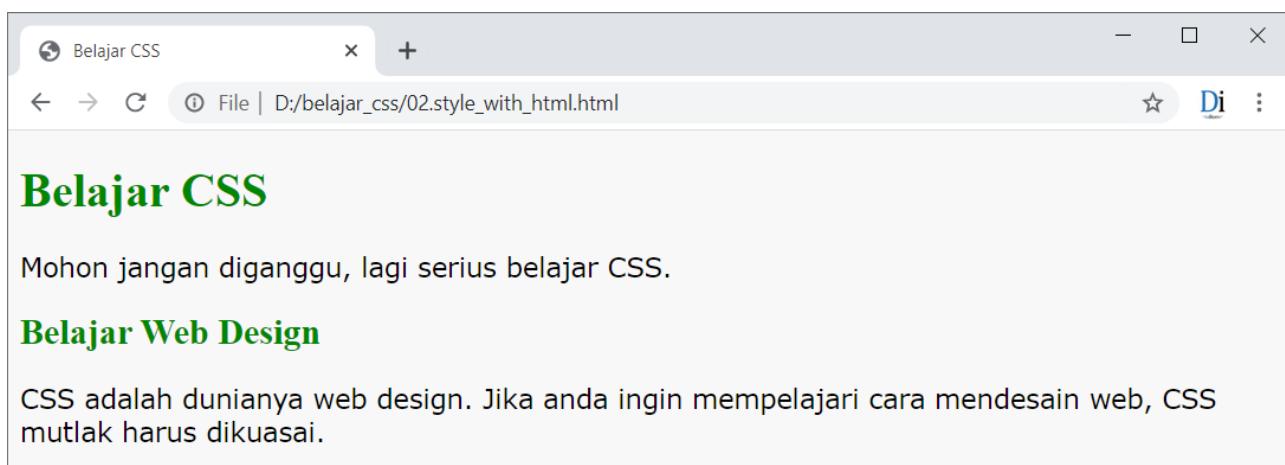


Gambar: Contoh halaman HTML Sederhana

Sekarang saya akan ubah design halaman HTML di atas. Terdapat 2 alternatif, menggunakan tag HTML atau dengan CSS. Jika menggunakan tag HTML, saya bisa memodifikasinya menjadi seperti berikut ini:

02.style_with_html.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6 </head>
7 <body bgcolor="#F7F7F7">
8   <h1><font color="green">Belajar CSS</font></h1>
9   <p><font face="verdana" size="4">Mohon jangan diganggu,
10  lagi serius belajar CSS.</font></p>
11  <h2><font color="green">Belajar Web Design</font></h2>
12  <p><font face="verdana" size="4">CSS adalah dunianya web design.
13  Jika anda ingin mempelajari cara mendesain web,
14  CSS mutlak harus dikuasai.</font></p>
15 </body>
16 </html>
```



Gambar: Hasil Style dengan HTML

Sekarang tampilan halaman sudah berubah, namun terdapat beberapa masalah dengan kode di atas. HTML yang seharusnya berfungsi untuk membuat struktur, juga dipakai untuk menangani tampilan web.

Apabila saya ingin menambah paragraf baru, harus ingat untuk selalu menulis tag `` serta atributnya (warna dan ukuran font). Bayangkan jika di dalam halaman tersebut akan ditambah 20 paragraf baru di waktu yang berbeda-beda.

Selain itu di HTML5, atribut yang berfungsi untuk membuat efek tampilan seperti `bgcolor` dan juga tag `` sudah *deprecated* (usang), dan tidak akan lolos validasi HTML5.

Menggunakan CSS, design yang sama bisa dibuat dengan kode yang lebih fleksibel:

03.style_with_css.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{
8              background-color:#F7F7F7;
9          }
10         h1,h2{
11             color:green;
12         }
13         p{
14             font-family: Verdana, Arial, Helvetica, sans-serif;
15             font-size: 18px;
16         }
17     </style>
18 </head>
19 <body>
20     <h1>Belajar CSS</h1>
21     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
22     <h2>Belajar Web Design</h2>
23     <p>CSS adalah dunianya web design.
24         Jika anda ingin mempelajari cara mendesain web,
25         CSS mutlak harus dikuasai.</p>
26 </body>
27 </html>
```

Perhatikan tambahan tag `<style>` di bagian `<head>` HTML. Inilah kode CSS yang dipakai pada halaman ini. Dengan CSS, masalah dari kode HTML sebelumnya bisa diatasi, yakni kode CSS sepenuhnya terpisah dari struktur HTML dan tidak saling tercampur.

Apabila saya ingin menambah 1, 2 atau 1000 paragraf baru menggunakan tag `<p>`, secara otomatis tag tersebut akan di design menyesuaikan dari kode CSS tanpa perlu mengubah paragraf satu per satu.

Jika di kemudian hari ingin menukar warna paragraf dari hijau menjadi biru, tinggal mengubah 1 baris kode CSS saja, dan seluruh teks akan ikut berubah.

Kode CSS di atas bisa 'diangkat' ke sebuah halaman khusus (external CSS), kemudian di-link kepada setiap halaman. Dengan cara ini, seluruh design website hanya butuh 1 file CSS saja.

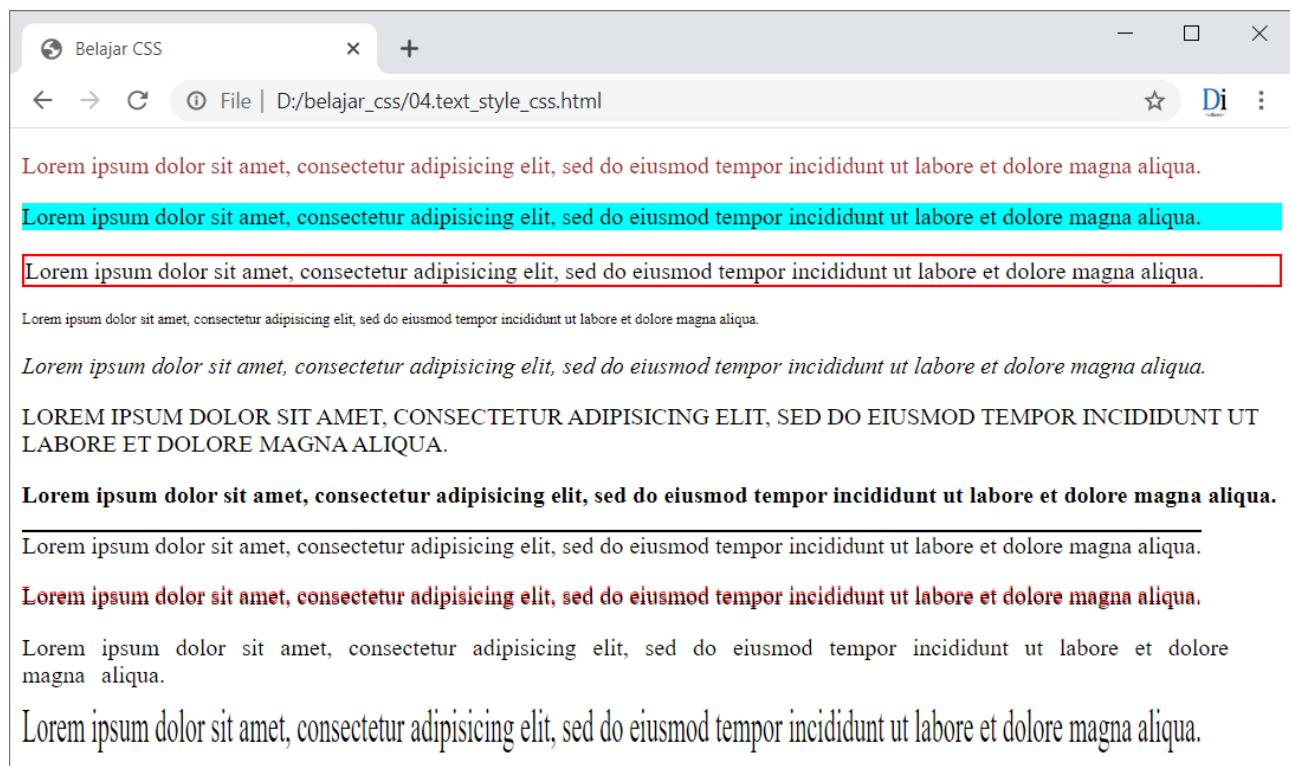
Sebagai contoh tambahan, dalam halaman HTML berikut ini saya tampilkan beberapa design teks yang bisa dihasilkan dari CSS:

04.text_style_css.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <p style="color:brown;">
9          Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
10         eiusmod tempor incididunt ut labore et dolore magna aliqua.
11     </p>
12     <p style="background-color:aqua;">
13         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
14         eiusmod tempor incididunt ut labore et dolore magna aliqua.
15     </p>
16     <p style="border: 2px solid red;">
17         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
18         eiusmod tempor incididunt ut labore et dolore magna aliqua.
19     </p>
20     <p style="font-size:10px;">
21         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
22         eiusmod tempor incididunt ut labore et dolore magna aliqua.
23     </p>
24     <p style="font-style:italic;">
25         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
26         eiusmod tempor incididunt ut labore et dolore magna aliqua.
27     </p>
28     <p style="text-transform: uppercase;">
29         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
30         eiusmod tempor incididunt ut labore et dolore magna aliqua.
31     </p>
32     <p style="font-weight:bold;">
33         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
34         eiusmod tempor incididunt ut labore et dolore magna aliqua.
35     </p>
36     <p style="text-decoration: overline;">
37         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
38         eiusmod tempor incididunt ut labore et dolore magna aliqua.
39     </p>
40     <p style="text-shadow: red 0 -2px;">
41         Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
42         eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

Berkenalan Dengan CSS

```
43 </p>
44 <p style="word-spacing: 7px;">
45 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
46 eiusmod tempor incididunt ut labore et dolore magna aliqua.
47 </p>
48 <p style="transform: scaleY(2);">
49 Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
50 eiusmod tempor incididunt ut labore et dolore magna aliqua.
51 </p>
52 </body>
53 </html>
```



Gambar: Berbagai variasi design teks menggunakan CSS

Tampilan design teks di atas (dan banyak efek lain) akan kita bahas secara detail sepanjang buku ini.

Dalam bab pembuka buku **CSS Uncover** ini kita telah membahas pengertian CSS dan melihat sekilas fungsi dan fitur dari CSS. Selanjutnya, saya akan mengajak anda mengenal lebih dalam sejarah lahirnya CSS serta melihat bagaimana perkembangan CSS hingga saat ini.

2. Sejarah dan Perkembangan CSS

Dalam bab kedua ini saya ingin mengajak anda melihat sejarah panjang perkembangan CSS mulai dari awal kemunculannya di tahun 1996 hingga CSS3 yang mendominasi dunia web desain saat ini. Setelah itu kita akan membahas apa saja fitur-fitur baru yang ada di CSS3.

2.1. Sejarah CSS

Kebutuhan akan Style Sheet Language

Sejak awal kemunculan HTML di tahun 1991, banyak keinginan dari web developer (pembuat web) untuk mengatur tampilan website. Perusahaan web browser menjawab hal ini dengan cara menambah kemampuan HTML, contohnya melalui tag .

Penggabungan fungsi HTML untuk struktur dan tampilan membuat HTML makin kompleks dan susah dikelola. Hal ini diperparah dengan implementasi kode yang berbeda-beda pada setiap web browser, karena saat itu belum ada sebuah standar resmi.

Melihat kondisi tersebut, berbagai web programmer mempertimbangkan suatu konsep pemisahan antara struktur web dengan tampilannya. Dari sekian banyak alternatif solusi, cara yang paling mungkin adalah menggunakan konsep *style sheet language*.

Style sheet language sebenarnya sudah ada sejak tahun 1980an ketika bahasa **SGML** (Standard Generalized Markup Language) dirilis untuk kali pertama. SGML merupakan 'induk' dari bahasa HTML yang kita gunakan sekarang. Dari SMGL-lah tanda kurung siku HTML berasal.

Awal Lahirnya CSS: Molis www-style

Tim Berners-Lee, ilmuwan yang merancang HTML juga melihat kebutuhan akan pemisahan struktur web dan struktur desain. Pada web browser **NeXT** yang dikembangkannya, Tim membuat sebuah konsep *style sheet* sederhana untuk memproses halaman website.

Akan tetapi, Tim Berners-Lee tidak pernah mempublikasikan *style sheet* ini karena ia berpendapat bahwa tampilan website sebaiknya ditentukan oleh web browser, bukan berada di tangan pembuat web.

Web browser lain kala itu juga menggunakan konsep *style sheet* yang sama, seperti web browser **Pei Wei's Viola** (1992) dan **Harmony browser** (1993). Pada kedua web browser ini, *style sheet* diatur oleh web browser itu sendiri.

Seperti yang bisa diperkirakan, kalangan web developer ingin agar kontrol tampilan halaman berada di mereka, bukan pada web browser saja. Sekitar tahun 1994, sekelompok programmer kemudian membentuk mailing list khusus (semacam forum) untuk membahas hal ini. Mailing list tersebut dinamakan **www-style**.

Diskusi dalam www-style banyak membahas konsep style sheet untuk HTML. Setelah beberapa waktu, dipilihlah 9 kandidat style sheet. Dari ke-9 pilihan ini, akhirnya mengerucut ke 2 konsep yang banyak menarik perhatian, yakni *Cascading HTML Style Sheets* dan *Stream-based Style Sheet Proposal (SSP)*.

Jika tertarik, mailing list www-style masih bisa diakses di alamat lists.w3.org/Archives/Public/www-style/, dan masih aktif hingga sekarang.

Kandidat pertama, **Cascading HTML Style Sheets** diusulkan oleh [Håkon Wium Lie](#)², seorang ilmuwan komputer asal Norwegia yang saat itu bekerja di **CERN** (tempat yang sama dimana HTML dikembangkan oleh Tim Berners-Lee). Ia mengimplementasikan Cascading HTML Style Sheets ke dalam web browser **Arena**.

Kandidat kedua, **Stream-based Style Sheet Proposal (SSP)** dikembangkan oleh [Bert Bos](#)³ yang di implementasikan ke dalam web browser **Argo**.

Standar CSS 1.0

Selang beberapa waktu kemudian, Håkon Wium Lie dan Bert Bos memutuskan bekerja sama menyempurnakan kedua konsep style sheet. Hasilnya, Cascading HTML Style Sheets dan Stream-based Style Sheet Proposal digabung menjadi sebuah bahasa style bernama **Cascading Style Sheets (CSS)**.

Kata 'HTML' dihapus dari Cascading HTML Style Sheets karena CSS ingin dirancang bukan untuk HTML saja, tapi juga bisa dipakai pada bahasa markup lain seperti XML.

Proposal CSS ini dipresentasikan di konferensi "Mosaic and the Web" (yang nantinya dikenal dengan konferensi WWW2) di Chicago, Illinois, Amerika Serikat pada tahun 1994 dan 1995.

Pada saat yang hampir bersamaan, W3C (saat itu baru berdiri 1 tahun) tertarik untuk ikut mengembangkan CSS. W3C kemudian membentuk badan khusus yang membahas CSS dan mengikutsertakan Håkon Wium Lie dan Bert Bos. Pada Desember 1996, **CSS level 1.0** resmi dipublikasikan dan menjadi standar W3C.

Walaupun CSS 1.0 dirumuskan oleh tim dari W3C, Håkon Wium Lie dan Bert Bos tetap dianggap sebagai "bapak" CSS. Keduanya juga aktif menulis buku tentang CSS, yang berjudul "Cascading Style Sheets: Designing for the Web".

2 https://en.wikipedia.org/wiki/H%C3%A5kon_Wium_Lie

3 https://en.wikipedia.org/wiki/Bert_Bos

Sebelumnya, pada bulan agustus 1996, Netscape Communication Corporation juga mengembangkan alternatif CSS yang disebut dengan **JavaScript Style Sheets (JSSS)**. Namun spesifikasi ini tidak pernah selesai dan akhirnya ditinggalkan.



Gambar: Håkon Wium Lie (kiri), Gijsbert "Bert" Bos (kanan)

Dukungan Web Browser untuk CSS 1.0

Sama seperti HTML, standar CSS yang dirilis W3C hanya "aturan rekomendasi". Maksudnya pembuat web browser boleh memilih apakah akan mengikuti standar ini atau tidak.

Sekurangnya butuh 3 tahun sejak CSS 1.0 diluncurkan hingga didukung penuh oleh web browser yang pada saat itu di dominasi oleh **Internet Explorer** dan **Netscape**.

Web browser Internet Explorer 5.0 yang dirilis tahun 2000 untuk sistem operasi Apple Macintosh adalah web browser pertama yang secara penuh telah mendukung standar CSS 1.0 (mengikuti lebih dari 99% standar yang ditetapkan).

Selain Internet Explorer dan Netscape, terdapat juga web browser alternatif: **Opera**. Walaupun memiliki pangsa pasar yang kecil, Opera paling aktif dalam menerapkan standar W3C.

Standar CSS 2.0

Sesaat setelah CSS 1.0 di rilis pada Desember 1996, W3C langsung bekerja untuk mengembangkan penerus CSS. **CSS 2.0** secara resmi menjadi standar W3C pada 12 May 1998.

Namun standar CSS 2.0 ini cukup rumit dan susah diimplementasikan oleh web browser. Oleh karena itu segera diadakan revisi untuk perbaikan yang dinamakan CSS 2.1.

Standar CSS 2.1

Standar CSS 2.1 merupakan versi revisi dari CSS 2.0. Setelah beberapa tahun pengembangan, pada 25 Februari 2004 CSS 2.1 direncanakan dirilis secara resmi, dimana dalam tahapan status W3C sudah sampai ke tahap "kandidat rekomendasi" (*candidate recommendation*).

Akan tetapi karena satu dan lain hal, CSS 2.1 kembali ditarik untuk direvisi ulang. Proses revisi ini butuh waktu beberapa tahun dan statusnya kembali menjadi "working draft". Kemudian diusulkan kembali menjadi candidate recommendation pada 19 Juli 2007.

Sekali lagi, karena masih butuh penambahan dan revisi, spesifikasi CSS 2.1 kembali diperbaiki dan butuh beberapa tahun lagi. Akhirnya pada tanggal 12 April 2011, **CSS 2.1** kembali masuk ke tahap *candidate recommendation* dan secara resmi menjadi standar pada 7 Juni 2011.

Seperti yang kita lihat, pengembangan CSS 2.1 yang bolak balik dari *working draft* ke *candidate recommendation* dan kembali lagi ke *working draft* memakan waktu hampir 7 tahun dan baru selesai di 2011.

Selama jangka waktu ini, web browser secara bertahap tetap mengimplementasikan standar CSS 2.1 yang masih belum selesai. Akibatnya terdapat beberapa perbedaan standar dari satu web browser dengan web browser lain.

Standar CSS 2.1 inilah yang menjadi standar resmi yang banyak dipakai oleh web browser sebelum lahirnya CSS3.

Standar CSS 3

Memetik pelajaran dari susahnya membuat spesifikasi CSS 2.1, W3C memakai konsep yang berbeda dalam mengembangkan kelanjutan CSS, yang segera dimulai setelah CSS 2.1 rilis.

Untuk CSS 3, W3C memutuskan memecah spesifikasi CSS menjadi modul-modul terpisah. Tujuannya agar setiap modul dapat dikembangkan secara independen tanpa harus menunggu modul lainnya selesai. Dengan cara seperti ini, fitur baru bisa hadir dengan lebih cepat.

Saat ini terdapat puluhan modul CSS 3 yang masing-masingnya dikembangkan secara terpisah.

Bagaimana cara penulisan CSS 3? Apakah 'CSS 3' atau 'CSS3'?

Dari situs W3C, cara penulisan yang digunakan adalah '**CSS3**' (tanpa spasi). Beberapa sumber menyebut bahwa nama resminya adalah: CSS level 3. Cara penulisan tanpa spasi ini sepertinya mengadopsi tren dimana HTML versi 5 juga ditulis tanpa spasi, yakni **HTML5**.

Standar CSS 4

Karena CSS 3 dikembangkan secara individu (modul), jadwal rilis setiap modul akan berbeda-beda. Bisa saja terdapat modul yang sudah selesai dan masuk ke tahap pengembangan, namun ada juga modul yang masih dalam tahap draft.

Sebagai contoh, modul "CSS Backgrounds and Borders" telah mencapai status candidate recommendation Level 3, bahkan saat ini *draft* untuk level 4 telah dirancang, itu artinya CSS 4!

Namun untuk modul lain seperti "CSS Tables" masih berada di tahap *draft* Level 3, dan mungkin masih perlu waktu lama untuk menjadi *candidate recommendation* Level 3.

Kedepannya CSS 3 masih terus dikembangkan dan modul-modul baru bisa dirilis dengan lebih cepat tergantung kebutuhan saat ini. Lebih lanjut tentang sejarah CSS ini bisa dibaca di: [wikipedia](#)⁴, dan [W3C: The CSS Saga](#)⁵.

Update: Dari [dokumen resmi W3C](#)⁶, dinyatakan bahwa tidak akan ada yang namanya CSS

4. Setiap modul bisa mencapai level 4 atau lebih, tapi CSS itu sendiri tidak akan disebut sebagai CSS 4. Istilah CSS 3 akan terus dipakai sebagai pembeda dengan versi non-modul CSS 2.1.

Teknik seperti ini mirip seperti konsep Living Standard yang dipakai oleh WHATWG untuk HTML5. Namun di masa depan bisa jadi akan ada evolusi lain dari penamaan CSS.

2.2. Perbedaan CSS 1.0, CSS 2.1 dan CSS3

Melihat dari sejarah CSS di atas, terdapat 3 standar spesifikasi CSS, yakni CSS 1.0, CSS 2.1 dan CSS3. Standar CSS 2.0 tidak saya masukkan karena langsung diganti oleh CSS 2.1.

Fitur dalam CSS 1.0

CSS 1.0 merupakan standar pertama CSS yang di rilis pada 17 Desember 1996. Di dalam spesifikasi ini dirumuskan beberapa konsep awal CSS:

- Font property, seperti *typeface* dan *emphasis*.
- Warna text, background, dan element lain.
- Property teks, seperti spasi antar text dan spasi antar huruf.
- Konsep margin, border, padding, dan aturan posisi element.

Hampir semua fitur CSS 1.0 ini sudah didukung web browser modern saat ini.

⁴ <https://en.wikipedia.org/wiki/CSS>

⁵ <https://www.w3.org/Style/LieBos2e/history/Overview.html>

⁶ <https://www.w3.org/TR/CSS/#css-level-4>

Fitur dalam CSS 2.1

CSS 2.1 yang mendapat status *recommendation* pada 7 Juni 2011, memiliki banyak fitur baru sebagai tambahan dari CSS 1.0. Beberapa diantaranya:

- Konsep posisi element seperti *absolute*, *relative*, dan *fixed*.
- Pengaturan z-index element.
- Penggunaan media type.
- Perbaikan konsep box model.

Walaupun butuh waktu yang panjang, saat ini web browser modern mayoritas juga sudah mendukung penuh standar CSS 2.1.

Fitur dalam CSS 3

Kemunculan CSS3 yang hampir bersamaan dengan HTML5 membuat efek marketing CSS3 menjadi sangat terasa. Penguasaan akan CSS3 menjadi nilai jual tersendiri. Beberapa fitur baru yang ditambahkan pada CSS3 adalah:

- Media query.
- Transparansi warna dengan opacity dan alpha channel.
- Rounding corner.
- Transformasi.
- Animasi.

Saat ini CSS3 dikembangkan sebagai "living standar" dan terus melahirkan fitur-fitur baru. Karena CSS3 mengadopsi sistem modul, Tidak ada satu standar yang bernama "CSS 3.0". CSS3 adalah kumpulan dari berbagai modul

Tahapan Status Spesifikasi W3C

Sebelum membahas modul CSS3, ada baiknya kita memahami tahapan status spesifikasi yang berlaku di W3C. Ini diperlukan karena modul CSS3 dikembangkan secara terpisah dan memiliki status yang berbeda-beda.

Dalam mengembangkan sebuah standar spesifikasi, W3C memiliki 5 tahap: **Working Draft**, **Last Call**, **Candidate Recommendation**, **Proposed Recommendation**, dan **Recommendation**.

W3C adalah organisasi yang membuat standar spesifikasi HTML, CSS dan hal lain terkait teknologi web. W3C beranggotakan berbagai programmer dari seluruh dunia, terutama pihak pengembang web browser dari Google, Apple, Mozilla, Microsoft dan Opera. Web resmi W3C beralamat di <https://www.w3.org>⁷.

⁷ <https://www.w3.org>

W3C tidak ada hubungan dengan web w3schools.com yang berisi materi-materi belajar web programming. Kemungkinan besar w3schools sengaja menggunakan nama yang mirip untuk "nompang tenar" dari nama besar W3C.

Tahap Working Draft

Fase **working draft** adalah tahap pertama dari sebuah spesifikasi di W3C. Pada tahap ini, berbagai programmer dari seluruh dunia saling bekerjasama dan memberikan ide-ide terkait hal yang ingin dibuat.

Berbagai aspek teknis akan di uji coba dan dimasukkan ke dalam draft. Dalam tahap inilah proses pembuatan sebuah standar dilakukan.

Tahap Last Call

Apabila draft yang dirancang dianggap sudah cukup, spesifikasi working draft akan masuk ke fase **last call**. Tahap ini sebenarnya hanya pengumuman bahwa penulisan draft telah selesai.

Biasanya tahapan last call memiliki jangka waktu yang apabila telah lewat statusnya otomatis berubah menjadi *candidate recommendation*. Selama berstatus last call, perubahan spesifikasi masih dimungkinkan.

Tahap Candidate Recommendation

Jika deadline tahap last call selesai, sebuah spesifikasi masuk ke fase **candidate recommendation**. Dalam tahap ini sebuah modul/spesifikasi bisa dikatakan sudah relatif stabil. Umumnya implementasi spesifikasi sudah dilakukan secara terbatas oleh web browser.

Perubahan yang bisa dilakukan dalam tahap ini hanyalah modifikasi minor seperti perbaikan bug. Jika butuh perombakan besar, statusnya harus kembali mundur ke working draft seperti yang terjadi pada kasus CSS 2.1.

Tahap Proposed Recommendation

Sama seperti last call, fase **proposed recommendation** hanya pengumuman bahwa spesifikasi akan segera disahkan. Biasanya dalam tahap ini spesifikasi sudah sepenuhnya selesai.

Tahap Recommendation

Recommendation adalah tahap akhir dari sebuah spesifikasi. Dengan status ini, sebuah standar baru siap dipakai dan secara resmi dinyatakan selesai oleh W3C. Jika ditemukan kesalahan/bug maka perbaikan akan dilakukan ke dalam versi berikutnya, seperti CSS 2.0 yang segera diperbaiki oleh CSS 2.1.

Kelima langkah proses standarisasi W3C ini bisa makan waktu mulai dari hitungan bulan, tahu, bahkan lebih. Sebagai contoh, CSS 2.1 baru selesai setelah pengembangan selama 17 tahun (1994–2011). Selain itu, sebuah spesifikasi juga bisa 'mundur' apabila diperlukan perubahan besar, yang sering adalah dari candidate recommendation kembali menjadi working draft.

2.3. Modul-modul dalam CSS3

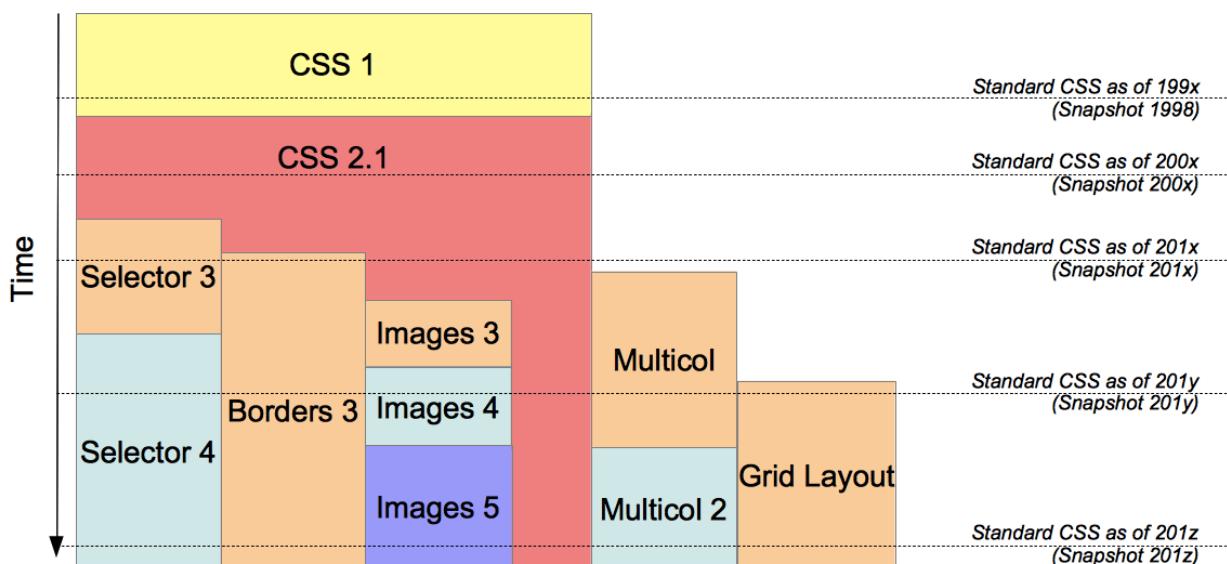
Dalam penjelasan sebelumnya beberapa kali saya menyenggung konsep modul dari CSS3. Tapi sebenarnya apa fungsi dari modul-modul ini? Dan apa pengaruhnya bagi kita sebagai web desainer?

Sebelum CSS3, pengembangan standar CSS dilakukan dalam 1 lembar spesifikasi yang terdiri dari seluruh aspek CSS. Konsep ini sebenarnya memudahkan kita dan tim pembuat web browser, yakni cukup mengikuti 1 standar spesifikasi dan selesai.

Akan tetapi konsep kesatuan seperti ini ternyata susah untuk dikembangkan. Jika terdapat masalah pada 1 komponen saja, pengembangan seluruh spesifikasi akan terhenti. Inilah yang terjadi ketika W3C mengembangkan CSS 2.1. Spesifikasi CSS 2.1 harus bolak balik dari *candidate recommendation* ke *working draft* dan kembali lagi ke *candidate recommendation*.

Agar pengembangan CSS tidak mengalami hal yang sama, W3C memutuskan bahwa CSS3 akan dipecah menurut fungsi dan perannya masing-masing.

Sebagai contoh, pengembangan modul warna (color) yang mencakup warna teks, warna background, dll, bisa dipisah dengan pengembangan selector. Jadi ketika spesifikasi tentang warna sudah selesai, modul warna bisa segera disahkan untuk menjadi standar tanpa harus menunggu spesifikasi selector selesai.



Gambar: Pengembangan CSS3 dipecah menjadi beberapa modul (sumber: developer.mozilla.org)

Tabel berikut berisi progress dari pengembangan berbagai modul CSS3:

Module	Specification title	Status	Date
css3-background	CSS Backgrounds and Borders Module Level 3 ⁸	Candidate Rec.	Dec 2020
css3-box	CSS Box Model Module Level 3 ⁸	Candidate Rec.	Dec 2020
css-cascade-3	CSS Cascading and Inheritance Level 3 ⁸	Recommendation	Feb 2021
css3-color	CSS Color Module Level 3 ⁸	Recommendation	Jun 2018
css3-content	CSS Generated Content Module Level 3 ⁸	Working Draft 2	Aug 2019
css3-fonts-3	CSS Fonts Module Level 3 ⁸	Recommendation	Sep 2018
css3-gcpm	CSS Generated Content for Paged Media Module ⁸	Working Draft	May 2014
css3-layout	CSS Template Layout Module ⁸	Note	Mar 2015
css3-mediaqueries	Media Queries ⁸	Recommendation	Jun 2012
mediaqueries-4	Media Queries Level 4 ⁸	Candidate Rec.	Jul 2020
css3-multicol	Multi-column Layout Module Level 1 ⁸	Working Draft	Feb 2021
css3-page	CSS Paged Media Module Level 3 ⁸	Working Draft	Oct 2018
selectors-3	Selectors Level 3 ⁸	Recommendation	Nov 2018
selectors-4	Selectors Level 4 ⁸	Working Draft	Nov 2018
css3-ui	CSS Basic User Interface Module Level 3 (CSS3 UI) ⁸	Recommendation	Jun 2018

Gambar: Tahap pengembangan berbagai modul CSS3 (sumber: en.wikipedia.org/wiki/CSS)

Jika tertarik, anda bisa mengunjungi halaman [CSS current work⁸](https://www.w3.org/Style/CSS/current-work) dari W3C untuk versi yang lebih update mengenai pengembangan modul-modul CSS3. Di halaman tersebut dapat dilihat secara langsung status tahap pengembangan dari setiap modul.

Dalam bab ini kita telah melihat sejarah CSS hingga memahami konsep modul dari CSS3. Berikutnya akan disambung dengan persiapan aplikasi yang diperlukan untuk mulai menulis kode CSS, yakni Web Browser dan Text Editor.

⁸ <https://www.w3.org/Style/CSS/current-work>

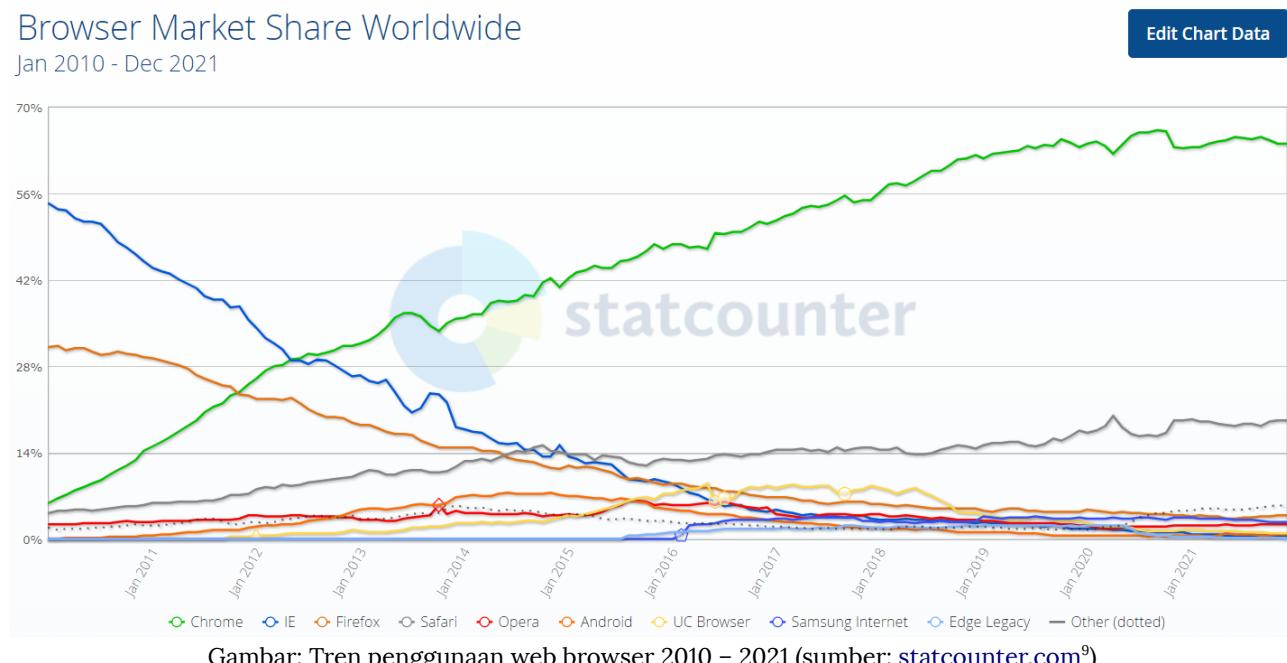
3. Web Browser dan Text Editor

Dalam bab ini akan dibahas tentang web browser dan text editor sebagai persiapan untuk menulis kode CSS.

3.1. Memilih Web Browser

Web browser adalah aplikasi yang dipakai untuk menampilkan halaman web. Sangat penting bagi kita sebagai web developer untuk memastikan kode yang ditulis bisa tampil sempurna di semua web browser, atau setidaknya tampil baik di mayoritas web browser yang mayoritas dipakai pengunjung.

Berdasarkan data dari statcounter.com, **Google Chrome** adalah web browser yang paling banyak dipakai saat ini. Berikut grafik pangsa pasar web browser dari 2010 hingga akhir 2021:



Gambar: Tren penggunaan web browser 2010 – 2021 (sumber: [statcounter.com](https://gs.statcounter.com/browser-market-share#monthly-201001-202012)⁹)

Terlihat penggunaan web browser Google Chrome sangat dominan (lebih dari 60%). Artinya, penting bagi kita merancang kode yang tampil baik di Google Chrome.

Selain Google Chrome, masih ada web browser lain seperti Mozilla Firefox, Apple Safari, Opera, dan Microsoft Edge. Untungnya, web browser ini sudah mengikuti standar kode dari

⁹ <https://gs.statcounter.com/browser-market-share#monthly-201001-202012>

W3C. Cukup jarang kasus tampilan web yang berbeda antara satu web browser dengan web browser lain seperti era tahun 2000-an. Kecuali untuk beberapa kode CSS3 terbaru yang belum final.

Saya sangat sarankan untuk menginstall beberapa web browser untuk proses testing. Pilihannya adalah Google Chrome, Mozilla Firefox, Opera, dan Microsoft Edge atau Apple Safari (tergantung sistem operasi yang digunakan). Ini untuk memastikan tampilan web yang kita rancang bisa tampil sempurna di mayoritas web browser.

Faktor lain yang juga harus dipertimbangkan adalah pengguna smartphone. Jumlah pengunjung yang mengakses web dari smartphone saat ini sudah mengalahkan pengunjung dari desktop / PC.

Banyaknya pengunjung dari smartphone mengharuskan kita merancang halaman web yang tampil baik di layar kecil. Salah satu solusinya adalah menggunakan teknik **web responsive**.

3.2. Mengenal Web Browser Layout Engine

Ketika membahas web browser dan CSS, tidak ada salahnya kita berkenalan dengan *web browser layout engine*.

Layout Engine adalah sebuah komponen khusus di dalam web browser yang bertanggung jawab menangani tampilan halaman web. *Layout engine* inilah yang membaca kode CSS dan memprosesnya menjadi halaman website. Layout engine yang baik harus mengikuti standar spesifikasi CSS yang ditetapkan oleh W3C.

Berikut daftar layout engine yang digunakan oleh 5 web browser populer:

- Google Chrome: Webkit (hingga 2013), Blink (dari 2013 ke atas)
- Mozilla Firefox: Gecko
- Apple Safari: Webkit
- Opera: Presto (hingga 2013), Blink (dari 2013 ke atas)
- Microsoft Edge: EdgeHTML (hingga 2018), Blink (dari 2018 ke atas)

Bagi kita sebagai web developer, sebenarnya tidak perlu ambil pusing dengan layout engine. Namun dalam kasus tertentu, ini perlu menjadi perhatian. Beberapa fitur CSS bisa tampil berbeda tergantung versi layout engine yang digunakan.

Sepanjang mempelajari CSS, kita akan menemui materi yang menyebut web browser berdasarkan layout enginennya. Sebagai contoh, browser *blink based* berarti web browser yang menggunakan **blink** sebagai layout engine, yakni Chrome, Opera dan Edge. Sedangkan web browser *gecko based* berarti web browser Mozilla Firefox.

Engine	Status	Embedded in
WebKit	Active	Safari browser, plus all browsers hosted on the iOS App Store.
Blink	Active	Google Chrome and other web browsers based on Chromium, such as Microsoft Edge, Opera, and Brave.
EdgeHTML	Active	Universal Windows Platform apps; formerly in the Edge browser ^[1] .
Gecko	Active	Firefox browser and Thunderbird email client, plus forks like SeaMonkey and Waterfox.
KHTML	Active	Konqueror browser
Presto	Discontinued	Formerly used as Opera's browser engine, prior to the migration to Chromium and Blink.
Trident	Discontinued	Internet Explorer and versions of Microsoft Outlook prior to Outlook 2007.

Gambar: Jenis jenis web browser layout engine (sumber: [wikipedia](#))

Konsep Modul dalam Web Browser

Pada awalnya, aplikasi web browser dikembangkan sebagai *monolithic application*, yaitu semua kode program dibuat dari awal hingga akhir dalam satu kesatuan. Arsitektur seperti ini membuat aplikasi tidak fleksibel karena ketika ingin melakukan update, seluruh kode program harus dibuat kembali dari awal.

Saat ini pengembang web browser mulai mengadopsi konsep *multi-tier architecture*, dimana setiap komponen terdiri dari modul-modul terpisah yang saling terhubung. Konsep ini membuat perkembangan aplikasi web browser menjadi lebih efisien.

Daripada membuat semuanya dari nol, web browser bisa menggunakan modul-modul yang sudah tersedia atau saling bekerjasama mengembangkan modul baru. Contohnya layout engine **Blink** yang sama-sama di kerjakan oleh Google, Opera dan Microsoft.

3.3. Dukungan Web Browser untuk CSS3

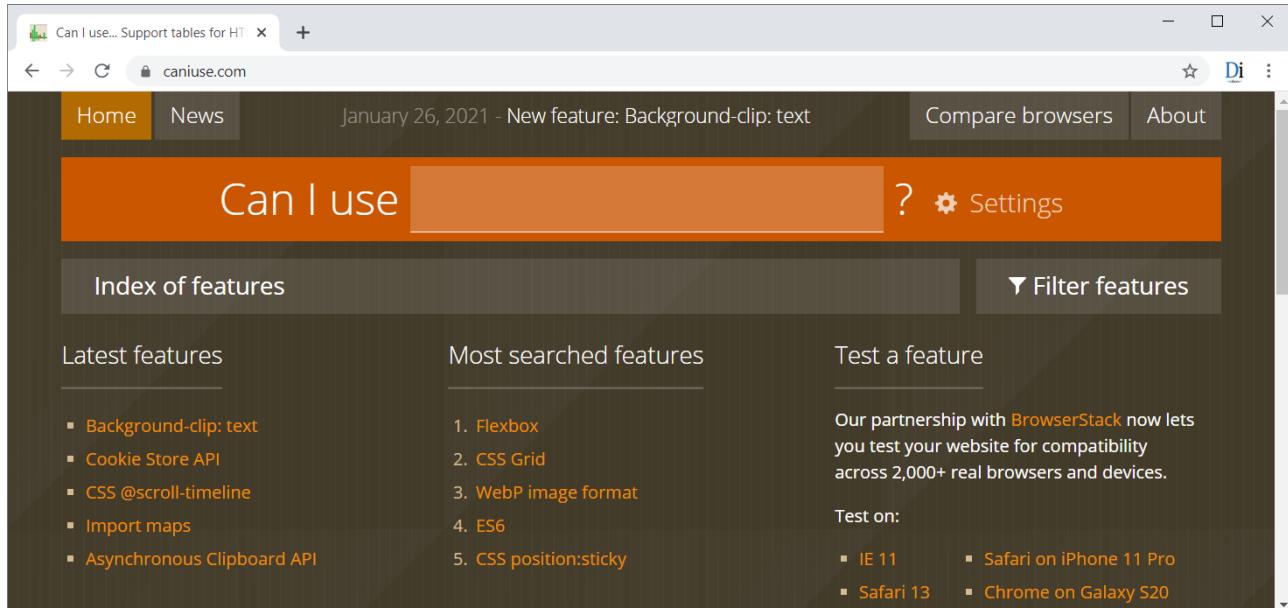
Pada bab sebelumnya kita telah bahas konsep modul di dalam CSS3. Melihat dari website W3C, tidak semua modul ini telah resmi menjadi standar. Banyak diantaranya yang masih dalam tahap *working draft*, dan masih berada di awal pengembangan.

Walaupun masih dalam status *working draft*, web browser modern sudah banyak yang mendukung modul-modul ini, terlepas dari status spesifikasinya.

Namun lain halnya dengan web browser lawas. Teknologi CSS3 baru booming beberapa tahun belakangan, oleh karena itu tidak semua web browser mendukung CSS3. Web browser "tua" (terutama Internet Explorer), belum mendukung CSS3. Tapi untungnya, populasi web browser tersebut sudah sangat jarang.

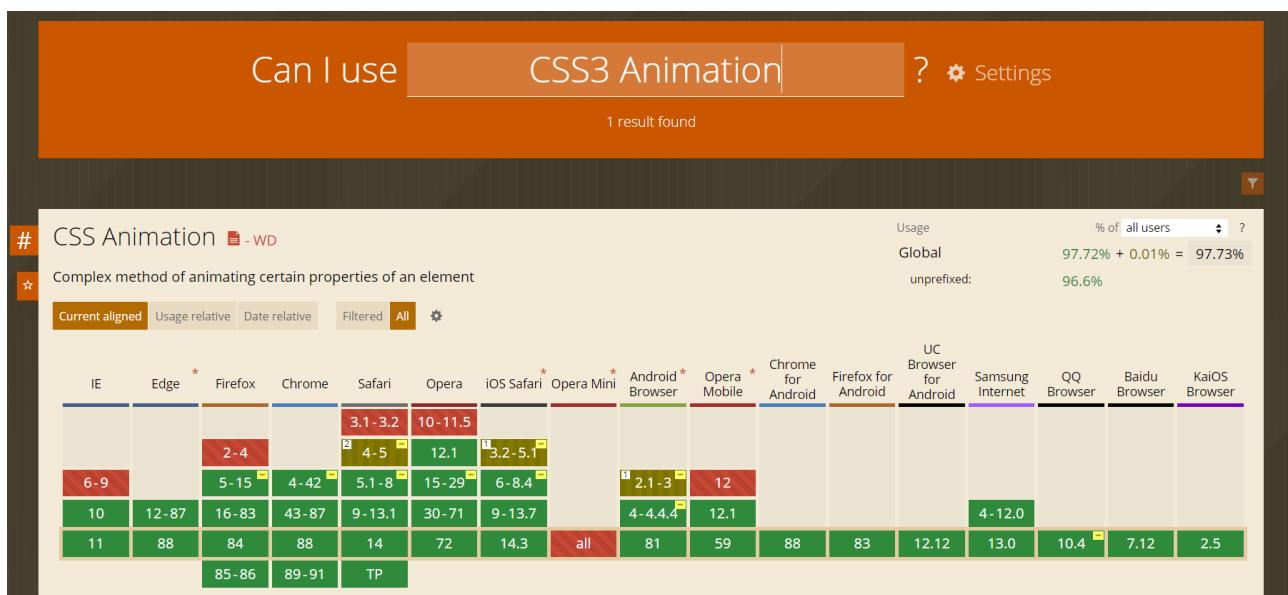
Mencari tahu apakah sebuah web browser telah mendukung CSS3 atau tidak bukanlah hal yang mudah. CSS3 terdiri dari berbagai modul, sehingga sangat mungkin satu modul bisa berjalan di web browser tertentu, tapi belum didukung oleh web browser lain. Ditambah lagi terdapat perbedaan versi untuk web browser yang sama.

Web www.caniuse.com¹⁰, bisa membantu kita menggali informasi ini. Sesuai dengan namanya, **Can I Use** dipakai untuk bertanya apakah sebuah fitur CSS3 (dan HTML5) telah tersedia, dan web browser apa saja yang telah mendukung fitur tersebut.



Gambar: Tampilan web www.caniuse.com

Sebagai contoh, jika saya ingin memeriksa web browser apa saja yang telah mendukung modul CSS3 Animation. Tinggal ketik "CSS3 Animation", dan akan tampil jenis serta versi web browser yang mendukung modul ini. Informasi yang disajikan sangat penting ketika kita ingin menggunakan modul CSS3 yang relatif baru.



Gambar: Mencari informasi web browser apa saja yang mendukung CSS3 animation

10 <https://www.caniuse.com>

3.4. Text Editor

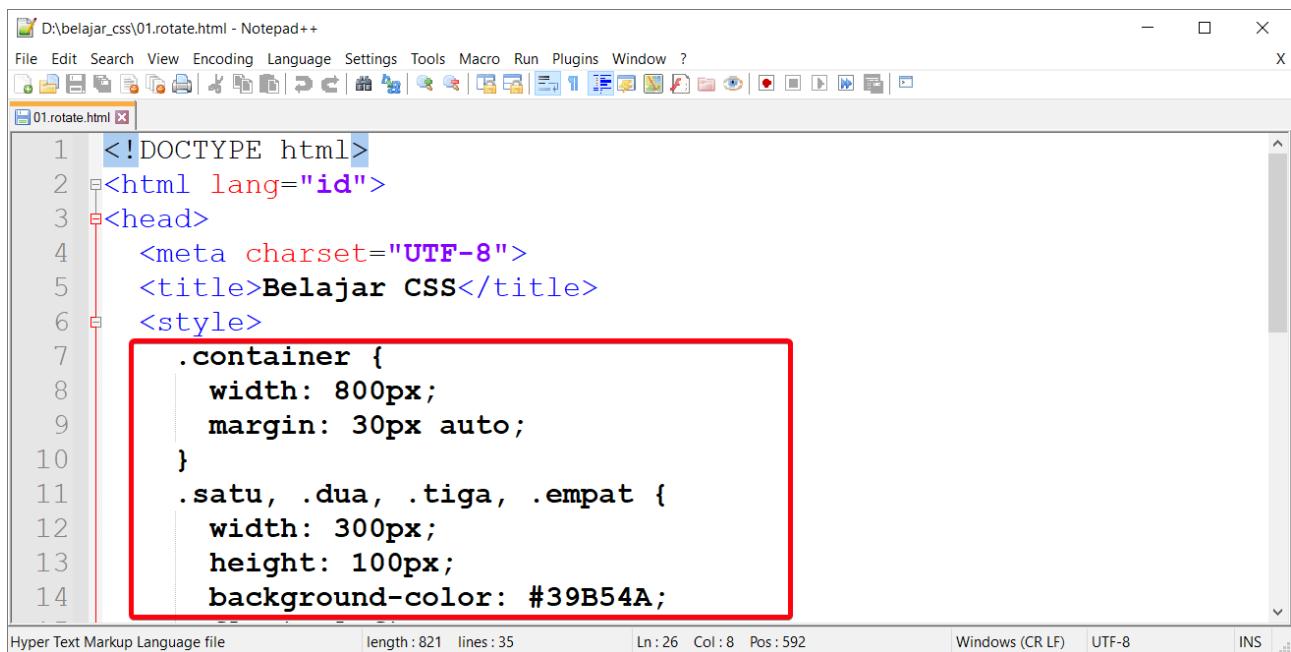
Sama seperti HTML, kode CSS terdiri dari teks biasa. Untuk membuatnya bisa menggunakan teks editor apapun yang dirasa cocok. Dalam buku pertama duniaIlkom: **HTML Uncover**, saya merekomendasikan **Notepad++** dan **Visual Studio Code**.

Notepad++

Notepad++ adalah teks editor 'generic' yang bisa dipakai untuk hampir seluruh bahasa pemrograman. Selain ringan, aplikasi ini sangat populer di kalangan web programmer.

Karena sifatnya yang sederhana, Notepad++ juga memiliki beberapa kekurangan. Sebagai contoh, di dalam Notepad++ kita harus memilih salah satu bahasa pemrograman yang akan menjadi bahasa utama. Bahasa pemrograman utama inilah yang akan di-warnai (fitur *syntax highlighting*).

Jika di dalam sebuah halaman terdapat 2 bahasa seperti HTML dan CSS, Notepad++ tidak bisa mewarnai keduanya, seperti contoh berikut:



```

D:\belajar_css\01.rotate.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
01.rotate.html

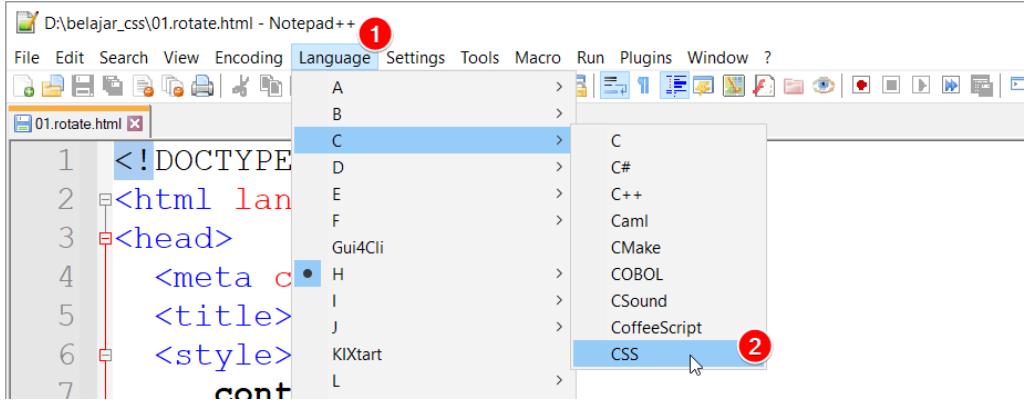
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .container {
8       width: 800px;
9       margin: 30px auto;
10      }
11      .satu, .dua, .tiga, .empat {
12        width: 300px;
13        height: 100px;
14        background-color: #39B54A;

```

The screenshot shows the Notepad++ interface with a file named "01.rotate.html". The code consists of HTML and CSS. The CSS part (lines 7-14) is highlighted with a red rectangle, indicating that Notepad++ is unable to correctly syntax-highlight the CSS code within the HTML file.

Gambar: Notepad++ tidak bisa 'mewarnai' kode CSS di dalam file HTML

Agar fitur *syntax highlighting* aktif untuk kode CSS, kita harus mengubah bahasa utama dari HTML menjadi CSS, namun kali ini giliran kode HTML yang tidak memiliki fitur *syntax highlighting*:



Gambar: Mengubah bahasa utama Notepad++

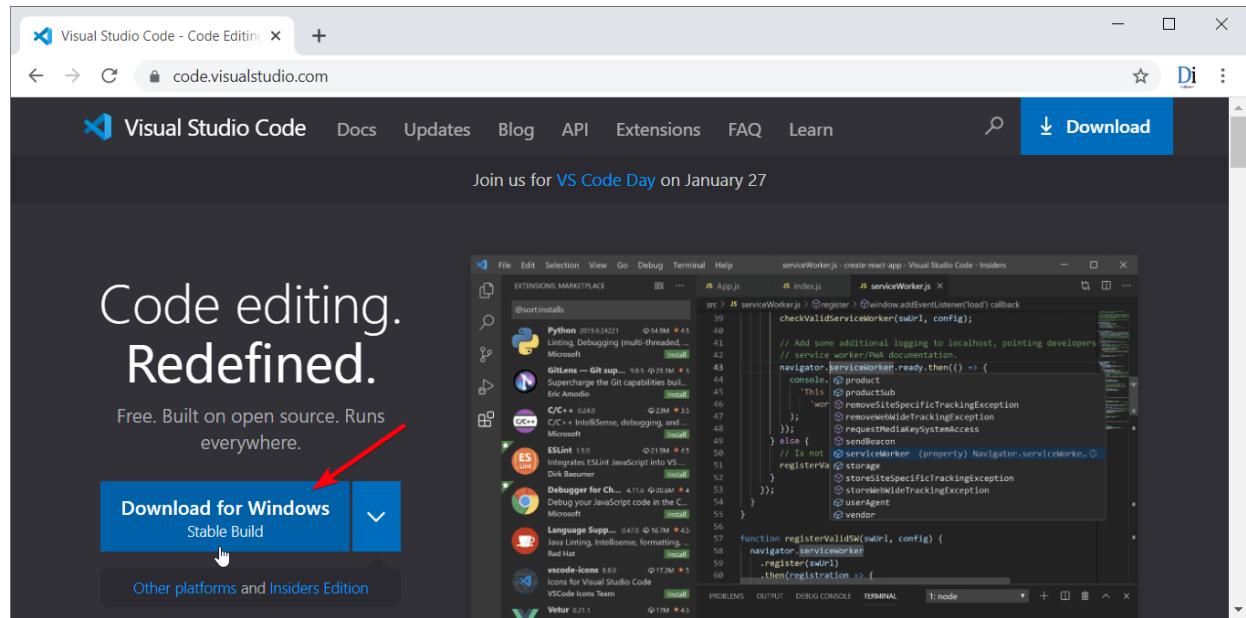
Meskipun dengan keterbatasan ini, saya masih sering menggunakan Notepad++ karena sangat ringan. Saat buku ini di update, versi terakhir Notepad ++ adalah versi 4.9.2 yang berukuran hanya 3,9 MB.

Notepad++ bisa di download dari web resminya di notepad-plus-plus.org/downloads¹¹.

Visual Studio Code

Selain Notepad++, teks editor lain yang saya sarankan adalah **Visual Studio Code** (sering disingkat sebagai **VS Code**). Teks editor ini juga bisa digunakan dengan gratis di: code.visualstudio.com. Saat buku ini saya revisi, versi terakhir adalah 1.52 yang berukuran sekitar 60 MB.

Untuk mendapatkan file installer, langsung saja klik tombol **Download** di sisi kiri web VS Code. Atau jika 'tebakan' website VS Code salah, bisa buka menu Download untuk menyesuaikan dengan sistem operasi yang dipakai (tersedia untuk OS Windows, Linux dan Mac).

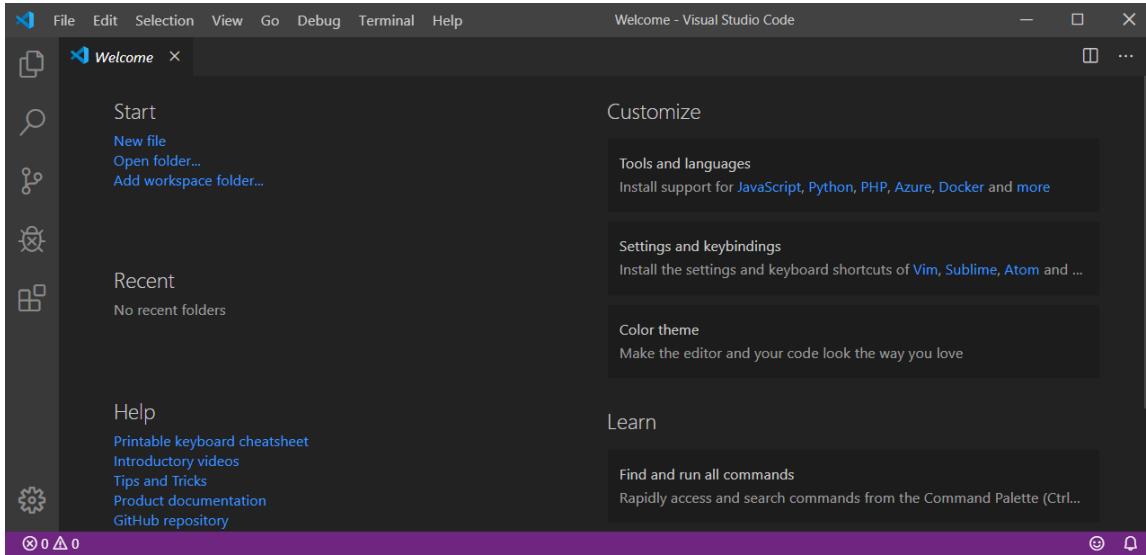


Gambar: Download aplikasi VS Code

11 <https://notepad-plus-plus.org/downloads>

Web Browser dan Text Editor

Silahkan ikuti proses instalasi seperti biasa, biarkan semua pengaturan dalam keadaan default.



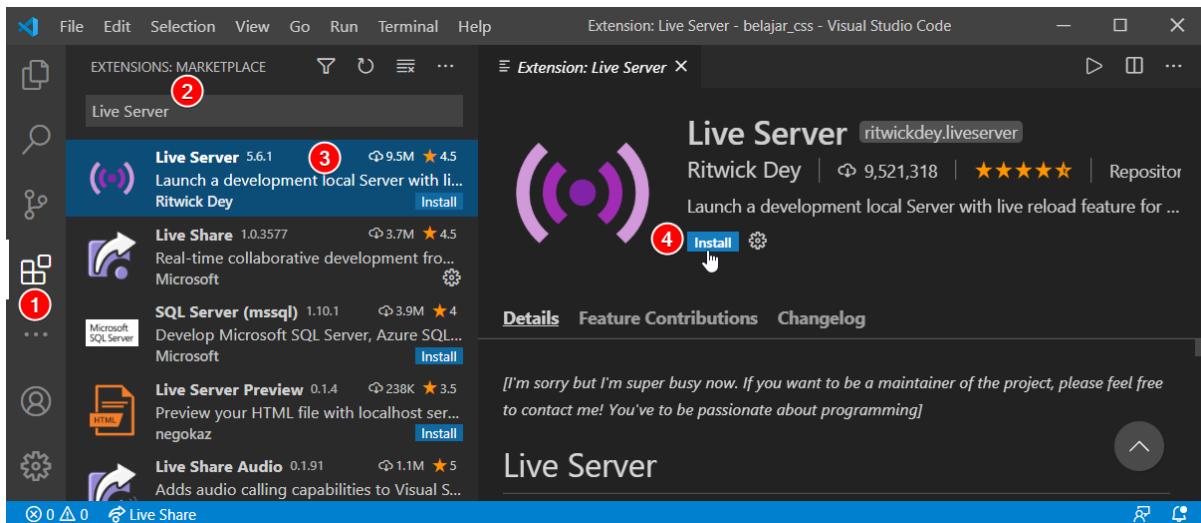
Gambar: Tampilan awal VS Code

VS Code terlihat lebih modern daripada Notepad++. Selain tampilan, keunggulan lain dari VS Code adalah banyaknya extension atau plugin yang bisa kita tambah.

Salah satu extension yang sangat menarik untuk proses pembuatan kode HTML dan CSS adalah **Live Server**. Extension ini menyediakan fitur *live update*, yakni pada saat file HTML disimpan (save), hasilnya langsung ter-update ke web browser tanpa perlu men-klik tombol *refresh* atau tombol *reload* di web browser.

Extension Live Server ini tidak wajib di install, hanya sekedar melihat fitur tambahan yang tersedia di VS Code.

Untuk menginstall extension ke dalam VS Code, klik shortcut **Extension** dari sidebar kiri (1), lalu ketik "Live Server" (2) di kotak pencarian:

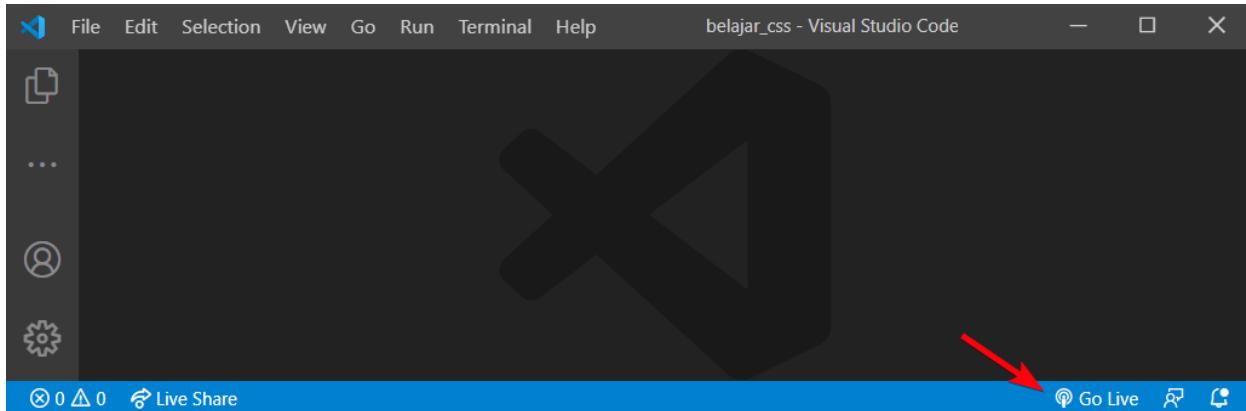


Gambar: Cara install Extension Live Server

Web Browser dan Text Editor

Dari hasil pencarian yang tampil, klik extension **Live Server** (3), lalu klik tombol Install di sebelah kanan (4). Proses Instalasi akan berjalan beberapa saat.

Setelah selesai, tutup jendela instalasi extension dan di sudut kanan bawah akan tampil icon **Go Live** yang menandakan extension **Live Server** sudah stand by dan siap digunakan.



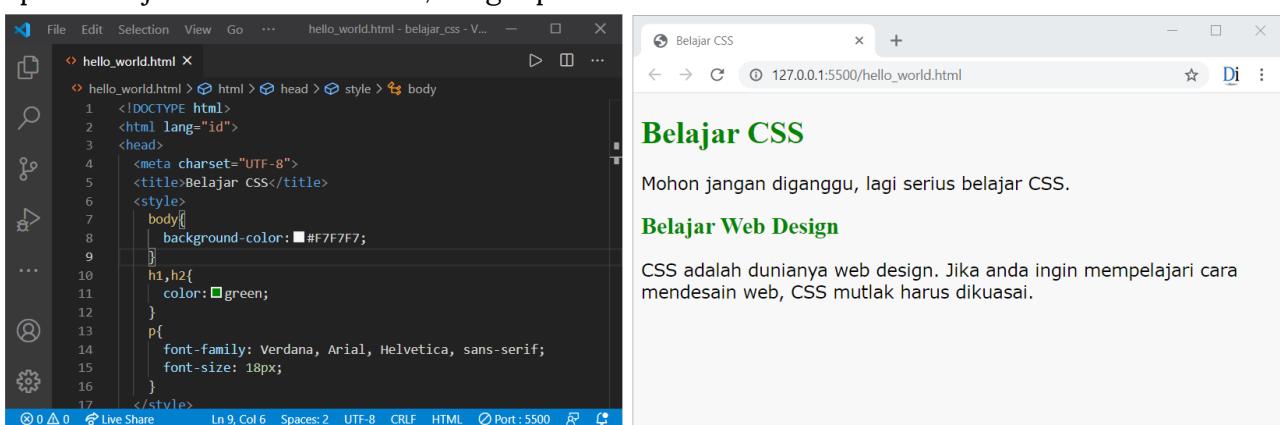
Gambar: Icon Go Live yang menandakan Live Server siap dipakai

Terdapat beberapa cara menjalankan **Live Server**. Yang paling praktis adalah buat sebuah file HTML, lalu klik icon **Go Live** di sudut kanan bawah.

Sesaat kemudian web browser akan terbuka yang langsung menampilkan file saat ini. Web browser yang dipakai adalah web browser *default* yang tersedia di komputer. Jika ingin menggunakan web browser lain, tinggal copy paste alamat URL di web browser.

Live Server ini akan menjalankan server lokal dengan menggunakan port khusus seperti http://127.0.0.1:5500/hello_world.html. Angka 5500 adalah nomor port yang dipakai.

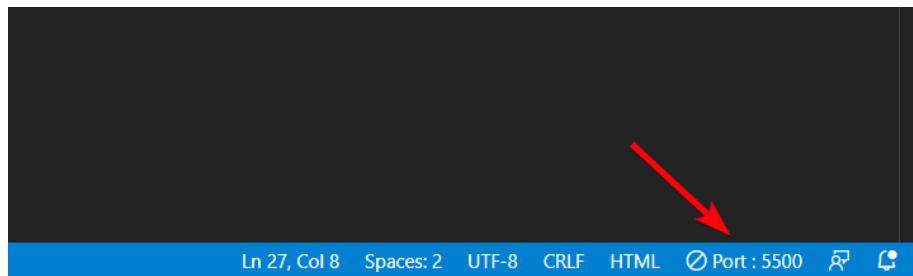
Untuk menguji fitur *live update*, atur agar jendela VS Code berdampingan dengan jendela web browser dalam 1 layar, lalu ketik beberapa kode HTML ke dalam editor VS Code. Pada saat kita menyimpan file tersebut (misalnya dengan menekan tombol CRTL + S), hasilnya langsung di update ke jendela web browser, sangat praktis!



Gambar: Fitur live update dengan extension Live Server di VS Code

Untuk menghentikan **Live Server**, klik icon nomor port yang terdapat di sisi kanan bawah VS

Code (tempat icon **Go Live** sebelumnya berada):



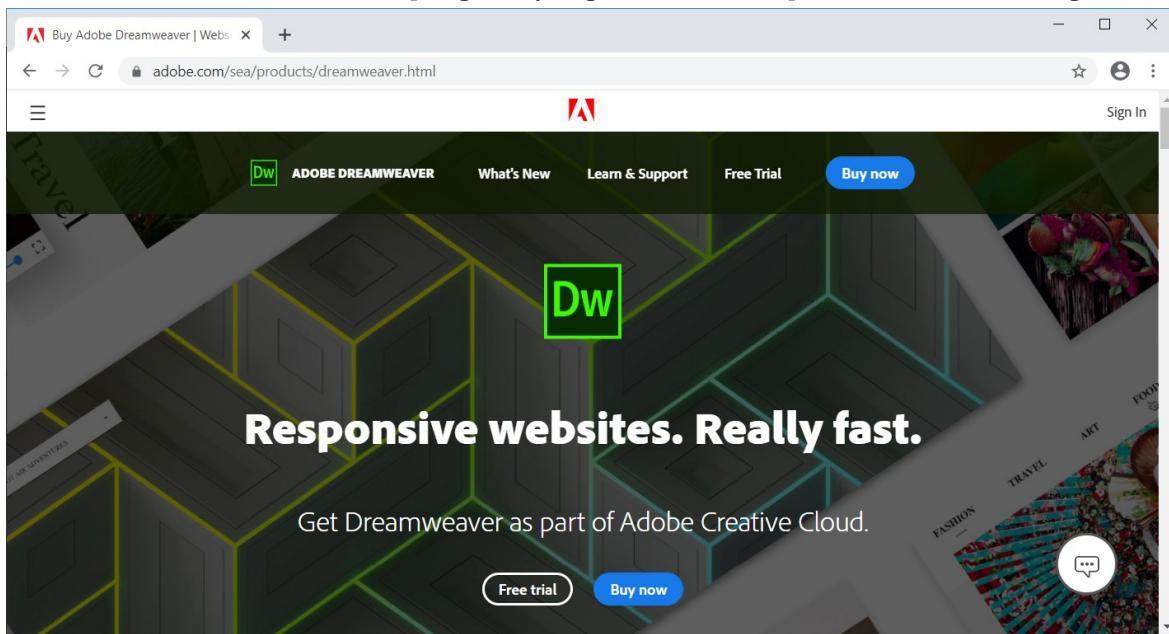
Gambar: Menghentikan Live Server

Bagaimana dengan Adobe Dreamweaver?

Salah satu aplikasi populer yang cukup 'melegenda' terutama bagi pemula web programming adalah **Adobe Dreamweaver**. Dreamweaver merupakan aplikasi canggih dengan fitur melimpah untuk pembuatan web.

Dreamweaver termasuk kelompok aplikasi yang dikenal dengan sebutan **WYSIWYG** (What You See Is What You Get), dimana kita bisa merancang tampilan website dengan cara 'drag and drop', yakni menggambar tampilan web secara visual tanpa harus mengetahui kode program dibalik itu (walaupun Dreamweaver juga menyediakan fitur coding yang sangat lengkap).

Dibalik keunggulannya, menurut saya Dreamweaver kurang cocok untuk proses belajar. Aplikasi ini lumayan berat dan cukup mahal untuk versi legalnya. Dreamweaver lebih pas jika anda telah memahami kode-kode program yang ada dan mampu membeli versi original.



Gambar: Website resmi Adobe Dreamweaver CC

Saat ini versi terakhir dari Dreamweaver adalah **Adobe Dreamweaver CC**. Aplikasi ini hanya bisa didapat dengan cara berlangganan dengan biaya Rp 280.000 per bulan.

Daripada menggunakan program bajakan, lebih baik mencari alternatif lain. Saya sendiri sangat jarang melihat lowongan kerja programmer dengan syarat "bisa menggunakan Dreamweaver". Malah akan menjadi nilai minus jika seorang programmer hanya bisa membuat program dari Dreamweaver saja.

Dalam bab ini kita telah membahas tentang web browser, layout engine, dukungan web browser untuk CSS3 serta pilihan teks editor.

Untuk bisa mengikuti semua materi di buku ini, anda bebas ingin memakai teks editor apa saja, apakah itu **Notepad++** yang ringan atau **VS Code** yang lebih modern.

Saya sendiri menginstall kedua editor ini dan memakainya secara bergantian. Untuk keperluan edit-edit sederhana lebih menyukai Notepad++ karena cepat. Tapi jika sedang membuat project yang agak besar dan perlu membuka banyak file, VS Code lebih sesuai.

Atau jika sebelumnya anda pernah mencoba web programming dan sudah punya teks editor pilihan sendiri, itu pun tetap bisa dipakai. Beberapa teks editor lain (yang juga gratis) adalah [Atom](#), [Komodo Edit](#), dan [Bracket](#).

Berikutnya kita akan mulai masuk ke pembahasan mengenai aturan dasar penulisan CSS.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

4. Aturan Dasar Penulisan CSS

Dalam 3 bab awal sebelumnya kita telah bahas pengertian CSS, sejarah CSS, serta menyiapkan perangkat perang (web browser dan text editor). Mulai dari bab ini hingga akhir buku nanti kita akan masuk ke materi *coding* CSS, yang dibuka dengan **aturan dasar penulisan kode CSS**.

4.1. Pengertian Selector, Declaration, Property dan Value CSS

Selector, declaration, property dan **value** adalah inti dari CSS. Hampir semua kode CSS yang kita buat hanya terdiri dari ke-4 element ini. Agar lebih mudah dipahami, perhatikan potongan kode CSS berikut:

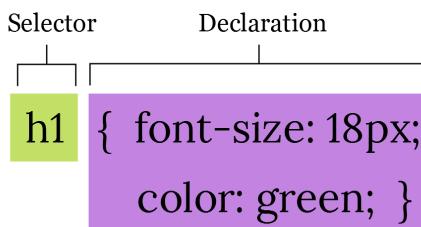
```
h1 {  
    font-size: 18px;  
    color: green;  
}
```

Arti dari kode di atas adalah: temukan seluruh tag `<h1>` di dalam halaman HTML, kemudian set ukuran font sebesar 18 pixel dan set warna teks menjadi hijau.

Kode "`h1`" dalam contoh di atas adalah **selector CSS**. Selector dipakai untuk mencari bagian mana dari HTML yang ingin di-style. CSS menyediakan beragam jenis selector mulai dari yang sederhana seperti *element selector*, hingga yang cukup rumit seperti *pseudo class selector*.

Dalam contoh di atas, "`h1`" dikenal sebagai **element selector** atau **tag selector**. Element selector dipakai untuk mencari seluruh element HTML atau tag HTML, yang dalam contoh kita adalah tag `<h1>`. Lebih jauh tentang selector akan dibahas pada bab setelah ini.

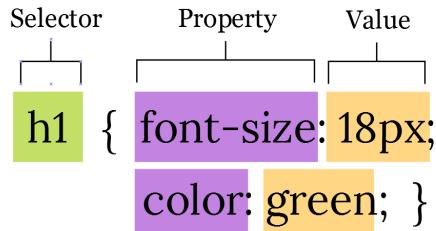
Setelah penulisan selector, berikutnya masuk ke **declaration**. Declaration adalah kumpulan aturan style CSS yang berada di antara tanda kurung kurawal. Gambar berikut memperjelas pengertian selector dan declaration:



Gambar: Selector dan Declaration Block CSS

Di dalam *declaration*, terdapat pasangan *property* dan *value*. **Property** adalah jenis *style*, atau bagian apa yang ingin diubah dari sebuah kode HTML. Misalnya jenis font, ukuran huruf, warna background, dll. Terdapat ratusan *property* di dalam CSS dan terus bertambah.

Agar bisa dipakai, sebuah *property* memiliki nilai atau *value*. **Value** ini sangat bergantung kepada jenis property. Misalnya untuk ukuran font bisa di isi satuan *pixel* atau *point*, tapi untuk property warna (*color*) kita harus mengisi nilai berupa nama warna atau kode **RGB**.



Gambar: Selector, Property dan Value dalam CSS

Keempat elemen inilah yang membangun CSS. Sepanjang buku nanti kita akan bahas lebih detail tentang **selector**, **property** dan **value** CSS.

Cara Penulisan Selector, Property dan Value

Seperti yang telah dilihat sebelumnya, penulisan kode CSS mencakup aturan berikut:

- Sebuah style di dalam CSS diawali dengan penulisan **selector**, yakni bagian apa dari HTML yang ingin diubah tampilannya.
- Setelah penulisan **selector**, seluruh **property** dan **value** (*declaration*) dari selector tersebut harus berada di dalam tanda kurung kurawal.
- Antara penulisan **property** dan **value** dipisah oleh tanda titik dua " : ".
- Antara satu property dengan property lain dipisah dengan tanda titik koma " ; ".
- Khusus untuk penulisan property terakhir, tanda titik koma " ; " boleh diabaikan, tapi sangat disarankan untuk tetap menulisnya.

4.2. Case Sensitivity

Case sensitivity adalah istilah yang membahas apakah sebuah bahasa membedakan penulisan huruf kecil dan huruf besar. Secara umum, CSS bersifat **case-insensitive**, yang artinya tidak membedakan antara huruf besar dan huruf kecil. Kedua kode berikut akan dianggap sama:

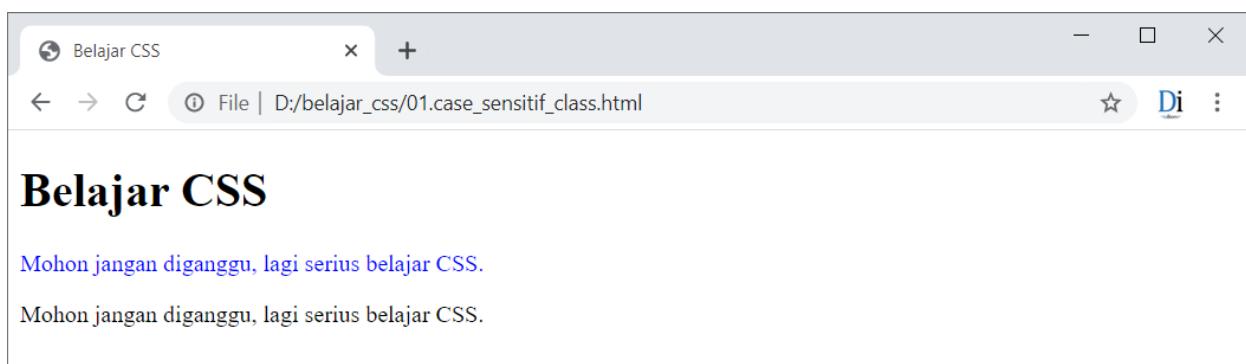
```
p {  
    font-size: 18px;  
}  
p {  
    foNT-SiZE: 18PX;  
}
```

Keduanya berfungsi untuk mengubah ukuran font element paragraf (tag <p>) menjadi 18 pixel.

Walaupun demikian, terdapat beberapa kasus dimana huruf besar dan kecil akan dibedakan. Ini sebenarnya bukan terletak pada CSS, tapi di HTML. Dalam HTML, penulisan atribut **class** bersifat *case sensitif* seperti contoh berikut:

01.case_sensitif_class.html

```
54  <!DOCTYPE html>
55  <html lang="id">
56  <head>
57      <meta charset="UTF-8">
58      <title>Belajar CSS</title>
59      <style>
60          .belajar {
61              color: blue;
62          }
63      </style>
64  </head>
65  <body>
66      <h1>Belajar CSS</h1>
67      <p class="belajar">Mohon jangan diganggu, lagi serius belajar CSS.</p>
68      <p class="Belajar">Mohon jangan diganggu, lagi serius belajar CSS.</p>
69  </body>
70  </html>
```



Gambar: Kasus case sensitive pada nama class HTML

Pada kode HTML di baris 14 dan 17 saya menulis 2 buah atribut **class** dengan nilai "belajar" dan "Belajar". Hasilnya hanya 1 paragraf yang akan berwarna biru. Ini terjadi karena penulisan atribut **class** di HTML akan membedakan huruf besar atau huruf kecil (*case-sensitif*).

Sebagai *best practice* atau cara yang paling disarankan adalah, selalu gunakan **huruf kecil** untuk **semua** penulisan *selector*, *property* dan *value* CSS. Jika butuh nama yang terdiri dari 2 kata atau lebih, gunakan tanda dash " - " seperti: *paragraf-pertama* atau *main-header*.

Materi tentang *class selector* `.belajar` serta berbagai jenis selector lain akan kita bahas detail pada bab berikutnya.

4.3. Whitespace

Whitespace adalah istilah yang merujuk kepada karakter *spasi* yang tidak tampil di layar, termasuk juga tab, dan enter (karakter *carriage returns*).

Dalam penulisan **declaration** (*property* dan *value*), umumnya whitespace akan diabaikan. Kedua penulisan kode CSS berikut akan dianggap sama:

```
p {  
    font-size: 18px;  
    color: black;  
    margin: 0;  
}  
  
p {font-size: 18px; color: black; margin: 0;}
```

Contoh penulisan pertama agak panjang namun lebih mudah dibaca. Dalam contoh kedua saya menyatukan penulisan selector dan property dalam 1 baris. Walaupun lebih hemat tempat, penulisan seperti ini tidak disarankan karena membuat kode CSS susah dibaca (kecuali hanya terdiri dari 1 atau 2 property singkat).

Namun pada penulisan **selector**, tambahan tanda spasi bisa membuat perbedaan besar seperti contoh berikut:

```
p.pertama {  
    font-size: 18px;  
    color: black;  
}  
  
p .pertama {  
    font-size: 18px;  
    color: black;  
}
```

Selector p.pertama artinya akan mencari seluruh paragraf yang memiliki atribut class= "pertama", sedangkan selector p .pertama artinya akan mencari seluruh tag HTML yang memiliki class=pertama **dan** berada di dalam sebuah paragraf. Kesimpulannya, tanda spasi sangat berpengaruh.

Untuk penulisan *value*, tanda spasi ini kadang juga berpengaruh:

```
p {  
    font-size: 18 px;  
}  
  
p {  
    font-size: 18px;  
}
```

Perhatikan cara penulisan pertama, terdapat satu spasi antara "18" dan "px". Ini menyebabkan

kode CSS tidak terbaca oleh web browser. Penulisan yang benar adalah langsung disambung seperti contoh pertama "18px". Pengecualian dari aturan ini adalah value yang tidak butuh karakter satuan, seperti nilai 0.

4.4. Baris Komentar

Materi yang akan kita bahas ini saya rasa lebih pas disebut sebagai **baris keterangan**. Namun dalam bahasa inggris memang disebut sebagai *comment* yang ketika diterjemahkan ke dalam bahasa indonesia menjadi **komentar**.

Dalam bahasa pemrograman secara umum, **baris komentar** atau *comment* adalah 'kode program' untuk memberi keterangan/penjelasan mengenai cara kerja program.

Komentar tidak akan diproses oleh web browser, tapi ditujukan kepada kita sebagai programmer atau desainer yang membuat kode tersebut. Baris komentar juga sangat berguna bagi programmer lain untuk menjelaskan maksud dari kode yang ada.

CSS hanya mendukung 1 cara penulisan komentar, yakni diawali karakter /* dan diakhiri dengan karakter */ . Berikut contohnya:

```
/* Ini adalah style untuk paragraf */
p {
    font-size: 18px; /* Set ukuran font sebesar 18 pixel */
    color: black;    /* Membuat teks berwarna hitam */
}
```

Karena tidak akan diproses, komentar bisa dipakai untuk menonaktifkan beberapa kode CSS. Teknik ini sangat berguna dalam proses *debugging* (mencari kesalahan kode program), atau jika kita ingin mencoba kombinasi style baru tapi tidak ingin menghapus kode lama. Berikut contoh penggunaannya:

```
/* dimatikan sementara
p {
    font-size: 18px;
    color: black;
} */

p {
    font-size: 16px;
    color: green;
}
```

Pada kode yang panjang, komentar sering dipakai untuk membagi kode CSS menjadi kelompok-kelompok, misalnya untuk bagian *header*, *footer*, *menu*, dll, seperti contoh berikut:

```
=====
=====HEADER=====
==========*/
```

```
.header{  
    font-size: 16px;  
    color: green;  
}  
  
/*=====  
 =====FOOTER=====  
 =====*/  
.footer{  
    font-size: 12px;  
    color: black;  
}
```

Pembagian seperti ini akan memudahkan kita (dan juga orang lain) untuk membaca dan mencari kode CSS karena sudah dikelompokkan menurut fungsinya masing-masing.

4.5. Cara Menginput kode CSS

Untuk menginput kode CSS ke dalam HTML, terdapat 3 cara: *inline style* CSS, *internal style* CSS dan *external style* CSS.

Inline Style CSS

Cara paling praktis untuk menginput kode CSS adalah dari **inline style CSS**. Dengan metode ini kita bisa menulis kode CSS langsung di dalam tag HTML menggunakan atribut **style**.

Sebagai contoh, untuk mengubah warna sebuah paragraf menjadi merah bisa menulis kode berikut:

```
<p style="color: red">  
    Mohon jangan diganggu, lagi serius belajar CSS.  
</p>
```

Atribut **style** pada tag **<p>** di atas berfungsi sebagai pembuka kode CSS. Isi dari atribut ini berupa property CSS beserta nilainya. Property "color: red" akan membuat teks berwarna merah.

Jika ingin menulis 2 property atau lebih, tambah tanda titik koma di antara kedua property:

```
<p style="color: red; font-size: 18px">  
    Mohon jangan diganggu, lagi serius belajar CSS.  
</p>
```

Sekarang paragraf di atas akan tampil dengan warna merah dan berukuran 18 pixel. Berikut contoh lengkapnya di dalam sebuah file HTML:

02. contoh_inline_css.html

```
1  <!DOCTYPE html>  
2  <html lang="id">
```

```
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6 </head>
7 <body>
8   <h1>Belajar CSS</h1>
9   <p style="color: red; font-size: 18px">
10    Mohon jangan diganggu, lagi serius belajar CSS.
11   </p>
12   <p style="font-weight: bold">
13    Mohon jangan diganggu, lagi serius belajar CSS.
14   </p>
15 </body>
16 </html>
```



Gambar: Contoh tampilan hasil inline style CSS

Walaupun praktis, penggunaan inline style CSS tidak terlalu disarankan karena membuat kode CSS tercampur dengan HTML. Sehingga sedikit sulit untuk membedakan mana kode HTML dan mana kode CSS.

Dalam contoh di atas saya membuat 1 paragraf berwarna merah. Umumnya sebuah web memiliki ratusan paragraf yang berada di banyak file HTML. Jika suatu saat ingin mengganti warna merah ini menjadi hijau, maka terpaksa di ganti satu per satu.

Namun ada beberapa kasus dimana inline style boleh atau "terpaksa" dipakai, yakni jika kita tidak memiliki akses langsung ke dokumen HTML. Contohnya seperti di kolom komentar blog, di dalam forum, atau pada pesan email.

Khusus untuk kolom komentar blog dan forum, biasanya admin website tidak mengizinkan penggunaan inline style CSS atau kode HTML apapun. Alasannya karena pengunjung yang iseng bisa menyisipkan kode yang akan mengganggu tampilan web, seperti membuat huruf besar atau warna yang sangat mencolok.

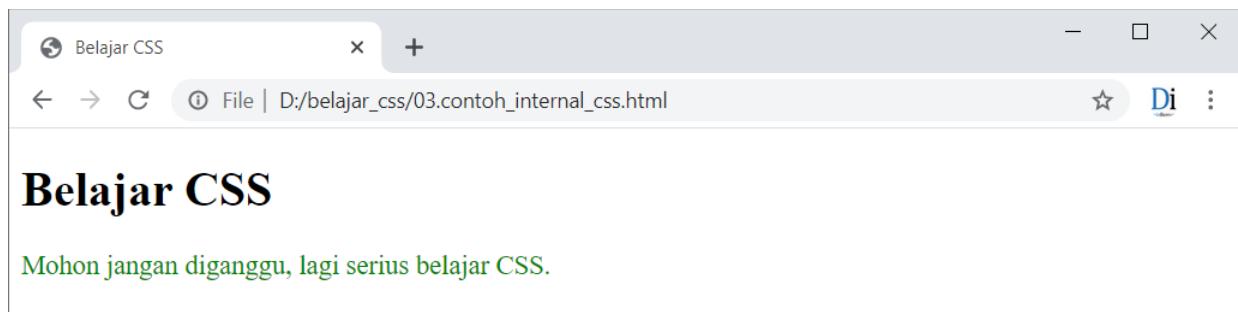
Internal Style CSS

Cara penggunaan CSS yang lebih disarankan adalah dengan metode **internal style CSS**. Untuk menggunakan, seluruh kode CSS ditulis di dalam dalam tag `<style>` pada bagian `<head>` kode HTML.

Berikut contoh penggunaan dari internal style CSS:

03.contoh_internal_css.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p {
8       font-size: 18px;
9       color: green;
10    }
11  </style>
12 </head>
13 <body>
14   <h1>Belajar CSS</h1>
15   <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
16 </body>
17 </html>
```



Gambar: Contoh tampilan hasil internal style CSS

Perhatikan kode di baris 6 – 11. Itulah cara penulisan kode CSS dengan metode internal style. Metode internal style CSS ini kadang disebut juga sebagai **embedded style CSS**.

Anda juga mungkin akan menemukan penulisan tag `<style>` dengan tambahkan atribut `type="text/css"` seperti contoh berikut:

```
<style type="text/css">
  p {
    font-size: 18px;
    color: green;
  }
</style>
```

Penulisan seperti ini juga bisa dipakai. Atribut `type="text/css"` berfungsi untuk menegaskan bahwa kode style yang digunakan adalah CSS. Saat ini mayoritas web browser sudah menjadikan CSS sebagai bahasa style *default* (karena memang tidak ada pilihan bahasa style lain) sehingga atribut `type="text/css"` ini tidak perlu ditulis lagi.

Dengan metode **internal style**, semua kode CSS akan berada di satu tempat khusus, terpisah dari konten website (kode HTML). Walaupun begitu, kode CSS tersebut masih berada di dalam

1 halaman HTML saja. Jika kita memiliki beberapa halaman yang ingin memakai style yang sama, maka kode CSS harus ditulis ulang pada setiap halaman.

Alternatif yang paling baik terutama jika website terdiri dari beberapa halaman, adalah dengan *external style CSS*.

External Style CSS

Cara penggunaan CSS yang paling paling fleksibel adalah **external style CSS**. Dengan metode ini, seluruh kode CSS ditempatkan pada file terpisah (file .css). Setiap halaman HTML yang butuh style CSS tinggal memanggil file ini menggunakan tag `<link>` atau perintah `@import`.

External style CSS dengan `<link>` element

Cara penggunaan external style CSS yang pertama adalah dengan tag `<link>`. Agar lebih mudah dipahami, langsung saja kita praktik dengan contoh kode program.

Silahkan buat satu file teks baru menggunakan text editor, kemudian ketik kode berikut:

style.css

```
1  h1 {  
2      color: red;  
3  }  
4  
5  p {  
6      font-size: 18px;  
7      color: silver;  
8  }
```

Save kode di atas dengan nama **style.css** dan simpan di satu folder yang sama dengan file HTML. Anda boleh menggunakan nama apa saja selama akhiran file-nya *.css.

Untuk dapat menjalankan kode CSS ini, tulis kode berikut di bagian `<head>` HTML:

```
<link href="style.css" rel="stylesheet">
```

Tag `<link>` dipakai untuk menghubungkan halaman HTML saat ini dengan file external, dimana salah satu dari file external tersebut adalah file CSS.

Atribut **href** berfungsi sebagai alamat dari file CSS yang akan diinput. Karena file **style.css** berada di dalam folder yang sama, maka kita tinggal menulis alamat `href="style.css"`.

Jika anda meletakkan file `style.css` di lokasi lain (misalnya di dalam sebuah folder), maka penulisan nilai `href` harus sesuai dengan alamat file tersebut. Dalam istilah HTML, kita bisa menggunakan *alamat relatif* seperti `href="css/style.css"` maupun *alamat absolute* seperti `href="http://www.duniaIlkom.com/style.css"`.

Atribut **rel** merupakan singkatan dari *relationship*. Ini dipakai untuk menjelaskan kepada web browser (dan juga mesin pencari) hubungan file yang akan di-link kan. Nilai `rel="stylesheet"`

artinya file yang akan di-link adalah sebuah file **stylesheet**.

Selain atribut **href** dan **rel**, kita juga bisa menulis atribut **type="text/css"**. Sama seperti penjelasan di internal style CSS, atribut **type="text/css"** berfungsi untuk menegaskan bahwa kode style yang dipakai adalah CSS. Untuk web browser modern, kode ini tidak wajib dituliskan:

```
<link href="style.css" rel="stylesheet" type="text/css">
```

Perhatikan bahwa tag `<link>` adalah sebuah *self closing tag*, sehingga kita tidak perlu menulis tag penutup `</link>`.

Berikut kode HTML lengkap dengan penambahan tag `<link>` ini:

04.contoh_external_style_css_link.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <link href="style.css" rel="stylesheet" type="text/css">
7  </head>
8  <body>
9      <h1>Belajar CSS</h1>
10     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
11 </body>
12 </html>
```



Gambar: Contoh tampilan hasil external style CSS dengan tag `<link>`

Apabila tidak terjadi perubahan apa-apa (warna teks tetap berwarna hitam), kemungkinan pertama adalah file **style.css** tidak bisa diakses. Pastikan anda telah menulis lengkap tag `<link>` dan alamat file `style.css` sudah benar terutama jika berada di dalam folder lain.

Kemungkinan kedua adalah masalah dengan fitur **web browser cache**.

Mengenal Web Browser "Cache"

Untuk mempercepat proses menampilkan website (dan juga menghemat *bandwidth*), web browser modern menyediakan fitur **cache**.

Fungsi dari **cache** adalah menyimpan sementara file gambar, file CSS, dan file JavaScript di dalam komputer. Ketika kita mengunjungi situs yang sama di lain waktu, web browser tinggal mengambil file dari **cache** tanpa harus mendownload ulang file-file tersebut.

Fitur cache ini sebenarnya sangat berguna, akan tetapi bagi kita sebagai web developer cache bisa membuat sakit kepala terutama pada saat proses pembuatan website.

Masalahnya adalah, saat kita memodifikasi kode CSS dan ingin melihat hasilnya, web browser kadang masih menggunakan file CSS dari cache, bukan dari file CSS yang 'asli'.

Beberapa kali saya mengalami hal seperti ini dan heran kenapa tampilan web tidak berubah. Solusi yang sering dilakukan adalah mencoba memperbaiki kembali kode CSS yang sebenarnya tidak salah.

Jika anda yakin tidak ada yang salah dengan kode CSS maupun penulisan tag `<link>`, maka kemungkinan besar **cache** inilah penyebabnya.

Untuk memaksa web browser menggunakan file CSS baru, refresh halaman web dari tombol **Crtl+F5**, atau tahan tombol **Shift** kemudian klik icon **refresh** di web browser. Cara ini akan membuat web browser mengambil file CSS baru dan bukan dari cache.

Untuk lebih memastikan, bisa test buka halaman dengan web browser lain atau dari *incognito mode*. Jika tetap tidak berubah, maka besar kemungkinan masalahnya memang ada di dalam kode CSS.

External style CSS dengan `@import`

Cara input file CSS external kedua adalah dengan perintah **@import** (dibaca: *at import*). Untuk menggunakannya, perintah `@import` ditulis dalam tag `<style>` seperti **internal style CSS**.

Berikut contoh penggunaan perintah `@import`:

```
05.contoh_external_style_css_import.html
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          @import url("style.css");
8      </style>
9  </head>
10 <body>
11     <h1>Belajar CSS</h1>
12     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
13 </body>
14 </html>
```



Gambar: Contoh tampilan hasil external style CSS dengan perintah @import

Cara penulisan perintah `@import` ada di baris 7, dengan menulis nama file CSS dalam tanda kurung. Karena nama file CSS yang saya gunakan adalah `style.css` yang sudah dibuat sebelumnya, maka kodennya adalah `@import url(style.css)`.

Penulisan perintah `@import` harus diletakkan di baris paling awal kode CSS, yakni sebelum penulisan `selector` dan `property`.

Kelebihan dari perintah `@import` dibandingkan dengan tag `<link>` adalah, perintah `@import` bisa dipakai langsung dari dalam file CSS. Dengan demikian, kita bisa memecah kode CSS dan menyatukannya kembali.

Misalkan saya memiliki 3 file CSS: `contoh1.css`, `contoh2.css`, dan `contoh3.css`. Di dalam file `style.css` semua ini bisa disatukan dengan kode berikut:

```
@import url("contoh1.css");
@import url("contoh2.css");
@import url("contoh3.css");
h1 {
    color: red;
}
p {
    font-size: 18px;
    color: brown;
}
```

Cara ini tidak bisa dilakukan dengan `<link>` karena tag `<link>` harus di dalam kode HTML.

Dalam prakteknya, menggunakan perintah `@import` bukanlah *best practice*. Apabila dimungkinkan, sebaiknya tetap pakai tag `<link>` untuk menginput kode external CSS.

Ini terkait dengan cara web browser memproses file CSS dimana tag `<link>` sedikit lebih efisien dibandingkan dengan perintah `@import`. Untuk penjelasan teknisnya bisa mengunjungi blog Steve Souders: [Don't use @import¹²](#).

Namun tetap tidak salah jika ingin menggunakan `@import`. Seperti yang kita bahas, perintah `@import` memiliki keunggulan tersendiri untuk memecah file kode CSS.

¹² <https://www.stevesouders.com/blog/2009/04/09/dont-use-import/>

Manakah yang Sebaiknya Dipakai?

Metode *external style* CSS adalah cara menginput kode CSS yang paling disarankan, terutama jika website kita punya lebih dari 1 halaman.

Dengan *external style* CSS, setiap file HTML yang butuh *style* tinggal memanggil file `style.css`. Dengan demikian, tampilan seluruh website akan berada di dalam 1 file saja, tidak peduli berapa pun jumlah file HTML yang akan menggunakan file ini.

Jika kita ingin mengganti sebuah *style* seperti mengubah warna teks dari hitam menjadi abu-abu, cukup mengubah di 1 file CSS dan tampilan dari semua file HTML yang menggunakan file `style.css` ini juga ikut berubah. Kita tidak perlu mengubah satu per satu kode CSS di masing-masing halaman seperti halnya jika menggunakan *inline style* atau *internal style* CSS.

Namun dalam beberapa kasus, *internal* atau *embedded* style CSS juga bisa dipakai. Terutama apabila kode tersebut hanya digunakan pada satu file HTML saja.

Untuk menghemat tempat, dalam buku ini saya akan banyak menggunakan **internal style CSS**. Ini semata-mata agar kode program menjadi lebih singkat saja. Selain itu rata-rata contoh kode hanya untuk 1 halaman.

Menggabung Ketiga Metode Input CSS

Dalam prakteknya, kita bisa menggabung ketiga cara input CSS secara bersamaan. Berikut contoh yang dimaksud:

06.tiga_metode_input_css.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <link href="style.css" rel="stylesheet">
7   <style>
8     p {
9       font-style: italic;
10    }
11   </style>
12 </head>
13 <body>
14   <h1 style="text-decoration: underline">Belajar CSS</h1>
15   <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
16 </body>
17 </html>
```



Gambar: Tiga cara input kode CSS dalam 1 halaman HTML

Pada contoh ini saya menggunakan **inline**, **internal** dan **external style** CSS secara bersamaan pada 1 file HTML.

4.6. CSS Media Type

Menutup pembahasan bab ini, kita akan bahas cara penggunaan **media type CSS**. Dengan *media type*, kita bisa mengatur di *media* atau perangkat mana saja sebuah style CSS akan berjalan.

CSS mendukung berbagai jenis media, seperti **screen** untuk jenis perangkat yang menggunakan layar, atau **print** untuk perangkat mesin cetak.

Dalam spesifikasi CSS 2.1 terdapat 9 nilai media yang di dukung, yakni: **all**, **aural**, **braille**, **handheld**, **projection**, **print**, **screen**, **tty**, dan **tv**. Nilai **all** merupakan nilai bawaan (*default*) jika kita tidak menulis media type apapun. Ini berarti kode CSS akan tampil di semua media.

Selain **all**, jenis media yang sering dipakai adalah **screen** (untuk monitor) dan **print** (untuk media print). Dengan ini, kita bisa merancang kode CSS yang tampil berbeda saat diakses dari sebuah monitor, dan ketika di print.

Terdapat 2 cara penggunaan media type, yakni dari **atribut media** pada tag **<link>** atau perintah **@media** dari dalam tag **<style>**.

Cara paling umum adalah menggunakan atribut media pada tag **<link>**, dan inilah yang akan kita bahas terlebih dahulu. Berikut contoh penggunaannya:

```
<link href="style.css" rel="stylesheet" media="screen">
```

Tag **<link>** ini akan menginstruksikan perangkat yang mengakses halaman web (apapun itu) bahwa kode CSS yang ada di dalam file **style.css** hanya ditujukan untuk media **screen** saja.

Dengan cara yang sama, kita bisa membuat file CSS lain untuk media **print**:

```
<link href="print.css" rel="stylesheet" media="print">
```

Dengan kode ini, file **print.css** akan dipakai saat halaman HTML tersebut di print (ketika dicetak).

Sebagai contoh praktek, kita akan lihat perbedaan dari media screen dan print ini. Sebelumnya kita sudah memiliki file `style.css` yang berisi kode berikut:

```
h1 {  
    color: red;  
}  
p {  
    font-size: 18px;  
    color: silver;  
}
```

Silahkan buat file CSS kedua dengan nama `print.css` dan isi dengan kode berikut:

```
h1 {  
    color: green;  
}  
p {  
    font-size: 18px;  
    color: blue;  
}
```

Jika diperhatikan, pada kedua file ini saya hanya menukar warna teks untuk tag `<h1>` dan `<p>`. Mari coba jalankan dengan kode HTML di bawah ini:

07.media_type_link.html

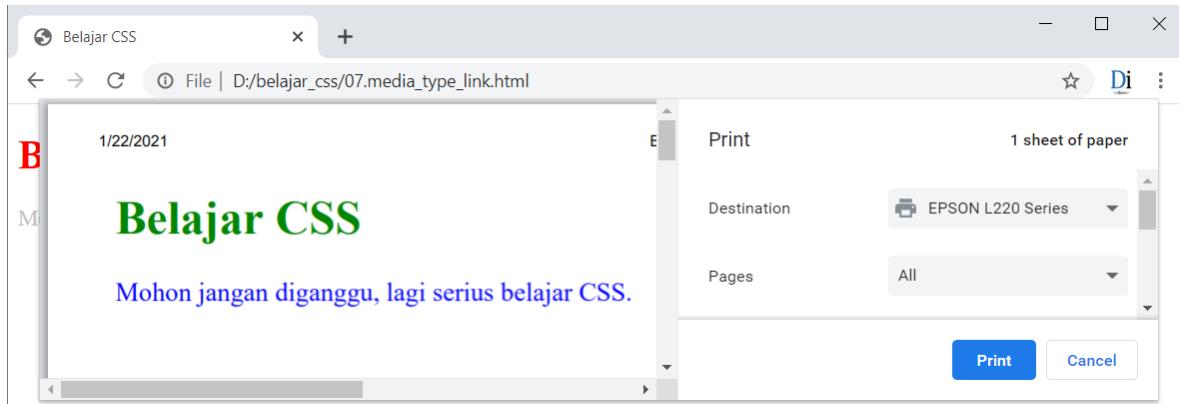
```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <link href="style.css" rel="stylesheet" media="screen">  
7      <link href="print.css" rel="stylesheet" media="print">  
8  </head>  
9  <body>  
10     <h1>Belajar CSS</h1>  
11     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>  
12  </body>  
13 </html>
```



Gambar: Tampilan pada media screen

Hasilnya, tag `<h1>` akan berwarna merah dan tag `<p>` berwarna cokelat. Kedua style ini berasal dari file `style.css`.

Bagaimana ketika di print? Silahkan klik menu **print** di web browser dan akan tampil jendela 'print preview'. Berikut hasil yang saya dapat di web browser Google Chrome:



Gambar: Tampilan halaman ketika akan di print

Perhatikan bahwa kali ini tag `<h1>` akan berwarna hijau dan tag `<p>` berwarna biru. Inilah efek dari penggunaan `media="print"`.

Dalam implementasinya, kita bisa buat sebuah file style khusus agar halaman web tampil lebih rapi ketika di print, misalnya dengan menghapus gambar background, menyembunyikan menu navigasi, dan menghilangkan sidebar.

Contoh web yang menerapkan ini adalah situs **wikipedia**. Silahkan tes bandingkan hasil ketika di web browser dan pada saat di print:

Two screenshots of a Google Chrome window comparing Wikipedia's CSS styles. The top screenshot shows the regular screen view of the 'Cascading Style Sheets' article. The sidebar on the left contains links like 'Halaman Utama', 'Perubahan terbaru', and 'Artikel pilihan'. The main content area has a large heading 'Cascading Style Sheets' and some text. The bottom screenshot shows the 'Print' dialog box for the same page. The dialog indicates '7 sheets of paper' will be printed. The print preview shows the same content as the screen view, but with a different layout optimized for printing, such as a larger font size and a clean, sans-serif appearance.

Gambar: Perbedaan style CSS wikipedia: screen (atas), dan print preview (bawah)

Selain menghilangkan bagian header dan sidebar, tampilan di media **print** wikipedia juga menghapus warna biru link, sehingga seluruh teks berwarna hitam.

Sebagai alternatif penggunaan, CSS juga mendukung penulisan media type dengan perintah **@media** (dibaca: *at media*). Langsung saja kita lihat contohnya:

08.media_type_at.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          /* Style untuk media screen */
8          @media screen {
9              h1 {
10                  color: red;
11              }
12              p {
13                  font-size: 18px;
14                  color: silver;
15              }
16          }
17
18          /* Style untuk media print */
19          @media print {
20              h1 {
21                  color: green;
22              }
23              p {
24                  font-size: 18px;
25                  color: blue;
26              }
27          }
28      </style>
29  </head>
30  <body>
31      <h1>Belajar CSS</h1>
32      <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
33  </body>
34  </html>
```

Untuk membuat perintah **@media**, seluruh *declaration* harus ditempatkan di antara tanda kurung kurawal. Dalam contoh ini untuk media *screen* penulisannya adalah **@media screen**, sedangkan untuk media *print*, ditulis dengan **@media print**.

Walaupun terdapat media lain seperti *handheld*, *projection* dan *tv*, media type ini tidak banyak dipakai. Bahkan walaupun kita menggunakan **media="tv"**, file CSS tersebut bisa jadi tidak berjalan ketika diakses dari smart TV berlayar lebar. Begitu pula dengan

`media="handheld"`, dimana hanya sedikit perangkat mobile yang menggunakan metode ini.

Untuk perangkat modern dengan berbagai ukuran layar, **CSS3** membawa fitur tambahan yang disebut sebagai **media query**. Melalui *media query* kita bisa jalankan style yang berbeda tergantung lebar layar. **Media query** adalah dasar dari pembuatan **web responsive** yang akan di bahas dalam bab tersendiri.

Sepanjang bab ini kita telah mempelajari pengertian **selector**, **declaration**, **property** dan **value** CSS. Pada bab berikutnya akan dikupas dengan lebih dalam tentang jenis-jenis selector dalam CSS.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

5. CSS Selector

Selector merupakan salah satu aset terpenting untuk bisa memahami CSS. Mengetahui jenis dan kapan sebuah selector digunakan sangat diperlukan untuk proses desain web yang efektif dan efisien. Dalam bab kali ini kita akan membahas apa saja selector yang tersedia di CSS, dan bagaimana cara penggunaannya.

5.1. Pengertian Selector

Pengertian **selector** telah disinggung pada bab sebelum ini. Pada dasarnya **selector** adalah sebuah pola (*pattern*) yang digunakan untuk mencari suatu tag/element di dalam HTML.

Misalnya kita ingin mencari semua tag `<p>` di dalam halaman HTML, mencari seluruh tag `<h1>` yang memiliki atribut `class="judul"` atau mencari seluruh paragraf di dalam tag `<div>` yang memiliki `id="header"`.

Selector di dalam CSS sangat beragam, mulai dari selector dasar seperti *element selector*, *class selector*, *id selector*, hingga yang sangat spesifik seperti *pseudo class selector* atau *pseudo element selector*. Semua selector ini akan kita bahas dengan cukup detail.

Pentingnya Struktur HTML

Sebelum mulai membahas jenis-jenis selector di dalam CSS, saya ingin menekankan pentingnya membuat struktur kode HTML yang teratur dan konsisten, karena selector ini dipakai untuk mencari kode-kode HTML.

Sebagian web desainer pemula terlalu fokus memahami CSS tapi tidak memperhatikan kode HTML-nya. Padahal CSS dan HTML sangat berkaitan. Sebagai contoh, untuk membuat kutipan banyak yang memakai tag `<p>` biasa, padahal HTML sudah menyediakan tag `<blockquote>` atau tag `<quote>` yang memang ditujukan untuk membuat kutipan.

HTML5 juga membawa banyak semantic tag baru, yakni **tag yang memiliki makna**. Meskipun tetap tidak salah menggunakan tag `<div>`, tapi kita bisa memanfaatkan HTML5 semantic tag untuk membuat struktur halaman. Contoh semantic tag ini adalah `<header>`, `<footer>`, `<article>`, `<nav>`, `<section>` dan `<aside>`.

Selain itu jika halaman web yang dirancang terdiri dari banyak halaman, kita juga harus selalu konsisten dalam membuat struktur HTML. Jika tag `<aside>` sudah dipakai untuk tempat iklan pada halaman pertama, di halaman kedua juga harus menggunakan tag yang sama. Dengan

demikian, satu selector CSS bisa dipakai pada semua halaman.

Kemampuan untuk membuat struktur HTML yang baik akan meningkat seiring banyaknya latihan menulis kode CSS dan HTML.

5.2. Universal Selector

Selector pertama yang akan kita bahas adalah **universal selector**. Ini merupakan selector yang paling sederhana sekaligus paling powerful. Universal selector ditulis menggunakan karakter bintang: '*'.

Perhatikan contoh kode CSS berikut:

```
* {
  color: blue;
  background-color: white;
}
```

Kode di atas berarti: cari **seluruh tag HTML**, lalu ubah warna teks menjadi biru dan warna background menjadi putih.

Kata paling penting di sini adalah "**seluruh tag HTML**". Ini membuat universal selector relatif jarang dipakai karena kita tidak bisa mengontrol tag apa saja yang akan di-style. Sebagai contoh, perhatikan kode berikut:

```
01.masalah_universal_selector.html

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      * {
8        color: red;
9      }
10   div {
11     color: green;
12   }
13   </style>
14 </head>
15 <body>
16   <h1>Belajar CSS</h1>
17   <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
18   <div>
19     <h2>Header h2 di dalam tag div</h2>
20     <p>Sebuah paragraf di dalam tag div</p>
21   </div>
22 </body>
23 </html>
```



Gambar: Salah satu efek dari universal selector

Walaupun kita belum membahas **element selector** dan konsep **inheritance** (penurunan style), di baris 7 - 9 saya memakai *universal selector* untuk mengubah seluruh warna teks menjadi merah, kemudian selector **div** di baris 10 - 12 ingin mengubah warna teks di dalam tag `<div>` menjadi hijau.

Namun apa yang terjadi? *Universal selector* akan menimpa efek dari *element selector*. Efek inilah yang bisa mendatangkan masalah jika kita tidak mengetahui apa yang terjadi. Saya akan kembali membahas masalah ini ketika masuk kepada *inheritance* pada bab berikutnya.

5.3. Element Selector

Element selector atau disebut juga dengan **tag selector** atau **type selector** adalah selector CSS yang digunakan untuk mencari suatu tag HTML. Penulisannya mirip seperti tag HTML biasa tapi tanpa tanda kurung siku:

```
02.element_selector.html
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h1 {
8              color: purple;
9          }
10         p {
11             color: blue;
12         }
13     </style>
14 </head>
15 <body>
16     <h1>Belajar CSS</h1>
17     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
18     <div>
19         <h2>Header h2 di dalam tag div</h2>
```

CSS Selector

```
20    <p>Sebuah paragraf di dalam tag div</p>
21  </div>
22 </body>
23 </html>
```



Gambar: Contoh penggunaan element selector

Dalam contoh ini saya membuat selector **h1** dan **p** yang dipakai untuk mencari seluruh kode HTML dengan tag **<h1>** dan **<p>**. Kemudian warna teks dari kedua tag tersebut dengan property **color: purple** dan **color: blue**.

Semua tag HTML bisa dipakai sebagai element selector. Misalnya untuk mencari seluruh tag **<blockquote>** dan mengubah hurufnya menjadi miring, bisa memakai kode berikut:

```
blockquote {
  font-style: italic;
}
```

Salah satu element selector yang cukup sering di pakai adalah **body**. Tag **<body>** merupakan 'induk' dari seluruh tag HTML di bagian body. Jika kita membuat style pada element body, efeknya mirip seperti *universal selector*. Misalkan untuk mengubah seluruh warna teks pada halaman HTML menjadi biru, bisa menggunakan kode berikut:

```
body {
  color: blue;
}
```

Element selector **body** ini cukup sering di pakai. Namun karena sifatnya yang global, biasanya perlu digabung dengan **class selector** atau **id selector** jika ingin mencari kode HTML yang lebih spesifik.

5.4. Class Selector

Class Selector merupakan salah satu selector yang paling sering di pakai. Selector ini akan mencari setiap tag HTML yang memiliki atribut **class** dengan nama tertentu. Untuk menulis class selector, diawali dengan tanda titik: " . ".

Sebagai contoh, agar sebuah paragraf bisa diakses dengan **class selector**, kita harus tulis atribut **class** pada tag `<p>` seperti contoh berikut:

```
<p class="sukses">Proses registrasi berhasil</p>
```

Nama dari atribut `class` bisa diubah sesuka hati tergantung keperluan. Dalam contoh ini saya menggunakan class "sukses". Untuk mengaksesnya dari CSS, gunakan selector berikut:

```
.sukses {
    color: green;
}
```

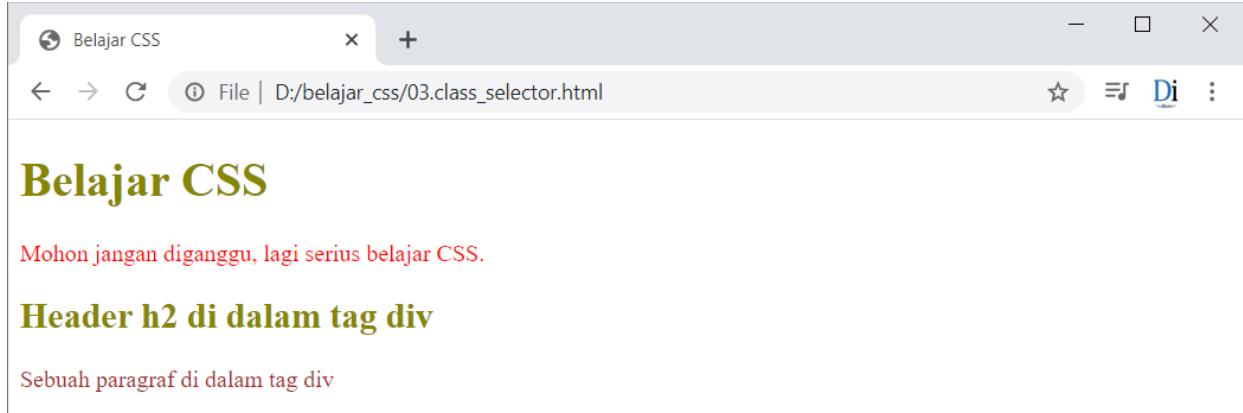
Perhatikan penambahan tanda titik pada kata `.sukses`, inilah yang menandakan bahwa kita ingin mencari sebuah (atau beberapa) tag HTML yang memiliki atribut class bernilai `sukses`.

Berbeda dengan **id selector** yang akan kita pelajari setelah ini, class selector bisa dipakai oleh lebih dari 1 tag HTML. Berikut contoh lain dari penggunaan **class selector**:

03.class_selector.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .judul {
8              color: olive;
9          }
10         .peringatan {
11             color: red;
12         }
13         .penjelasan {
14             color: brown;
15         }
16     </style>
17 </head>
18 <body>
19     <h1 class="judul">Belajar CSS</h1>
20     <p class="peringatan">Mohon jangan diganggu, lagi serius belajar CSS.</p>
21     <div>
22         <h2 class="judul">Header h2 di dalam tag div</h2>
23         <p class="penjelasan">Sebuah paragraf di dalam tag div</p>
24     </div>
25 </body>
26 </html>
```

CSS Selector



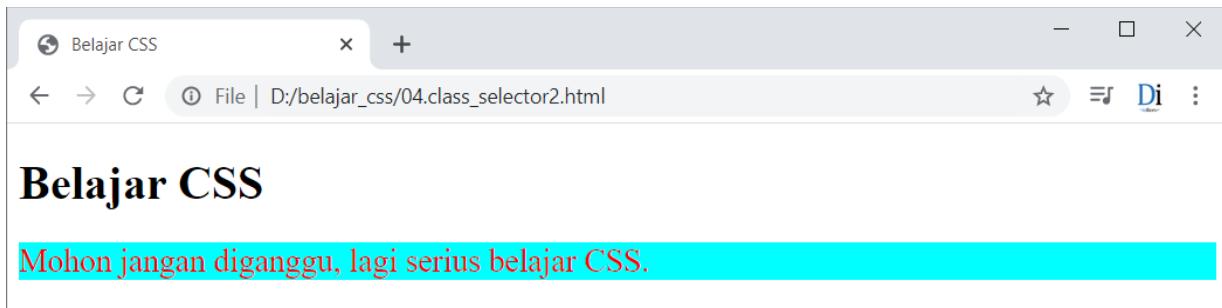
Gambar: Contoh penggunaan class selector

Di sini saya menggunakan 3 class selector yang masing-masingnya berpasangan dengan atribut `class`.

Praktek yang umum dipakai untuk class selector adalah menggunakan lebih dari satu selector pada satu tag HTML. Berikut contoh yang dimaksud:

04.class_selector2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .peringatan {
8              color: red;
9          }
10         .penting {
11             font-size:22px;
12         }
13         .pertama {
14             background-color: aqua;
15         }
16     </style>
17 </head>
18 <body>
19     <h1>Belajar CSS</h1>
20     <p class="peringatan penting pertama">
21         Mohon jangan diganggu, lagi serius belajar CSS.
22     </p>
23 </body>
24 </html>
```



Gambar: Contoh penggunaan multiple class selector

Pada tag <p> di baris 20, terdapat atribut `class="peringatan penting pertama"`. Ini artinya saya memakai 3 buah class untuk tag <p>, yakni class peringatan, class penting dan class pertama. Penulisan 3 class ini dipisah dengan spasi.

Hasilnya, tag <p> memiliki efek gabungan dari ketiga class, yakni teks akan berwarna merah, berukuran 22 pixel dan memiliki warna background *aqua*.

Berikut tips untuk menggunakan **class selector**:

- Carilah nama class yang bermakna (*semantic meaning*).

Walaupun kita bebas ingin menggunakan nama apa saja, tapi carilah nama yang memiliki makna, bukan bagaimana tampilannya. Pada contoh sebelum ini saya memakai nama class "peringatan". Nama ini mencerminkan makna dari class tersebut, yakni sebagai peringatan.

Sebaiknya tidak memakai nama class "merah" atau "judul-merah", karena jika suatu saat kita ingin mengubah warna teks menjadi kuning, nama class ini tidak relevan lagi.

- Gunakan tanda penghubung jika nama class terdiri dari 2 kata atau lebih.

Saran umum, cari nama class yang singkat. Namun tidak ada salahnya jika ingin menggunakan 2 kata atau lebih. Sebagai pemisah, pakai tanda minus atau strip " - ".

Tanda strip ini bukan keharusan, hanya saja sudah menjadi kebiasaan banyak desainer web. Sebagai contoh, kita bisa memakai nama class seperti *judul-artikel*, *kutipan-penting*, atau *menu-navigasi-utama*.

5.5. Id Selector

Cara penggunaan **id selector** mirip seperti **class selector**. Jika pada class selector kita menggunakan atribut **class** untuk menandai tag HTML, untuk id selector yang dipakai adalah atribut **id** seperti contoh berikut:

```
<p id="pesan">Proses registrasi berhasil</p>
```

Untuk mengakses paragraf di atas, cara penulisan selector CSS harus diawali dengan karakter

pagar: "# ". Jika saya ingin paragraf di atas berwarna biru, berikut kode CSSnya:

```
#pesan {
    color: blue;
}
```

Perbedaan paling mendasar antara **id selector** dengan **class selector** adalah: id selector hanya bisa digunakan 1 kali di dalam setiap halaman HTML. Atau dengan kata lain setiap id **harus unik**, tidak boleh terdapat 2 buah tag dengan atribut id yang sama.

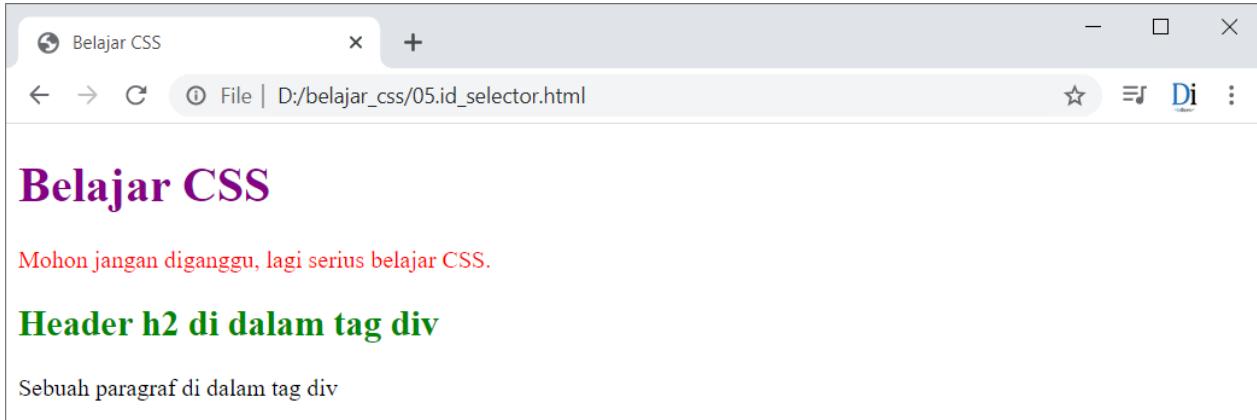
Dalam contoh di atas saya menggunakan `id="pesan"` pada tag `<p>`, oleh sebab itu tidak boleh ada tag HTML lain yang juga menggunakan atribut `id="pesan"`.

Pada prakteknya, kita akan lebih sering menggunakan **class selector** daripada **id selector**. Id selector memiliki urutan prioritas yang cukup tinggi dan sering menimpa style lain, termasuk class selector (pembahasan tentang urutan prioritas ini akan kita bahas dalam bab selanjutnya). Biasanya atribut **id** lebih banyak dipakai saat kode HTML digabung dengan bahasa pemrograman **JavaScript**.

Berikut contoh praktek dari penggunaan **id selector**:

05.id_selector.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          #judulUtama {
8              color: purple;
9          }
10         #pesan {
11             color: red;
12         }
13         #judulKedua {
14             color: green;
15         }
16     </style>
17 </head>
18 <body>
19     <h1 id="judulUtama">Belajar CSS</h1>
20     <p id="pesan">Mohon jangan diganggu, lagi serius belajar CSS.</p>
21     <div>
22         <h2 id="judulKedua">Header h2 di dalam tag div</h2>
23         <p>Sebuah paragraf di dalam tag div</p>
24     </div>
25 </body>
26 </html>
```



Gambar: Contoh penggunaan id class

Pada contoh ini saya menggunakan 3 buah atribut **id** pada tag HTML dan 3 buah **id selector** pada kode CSS. Property color di pakai untuk mengubah warna teks dari masing-masing id.

Berikut beberapa hal yang perlu diperhatikan terkait id selector:

- Id selector harus unik, oleh karena itu kita tidak bisa membuat 2 buah tag HTML yang memiliki id yang sama. Id selector hanya bisa digunakan untuk 1 element HTML saja.
- Sama seperti class selector, sedapat mungkin gunakan nama atribut id yang bermakna. Yakni yang menjelaskan apa fungsi tag tersebut dan bukan bagaimana tag itu akan ditampilkan. Memilih `id="pesanKesalahan"` lebih baik dari pada `id="teksMerah"`.
- Beberapa programmer cenderung menggunakan penulisan *camelCase* untuk atribut id. Ini karena kebanyakan atribut id juga banyak dipakai dalam JavaScript. **camelCase** adalah istilah penulisan yang menjadikan awalan kata kedua, ketiga dan seterusnya dengan huruf besar. Contoh: `id="pesanKesalahan"` atau `id="judulArtikelPertama"`.

Anda boleh mengikuti kebiasaan ini atau memakai cara penulisan dengan tanda hubung " - " yang sama seperti class selector, seperti `id="pesan-kesalahan"` atau `id="judul-artikel-pertama"`.

- Jika memungkinkan, gunakan **class selector** daripada **id selector**. Ini terkait dengan kekuatan id selector yang sangat mudah menimpa style selector lain (akan kita bahas pada bab berikutnya).

5.6. Attribute Selectors

Attribute selector adalah selector CSS yang di pakai untuk mencari tag HTML berdasarkan ada atau tidaknya sebuah atribut.

Attribute selector mirip dengan *class* dan *id* selector, namun **attribute selector** bersifat lebih global karena bisa dipakai untuk seluruh atribut HTML (tidak hanya *class* dan *id* saja). Sebagai contoh, perhatikan potongan kode HTML berikut:

```
<a href="https://en.wikipedia.org/wiki/Jakarta">Profil kota Jakarta</a>
```

Untuk mengakses tag di atas, kita bisa menggunakan *element selector* "a", namun kali ini saya ingin menggunakan *attribute selector*:

```
[href]{  
    color: brown;  
}
```

Kode CSS di atas berarti: *ubah warna teks menjadi cokelat (brown) untuk seluruh tag HTML yang memiliki atribut href*. Tanda kurung siku inilah yang menandakan **attribute selector**. Atribut yang ingin dicari ditulis dalam tanda kurung siku.

Jika yang kita inginkan adalah mencari seluruh atribut href yang berada di dalam tag , maka selector CSSnya menjadi sebagai berikut:

```
a[href]{  
    color: brown;  
}
```

Selector CSS di atas hanya akan mencari tag yang memiliki atribut href. Di antara huruf "a" dan kurung siku tidak boleh terdapat spasi.

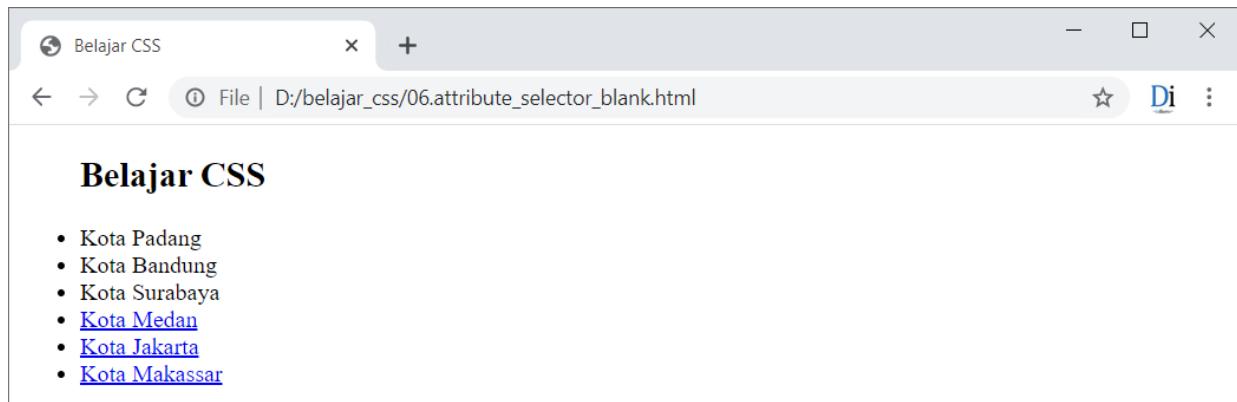
Attribute selector menyediakan berbagai fitur lain untuk memudahkan proses pencarian, seperti mencari tag HTML berdasarkan nilai awalan (*prefix*), akhiran (*suffix*), dan ada atau tidaknya sebuah kata tertentu di dalam atribut. Tambahan fitur *attribute selector* seperti ini berasal dari CSS3.

Perhatikan kode HTML berikut:

```
06.attribute_selector_blank.html  
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <style>  
7  
8          </style>  
9      </head>  
10     <body>  
11     <ul>  
12         <h2 title="belajar css">Belajar CSS</h2>  
13         <li>Kota Padang</li>  
14         <li title="kota bandung">Kota Bandung</li>  
15         <li title="surabaya">Kota Surabaya</li>  
16         <li><a title="kota medan" href="medan.html">Kota Medan</a></li>  
17         <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>  
18         <li><a title="kota makassar" href="makassar.pdf">Kota Makassar</a></li>  
19     </ul>
```

CSS Selector

```
20 </body>
21 </html>
```



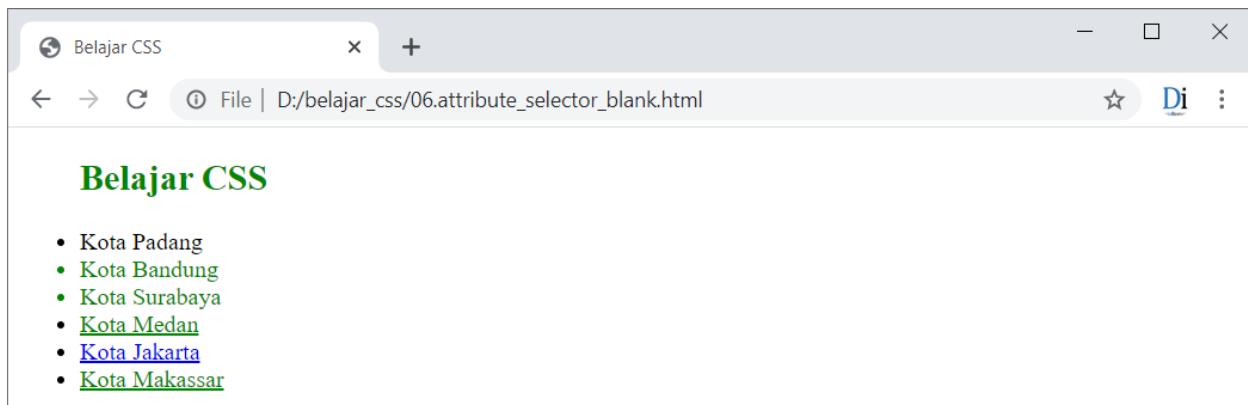
Gambar: Tampilan kode latihan attribute selector

Kode HTML di atas terdiri dari 1 judul `<h2>`, dan enam daftar nama kota yang dibuat dari tag ``. Dalam daftar ini (*unordered list*) saya menulis berbagai atribut yang berbeda.

Diantaranya merupakan link, dan sebagian lain hanya teks biasa dengan atribut HTML. Bagian kode CSS sengaja belum diisi karena kita akan lakukan beberapa eksperimen **attribute selector**.

Percobaan pertama, saya ingin mengubah seluruh tag HTML yang memiliki atribut **title** menjadi warna hijau, maka berikut kode CSSnya:

```
[title]{
  color: green;
}
```



Gambar: Contoh penggunaan attribute selector [title]

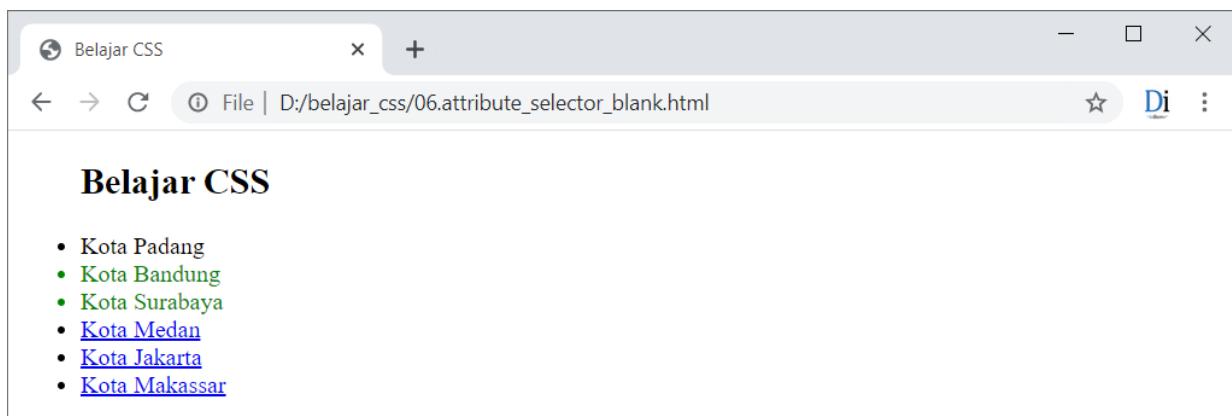
Terlihat selain "Kota Padang" dan "Kota Jakarta", semua list berwarna hijau. Ini terjadi karena di dalam kode HTML, tag `` untuk "Kota Padang" dan "Kota Jakarta" memang tidak memiliki atribut **title**.

Perhatikan pula judul "Belajar CSS" yang ikut berwarna hijau meskipun bukan bagian dari list. Ini karena atribut selector akan mencari **seluruh** tag HTML. Jika saya ingin lebih spesifik lagi

CSS Selector

bahwa hanya atribut `title` di dalam tag `` saja, kode yang digunakan adalah sebagai berikut:

```
li[title]{  
    color: green;  
}
```

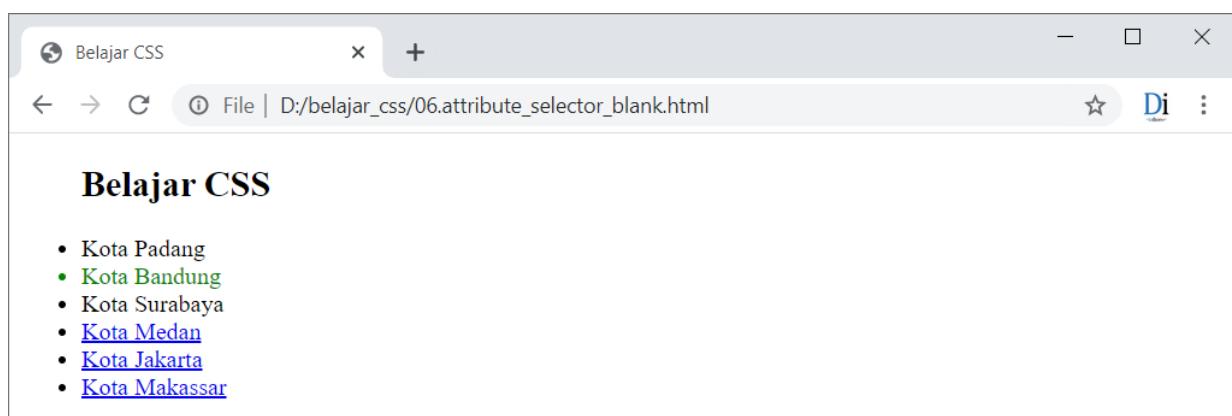


Gambar: Contoh penggunaan attribute selector `li[title]`

Selector `li[title]` hanya akan cocok dengan "Kota Bandung" dan "Kota Surabaya", karena di sinilah atribut `title` berada langsung di dalam tag ``. Khusus untuk "Kota Medan" dan "Kota Makassar", tidak lagi berwarna hijau karena atribut `title` berada di dalam tag `<a>`, bukan langsung di dalam tag ``.

Selain mengakses dari nama atribut, kita juga bisa mengakses element HTML dari nilai atribut tersebut. Sebagai contoh, jika saya ingin mencari tag HTML yang memiliki atribut `title` dengan nilai "kota bandung", bisa menggunakan selector berikut:

```
[title="kota bandung"]{  
    color: green;  
}
```

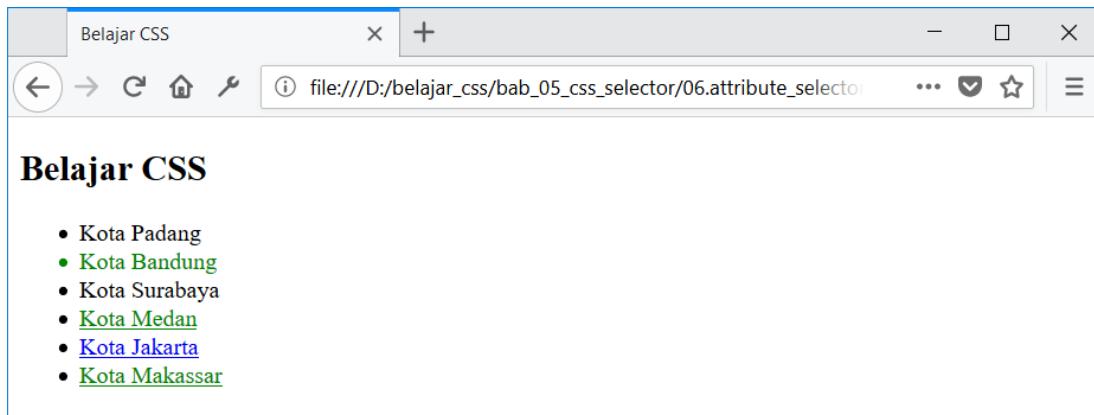


Gambar: Contoh penggunaan attribute selector `[title="kota bandung"]`

Selector `[title="kota bandung"]` hanya akan mengubah warna teks dari "Kota Bandung" saja, karena hanya di sini terdapat atribut `title` dengan nilai "kota bandung".

Bagaimana jika kita ingin mencari seluruh atribut **title** yang mengandung kata "kota" ? berikut cara penulisan selector yang bisa dipakai:

```
[title~="kota"]{  
    color: green;  
}
```



Gambar: Contoh penggunaan attribute selector [title~="kota"]

Perhatikan tambahan karakter *tilde* " ~ " sebelum tanda sama dengan. Karakter tilde ini dipakai untuk mencari kata **kota** di dalam atribut **title**. Sebuah "kata" didefinisikan sebagai kumpulan huruf yang dipisah dengan spasi.

Dalam contoh kita, hanya ada tiga kota yang miliki kata "kota" di dalam atribut **title**, yakni "kota bandung", "kota medan", dan "kota makassar".

Karakter tilde " ~ " berada di sebelah kiri angka 1 pada keyboard, untuk mengetiknya harus menahan tombol **Shift**.

Bagaimana jika kita ingin mencari atribut **title** yang mengandung kata "**ba**"? Ini tidak bisa dilakukan dengan karakter *tilde*, karena tilde butuh tanda pemisah spasi. Untuk keperluan ini, bisa menggunakan karakter bintang " * " seperti contoh berikut:

```
[title*="ba"]{  
    color: green;  
}
```

CSS Selector



Gambar: Contoh penggunaan attribute selector [title*="ba"]

Selector di atas akan mencari seluruh atribut `title` yang nilainya mengandung karakter "ba". Dalam kode HTML, ini cocok dengan `title="kota bandung"` dan `title="surabaya"`.

Untuk mencari nilai atribut berdasarkan posisi, bisa menggunakan karakter pangkat (*caret*): "`^`" dan karakter dollar: "`$`". Karakter "`^`" akan mencari di awal atribut, sedangkan karakter "`$`" akan mencari di akhir atribut. Ini akan lebih mudah dijelaskan dengan contoh.

Untuk menyeleksi hanya atribut `href` yang diawali dengan karakter "`https`", saya bisa menggunakan selector berikut ini:

```
[href^="https"]{  
    color: green;  
}
```

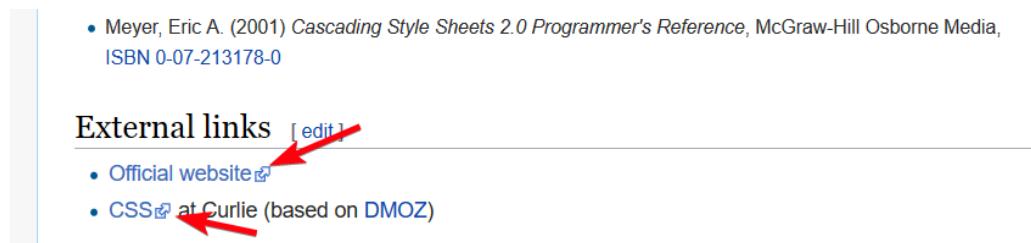


Gambar: Contoh penggunaan attribute selector [href^="http"]

Dengan kode di atas, hanya "Kota Jakarta" yang berwarna hijau karena atribut `href` pada tag `<a>` diawali dengan "`https`", yakni `href="https://en.wikipedia.org/wiki/Jakarta"`.

Penggunaan *attribute selector* seperti ini bisa dimanfaatkan untuk membedakan antara tampilan **internal link** dengan **external link** dalam website kita. Sebagai contoh, di situs wikipedia setiap link yang menuju keluar (bukan ke dokumen lain di dalam wikipedia) memiliki icon khusus.

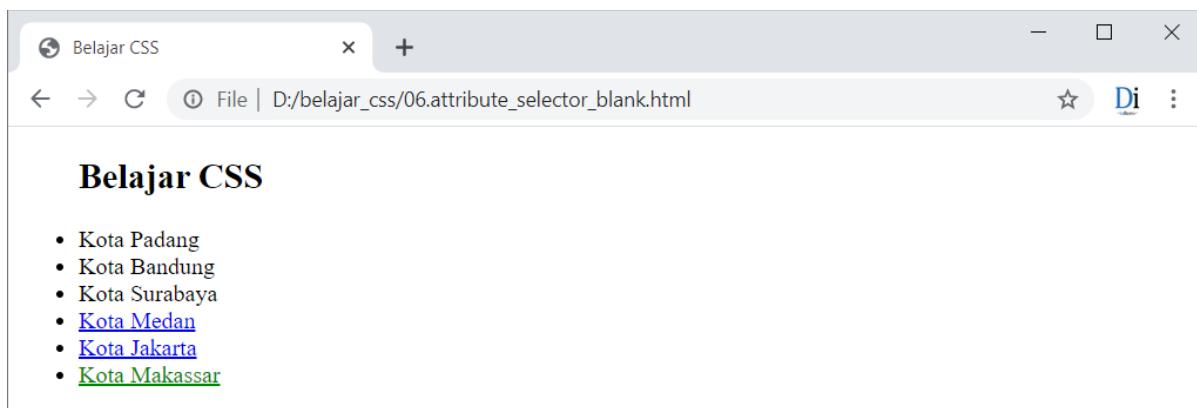
CSS Selector



Gambar: Link keluar di wikipedia memiliki icon khusus (panah merah)

Kita juga bisa membuat efek yang mirip. Misalnya agar setiap link yang menuju file PDF berwarna hijau, saya bisa menggunakan selector berikut:

```
[href$=".pdf"]{  
    color: green;  
}
```



Gambar: Contoh penggunaan attribute selector [href\$=".pdf"]

Selector [href\$=".pdf"] akan cocok dengan "Kota Makassar" karena atribut href-nya diakhiri dengan ".pdf", yang tidak lain adalah dokumen PDF. Trik seperti ini bisa dipakai untuk membedakan apakah sebuah link menuju dokumen **pdf** atau ke halaman html lain. Misalnya untuk link yang menuju file PDF, kita bisa menambahkan icon **pdf** di akhir link.

Attribute selector sangat powerful karena bisa dipakai untuk menyeleksi struktur HTML dengan detail. Pada proses desain form HTML, selector ini sangat membantu. Struktur form HTML yang sebagian besar menggunakan tag `<input>` hanya dibedakan berdasarkan atribut **type**. Misalnya untuk mencari seluruh **checkbox** bisa memakai attribute selector: `input[type="checkbox"]`. Ini membuat proses design form menjadi lebih mudah.

Lima selector yang sudah kita bahas, yakni universal selector, element selector, class selector, id selector dan attribute selector termasuk ke dalam kelompok **simple selectors** karena penggunaannya cukup sederhana.

5.7. Group Selector

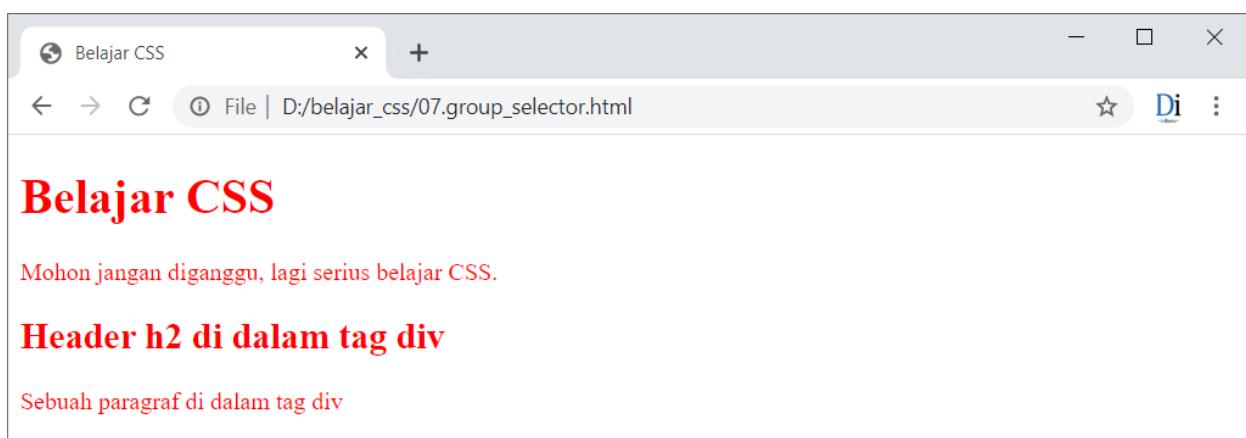
Group selector sebenarnya bukanlah 'selector murni', tapi hanya sebutan untuk cara menggabungkan beberapa selector ke dalam satu perintah.

Agar lebih mudah dipahami, perhatikan kode HTML berikut:

07.group_selector.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h1 {
8              color: red;
9          }
10         h2 {
11             color: red;
12         }
13         p {
14             color: red;
15         }
16     </style>
17 </head>
18 <body>
19     <h1>Belajar CSS</h1>
20     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
21     <div>
22         <h2>Header h2 di dalam tag div</h2>
23         <p>Sebuah paragraf di dalam tag div</p>
24     </div>
25 </body>
26 </html>
```



Gambar: Hasil tampilan beberapa element selector

Dalam kode ini saya menggunakan 3 buah element selector, yakni **h1**, **h2** dan **p**. Ketiganya dipakai untuk mengubah warna teks menjadi merah. Walaupun penulisan tersebut tidak salah,

tapi kita bisa menggabungkan ketiga selector ke dalam 1 baris sebagai berikut:

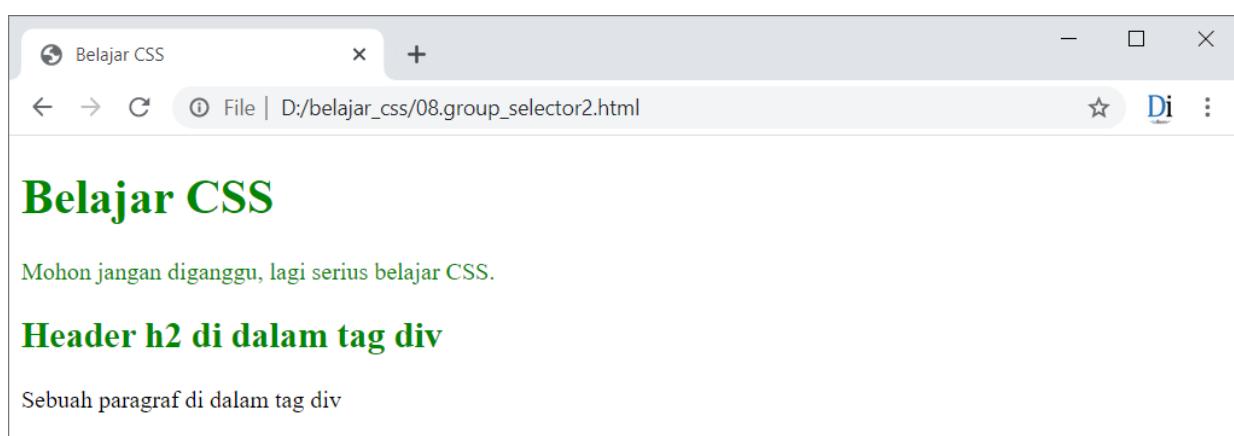
```
h1, h2, p {
    color: red;
}
```

Kode CSS di atas akan mengubah warna teks menjadi merah untuk tag `<h1>`, `<h2>`, dan `<p>` atau sama dengan kode CSS sebelum ini. Tanda koma: `" , "` dipakai sebagai tanda pemisah dari **group selector**. Tambahan karakter spasi di antara group selector ini tidak berpengaruh, sehingga bisa ditulis sebagai `h1,h2,p` maupun `h1 , h2 , p`.

Penggunaan **group selector** tidak terbatas kepada *element selector* saja, tapi juga bisa untuk menyatukan selector lain seperti *class selector*, *id selector*, atau *attribute selector*, seperti contoh berikut:

08.group_selector2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .judul-utama, #headerKedua, [title~="serius"] {
8              color: green;
9          }
10     </style>
11 </head>
12 <body>
13     <h1 class="judul-utama">Belajar CSS</h1>
14     <p title="lagi serius">Mohon jangan diganggu, lagi serius belajar CSS.</p>
15     <div>
16         <h2 id="headerKedua">Header h2 di dalam tag div</h2>
17         <p>Sebuah paragraf di dalam tag div</p>
18     </div>
19 </body>
20 </html>
```



Gambar: Contoh penggunaan group selector

Kali ini saya menggabung 3 jenis selector dalam 1 baris, ketiganya akan membuat teks berwarna hijau. CSS tidak membatasi berapa banyak selector yang bisa digabung ke dalam 1 group, bisa 3, 5, bahkan 80 selector jika memang di butuhkan.

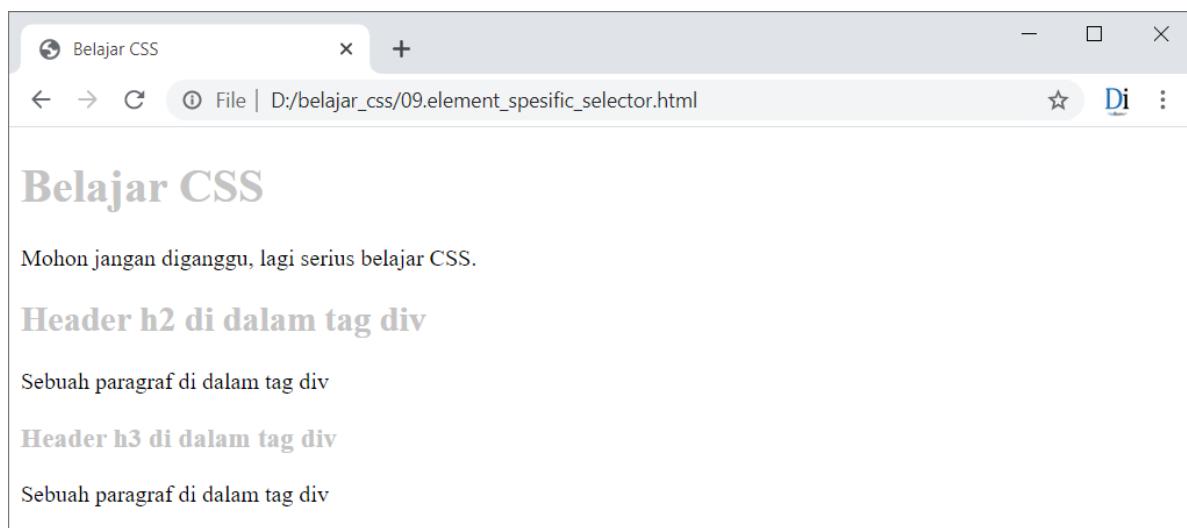
5.8. Element Spesific Selector

Sama seperti *group selector*, **element spesific selector** juga berupa penggabungan beberapa selector untuk membuat selector yang lebih spesifik. Langsung saja kita lihat contoh penggunaannya:

09.element_spesific_selector.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .judul {
8              color: silver;
9          }
10     </style>
11 </head>
12 <body>
13     <h1 class="judul">Belajar CSS</h1>
14     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15     <div>
16         <h2 class="judul">Header h2 di dalam tag div</h2>
17         <p>Sebuah paragraf di dalam tag div</p>
18         <h3 class="judul">Header h3 di dalam tag div</h2>
19         <p>Sebuah paragraf di dalam tag div</p>
20     </div>
21 </body>
22 </html>
```



Gambar: Hasil tampilan class selector

Selector `.class="judul"` adalah sebuah **class selector**. Pada kode HTML, terdapat atribut `class="judul"` di dalam tag `<h1>`, `<h2>` dan `<h3>`. Akibatnya semua tag header ini akan berwarna silver sesuai dengan kode CSS `color: silver`.

Bagaimana jika saya ingin mengubah warna teks hanya untuk tag `<h2>` yang memiliki `class="judul"` saja? Caranya, tambah element selector `h2` di depan class selector seperti contoh berikut:

```
h2.judul {  
    color: silver;  
}
```

Selector ini hanya akan mencari tag `<h2>` yang memiliki class `judul`. Perhatikan bahwa antara "`h2`" dan "`.judul`" **tidak boleh ada spasi**. Ini sangat penting karena jika terdapat spasi di antara keduanya, selector itu akan berubah fungsi.

Element spesific selector juga bisa dipakai untuk selector lain seperti **id selector** dan **atribut selector** seperti kode berikut:

```
p#penting {  
    color: yellow;  
}
```

Kode CSS di atas berarti: Ubah warna teks menjadi kuning untuk seluruh paragraf yang memiliki `id="penting"`.

Lebih jauh lagi, kita bisa mengombinasikan *element spesific selector* dengan *group selector* seperti contoh berikut:

```
p.pesan, h2#judul, span[title] {  
    color: blue;  
}
```

Walaupun terkesan kompleks, tapi saya yakin anda sudah bisa memahami maksud dari kode CSS ini.

5.9. Descendant Selector

Descendant selector adalah sebutan untuk selector yang mencari tag HTML berdasarkan struktur hubungan induk-anak (**parent-child relationship**).

Di buku **HTML Uncover** saya sempat menyinggung **DOM** (Document Object Model). DOM adalah sebutan keren untuk struktur halaman HTML, dimana setiap tag bisa menjadi **induk** (*parent*) dan **anak** (*child*).

Sebagai contoh, perhatikan kode HTML berikut:

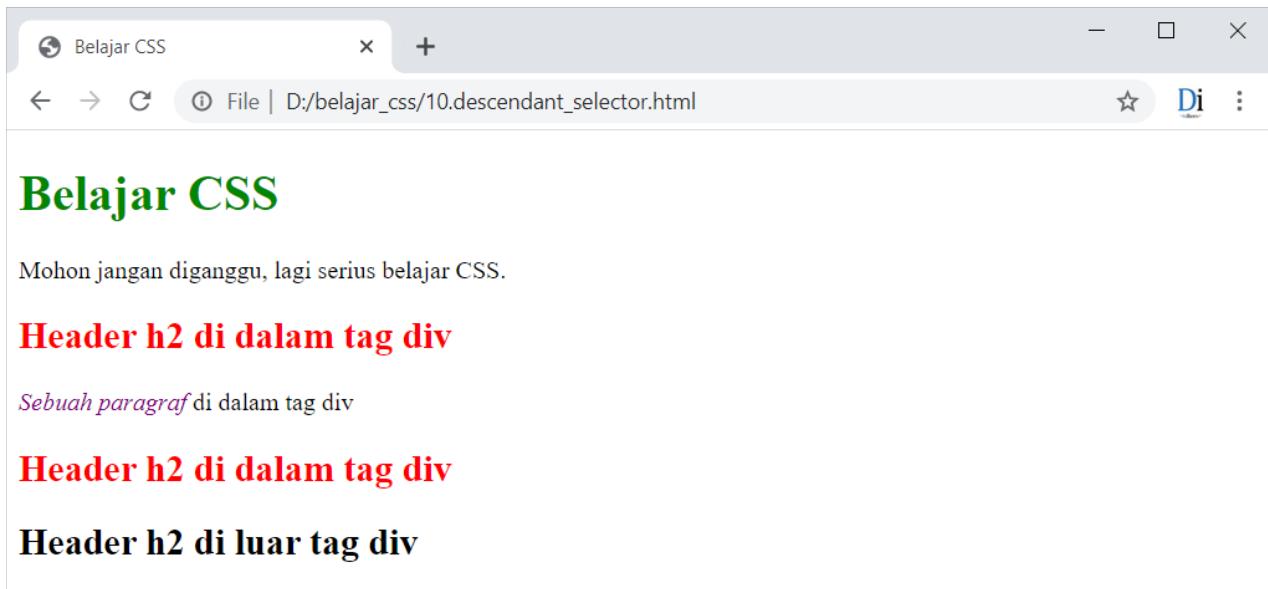
```
<div>
  <h2>Header h2 di dalam tag div</h2>
  <p><em>Sebuah paragraf </em>di dalam tag div</p>
</div>
```

Kode ini terdiri dari tag `<div>`, `<h2>`, `<p>`, dan `` yang dapat dilihat sebagai sebuah pohon keluarga. Tag `<div>` merupakan induk (parent) dari tag `<h2>` dan `<p>`, karena kedua tag ini berada di dalam tag `<div>`. Begitu pula dengan tag `` yang menjadi anak (child) dari tag `<p>`, karena berada di dalamnya.

Descendant selector memungkinkan kita untuk mencari tag HTML berdasarkan struktur seperti ini. Untuk menggunakannya, gabung penulisan beberapa selector yang dipisah dengan sebuah **spasi** seperti contoh berikut:

10.descendant_selector.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      div h2 {
8        color: red;
9      }
10   p em {
11     color: purple;
12   }
13   body h1 {
14     color: green;
15   }
16  </style>
17 </head>
18 <body>
19  <h1>Belajar CSS</h1>
20  <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
21  <div>
22    <h2>Header h2 di dalam tag div</h2>
23    <p><em>Sebuah paragraf </em>di dalam tag div</p>
24    <h2>Header h2 di dalam tag div</h2>
25  </div>
26  <h2>Header h2 di luar tag div</h2>
27 </body>
28 </html>
```



Gambar: Contoh penggunaan descendant_selector

Agar dapat menggunakan *descendant selector*, kita harus paham bagaimana struktur HTML yang ada. Jika tidak, maka tidak akan bisa menggunakan *descendant selector*.

Selector pertama:

```
div h2 {
    color: red;
}
```

Bisa dibaca: *ubah warna teks menjadi merah untuk setiap tag <h2> yang berada di dalam tag <div>*. Perhatikan di antara `div` dan `h2` dipisah dengan sebuah spasi. Di dalam struktur HTML, terdapat 2 buah tag `<h2>` yang ada di dalam tag `<div>`, sehingga keduanya akan berwarna merah.

Selector kedua:

```
p em {
    color: purple;
}
```

Berarti: *ubah warna teks menjadi ungu untuk setiap tag yang berada di dalam tag <p>*. Hasilnya, kata "Sebuah paragraf" akan berwarna ungu (*purple*) karena kata tersebut berada di dalam tag `` yang menjadi *child* dari tag `<p>`.

Selector terakhir:

```
body h1 {
    color: green;
}
```

Berarti: *ubah warna teks menjadi hijau untuk seluruh tag <h1> yang berada di dalam tag*

<body>. Sebenarnya selector "body" tidak perlu kita tulis karena seluruh element HTML yang tampil adalah *child* dari tag <body>.

Sebagai latihan tambahan, dapatkah anda menjelaskan maksud dari selector berikut?

```
p .keterangan {  
    color: gray;  
}
```

Selector ini akan *mengubah warna teks menjadi gray untuk setiap tag HTML yang memiliki atribut class="keterangan" dan berada di dalam tag <p>*. Jika terdapat tag HTML yang juga memiliki class="keterangan" tetapi tidak berada di dalam tag <p>, maka tidak akan dikenai oleh selector ini.

Untuk lebih memastikan pemahaman, bisakah anda membedakan maksud kedua selector ini?

```
div.pesan {  
    color: green;  
}  
  
div .pesan {  
    color: purple;  
}
```

Penulisan kedua selector hanya berbeda oleh sebuah **spasi!** Dimana salah satunya adalah *element spesific selector* dan yang satunya adalah *descendant selector*.

Selector pertama akan mengubah warna teks untuk tag <div> dengan class="pesan". Sedangkan selector kedua akan mengubah warna teks untuk tag apapun dengan atribut class="pesan" yang berada di dalam tag <div>, tapi bukan untuk tag <div> itu sendiri.

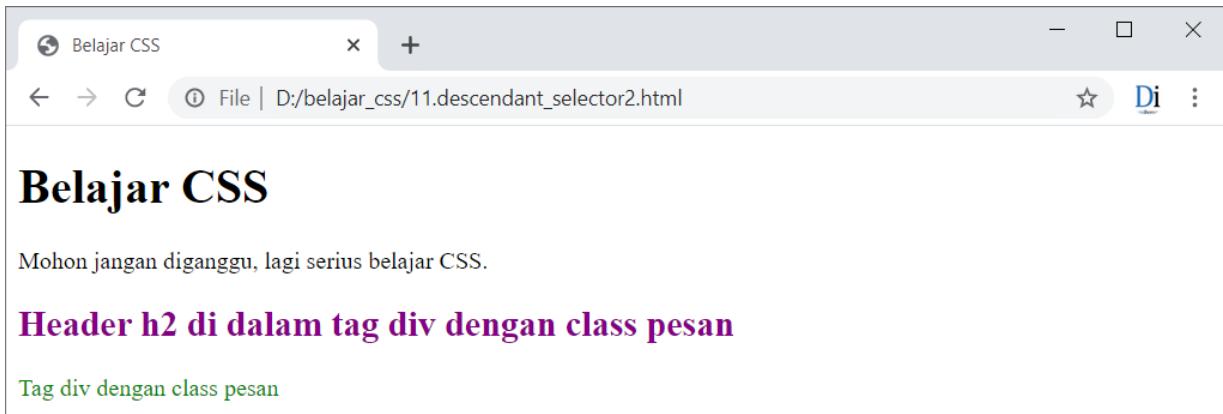
Berikut contoh hasilnya di dalam sebuah halaman HTML:

11. descendant_selector2.html

```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <style>  
7          div.pesan {  
8              color: green;  
9          }  
10         div .pesan {  
11             color: purple;  
12         }  
13     </style>  
14 </head>  
15 <body>  
16     <h1>Belajar CSS</h1>  
17     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
```

CSS Selector

```
18 <div>
19   <h2 class="pesan">Header h2 di dalam tag div dengan class pesan</h2>
20 </div>
21 <div class="pesan">Tag div dengan class pesan</div>
22 </body>
23 </html>
```



Gambar: Beda element spesific selector dengan descendant selector

Perbedaan antara *element specific selector* dengan *descendant selector* ini sangat penting untuk dipahami karena keduanya cukup sering dipakai.

Kita juga tidak dibatasi seberapa banyak element yang bisa digabung. Selector berikut dianggap valid dan diperbolehkan:

```
body div p .pesan #penting .utama em {
  color: purple;
}
```

Akan tetapi selector ini kurang efisien karena web browser butuh waktu beberapa saat dalam menelusuri semua struktur DOM untuk mencari tag ``. Best practice-nya, gunakan maksimal 3 level untuk *descendant selector* (3 tingkat element).

5.10. Child Selector

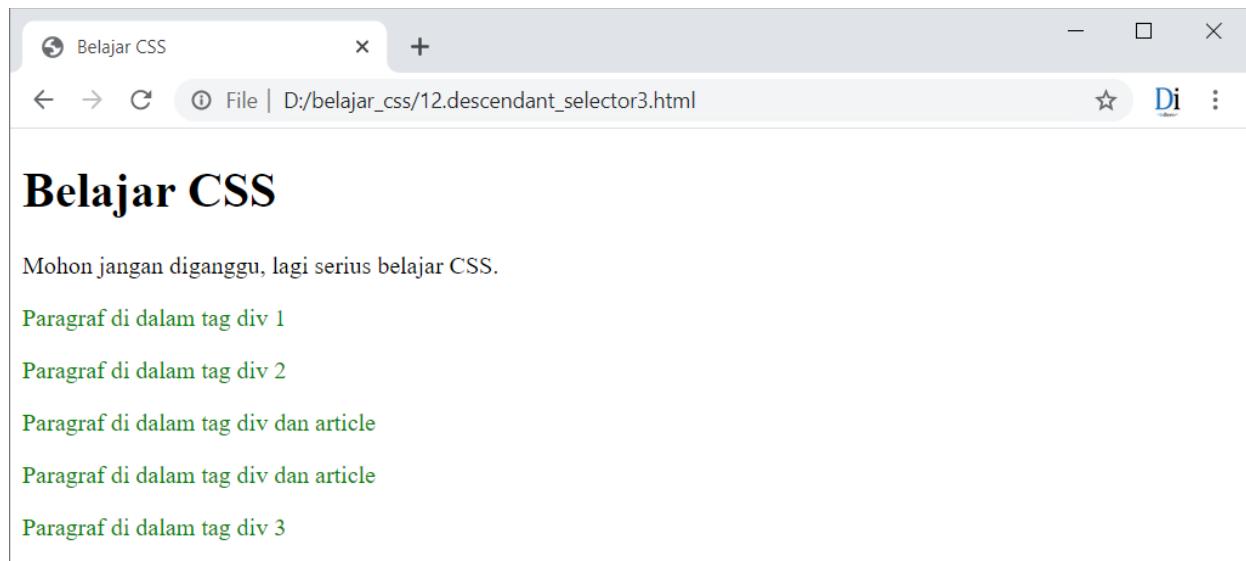
Mirip dengan *descendant selector*, **child selector** juga menggunakan struktur DOM dan *parent child relationship*. Bedanya, *child selector* hanya akan mencari tag HTML yang menjadi anak pertama (**first child**) dari struktur DOM. Penjelasan ini lebih mudah dengan menggunakan contoh:

12.descendant_selector3.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
```

CSS Selector

```
7      div p {
8          color: green;
9      }
10     </style>
11 </head>
12 <body>
13     <h1>Belajar CSS</h1>
14     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15     <div>
16         <p>Paragraf di dalam tag div 1</p>
17         <p>Paragraf di dalam tag div 2</p>
18         <article>
19             <p>Paragraf di dalam tag div dan article</p>
20             <p>Paragraf di dalam tag div dan article</p>
21         </article>
22         <p>Paragraf di dalam tag div 3</p>
23     </div>
24 </body>
25 </html>
```



Gambar: Contoh penggunaan child selector ?

Silahkan pelajari sejenak struktur HTML di atas. Saya membuat beberapa tag `<p>` di dalam tag `<div>` serta tag `<article>` yang juga memiliki tag `<p>` tambahan.

Selector berikut:

```
div p {
    color: green;
}
```

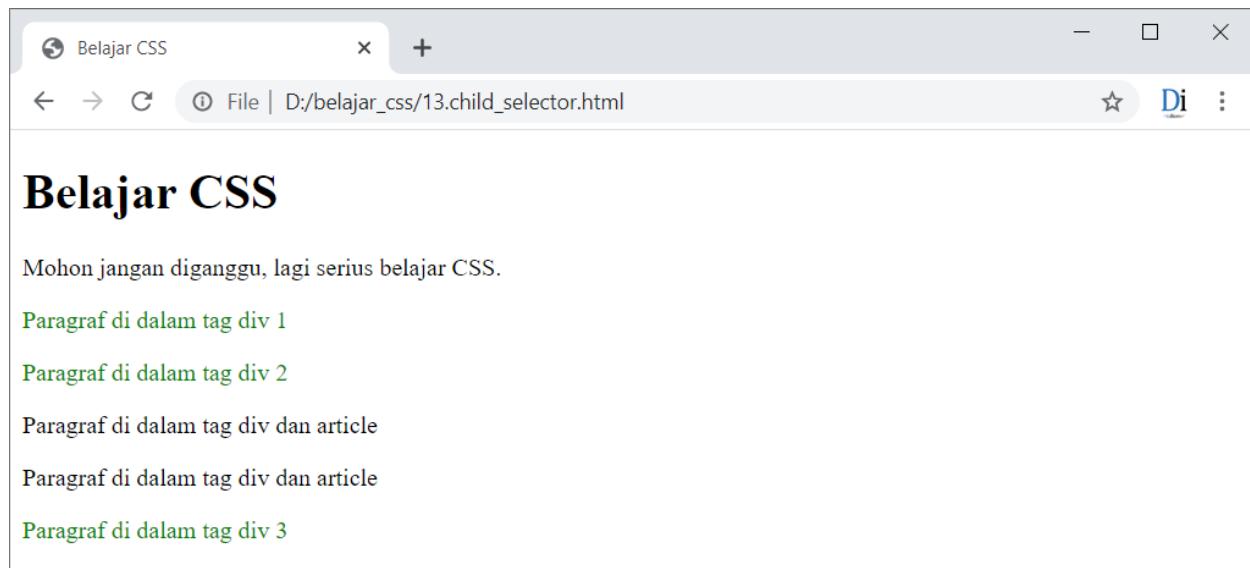
Akan mengubah warna teks untuk seluruh tag `<p>` yang berada di dalam tag `<div>`, berapa pun 'dalamnya'. Ini adalah *descendant selector* yang kita pelajari sebelumnya.

Berbeda dengan *descendant selector*, **child selector** hanya akan mencari anak pertama (*first child*) dari tag induk (*parent*). Ini ditulis menggunakan tanda besar dari "`>`" seperti berikut ini:

CSS Selector

13.child_selector.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div > p {
8       color: green;
9     }
10  </style>
11 </head>
12 <body>
13  <h1>Belajar CSS</h1>
14  <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15  <div>
16    <p>Paragraf di dalam tag div 1</p>
17    <p>Paragraf di dalam tag div 2</p>
18    <article>
19      <p>Paragraf di dalam tag div dan article</p>
20      <p>Paragraf di dalam tag div dan article</p>
21    </article>
22    <p>Paragraf di dalam tag div 3</p>
23  </div>
24 </body>
25 </html>
```

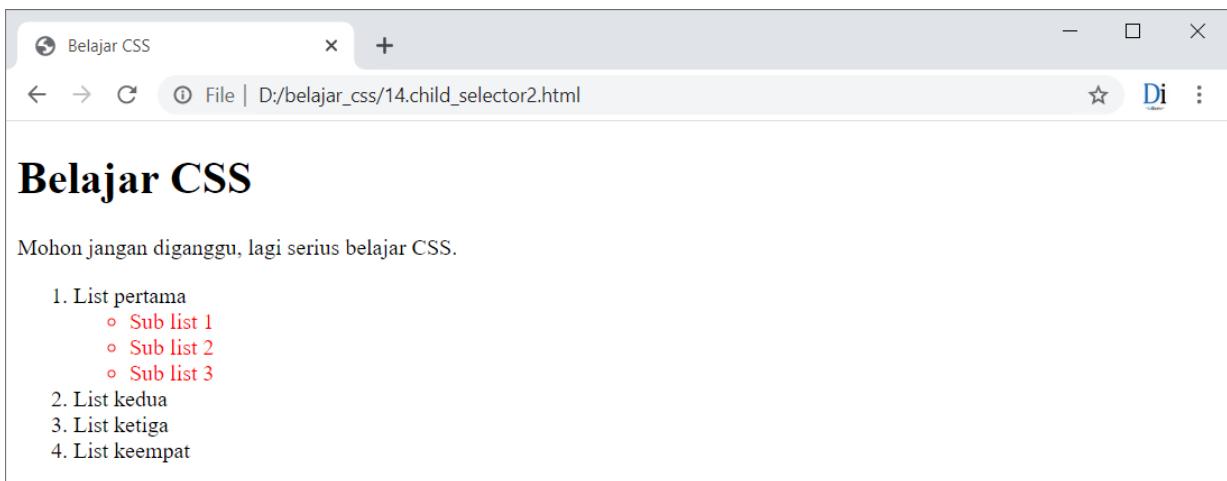


Sebagai contoh lain, dalam kode berikut saya menerapkan penggunaan *child selector* dalam list bersarang (*nested list*), yakni list di dalam list:

14.child_selector2.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          ol ul>li {
8              color: red;
9          }
10     </style>
11 </head>
12 <body>
13     <h1>Belajar CSS</h1>
14     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15     <ol>
16         <li>List pertama
17             <ul>
18                 <li>Sub list 1</li>
19                 <li>Sub list 2</li>
20                 <li>Sub list 3</li>
21             </ul>
22         </li>
23         <li>List kedua</li>
24         <li>List ketiga</li>
25         <li>List keempat</li>
26     </ol>
27 </body>
28 </html>
```



Gambar: Contoh penggunaan child selector pada nested list

Selector berikut:

```
ol ul>li {
  color: red;
}
```

Bisa dibaca: Ubah warna teks menjadi merah untuk tag `` yang merupakan first child dari tag ``, dimana tag `` harus berada di dalam tag ``.

5.11. Adjacent Selector

Adjacent selector masih berhubungan dengan struktur DOM, kali ini yang akan disinggung adalah *sibling relationship* (hubungan saudara). Perhatikan kode HTML berikut:

```
<div>
  <h2>Header h2 di dalam tag div</h2>
  <p>Sebuah paragraf di dalam div</p>
  <p>Sebuah paragraf lain di dalam div</p>
</div>
```

Kode ini berisi tag `<h2>` dan `<p>` yang berada di dalam tag `<div>`. Menggunakan *parent-child relationship*, tag `<h2>` dan `<p>` adalah *child* dari tag `<div>`. Sebaliknya, tag `<div>` adalah *parent* dari tag `<h2>` dan `<p>`. Karena memiliki parent yang sama, tag `<h2>` dan `<p>` adalah *sibling* (saudara).

Adjacent selector bisa dipakai untuk mencari tag HTML berdasarkan *sibling relationship* dan harus saling berdekatan. Untuk menulisnya, pisah dua selector dengan karakter tambah "+" seperti contoh berikut:

```
h2 + p {
  color: green;
}
```

Selector ini berarti: cari 1 tag `<p>` yang berada setelah tag `<h2>` dan merupakan *sibling* (saudara)

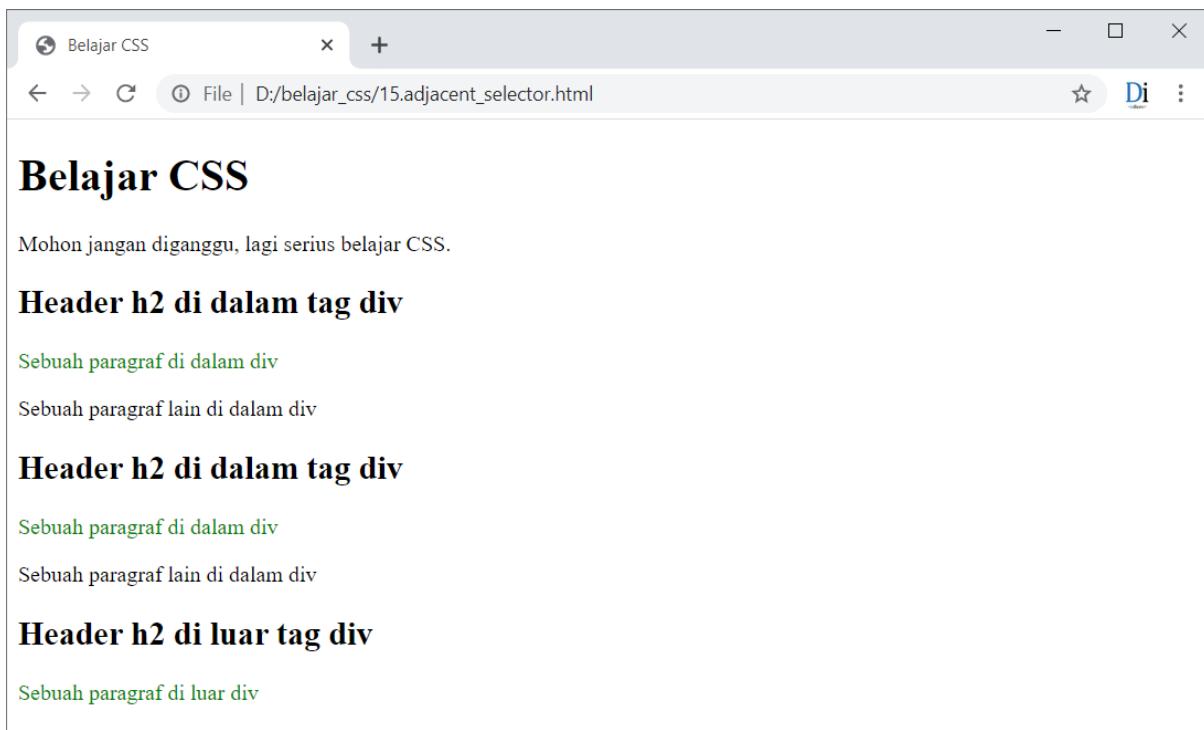
Berikut praktik lengkapnya:

15.adjacent_selector.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      h2 + p {
8        color: green;
9      }
10   </style>
11 </head>
```

CSS Selector

```
12 <body>
13   <h1>Belajar CSS</h1>
14   <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15
16   <div>
17     <h2>Header h2 di dalam tag div</h2>
18     <p>Sebuah paragraf di dalam div</p>
19     <p>Sebuah paragraf lain di dalam div</p>
20   </div>
21
22   <div>
23     <h2>Header h2 di dalam tag div</h2>
24     <p>Sebuah paragraf di dalam div</p>
25     <p>Sebuah paragraf lain di dalam div</p>
26   </div>
27
28   <h2>Header h2 di luar tag div</h2>
29   <p>Sebuah paragraf di luar div</p>
30 </body>
31 </html>
```



Gambar: Contoh penggunaan adjacent selector

Hasilnya, setiap tag `<p>` pertama yang ada setelah tag `<h2>` akan berwarna hijau.

Bagi pemula yang baru belajar CSS sering dibuat bingung dengan perbedaan antara *adjacent selector* dan *descendant selector*, setidaknya saya dulu demikian. Untuk memastikan pemahaman, bisakah anda membedakan arti kedua selector ini?

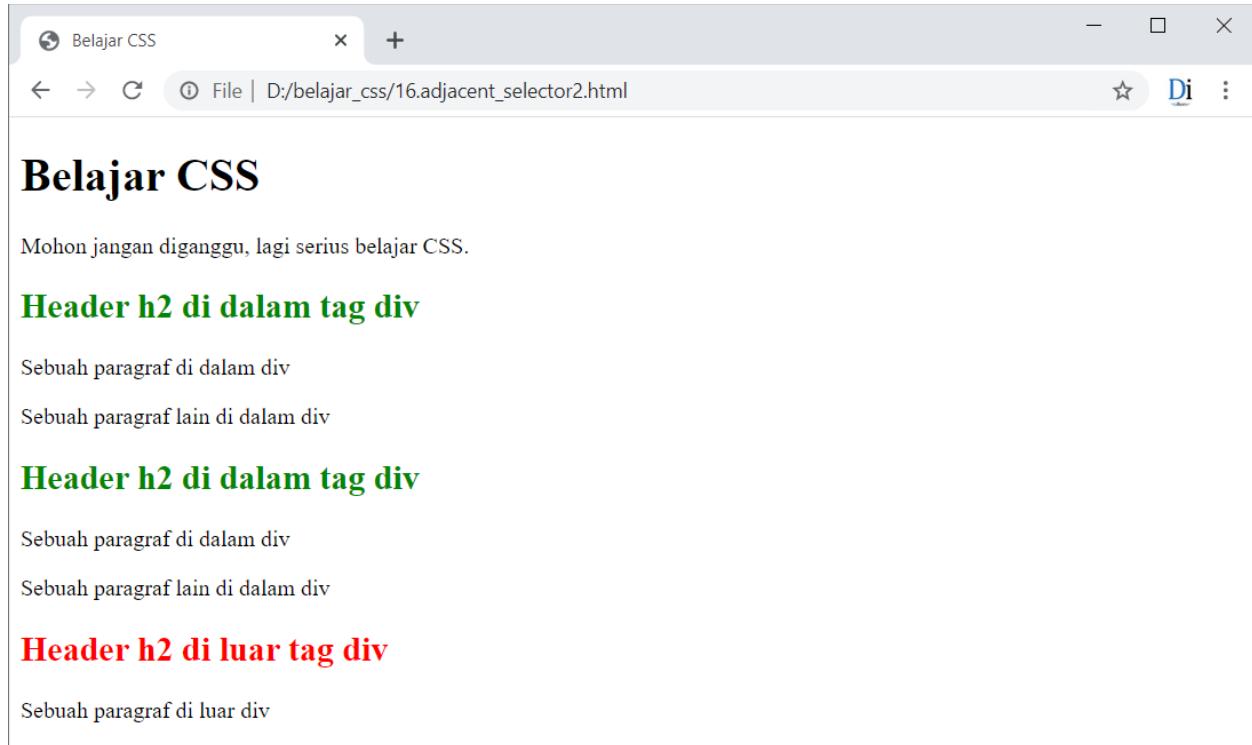
```
div h2 { ... }
div + h2 { ... }
```

Selector `div h2` akan mencari **seluruh** tag `<h2>` yang berada di dalam tag `<div>`, sedangkan selector `div + h2` akan mencari **satu** tag `<h2>` yang berada persis setelah tag `<div>`.

Contoh berikut bisa memperjelas maksud kedua selector ini:

16.adjacent_selector2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div h2 {
8              color: green;
9          }
10         div + h2 {
11             color: red;
12         }
13     </style>
14 </head>
15 <body>
16     <h1>Belajar CSS</h1>
17     <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
18
19     <div>
20         <h2>Header h2 di dalam tag div</h2>
21         <p>Sebuah paragraf di dalam div</p>
22         <p>Sebuah paragraf lain di dalam div</p>
23     </div>
24
25     <div>
26         <h2>Header h2 di dalam tag div</h2>
27         <p>Sebuah paragraf di dalam div</p>
28         <p>Sebuah paragraf lain di dalam div</p>
29     </div>
30
31     <h2>Header h2 di luar tag div</h2>
32     <p>Sebuah paragraf di luar div</p>
33 </body>
34 </html>
```



Gambar: Contoh beda adjacent selector dengan descendant selector

Dua tag `<h2>` pertama akan berwarna hijau karena cocok dengan selector `div h2`. Serta tag `<h2>` terakhir akan berwarna merah karena cocok dengan selector `div + h2`.

5.12. General Sibling Selector

General sibling selector sangat mirip dengan *adjacent selector*. Bedanya, jika pada *adjacent selector* hanya cocok dengan satu tag yang menjadi sibling, *general sibling selector* akan cocok dengan seluruh sibling.

Karakter **tilde** " ~ " dipakai untuk membuat *general sibling selector*. Tanda ini berada di sisi kiri atas keyboard (sebelum angka 1), dan ditekan menggunakan SHIFT. Berikut contoh penulisannya:

`h2 ~ p { ... }`

Selector ini bisa dibaca: *cari seluruh tag <p> yang berada setelah tag <h2> dan merupakan sibling (saudara)*.

16a.general_sibling_selector.html

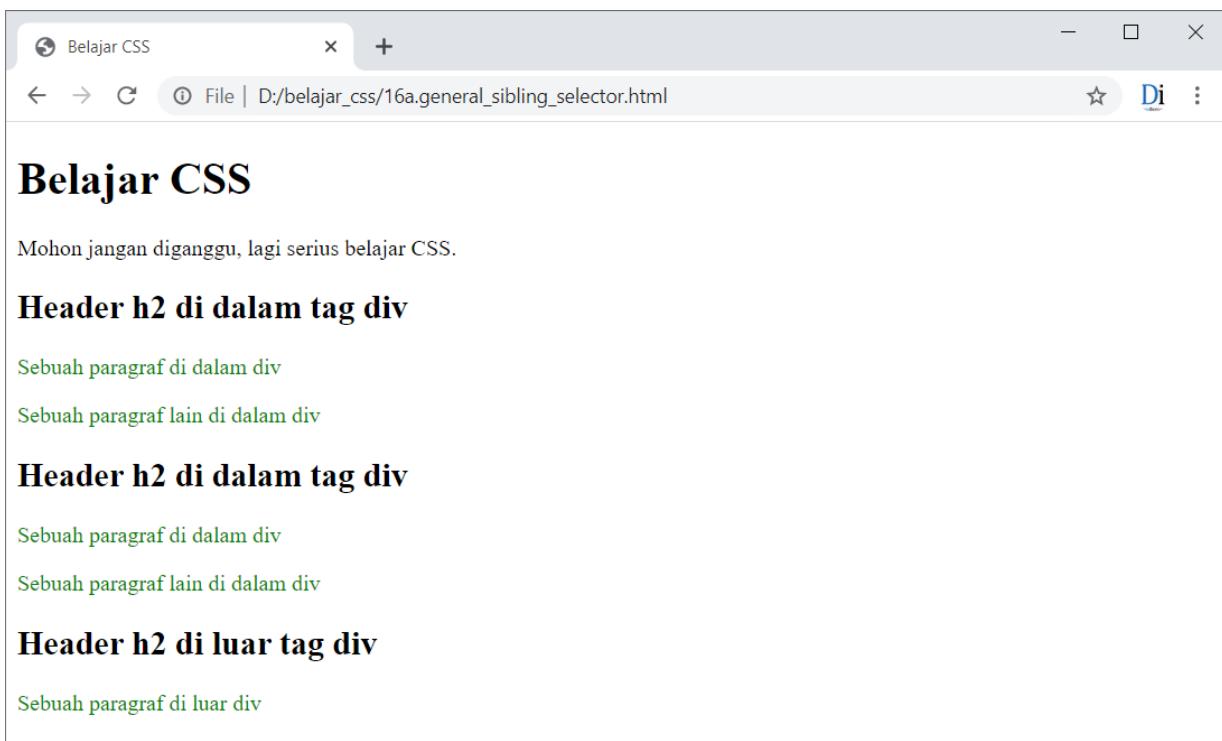
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>

```

CSS Selector

```
6  <style>
7      h2 ~ p {
8          color: green;
9      }
10 </style>
11 </head>
12 <body>
13 <h1>Belajar CSS</h1>
14 <p>Mohon jangan diganggu, lagi serius belajar CSS.</p>
15
16 <div>
17     <h2>Header h2 di dalam tag div</h2>
18     <p>Sebuah paragraf di dalam div</p>
19     <p>Sebuah paragraf lain di dalam div</p>
20 </div>
21
22 <div>
23     <h2>Header h2 di dalam tag div</h2>
24     <p>Sebuah paragraf di dalam div</p>
25     <p>Sebuah paragraf lain di dalam div</p>
26 </div>
27
28 <h2>Header h2 di luar tag div</h2>
29 <p>Sebuah paragraf di luar div</p>
30 </body>
31 </html>
```



Gambar: Contoh penggunaan general sibling selector

Kali ini kedua tag `<p>` yang berada setelah tag `<h2>` akan berwarna hijau, tidak hanya satu seperti *adjacent selector*. Untuk paragraf "Mohon jangan diganggu, lagi serius belajar CSS"

tetap berwarna hitam karena berada sebelum tag <h2>.

5.13. Dynamic Pseudo Class Selector

Dari seluruh selector yang sudah kita bahas sejauh ini, semuanya digunakan untuk mencari element yang ada di dalam struktur DOM, yakni kode HTML yang memang terlihat secara langsung.

Sekarang kita akan masuk ke jenis lain dari selector, yakni *pseudo class selector*. **Pseudo class selector** adalah selector CSS untuk mencari element HTML yang tidak terlihat secara langsung (tidak ada di dalam struktur DOM).

Salah satu hal unik dengan *pseudo class selector* adalah penulisannya diawali dengan tanda titik dua ":" seperti :hover, :active, atau :visited.

Pseudo class selector yang akan kita bahas adalah:

- **Dynamic pseudo class selector**
- **UI element state pseudo class selector**
- **Structural pseudo class selector**

Kita akan mulai dari *dynamic pseudo class selector* terlebih dahulu.

Dynamic pseudo class selector dipakai untuk mencari struktur HTML berdasarkan suatu hal yang berubah karena interaksi user.

Terdapat beberapa jenis *dynamic pseudo class selector* dalam CSS. Yang akan kita bahas adalah:

- :link
- :visited
- :hover
- :active
- :target
- :focus

Jenis-jenis *pseudo class selector* terus bertambah dari waktu ke waktu sesuai standar baru yang dikembangkan oleh W3C. Agar pembahasan kita tidak terlalu panjang, saya memilih beberapa selector yang cukup sering dipakai saja. Jika anda tertarik mempelajari semua *pseudo selector* ini, bisa lanjut ke [MDN Web Docs: Pseudo-classes](#)¹³.

Selector :link

Selector :link akan mencari seluruh link di dalam HTML. Fungsinya mirip dengan selector a

¹³ <https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes>

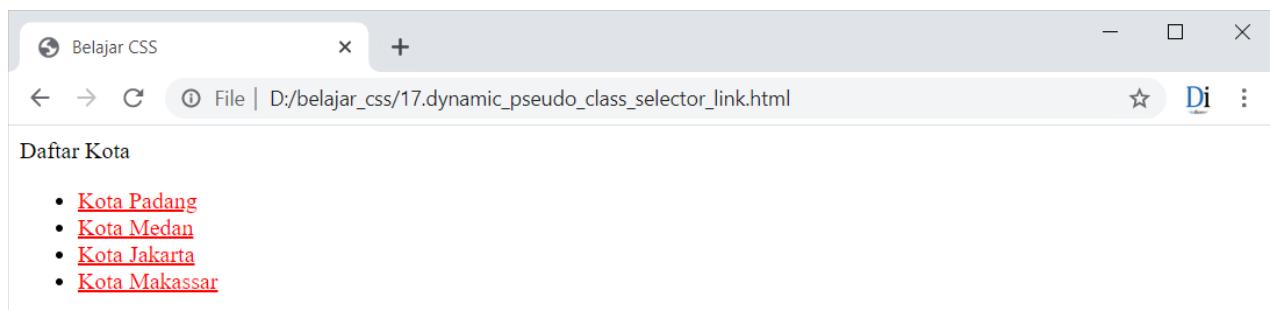
yang akan mencari seluruh tag `<a>`. Namun perlu diingat bahwa tidak semua tag `<a>` adalah link:

17. dynamic_pseudo_class_selector_link.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          :link { color:red; }
8      </style>
9  </head>
10 <body>
11     <a id="daftar_kota">Daftar Kota</a>
12     <ul>
13         <li><a href="padang.html">Kota Padang</a></li>
14         <li><a href="medan.html">Kota Medan</a></li>
15         <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
16         <li><a href="makassar.pdf">Kota Makassar</a></li>
17     </ul>
18 </body>
19 </html>

```



Gambar: Selector `:link` hanya mencari tag `<a>` dengan atribut `href`

Dalam kode ini saya membuat beberapa link di dalam unordered list. Selector `:link` di baris 7 akan mencari seluruh tag `<a>` yang memiliki atribut `href`. Di dalam list, semua tag `<a>` memiliki atribut `href` sehingga dianggap sebagai link.

Sedangkan tag `Daftar Kota` tidak ikut berwarna merah karena tidak memiliki atribut `href` (tidak dianggap sebagai link).

Selector `:visited`

Lanjut ke *dynamic pseudo class selector* kedua, selector `:visited` dipakai untuk mencari sebuah link yang telah dikunjungi. Mari langsung kita lihat contoh penggunaannya:

18. dynamic_pseudo_class_selector_visited.html

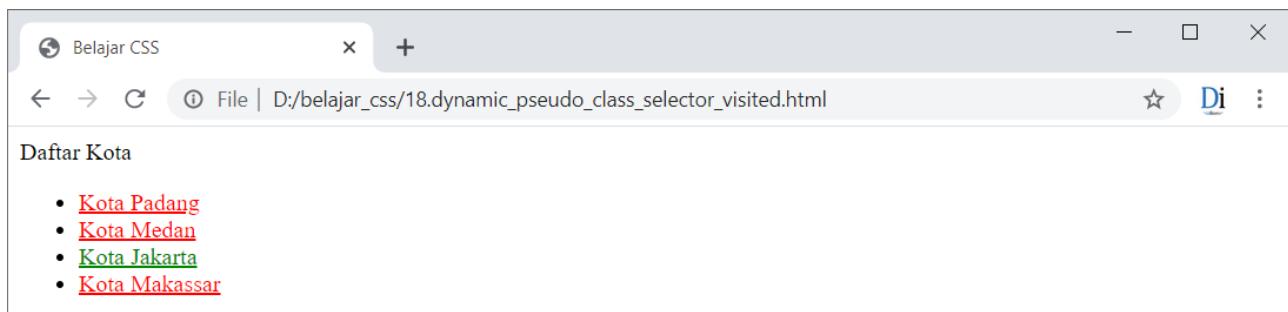
```

1  <!DOCTYPE html>
2  <html lang="id">

```

CSS Selector

```
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     :link    { color:red;   }
8     :visited { color:green; }
9   </style>
10 </head>
11 <body>
12   <a id="daftar_kota">Daftar Kota</a>
13   <ul>
14     <li><a href="padang.html">Kota Padang</a></li>
15     <li><a href="medan.html">Kota Medan</a></li>
16     <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
17     <li><a href="makassar.pdf">Kota Makassar</a></li>
18   </ul>
19 </body>
20 </html>
```



Gambar: Selector :visited akan mencari link yang sudah dikunjungi

Saat halaman tampil pertama kali, kemungkinan besar semua link akan berwarna merah. Ini berasal dari selector :link di baris 7. Kemudian silahkan klik link "Kota Jakarta" agar terbuka halaman penjelasan di wikipedia. Web browser secara otomatis mencatat web ini sebagai salah satu alamat yang pernah dikunjungi (*visited*).

Lalu buka kembali file HTML di atas dan sekarang link "Kota Jakarta" akan tampil dengan warna hijau. Inilah efek dari selector :visited.

Selector :hover

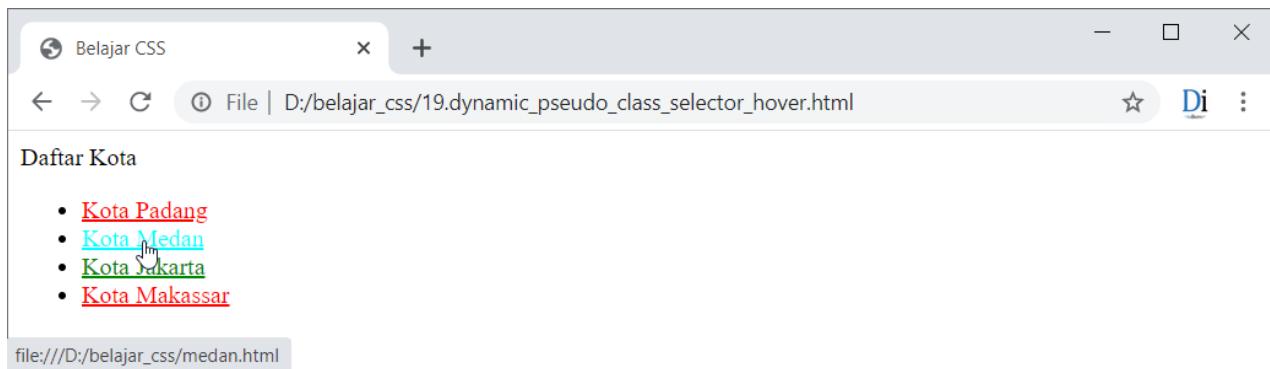
Dari semua *dynamic pseudo class selector*, :hover adalah yang paling menarik dan sering dipakai. Selector ini akan aktif ketika cursor mouse berada di atas sebuah element. Berikut contoh penggunaannya:

19.dynamic_pseudo_class_selector_hover.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
```

CSS Selector

```
6  <style>
7      :link    { color:red;    }
8      :visited { color:green; }
9      a:hover { color:aqua;   }
10 </style>
11 </head>
12 <body>
13     <a id="daftar_kota">Daftar Kota</a>
14     <ul>
15         <li><a href="padang.html">Kota Padang</a></li>
16         <li><a href="medan.html">Kota Medan</a></li>
17         <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
18         <li><a href="makassar.pdf">Kota Makassar</a></li>
19     </ul>
20 </body>
21 </html>
```



Gambar: Selector :hover akan mewarnai link ketika mouse berada diatasnya (mouse over)

Saat kita menggerakkan cursor mouse ke salah satu link, warna teks akan berubah menjadi biru aqua. Inilah fungsi dari selector :hover.

Juga bisa di perhatikan bahwa saya menulis selector ini dengan `a:hover`. Tanda "a" di awal selector adalah implementasi dari *element specific selector* yang telah kita pelajari sebelumnya. Dengan menambahkan selector "a", maka efek hover hanya akan aktif untuk tag "a" saja.

Selector :active

Selector :active aktif hanya sepersekian detik, yakni ketika kita men-klik tombol mouse. Agar bisa melihat efeknya, tahan tombol mouse saat men-klik link berikut:

20. dynamic_pseudo_class_selector_active.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          :link    { color:red;    }
8          :visited { color:green; }
9          a:hover { color:aqua;   }
```

CSS Selector

```
10      a:active { color:yellow; }
11  
```

```
</style>
```

```
</head>
```

```
<body>
```

```
14    <a id="daftar_kota">Daftar Kota</a>
```

```
15  
```

```
<ul>
```

```
16    <li><a href="padang.html">Kota Padang</a></li>
```

```
17    <li><a href="medan.html">Kota Medan</a></li>
```

```
18    <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
```

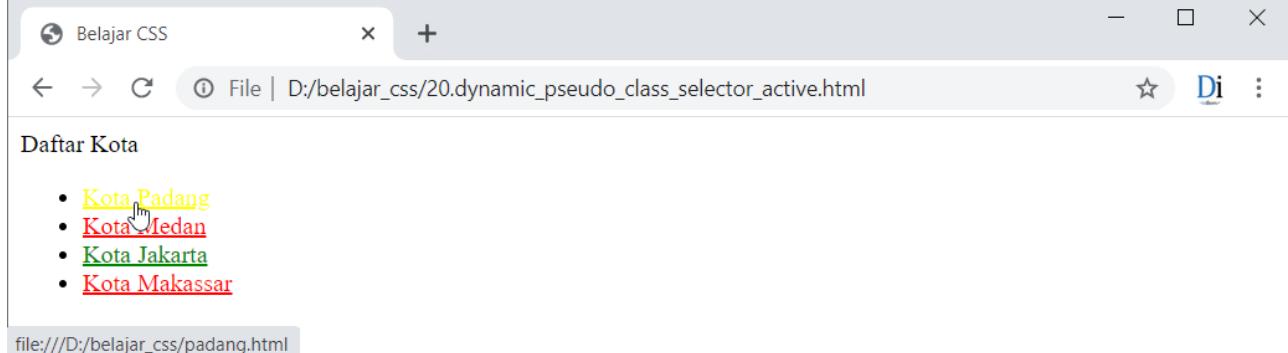
```
19    <li><a href="makassar.pdf">Kota Makassar</a></li>
```

```
20  
```

```
</ul>
```

```
21 </body>
```

```
22 </html>
```



Gambar: Selector :active akan aktif ketika link sedang di klik

Efek warna kuning bisa terlihat karena saya menahan tombol mouse pada saat men-klik link "Kota Padang", jika tidak, efek yang terjadi terlalu cepat untuk bisa dilihat.

Hal lain yang perlu diingat terkait penggunaan *dynamic pseudo class selector* adalah: posisi penulisan selector mempengaruhi efek yang tampil.

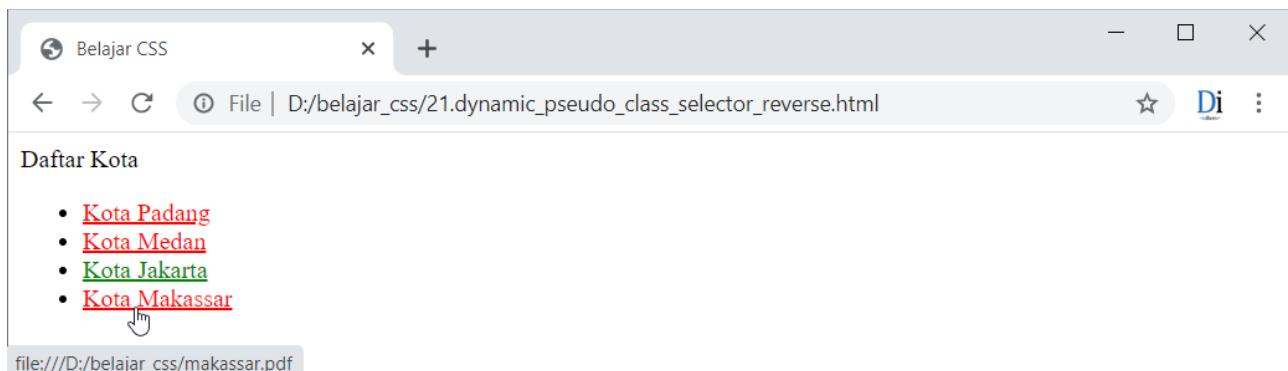
Ini terjadi karena setiap selector memiliki prioritas yang berbeda-beda. Dalam contoh di atas saya membuat urutan penulisan mulai dari :link, :visited, :hover, dan :active. Bagaimana jika urutannya dibalik?

21. dynamic_pseudo_class_selector_reverse.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      a:active { color:yellow; }
8      a:hover { color:aqua; }
9      a:visited { color:green; }
10     a:link { color:red; }
11   </style>
12 </head>
13 <body>
14   <a id="daftar_kota">Daftar Kota</a>
```

CSS Selector

```
15  <ul>
16    <li><a href="padang.html">Kota Padang</a></li>
17    <li><a href="medan.html">Kota Medan</a></li>
18    <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
19    <li><a href="makassar.pdf">Kota Makassar</a></li>
20  </ul>
21 </body>
22 </html>
```



Gambar: Efek :hover dan :active tidak lagi berjalan

Ketika saya menjalankan kode diatas, style untuk :hover dan :active tidak akan berjalan. Ini terjadi karena selector :link dan :visited akan menimpa efek dari :hover dan :active.

Oleh karena itu urutan penulisan *dynamic pseudo class selector* haruslah mulai dari :link, :visited, :hover, dan :active. Web desainer memiliki trik tersendiri untuk menghafal urutan ini: "**L**ove **H**ate", dimana awalan setiap selector tersusun dari huruf "l, v, h, dan a".

Meskipun dalam contoh ini saya menggunakan link atau tag `<a>` sebagai contoh, *dynamic pseudo class selector* tetap bisa dipakai untuk element HTML lain. Selector :hover misalnya, banyak dipakai untuk membuat efek interaktif pada gambar atau tag ``. Sebagai contoh, jika cursor mouse berada di atas gambar, jalankan efek blur. Praktek seperti ini akan kita lihat pada bab-bab selanjutnya.

Selector :target

Selector :target sangat spesifik karena baru akan aktif ketika kita mengakses sebuah "anchor link", yakni internal link pada sebuah halaman.

Menggunakan contoh kode program sebelum ini, saya menulis salah satu tag `<a>` dengan atribut `id="daftar_kota"`. Atribut `id="daftar_kota"` bisa berfungsi juga sebagai *anchor link*. Berikut modifikasi kode CSS dengan penambahan selector :target:

22.dynamic_pseudo_class_selector_target.html

```
1  <!DOCTYPE html>
```

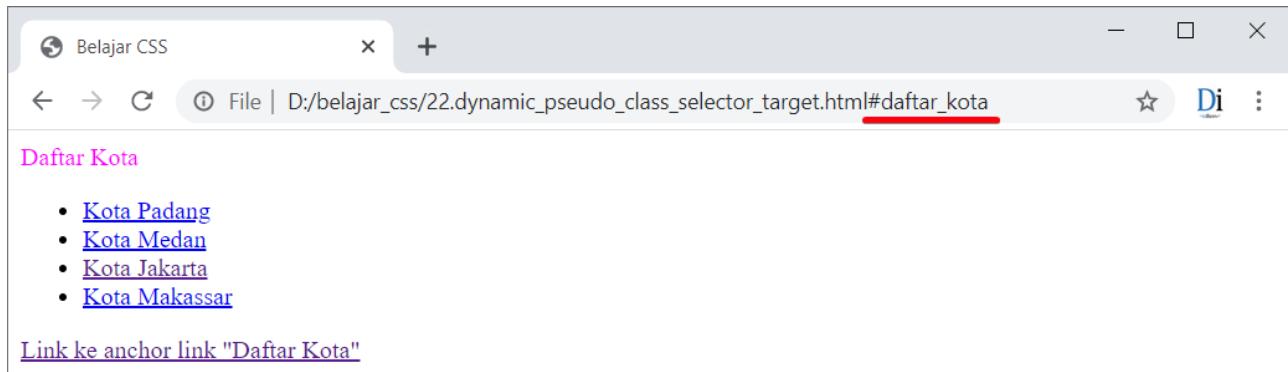
CSS Selector

```
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     :target { color: magenta; }
8   </style>
9 </head>
10 <body>
11   <a id="daftar_kota">Daftar Kota</a>
12   <ul>
13     <li><a href="padang.html">Kota Padang</a></li>
14     <li><a href="medan.html">Kota Medan</a></li>
15     <li><a href="https://en.wikipedia.org/wiki/Jakarta">Kota Jakarta</a></li>
16     <li><a href="makassar.pdf">Kota Makassar</a></li>
17   </ul>
18   <a href="#daftar_kota">Link ke anchor link "Daftar Kota"</a>
19 </body>
20 </html>
```

Saat kode di atas dijalankan, tidak akan terlihat perubahan apa-apa kecuali warna link menjadi biru (default style web browser).

Agar efek selector `:target` bisa terlihat, kita harus mengakses anchor link. Caranya, tambah `#daftar_kota` di akhir alamat URL atau klik **"Link ke anchor link "Daftar Kota"**.

Halaman web akan refresh dan kali ini anchor link "Daftar Kota" akan berwarna pink yang menandakan selector `:target` telah aktif.



Gambar: Selector `:target` akan aktif ketika sebuah anchor link dikunjungi

Selector `:focus`

Selector lain yang juga bertipe *dynamic* adalah `:focus`. Selector ini akan aktif ketika sebuah element HTML di-fokuskan, apakah menggunakan klik mouse atau tab keyboard.

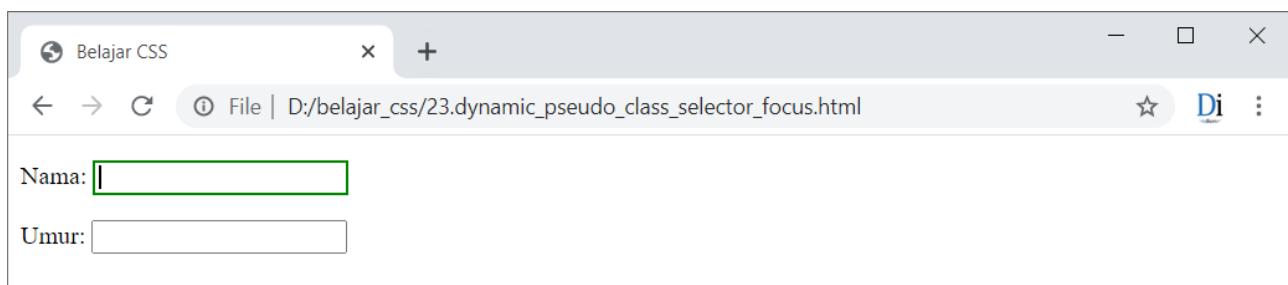
Penggunaan selector `:focus` lebih cocok untuk element form yang butuh banyak interaksi. Berikut contohnya:

23.dynamic_pseudo_class_selector_focus.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          input:focus {
8              border: solid green 2px;
9              outline: none;
10         }
11     </style>
12 </head>
13 <body>
14     <form action="register.php" method="get">
15         <p>Nama: <input type="text" name="nama" ></p>
16         <p>Umur: <input type="text" name="umur" ></p>
17     </form>
18 </body>
19 </html>

```



Gambar: Selector :active akan aktif ketika sebuah objek form di select/di klik

Ketika kita men-klik form input, warna border akan berubah sesuai dengan style yang dibuat dari selector :focus.

5.14. UI Element State Pseudo Class Selector

Dibalik namanya yang cukup panjang, **UI element state pseudo class selector** adalah jenis selector CSS yang dipakai untuk mencari element HTML berdasarkan kondisi dari element tersebut.

Umumnya selector ini digunakan untuk element form HTML. Beberapa selector yang akan kita bahas adalah :disabled, :enabled dan :checked. Sesuai dengan namanya, ketiga selector ini akan mencari form HTML dengan kondisi-kondisi tersebut. Berikut contoh penggunaannya:

24.ui_element_state_pseudo_class_selector.html

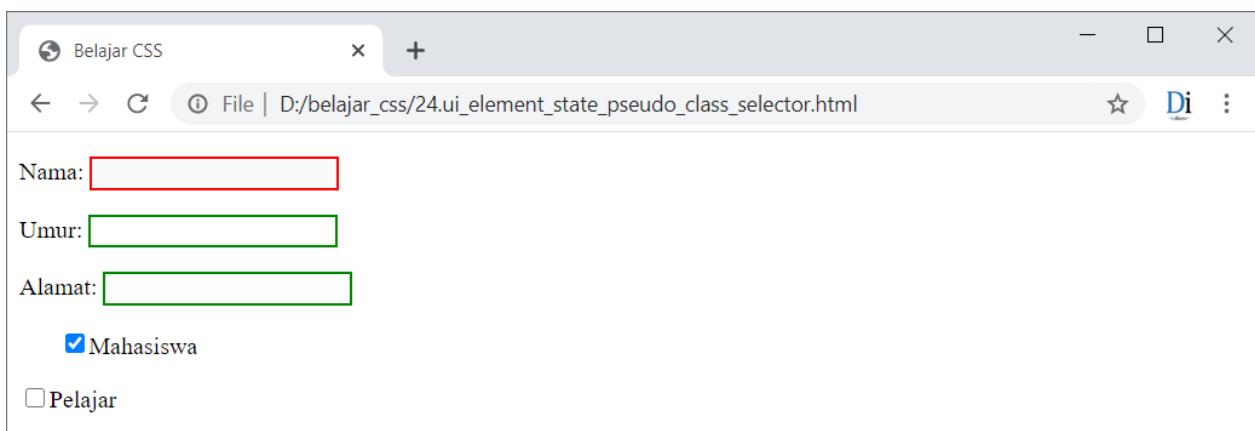
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">

```

CSS Selector

```
5 <title>Belajar CSS</title>
6 <style>
7   input:disabled {
8     border: solid red 2px;
9   }
10  input:enabled {
11    border: solid green 2px;
12  }
13  input:checked{
14    margin-left: 30px;
15  }
16 </style>
17 </head>
18 <body>
19 <form action="register.php" method="get">
20   <p>Nama: <input type="text" name="nama" disabled></p>
21   <p>Umur: <input type="text" name="umur" ></p>
22   <p>Alamat: <input type="text" name="alamat"></p>
23   <p><input type="checkbox" name="mahasiswa">Mahasiswa</p>
24   <p><input type="checkbox" name="pelajar">Pelajar</p>
25 </form>
26 </body>
27 </html>
```



Gambar: Contoh penggunaan selector :disabled, :enabled dan :checked

Selector `input:disabled` akan mencari tag `<input>` dengan atribut `disabled`. Atribut `disabled` menandakan objek form sedang tidak aktif (tidak bisa diinput). Untuk mengaktifkannya secara interaktif, bisa menggunakan kode JavaScript.

Selector `input:enabled` akan mencari seluruh objek form dengan kondisi aktif. Walaupun kita tidak menuliskan atribut `enabled` di dalam tag `<input>`, namun web browser akan menganggap semua element form sebagai enabled, kecuali ditulis `disabled`.

Terakhir, `input:checked` akan cocok dengan objek form yang berstatus `checked`. Dalam contoh di atas, jika "Mahasiswa" atau "Pelajar" dipilih, kotak checkbox akan bergeser beberapa pixel.

Selain untuk checkbox, selector `:checked` juga bisa digunakan untuk objek form lain yang juga memiliki status `checked`, seperti `radio`.

5.15. Structural Pseudo Class Selectors

Structural pseudo class selectors dipakai untuk menemukan element HTML yang cukup kompleks dimana selector biasa tidak bisa digunakan. Dalam beberapa hal, kita juga bisa melakukan 'pencocokan pola' (*pattern matching*) untuk mencari element HTML.

Berbeda dengan *dynamic pseudo selector*, **structural pseudo class selectors** tetap mencari element HTML yang ada di dalam struktur DOM.

Kita akan bahas 8 *structural pseudo class selectors*, yakni:

- :first-child
- :last-child
- :first-of-type
- :last-of-type
- :nth-child()
- :nth-last-child()
- :nth-of-type()
- :nth-last-of-type()
- :not()

Sebagai contoh kode program, saya telah menyiapkan sebuah unordered list yang berisi 15 element:

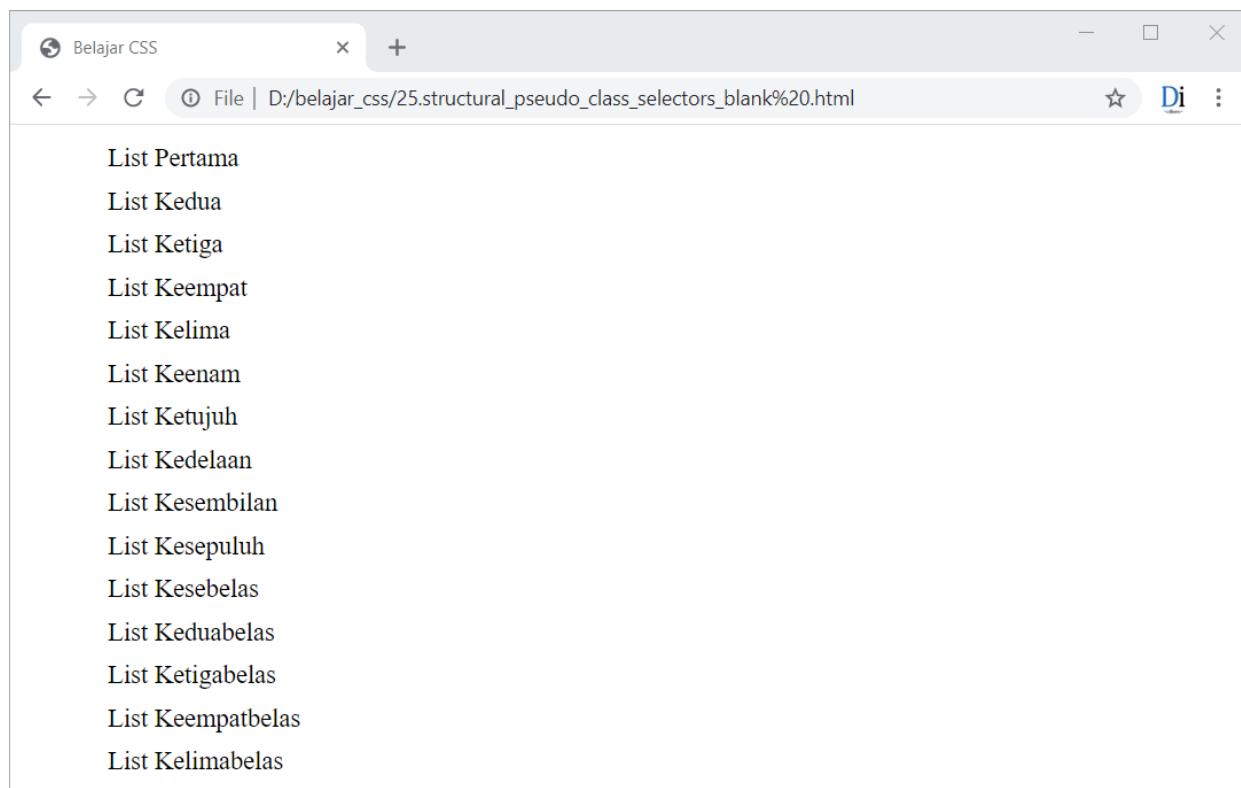
25.structural_pseudo_class_selectors_blank.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          ul {
8              list-style: none;
9              margin: 0;
10             padding: 0;
11         }
12         li {
13             width: 150px;
14             height: 30px;
15             padding: 0 10px;
16             font-size: 18px;
17             margin-left: 50px;
18             line-height: 30px;
19         }
20     </style>
21 </head>
22 <body>
23 <ul>
```

CSS Selector

```
24 <li>List Pertama</li>
25 <li>List Kedua</li>
26 <li>List Ketiga</li>
27 <li>List Keempat</li>
28 <li>List Kelima</li>
29 <li>List Keenam</li>
30 <li>List Ketujuh</li>
31 <li>List Kedelaan</li>
32 <li>List Kesembilan</li>
33 <li>List Kesepuluh</li>
34 <li>List Kesebelas</li>
35 <li>List Keduabelas</li>
36 <li>List Ketigabelas</li>
37 <li>List Keempatbelas</li>
38 <li>List Kelimabelas</li>
39 </ul>
40 </body>
41 </html>
```



Gambar: Tampilan list untuk contoh praktek structural pseudo class selector

Untuk saat ini boleh abaikan maksud dari property CSS yang ada pada selector `ul` dan `li`. Ini saya pakai agar tampilan list menjadi lebih rapi sehingga mudah untuk mengimplementasikan `pseudo class selector`. Kita akan bahas property CSS ini dengan lebih detail dalam bab-bab selanjutnya.

Selector :first-child dan :last-child

Sesuai dengan namanya, selector **:first-child** dan **:last-child** dipakai untuk mencari anak pertama (*first child*) dan anak terakhir (*last child*) dari struktur DOM. Umumnya selector ini digunakan bersama *element specific* selector untuk menentukan tag HTML yang akan dicari.

Misalkan, **li:first-child** berarti: *cari seluruh tag yang menjadi first-child dari parent element-nya*. Sedangkan **p:last-child** berarti: *cari seluruh tag <p> yang menjadi last-child dari parent element-nya (apapun parent element tersebut)*.

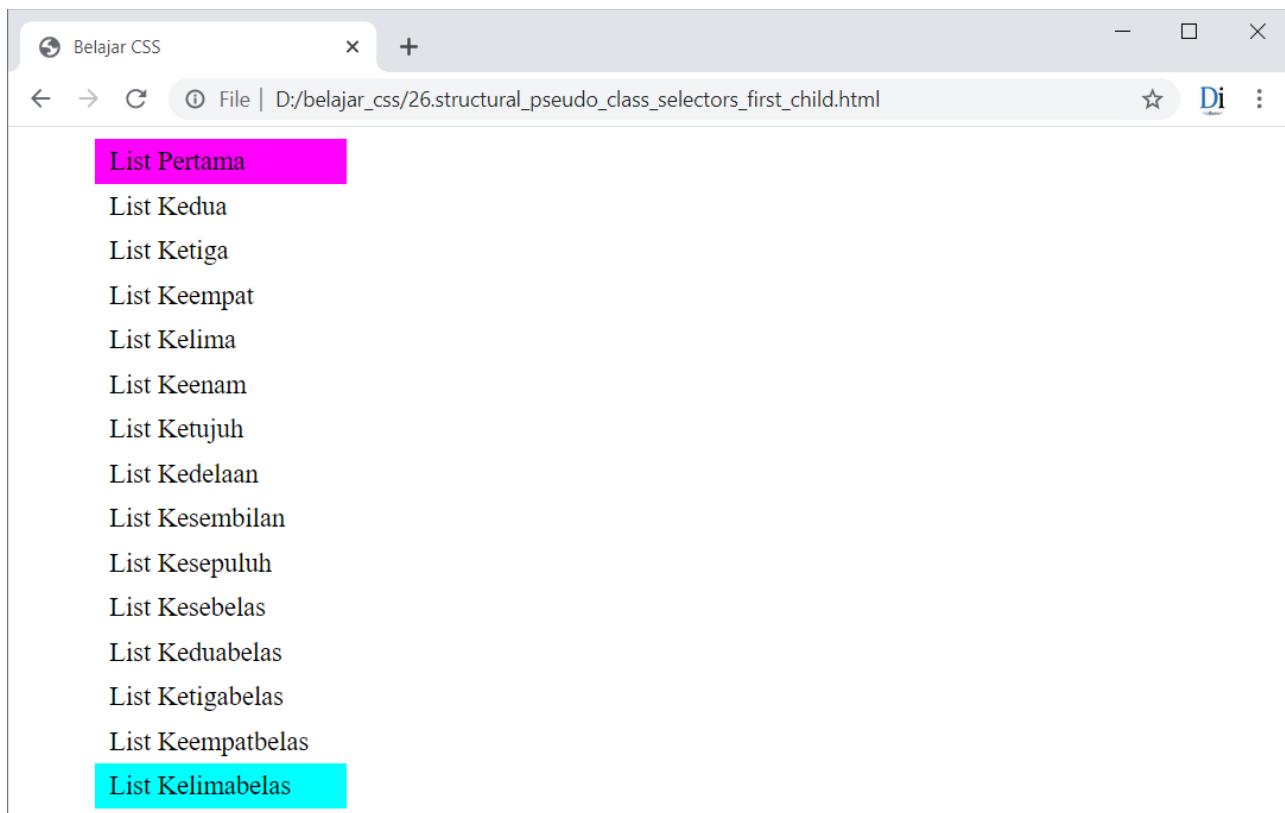
Perhatikan pula bahwa penulisan selector ini **tidak boleh ada spasi** antara nama element dengan *pseudo selector*.

Berikut contoh penggunaan selector **:first-child** dan **:last-child**:

26.structural_pseudo_class_selectors_first_child.html

```

1  <style>
2    ...
3    li:first-child{
4      background-color: magenta;
5    }
6    li:last-child{
7      background-color: aqua;
8  }
9  </style>
```



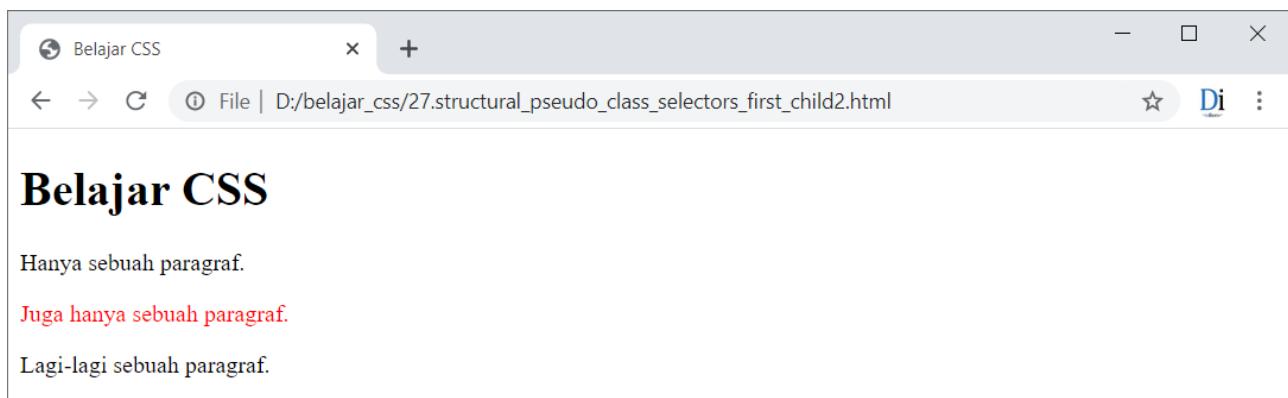
Gambar: Contoh penggunaan :first-child dan :last-child pseudo selector

Seperti yang terlihat, selector ini akan mengubah warna dari list pertama dan terakhir.

Agar lebih memahami makna dari selector `:first-child` dan `:last-child`, dapatkah anda menebak kenapa dalam contoh berikut paragraf kedua berwarna merah, namun paragraf pertama tidak?

27.structural_pseudo_class_selectors_first_child2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p:first-child{
8              color: red;
9          }
10     </style>
11 </head>
12 <body>
13 <div>
14     <h1>Belajar CSS</h1>
15     <p>Hanya sebuah paragraf.</p>
16 </div>
17 <div>
18     <p>Juga hanya sebuah paragraf.</p>
19     <p>Lagi-lagi sebuah paragraf.</p>
20 </div>
21 </body>
22 </html>
```



Gambar: Kenapa paragraf kedua berwarna dan paragraf pertama tidak?

Dalam kode HTML ini, paragraf pertama **bukanlah** sebagai *first child* dari tag `<div>`, oleh karena itu ia tidak berwarna merah. Sebaliknya pada tag `<div>` kedua, terdapat 2 buah paragraf, namun hanya paragraf pertama yang menyandang status anak pertama.

Selector `:first-of-type` dan `:last-of-type`

Selector `:first-of-type` dan `:last-of-type` hampir mirip seperti selector `:first-child` dan

`:last-child`. Bedanya, kali ini ada tambahan bahwa element HTML tersebut harus menjadi *first child* dari tipenya sendiri. Jadi walaupun sebuah tag tidak berada di posisi pertama, tapi selama itu adalah yang pertama dari jenisnya sendiri, tetap akan ditangkap oleh selector `:first-of-type`.

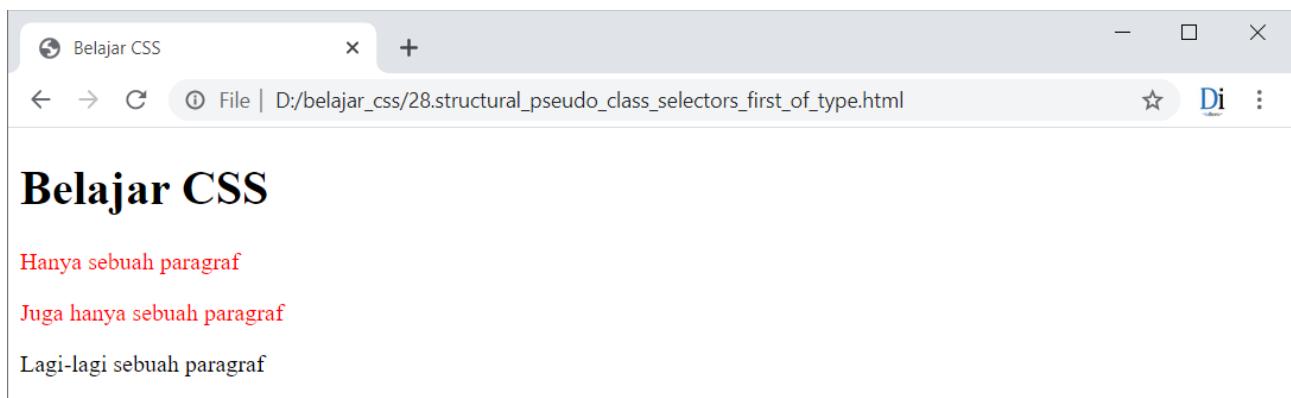
Jika kita menggunakan contoh list sebelumnya, tidak akan tampak perbedaan antara `li:first-child` dengan `li:first-of-type`, karena tag `` memang sebagai *first child* dan juga sebagai tipe pertama dari tag ``. Mari kita pakai contoh kedua tentang paragraf. Kali ini saya akan mengubahnya menjadi `p:first-of-type`:

28.structural_pseudo_class_selectors_first_of_type.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p:first-of-type{
8              color: red;
9          }
10     </style>
11 </head>
12 <body>
13 <div>
14     <h1>Belajar CSS</h1>
15     <p>Hanya sebuah paragraf</p>
16 </div>
17 <div>
18     <p>Juga hanya sebuah paragraf</p>
19     <p>Lagi-lagi sebuah paragraf</p>
20 </div>
21 </body>
22 </html>

```



Gambar: Contoh penggunaan `:first-of-type` pseudo selector

Hasilnya, paragraf pertama juga ikut menjadi merah. Ini terjadi karena tag `<p>` pertama adalah *first child* dari satu jenisnya, yakni tag `<p>`. Jadi selama paragraf "Hanya sebuah paragraf" tetap

menjadi tag `<p>` pertama di dalam tag `<div>`, maka akan cocok dengan selector `:first-of-type`, walaupun sebenarnya element pertama adalah tag `<h2>`.

Pseudo Selector `:nth-child()` dan `:nth-last-child()`

Tanda kurung di dalam selector `:nth-child()` dan `:nth-last-child()` adalah tempat bagi angka yang dipakai untuk mencari tag HTML berdasarkan urutannya atau pola tertentu. Ini mirip seperti *argument function* pada bahasa pemrograman.

Pola yang bisa diterapkan untuk kedua selector ini cukup beragam. Selain angka, kita bisa menulis *keyword* khusus seperti "**odd**" atau "**even**", serta pola matematis seperti penambahan.

Sebagai contoh, berikut penjelasan dari beberapa pola untuk selector `:nth-child()`:

- `li:nth-child(2)`: cari tag `` yang berada pada urutan kedua dalam sebuah parent element.
- `li:nth-child(9)`: cari tag `` yang berada pada urutan kesembilan dalam sebuah parent element.
- `li:nth-child(even)`: cari tag `` yang berada pada urutan genap (even) dalam sebuah parent element.
- `li:nth-child(odd)`: cari tag `` yang berada pada urutan ganjil (odd) dalam sebuah parent element.
- `li:nth-child(2n+1)`: cari tag `` yang berada pada urutan pertama, kemudian 2 tag setelahnya hingga akhir. Hasilnya sama seperti `li:nth-child(odd)`.
- `li:nth-child(2n+0)`: cari tag `` yang berada pada urutan 0, kemudian 2 tag setelahnya secara berulang hingga akhir. Hasilnya sama seperti `li:nth-child(even)`.
- `li:nth-child(2n+5)`: cari tag `` yang berada pada urutan kelima, kemudian 2 tag setelahnya hingga akhir.
- `li:nth-child(4n+3)`: cari tag `` yang berada pada urutan ketiga, kemudian 4 tag setelahnya secara berulang hingga akhir.
- `li:nth-child(4n-2)`: cari tag `` yang berada pada setiap perulangan ke 4, tapi mulai dari tag kedua (hasil dari $4-2=2$).
- `li:nth-child(5n-3)`: cari tag `` yang berada pada setiap perulangan ke 5, tapi mulai dari tag kedua (hasil dari $5-3=2$).

Seperti yang terlihat, terdapat banyak pola yang bisa kita pakai, beberapa diantaranya lumayan 'njelmet'.

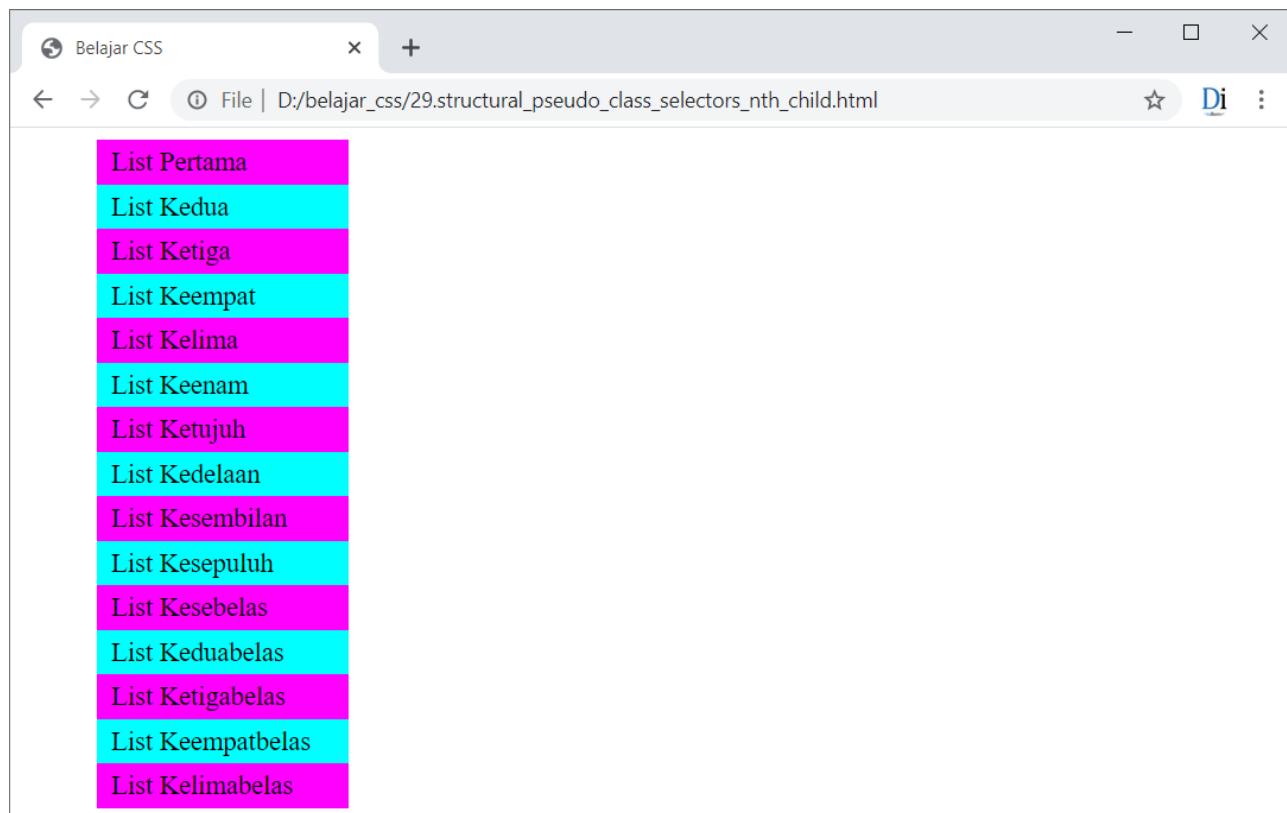
Sebagai sarana latihan, saya sarankan untuk mencoba seluruh selector ini ke dalam kode program daftar list kita sebelumnya. Sebagai contoh, saya akan pakai 2 selector, yakni `li:nth-`

CSS Selector

child(even) dan li:nth-child(odd):

29.structural_pseudo_class_selectors_nth_child.html

```
1 <style>
2 ...
3     li:nth-child(odd) {
4         background-color: magenta;
5     }
6     li:nth-child(even) {
7         background-color: aqua;
8     }
9 </style>
```



Gambar: Contoh penggunaan li:nth-child(odd) dan li:nth-child(even) pseudo selector

Terlihat warna list berselang seling antara *magenta* dan *aqua*. Efek stripes seperti ini sangat cocok digunakan dalam tabel, dimana kita bisa membedakan warna setiap baris agar mudah dibaca. Selector yang diperlukan untuk keperluan ini adalah `tr:nth-child(even)` atau `tr:nth-child(odd)`.

Pasangan selector `:nth-child()`, yakni `:nth-last-child()` sebenarnya hampir sama, tetapi perhitungan urutan kolom di lakukan dari bawah list.

Pseudo Selector `:nth-of-type()` dan `:nth-last-of-type()`

Jika anda sudah memahami beda selector `:first-child` dan `:first-of-type`, tentunya bisa

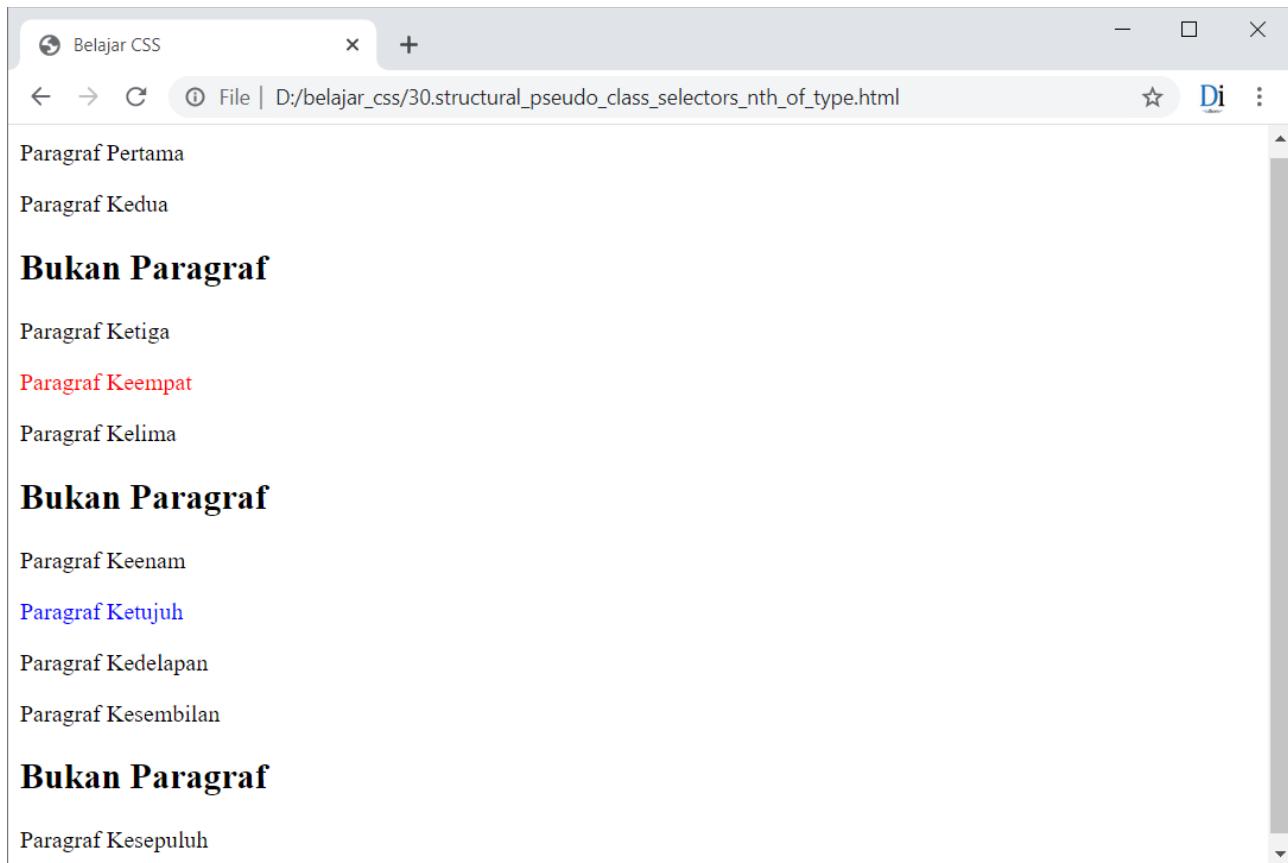
menebak arti dari selector :nth-of-type() dan :nth-last-of-type().

Selector :nth-of-type() dan :nth-last-of-type() adalah versi "tipe" dari :nth-child() dan :nth-last-child(). Jika dalam :nth-child() kita mencari pola yang merupakan anak (child), maka dalam :nth-of-type() anak tersebut juga harus dari tipe yang sama.

Agar dapat memahami perbedaannya, perhatikan kode program berikut ini:

30.structural_pseudo_class_selectors_nth_of_type.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p:nth-child(3){
8              color: red;
9          }
10         p:nth-child(5){
11             color: red;
12         }
13         p:nth-of-type(7){
14             color: blue;
15         }
16     </style>
17 </head>
18 <body>
19 <div>
20     <p>Paragraf Pertama</p>
21     <p>Paragraf Kedua</p>
22     <h2>Bukan Paragraf</h2>
23     <p>Paragraf Ketiga</p>
24     <p>Paragraf Keempat</p>
25     <p>Paragraf Kelima</p>
26     <h2>Bukan Paragraf</h2>
27     <p>Paragraf Keenam</p>
28     <p>Paragraf Ketujuh</p>
29     <p>Paragraf Kedelapan</p>
30     <p>Paragraf Kesembilan</p>
31     <h2>Bukan Paragraf</h2>
32     <p>Paragraf Kesepuluh</p>
33 </div>
34 </body>
35 </html>
```



Gambar: Perbedaan antara li:nth-child dengan :nth-of-type() pseudo selector

Kode HTML di atas terdiri dari 10 tag `<p>` yang diselingi oleh 3 tag `<h2>`. Di dalam bagian `<style>` saya membuat 3 selector CSS, namun seperti yang terlihat, hanya 2 paragraf yang ikut 'diwarnai'. Ini terjadi karena perbedaan antara `:nth-child()` dan `:nth-of-type()`.

Selector `p:nth-child(3)` akan mencari tag `<p>` yang menjadi anak ke-3. Perhatikan bahwa pada urutan ke-3 dari `<div>`, bukanlah sebuah paragraf, melainkan header `<h2>`. Oleh karena itu selector ini tidak cocok dengan struktur HTML dan akan diabaikan.

Selector `p:nth-child(5)` bisa memperjelas hal ini. Walaupun saya mencari tag `<p>` kelima, tapi yang akan diwarnai adalah "Paragraf Keempat". Hal ini disebabkan selector `:nth-child` ikut menghitung element anak lain, dimana anak ke-3 sudah ditempati oleh tag `<h2>`.

Bagaimana dengan selector `p:nth-of-type(7)`? Kali ini yang diwarnai adalah "Paragraf Ketujuh". Dengan demikian, kita bisa mengambil kesimpulan bahwa selector `:nth-of-type` akan mengabaikan element lain, dan hanya menghitung tag `<p>` yang ada.

Selector `:nth-last-of-type()` adalah varian lain dari `:nth-of-type`, dimana proses perhitungan akan dimulai dari bawah.

Pseudo Selector `:not()`

Pseudo selector `:not()` cukup unik, yakni berfungsi untuk mencari element HTML selain yang

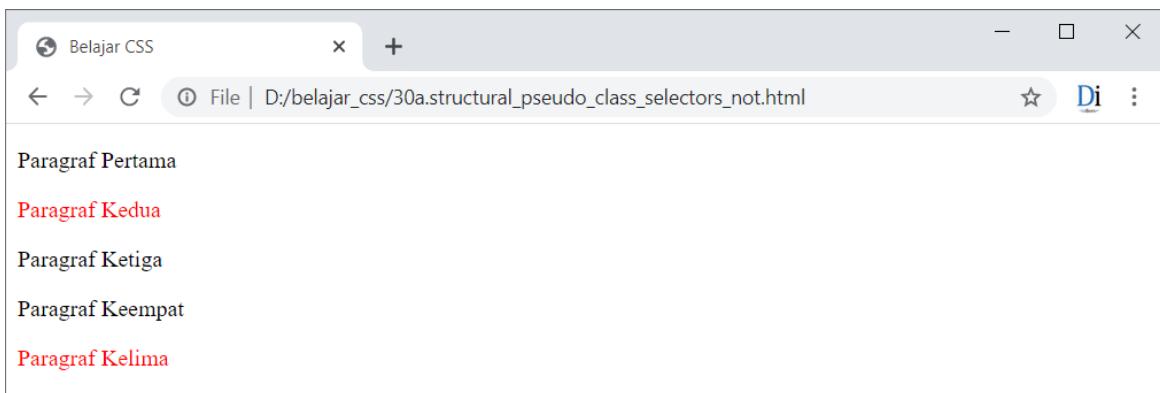
ditulis. Selector ini dikenal juga sebagai *negation pseudo-class selector*.

Sebagai contoh, selector `p:not(.tes)` akan cocok dengan semua element selain tag `p` yang memiliki class `.tes`. Berikut percobaannya:

30a.structural_pseudo_class_selectors_not.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p:not(.tes){
8              color: red;
9          }
10     </style>
11 </head>
12 <body>
13     <p class="tes">Paragraf Pertama</p>
14     <p>Paragraf Kedua</p>
15     <p class="tes">Paragraf Ketiga</p>
16     <p class="tes">Paragraf Keempat</p>
17     <p>Paragraf Kelima</p>
18 </body>
19 </html>
```



Gambar: Contoh penggunaan pseudo selector :not()

Paragraf kedua dan kelima berwarna merah karena keduanya tidak memiliki tambahan atribut `class="tes"` di dalam tag `<p>`.

Selector `:not()` memiliki beberapa efek yang mungkin tidak kita sangka. Misalnya jika selector `p:not(.tes)` diganti menjadi `:not(.tes)`, maka semua paragraf akan berwarna merah. Sebab selector `:not(.tes)` juga akan cocok dengan tag `<html>` dan `<body>` yang menjadi induk dari semua element HTML.

5.16. Pseudo Element Selector

Jika pembahasan mengenai *pseudo class selector* cukup membuat anda pusing, masih terdapat selector lain yang menggunakan nama 'pseudo'. Yakni **pseudo element selector**.

Walaupun sama-sama menggunakan nama *pseudo*, **pseudo class selector** dan **pseudo element selector** merupakan 2 jenis selector yang berbeda.

Pseudo element lebih fokus kepada menyeleksi konten atau isi element daripada struktur HTML. *Pseudo element* juga memiliki fitur *generated content* dimana kita bisa menambah sesuatu ke dalam HTML menggunakan CSS.

Pseudo element selector sebenarnya telah ada sejak CSS 2.1, namun pada CSS3 format penulisannya sedikit berbeda. Pada CSS 2.1 penulisan *pseudo element* mirip dengan *pseudo class*, yakni menggunakan tanda titik dua ':'. Pada spesifikasi CSS3, penulisan *pseudo element* menggunakan dua buah tanda titik dua '::'. Ini dilakukan untuk membedakan antara *pseudo class selector* dengan *pseudo element selector*.

Kali ini saya akan bahas 4 jenis *pseudo element selector*:

- ::first-line
- ::first-letter
- ::before
- ::after

Pseudo Element Selector ::first-line

Sesuai dengan namanya, selector **::first-line** dipakai untuk mengakses baris pertama dari sebuah element HTML. Langsung saja kita lihat contoh penggunaannya:

31.pseudo_element_first_line.html

```

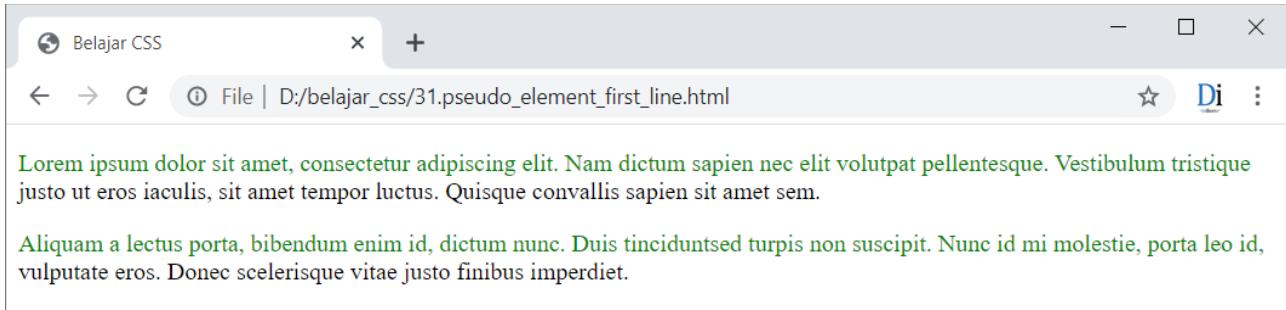
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p::first-line{
8              color:green;
9          }
10     </style>
11 </head>
12 <body>
13     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam dictum sapien
14         nec elit volutpat pellentesque. Vestibulum tristique justo ut eros iaculis,
15         sit amet tempor luctus. Quisque convallis sapien sit amet sem.</p>
16     <p>Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt sed
17         turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate eros.

```

```

18     Donec scelerisque vitae justo finibus imperdiet.</p>
19 </body>
20 </html>

```



Gambar: Contoh penggunaan ::first-line pseudo element selector

Terlihat pada baris pertama tag `<p>` akan berwarna hijau. Dan jika kita mengubah lebar jendela web browser, style pada baris pertama juga akan disesuaikan.

Pseudo Element Selector ::first-letter

Apabila selector `::first-line` dipakai untuk mengakses baris pertama sebuah element, maka selector `::first-letter` berguna untuk mengakses huruf/karakter pertama dari sebuah element.

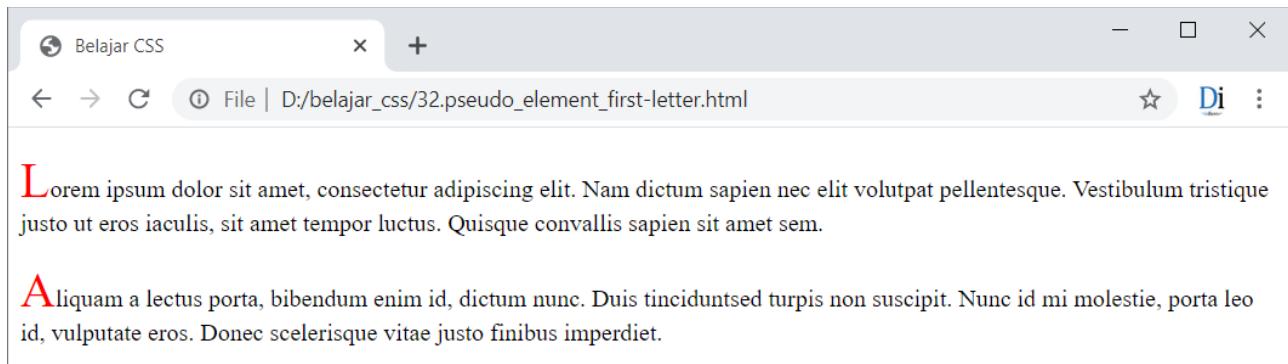
Dengan `::first-letter`, kita bisa membuat efek *drop cap*, dimana huruf pertama dibuat besar seperti contoh berikut:

32.pseudo_element_first-letter.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p::first-letter{
8              color:red;
9              font-size: 2em;
10         }
11     </style>
12 </head>
13 <body>
14     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam dictum sapien
15         nec elit volutpat pellentesque. Vestibulum tristique justo ut eros iaculis,
16         sit amet tempor luctus. Quisque convallis sapien sit amet sem.</p>
17     <p>Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
18         turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate eros.
19         Donec scelerisque vitae justo finibus imperdiet.</p>
20 </body>
21 </html>

```



Gambar: Contoh penggunaan ::first-letter pseudo element selector

Dalam contoh ini saya membuat huruf pertama setiap tag `<p>` berukuran besar dan berwarna merah. Tanpa selector `::first-letter`, ini cukup sulit dilakukan karena untuk menyeleksi huruf pertama harus dilakukan secara manual, misalnya menambah tag `` ke setiap huruf pertama awal paragraf.

Pseudo Element Selector `::before` dan `::after`

Pseudo element selector `::before` dan `::after` dikenal juga sebagai selector *generated content*. Karena dari kedua selector ini kita bisa menyisipkan konten baru ke dalam struktur HTML, apakah itu berupa teks, gambar, atau bahkan sebuah artikel yang terdiri dari beberapa paragraf.

Selector `::before` dipakai untuk menambah konten **sebelum element HTML**, sedangkan selector `::after` digunakan untuk menambahkan konten **setelah element HTML**. Untuk dapat menambahkan konten ini, selector `::before` dan `::after` memiliki property khusus, yakni: **content**.

Mari kita lihat cara penggunaan selector `::before` dengan kode HTML berikut ini:

33.pseudo_element_before.html

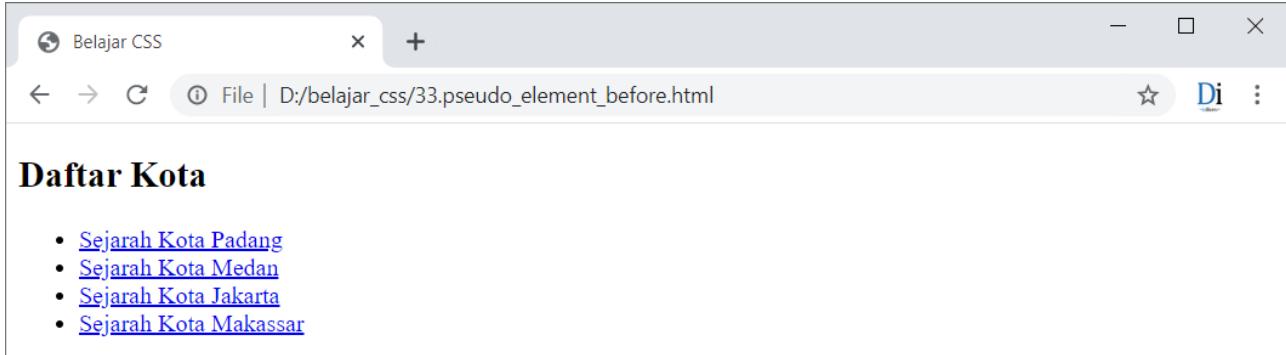
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          a::before{
8              content: "Sejarah ";
9          }
10     </style>
11 </head>
12 <body>
13     <h2>Daftar Kota</h2>
14     <ul>
15         <li><a href="padang.html">Kota Padang</a></li>
16         <li><a href="medan.html">Kota Medan</a></li>

```

CSS Selector

```
17     <li><a href="jakarta.html">Kota Jakarta</a></li>
18     <li><a href="makassar.html">Kota Makassar</a></li>
19   </ul>
20 </body>
21 </html>
```



Gambar: Contoh penggunaan ::before pseudo element selector

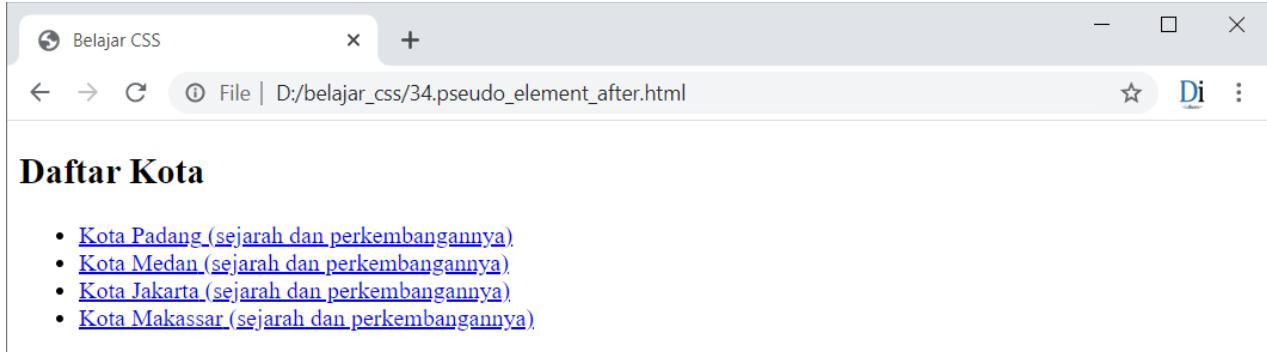
Jika melihat dari kode HTMLnya, setiap list hanya berisi nama-nama kota, namun ketika dijalankan di web browser, terdapat tambahan kata "Sejarah" di awal setiap list. Ini didapat dari penggunaan selector `::before` dan property `content: "Sejarah "`.

Selector `a::before` berarti: isi dari property content akan ditambahkan sebelum tag `<a>`.

Bagaimana dengan selector `::after`? Cara penggunaannya sangat mirip:

34.pseudo_element_after.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      a::after{
8        content: " (sejarah dan perkembangannya)";
9      }
10   </style>
11 </head>
12 <body>
13   <h2>Daftar Kota</h2>
14   <ul>
15     <li><a href="padang.html">Kota Padang</a></li>
16     <li><a href="medan.html">Kota Medan</a></li>
17     <li><a href="jakarta.html">Kota Jakarta</a></li>
18     <li><a href="makassar.html">Kota Makassar</a></li>
19   </ul>
20 </body>
21 </html>
```



Gambar: Contoh penggunaan ::after pseudo element selector

Kali ini saya menambah teks "(sejarah dan perkembangannya)" di akhir tag `<a>`.

Sebagai info tambahan, *generated content* ini berada di luar struktur DOM. Oleh karena itu konten yang dihasilkan kemungkinan besar akan susah diakses oleh mesin pencari seperti Google. Generated content sebaiknya hanya dipakai untuk mempercantik tampilan dan bukan sebagai konten utama.

CSS3 menyediakan beragam fitur lain untuk generated content, misalnya kita bisa menyisipkan gambar, mengambil nilai atribut hingga membuat nomor urut yang dihasilkan dari CSS.

Untuk mengambil nilai atribut, kita bisa menggunakan property **content: attr(nama-atribut)**. Misalnya untuk mengambil alamat link dari sebuah tag `<a>`, bisa menggunakan property `content:attr(href)` seperti contoh berikut:

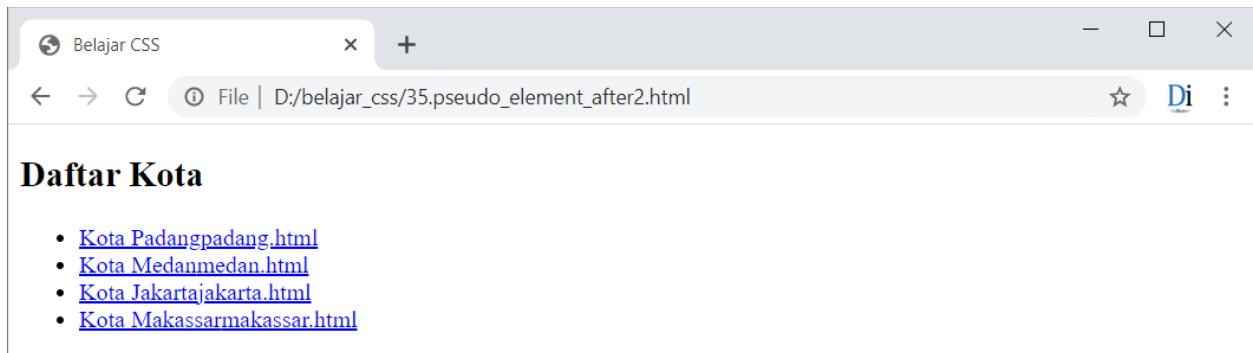
35.pseudo_element_after2.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          a::after{
8              content: attr(href);
9          }
10     </style>
11 </head>
12 <body>
13     <h2>Daftar Kota</h2>
14     <ul>
15         <li><a href="padang.html">Kota Padang</a><a href="padang.html" style="float:right">(sejarah dan perkembangannya)</a></li>
16         <li><a href="medan.html">Kota Medan</a><a href="medan.html" style="float:right">(sejarah dan perkembangannya)</a></li>
17         <li><a href="jakarta.html">Kota Jakarta</a><a href="jakarta.html" style="float:right">(sejarah dan perkembangannya)</a></li>
18         <li><a href="makassar.html">Kota Makassar</a><a href="makassar.html" style="float:right">(sejarah dan perkembangannya)</a></li>
19     </ul>
20 </body>
21 </html>

```

CSS Selector

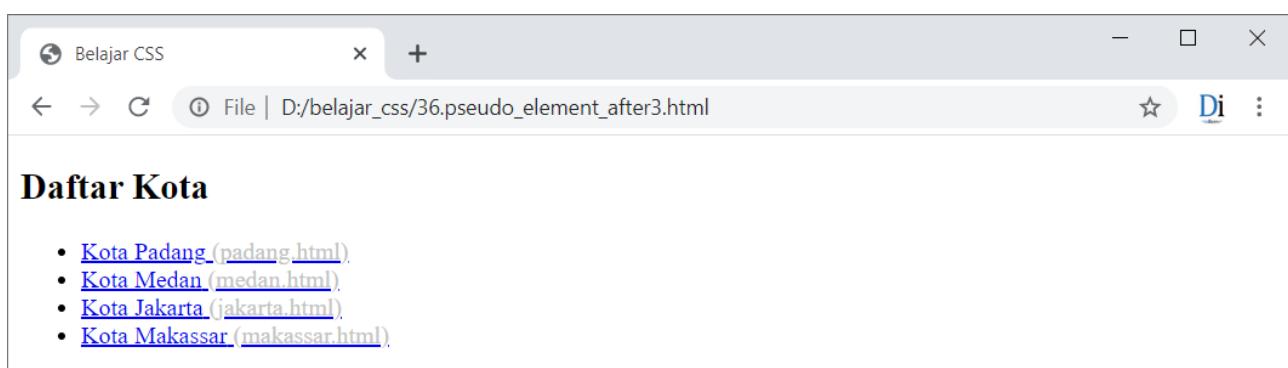


Gambar: Contoh penggunaan ::after dengan content: attr(href)

Setelah nama list, terdapat alamat link yang dihasilkan oleh property `content`. Agar tampilannya lebih rapi, saya akan memformat tulisan ini agar lebih baik lagi:

36.pseudo_element_after3.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          a::after{
8              content: " (" attr(href) ")";
9              color: silver;
10         }
11     </style>
12 </head>
13 <body>
14     <h2>Daftar Kota</h2>
15     <ul>
16         <li><a href="padang.html">Kota Padang</a></li>
17         <li><a href="medan.html">Kota Medan</a></li>
18         <li><a href="jakarta.html">Kota Jakarta</a></li>
19         <li><a href="makassar.html">Kota Makassar</a></li>
20     </ul>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan ::after dengan content: attr(href)

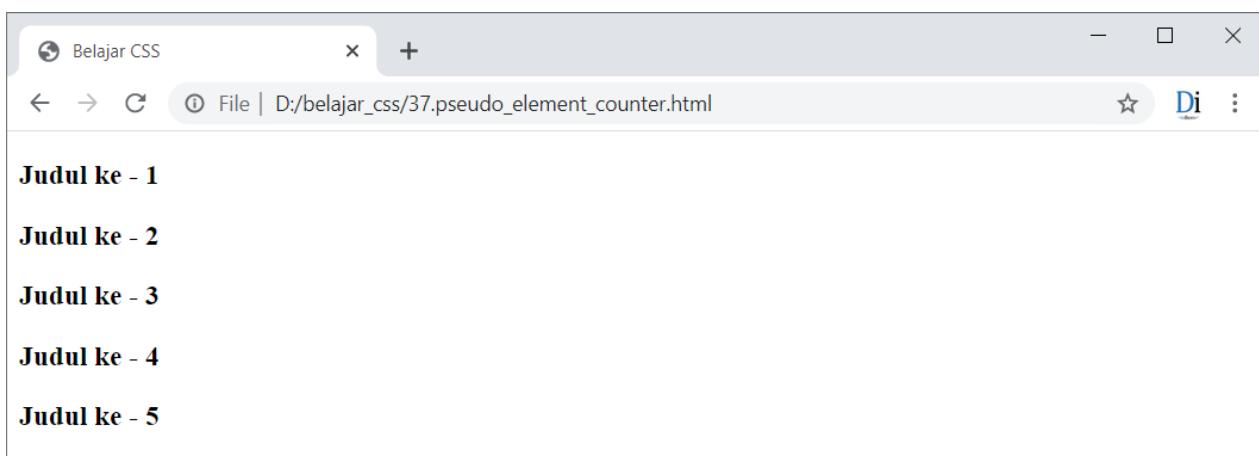
Kali ini setelah penulisan link terdapat alamat tujuan. Fitur seperti ini sangat berguna untuk halaman web yang akan di print.

Sebagai contoh terakhir, kita bisa menggunakan nilai property **counter** dan **counter-increment** untuk menghasilkan nomor urut dari sebuah element HTML. Berikut contoh penggunaannya:

37.pseudo_element_counter.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h3::after{
8              content: " ke - " counter(nourut);
9          }
10         h3 {
11             counter-increment: nourut;
12         }
13     </style>
14 </head>
15 <body>
16     <h3>Judul</h3>
17     <h3>Judul</h3>
18     <h3>Judul</h3>
19     <h3>Judul</h3>
20     <h3>Judul</h3>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan ::after dengan counter-increment

Saat halaman diakses, setiap tag `<h3>` akan langsung ditambahkan nomor urut. Padahal nomor ini tidak terdapat di dalam konten HTML. Silahkan tes tambah satu tag `<h3>` baru dibagian bawah halaman. Secara otomatis CSS akan menambahkan no urut selanjutnya. Ini dilakukan menggunakan *pseudo element selector* dari CSS3.

Dalam bab ini kita telah membahas cukup panjang tentang selector CSS. Pada prakteknya, selector yang banyak dipakai adalah *element selector*, *class selector*, dan *id selector* serta gabungan dari selector tersebut.

Walaupun demikian, cukup penting untuk mengetahui apa saja selector yang tersedia di dalam CSS. Sehingga jika menemui kasus yang cukup rumit, selector khusus seperti *pseudo class* atau *pseudo element* siap digunakan.

Dengan konsep modul dari CSS3, ke depannya akan terus hadir selector-selector baru yang bisa digunakan sebagai 'senjata' untuk mencari element HTML dengan lebih tertarget.

Berikutnya kita akan masuk ke materi tentang **cascading, inheritance dan specificity** di dalam CSS. Ketiganya membahas apa yang terjadi jika ada 2 atau lebih selector dan property CSS yang saling 'bentrok'.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

6. Cascade, Inheritance dan Specificity

Setelah menguasai berbagai selector yang ada di dalam CSS, kali ini kita akan pelajari apa yang terjadi ketika 1, 2 atau 10 selector saling berebut sebuah element HTML yang sama.

Konsep yang akan dibahas adalah: **Cascade**, **Inheritance** dan **Specificity**. Ketiganya merupakan aturan main yang dipakai CSS untuk menangani konflik antar selector.

6.1. "Cascade" dari Cascading Style Sheet

Seperti yang pernah kita bahas dalam bab 1 tentang pengertian CSS, huruf "**C**" dari **CSS** adalah singkatan **Cascading** (berasal dari kata *cascade*). Apa sebenarnya maksud dari cascade ini?

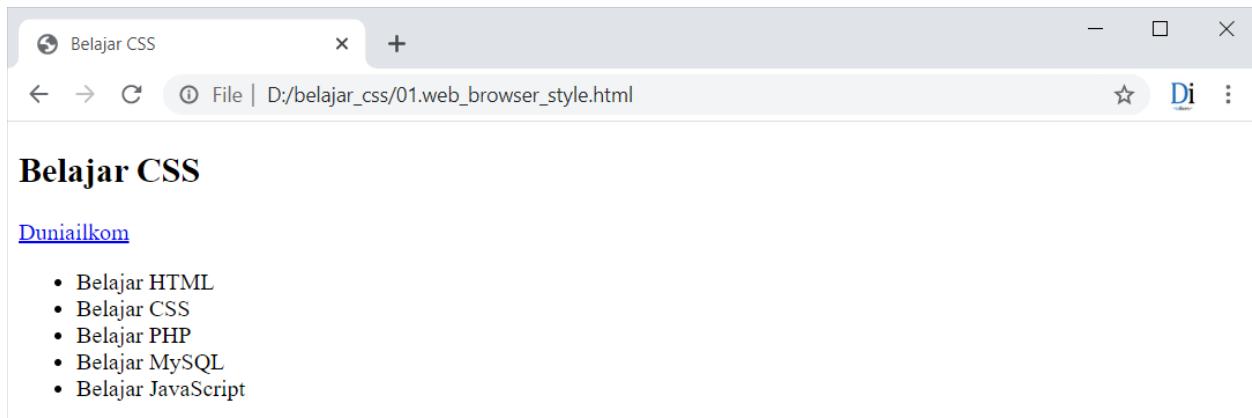
Cascade adalah sebuah aturan penurunan style di dalam CSS. Untuk dapat memahaminya, mari kembali bahas sekilas tentang cara menginput kode CSS ke dalam halaman HTML.

Sebagaimana yang telah kita pelajari, terdapat 3 cara memasukkan kode CSS ke dalam halaman HTML, yakni **inline style**, **internal style**, dan **external style CSS**. Selain ketiga metode ini sebenarnya terdapat 2 jenis style lain, yakni **web browser style** dan **user style**.

Web browser style adalah style CSS bawaan web browser. Sebagai contoh, perhatikan kode HTML berikut ini:

01.web_browser_style.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <h2>Belajar CSS</h2>
9      <a href="http://www.duniaIlkom.com">DuniaIlkom</a>
10     <ul>
11         <li>Belajar HTML</li>
12         <li>Belajar CSS</li>
13         <li>Belajar PHP</li>
14         <li>Belajar MySQL</li>
15         <li>Belajar JavaScript</li>
16     </ul>
17 </body>
18 </html>
```



Gambar: Tampilan default style web browser

Secara bawaan, tag `<h2>` tampil dengan ukuran font yang cukup besar, lalu tag `<a>` tampil dengan warna biru plus efek garis bawah, serta tag `` yang tampil dengan bulatan kecil di awal list. Inilah yang dimaksud dengan style bawaan web browser style (*web browser style*).

Setiap web browser membawa default style masing-masing. Jika di lihat sekilas, style ini terlihat sama antar satu web browser dengan web browser lain. Misalnya untuk link akan selalu berwarna biru, untuk tag header seperti `<h2>` akan tampil dengan ukuran font yang besar, dst.

Meskipun begitu, ada kemungkinan perbedaan style untuk setiap web browser. Hal ini bisa menjadi masalah bagi kita sebagai web desainer karena dapat membuat perbedaan tampilan antar web browser.

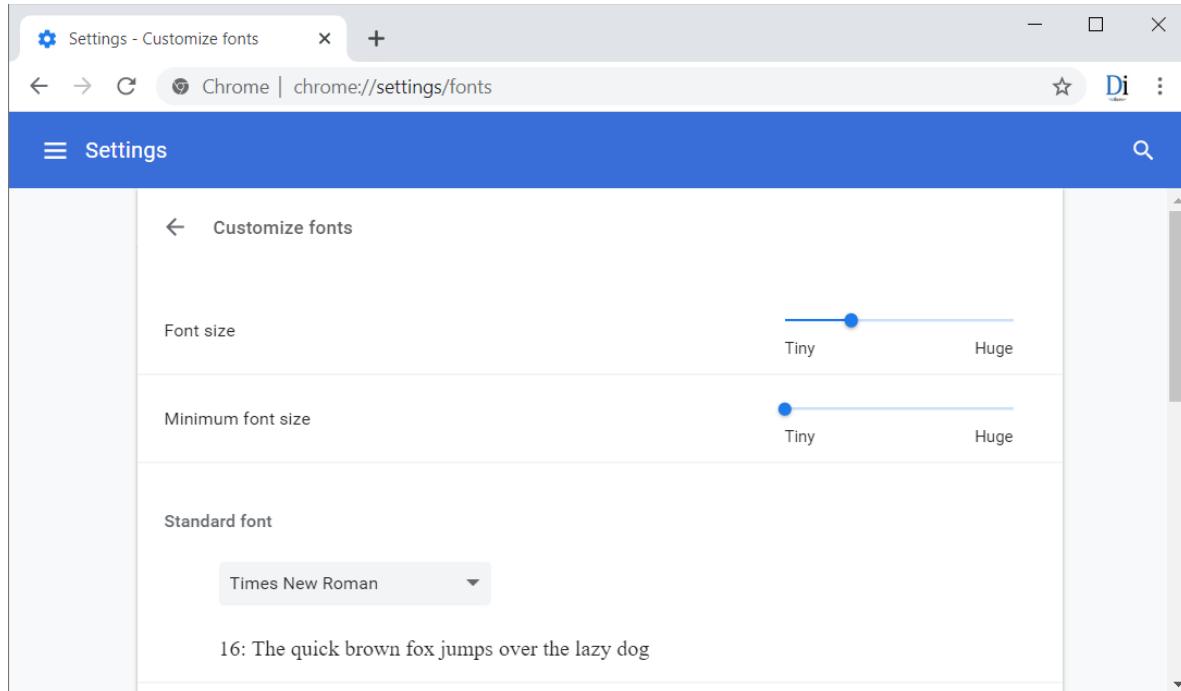
Untuk mengatasi masalah perbedaan web browser style, terdapat teknik yang bernama **CSS reset**. CSS reset dipakai untuk menghapus seluruh style bawaan web browser sehingga kita bisa mulai mendesain halaman HTML dari nol. Lebih lanjut tentang CSS reset akan dibahas pada bab tentang CSS box model.

Kembali ke pembahasan tentang jenis-jenis style, **user style** adalah style yang di buat oleh pengunjung website melalui web browser. Fitur ini memang relatif jarang dipakai, tapi disediakan di hampir setiap web browser.

Sebagai contoh, di web browser Google Chrome, user style bisa diakses dari menu **Settings -> Appearance -> Customize fonts**. Di sini kita bisa mengatur sendiri ukuran font, jenis font, warna teks dan warna background yang diinginkan.

User style umumnya dipakai oleh pengguna dengan keterbatasan dalam membaca konten website. Dengan user style, kita bisa memperbesar ukuran font atau mengganti warna font agar lebih kontras.

Cascade, Inheritance dan Specificity



Gambar: Cara mengubah pengaturan user style di Google Chrome

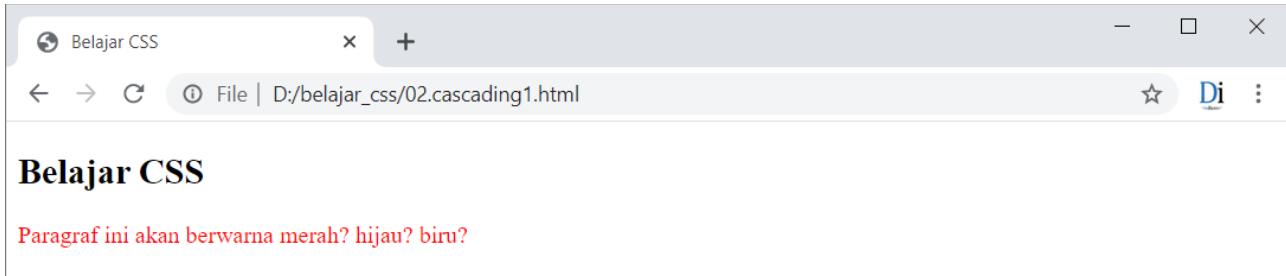
Konsep **cascading** dari CSS memiliki aturan mengenai style mana yang layak diprioritaskan. Berikut adalah urutan prioritas style pada CSS mulai dari yang paling kuat:

1. User Style
2. Inline Style
3. Internal Style dan External Style
4. Web browser style

Mari kita lihat implementasi dari efek cascading ini:

02.cascading1.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { color: green; }
8          p { color: blue; }
9          p { color: red; }
10     </style>
11 </head>
12 <body>
13     <h2>Belajar CSS</h2>
14     <p>Paragraf ini akan berwarna merah? hijau? biru?</p>
15 </body>
16 </html>
```



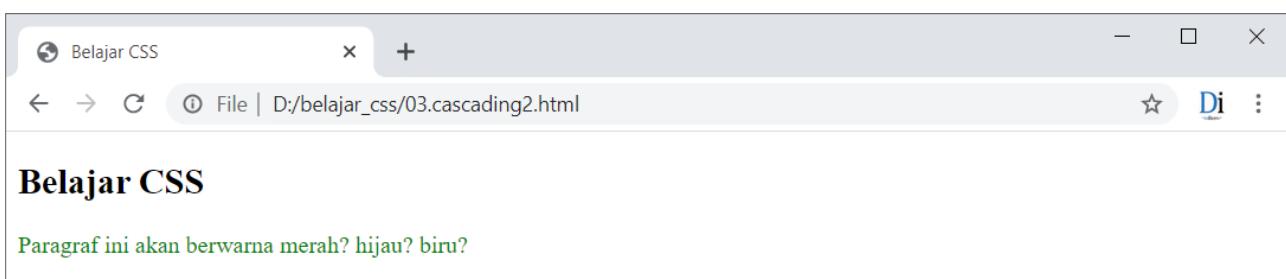
Gambar: Efek cascading *last rules apply*

Di baris 7-9 saya membuat 3 buah element type selector `p` yang sama-sama ingin mengubah warna teks dari tag `<p>`. Hasilnya, tag `<p>` akan berwarna merah (`red`). Ini terjadi karena jika ditemukan selector dan property yang sama pada style yang sejenis maka perintah terakhirlah yang akan digunakan, atau istilah keren-nya: *last rules apply*.

Sebagai contoh lain, saya akan menambah inline style CSS:

03.cascading2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { color: blue; }
8          p { color: red; }
9      </style>
10 </head>
11 <body>
12     <h2>Belajar CSS</h2>
13     <p style="color:green">Paragraf ini akan berwarna merah? hijau? biru?</p>
14 </body>
15 </html>
```



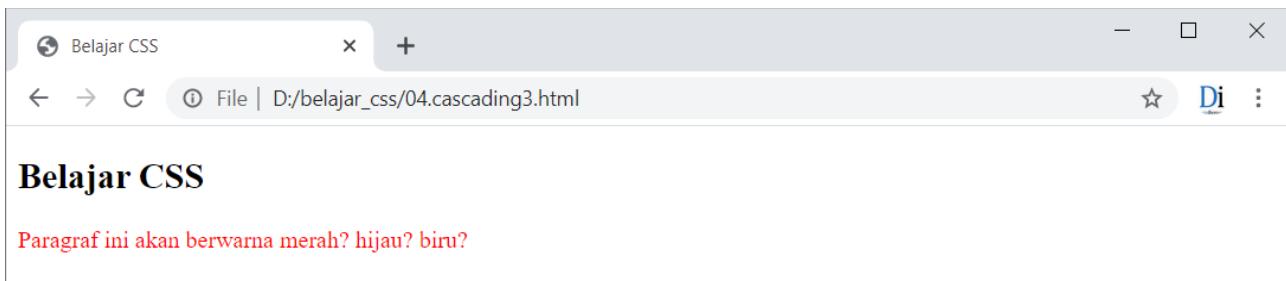
Gambar: Efek inline style CSS pada cascading

Sekarang paragraf akan berwarna hijau. Ini terjadi sesuai dengan urutan prioritas cascading, dimana inline style memiliki tingkat prioritas lebih tinggi dibandingkan embedded/internal style (perhatikan kembali daftar urutan prioritas kita sebelumnya).

Bagaimana dengan external style? Mari kita coba:

04.cascading3.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <link href="style_cascading.css" rel="stylesheet">
7   <style>
8     p { color: blue; }
9     p { color: red; }
10  </style>
11 </head>
12 <body>
13   <h2>Belajar CSS</h2>
14   <p>Paragraf ini akan berwarna merah? hijau? biru?</p>
15 </body>
16 </html>
```



Gambar: Efek internal style dan external style pada cascading

Kali ini saya memakai external style dan internal style untuk 'mewarnai' tag `<p>`. Dari urutan prioritas, external style dan internal style memiliki kekuatan prioritas yang sama, oleh karena itu aturan *last rules apply* akan berlaku.

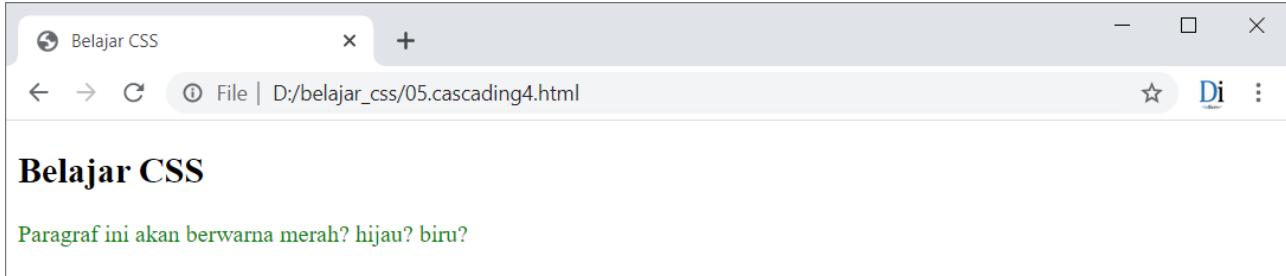
File external `style_cascading.css` hanya berisi 1 property, yakni `p {color:green}`, namun karena tag `<style>` (internal style) berada di bawah tag `<link>`, maka internal style-lah yang akan menang (paragraf jadi berwarna merah).

Bagaimana jika penulisannya kita balik?

05.cascading4.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p { color: blue; }
8     p { color: red; }
9   </style>
10  <link href="style_cascading.css" rel="stylesheet">
11 </head>
12 <body>
```

```
13 <h2>Belajar CSS</h2>
14 <p>Paragraf ini akan berwarna merah? hijau? biru?</p>
15 </body>
16 </html>
```



Gambar: Efek cascading last rules apply untuk internal dan external CSS

Sekarang paragraf akan berwarna hijau yang berasal dari external style CSS. Ini mengkonfirmasi bahwa internal style memiliki tingkat prioritas yang sama dengan external style, sehingga yang menang ditentukan dari aturan *last rules apply*.

Apa yang kita bahas memang kasus yang sangat sederhana. Seiring dengan banyaknya property yang ditambah, efek cascading akan berperan penting dalam menentukan style akhir dari sebuah element.

Dengan memahami konsep *cascading*, kita bisa merencanakan cara penulisan kode CSS. Apakah nantinya akan memakai external style, internal style, atau inline style.

Yang perlu diperhatikan adalah, inline style hampir selalu menang dan sangat susah untuk ditimpa oleh kode dari internal maupun external style CSS. Oleh karena itu sedapat mungkin pakai external style atau internal style CSS.

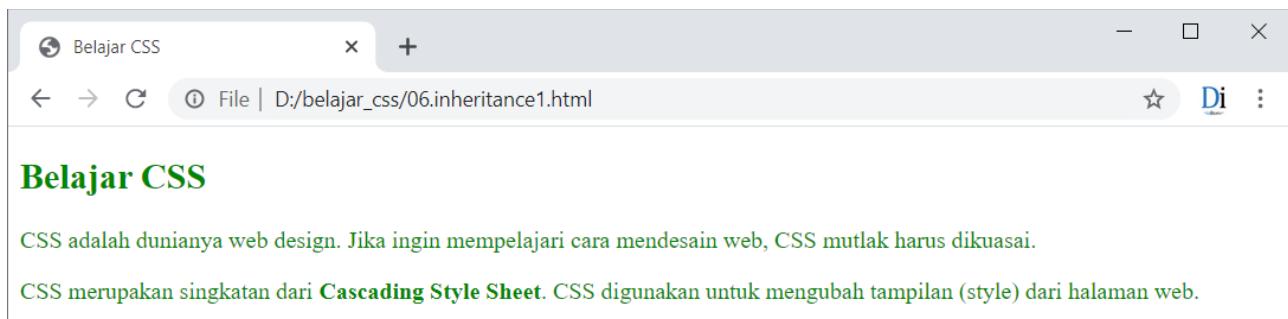
6.2. Inheritance Style CSS

Inheritance atau dalam Bahasa Indonesia diartikan sebagai penurunan/pewarisan adalah konsep dimana sebuah style bisa diturunkan dari parent (induk) kepada child element (anak). Berikut contoh kasusnya:

06.inheritance1.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     body { color: green; }
8   </style>
9 </head>
10 <body>
11   <h2>Belajar CSS</h2>
```

```
12 <p>CSS adalah dunianya web design. Jika ingin mempelajari cara mendesain web,  
13     CSS mutlak harus dikuasai.</p>  
14 <div>  
15     <p>CSS merupakan singkatan dari <strong>Cascading Style Sheet</strong>.  
16     CSS digunakan untuk mengubah tampilan (style) dari halaman web.</p>  
17 </div>  
18 </body>  
19 </html>
```



Gambar: Efek inheritance?

Kode HTML + CSS di atas hanya terdiri dari 1 selector, yakni body. Dalam selector ini saya menulis property `color` untuk mengubah warna teks dari tag `<body>`.

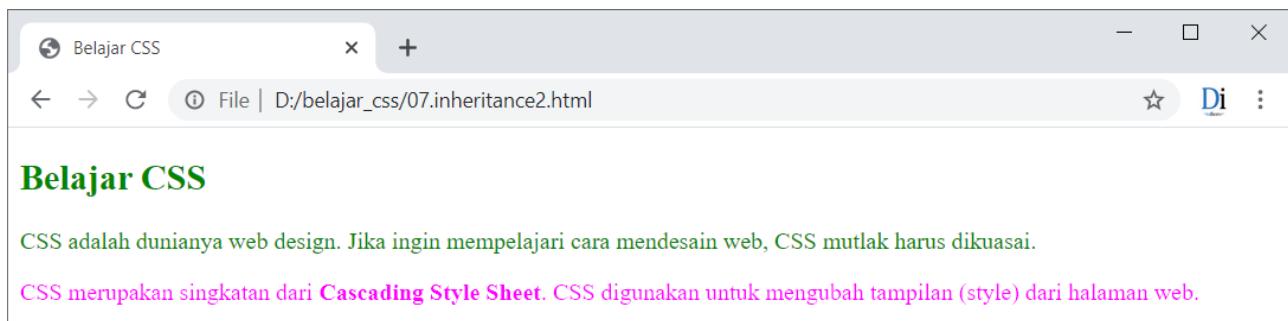
Perhatikan bahwa selector body seharusnya hanya mengubah warna teks untuk tag `<body>` saja, tetapi seluruh teks di dalam tag `<p>`, `<div>`, dan `` juga ikut berwarna hijau. Inilah yang dimaksud dengan penurunan (*inheritance*) style.

Karena tag `<p>`, `<div>` dan `` semuanya berada di dalam tag `<body>`, maka property `color: green` akan diturunkan dari tag `<body>` ke seluruh child element.

Efek *inheritance* bisa ditimpakan ketika kita membuat style yang lebih spesifik untuk child element seperti contoh berikut:

07.inheritance2.html

```
1 ...
2 <style>
3     body { color: green; }
4     div { color: magenta; }
5 </style>
```



Gambar: Menimpa efek inheritance dengan selector yang lebih spesifik

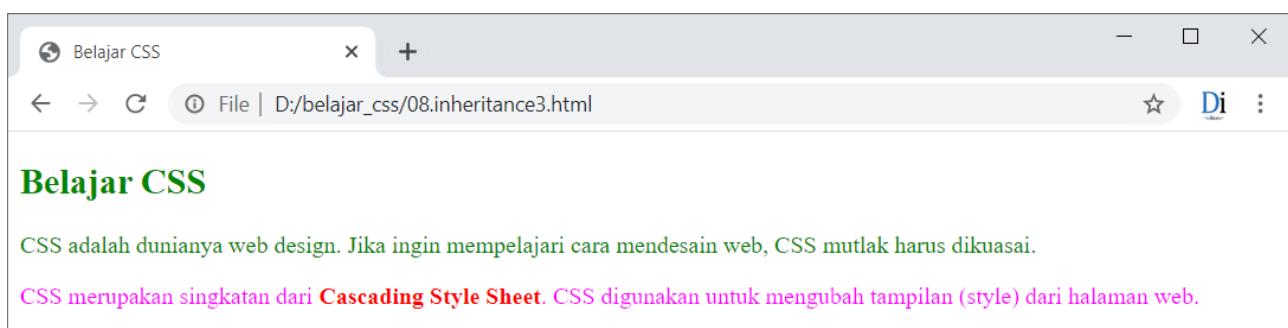
Kali ini saya menambah selector untuk tag `<div>` dengan property yang sama (`color`). Hasilnya, tag `<p>` yang berada di dalam tag `<div>` akan berwarna magenta. Di sini property `color` dari selector `body` ditimpa oleh selector `div` yang lebih spesifik.

Secara umum, proses 'timpa-menimpa' style ini akan dimenangkan oleh selector yang paling spesifik, yakni selector yang paling dekat dengan element target.

Sebagai contoh lain, dapatkah anda menebak apa warna tag `` dari kode berikut?

08.inheritance3.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          strong { color: red; }
8          div { color: magenta; }
9          body { color: green; }
10     </style>
11 </head>
12 <body>
13     <h2>Belajar CSS</h2>
14     <p>CSS adalah dunianya web design. Jika ingin mempelajari cara mendesain web,
15         CSS mutlak harus dikuasai.</p>
16     <div>
17         <p>CSS merupakan singkatan dari <strong>Cascading Style Sheet</strong>.
18             CSS digunakan untuk mengubah tampilan (style) dari halaman web.</p>
19     </div>
20 </body>
21 </html>
```



Gambar: Menimpa inheritance dengan selector yang lebih spesifik

Terlihat isi dari tag `` yakni teks "Cascading Style Sheet" akan berwarna merah.

Tapi tunggu dulu, bukankah style paling akhir yang menang (konsep *last rules apply*)? Selector `strong` ada di baris 7, tapi di baris 9 ada selector `body` yang juga akan mengubah warna teks.

Efek cascading yang kita pelajari sebelumnya hanya terjadi untuk selector yang sama. Pada contoh kali ini, selector yang dipakai sudah berbeda, sehingga yang terjadi adalah *inheritance*

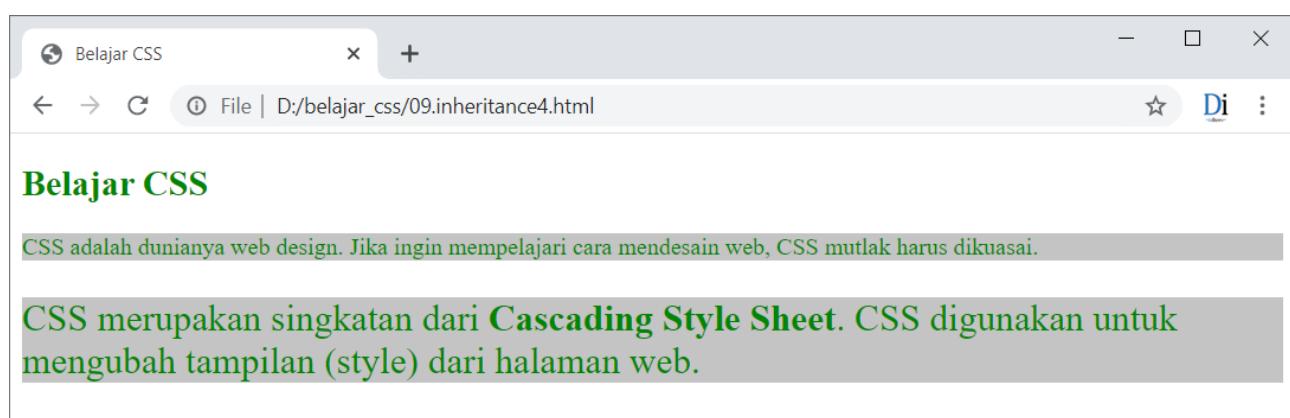
bukan cascading.

Memahami kedua perbedaan ini sangat penting agar kita bisa yakin kenapa sebuah style tidak berefek apa-apa, atau kenapa sebuah element malah 'terkena' style dari selector lain.

Ketika kita memakai selector yang berbeda dan property yang berbeda pula, efek inheritance akan menggabung hasil property. Berikut contoh yang dimaksud:

09.inheritance4.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{ color: green; }
8          div { font-size: 25px; }
9          p   { background-color: silver; }
10     </style>
11 </head>
12 <body>
13     <h2>Belajar CSS</h2>
14     <p>CSS adalah dunianya web design. Jika ingin mempelajari cara mendesain web,
15         CSS mutlak harus dikuasai.</p>
16     <div>
17         <p>CSS merupakan singkatan dari <strong>Cascading Style Sheet</strong>.
18         CSS digunakan untuk mengubah tampilan (style) dari halaman web.</p>
19     </div>
20 </body>
21 </html>
```



Gambar: Penggabungan style CSS dengan inheritance

Dari tampilan di atas, tag `<p>` yang berada di dalam tag `<div>` akan berwarna hijau, berukuran 25 pixel dan memiliki warna background silver. Dari manakah tag `<p>` mendapat style ini?

Seluruh style pada tag `<p>` berasal dari akumulasi style parent element-nya, yakni tag `<body>`, tag `<div>`, dan style dari element `<p>` itu sendiri. Inilah hasil dari efek *inheritance*.

Sebagai tambahan, perhatikan style pada tag ``. Selain memiliki warna dan ukuran yang

sama dengan tag `<p>`, tag `` juga tampil dengan huruf tebal. Style ini berasal dari *web browser style* dimana efeknya ikut bergabung dengan style yang digunakan.

Efek inheritance sebenarnya juga dipengaruhi oleh property yang dipakai. Kebanyakan property CSS diturunkan dari parent element ke child element, tapi tidak semua. Property `color` dan `font-size` adalah contoh property yang diturunkan (*inherit*), tapi property `border` tidak diturunkan.

Menggunakan contoh kode HTML sebelumnya, efek dari style berikut:

```
body{  
    color: green;  
    font-size: 25px;  
}
```

Akan sama hasilnya dengan:

```
body{  
    color: green;  
    font-size: 25px;  
}  
div{  
    color: green;  
    font-size: 25px;  
}  
p {  
    color: green;  
    font-size: 25px;  
}
```

Ini terjadi karena property `color` dan `font-size` termasuk property CSS yang diturunkan. Dengan memanfaatkan fitur ini, kita tidak perlu menulis style untuk seluruh element yang ada di dalam tag `<body>`.

Mari kita coba test dengan property `border` (style untuk membuat garis tepi/bingkai), efek border ini tidak termasuk ke dalam *inherit* property, sehingga style berikut:

```
body { border: 1px red solid; }
```

Tidak akan sama hasilnya dengan:

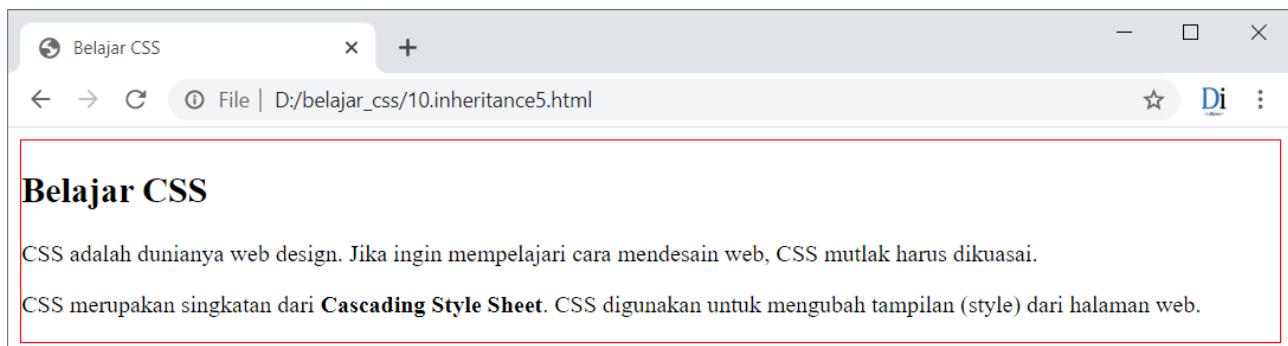
```
body { border: 1px red solid; }  
div { border: 1px red solid; }  
p { border: 1px red solid; }
```

Penyebabnya karena property `border` hanya 'melekat' ke selector yang ditulis (tidak diturunkan). Berikut contoh prakteknya:

10.inheritance5.html

```
1 <!DOCTYPE html>
```

```
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     body { border: 1px red solid; }
8   </style>
9 </head>
10 <body>
11   <h2>Belajar CSS</h2>
12   <p>CSS adalah dunianya web design. Jika ingin mempelajari cara mendesain web,
13     CSS mutlak harus dikuasai.</p>
14   <div>
15     <p>CSS merupakan singkatan dari <strong>Cascading Style Sheet</strong>.
16     CSS digunakan untuk mengubah tampilan (style) dari halaman web.</p>
17 </div>
18 </body>
19 </html>
```



Gambar: Property border tidak diturunkan (inherit)

Bingkai merah hanya terlihat di tag `<body>` saja, tidak diturunkan ke tag-tag lain

Nilai Property Inherit

Walaupun kita belum masuk ke materi tentang property, tapi saya ingin membahas satu nilai khusus, yakni: `inherit`. Nilai `inherit` bisa dipakai pada hampir seluruh property CSS. Berikut contoh penggunaannya:

```
p {
  color: inherit;
}
```

Kode di atas berarti: *ubah warna teks untuk tag `<p>` menjadi sama dengan nilai parent element-nya*. Dalam kebanyakan kasus, kita tidak perlu menulis nilai `inherit` ini karena menjadi nilai default dari mayoritas property CSS.

6.3. Specificity Selector

Konsep terakhir untuk menyelesaikan 'pertentangan antar selector' adalah **specificity**.

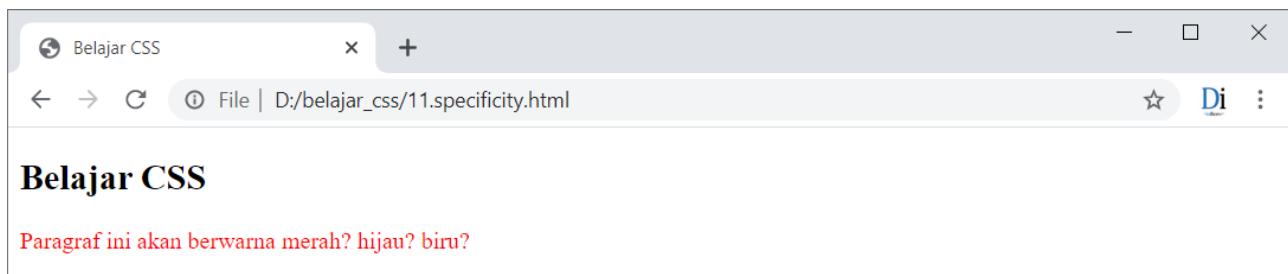
Specificity merupakan metode perhitungan nilai untuk menentukan selector mana yang akan didahulukan. Sesuai dengan namanya, semakin spesifik atau semakin detail sebuah selector, akan semakin tinggi nilainya.

Sebagai contoh, dapatkah anda menebak akan berwarna apa paragraf dalam dokumen ini?

11.specificity.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          #paragrafPertama { color: red; }
8          .paragraf { color: blue; }
9          p { color: green; }
10     </style>
11 </head>
12 <body>
13     <h2>Belajar CSS</h2>
14     <p class="paragraf" id="paragrafPertama">
15         Paragraf ini akan berwarna merah? hijau? biru?
16     </p>
17 </body>
18 </html>
```



Gambar: Efek dari specificity CSS

Teks yang ada di dalam tag `<p>` akan berwarna **merah**. Hal ini terjadi karena nilai *specificity* dari selector `#paragrafPertama` lebih kuat daripada kedua selector lain. Di dalam konsep specificity, *id* selector memiliki nilai yang lebih tinggi dari pada *class* selector maupun *element type* selector.

Sebagai aturan dasar, tabel berikut merangkum nilai specificity dari setiap selector CSS:

Jenis Selector	Contoh	Nilai Specificity
inline style	style="color:blue"	1000
id selector	#paragrafPertama	100
class selector	.paragraf	10
attribute selector	[href]	10
pseudo-class selector	:link	10
element selector	p	1
pseudo-element selector	::first-line	1
universal selector	*	0

Bagaimana dengan selector gabungan seperti *group selector*, *adjacent selectors* atau *child selectors*? Nilai specificity-nya dihitung dengan menambahkan setiap selector yang ada. Berikut contoh perhitungannya:

Selector	Jenis Selector	Nilai Specificity
body #paragrafPertama	element + id	1+100 = 101
p.paragraf	element + class	1+10 = 11
div > p	element + element	1+1 = 2
h2 + p	element + element	1+1 = 2
ul ol+li	element + element + element	1+1+1 = 3
ul ol li.red	element + element + element + class	1+1+1+10 = 13
body #myID .red	element + id + class	1+10+100 = 111

Dari tabel kita bisa lihat bahwa inline style dan id selector memiliki nilai specificity paling tinggi. Inline style agak jarang digunakan, tetapi id selector layak dibahas.

Id selector memiliki nilai specificity 100 yang membuatnya sangat sulit untuk 'dikalahkan' oleh selector lain (selain menggunakan id selector juga). Oleh karena inilah banyak web desainer menyarankan untuk menghindari penggunaan id selector untuk CSS.

Sebagai contoh, mari lihat kembali kode HTML + CSS dari praktek sebelumnya:

11.specificity.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     #paragrafPertama { color: red; }
8     .paragraf { color: blue; }
9     p { color: green; }
10  </style>
11 </head>
12 <body>
13   <h2>Belajar CSS</h2>
14   <p class="paragraf" id="paragrafPertama">
15     Paragraf ini akan berwarna merah? hijau? biru?
16   </p>
17 </body>
18 </html>
```

Bagaimana cara untuk menimpa efek style dari id selector `#paragrafPertama` tanpa harus menghapusnya? Walaupun saya menambahkan 3 selector gabungan seperti ini:

```
body p .paragraf { color: brown; }
```

Ini masih tidak kuat melawan nilai `#paragrafPertama`. Salah satu cara adalah dengan menggunakan id selector juga, misalnya:

```
body #paragrafPertama{ color: brown; }
```

Kali ini selector di atas akan memiliki nilai yang lebih kuat sehingga teks paragraf akan berwarna cokelat (*brown*).

Inilah masalah yang sering timbul dari penggunaan id selector. Jika anda belum banyak membuat kode CSS sebelumnya, ini tidak tampak sebagai masalah. Tapi seiring dengan seringnya latihan dengan CSS, penggunaan id selector bisa dipertimbangkan terlebih dahulu.

Namun juga bukan berarti kita harus menghindari penggunaan ID selector sama sekali. Untuk beberapa kasus, id selector dianggap lebih cocok dibandingkan selector lain. Misalnya sebagai identitas untuk setiap halaman seperti contoh berikut ini:

```
<!-- untuk halaman home: -->
<body id="homepage">
...
</body>

<!-- untuk halaman kategori: -->
<body id="category">
```

```
...
</body>

<!-- untuk halaman contact-us: -->
<body id="contactUs">
...
</body>
```

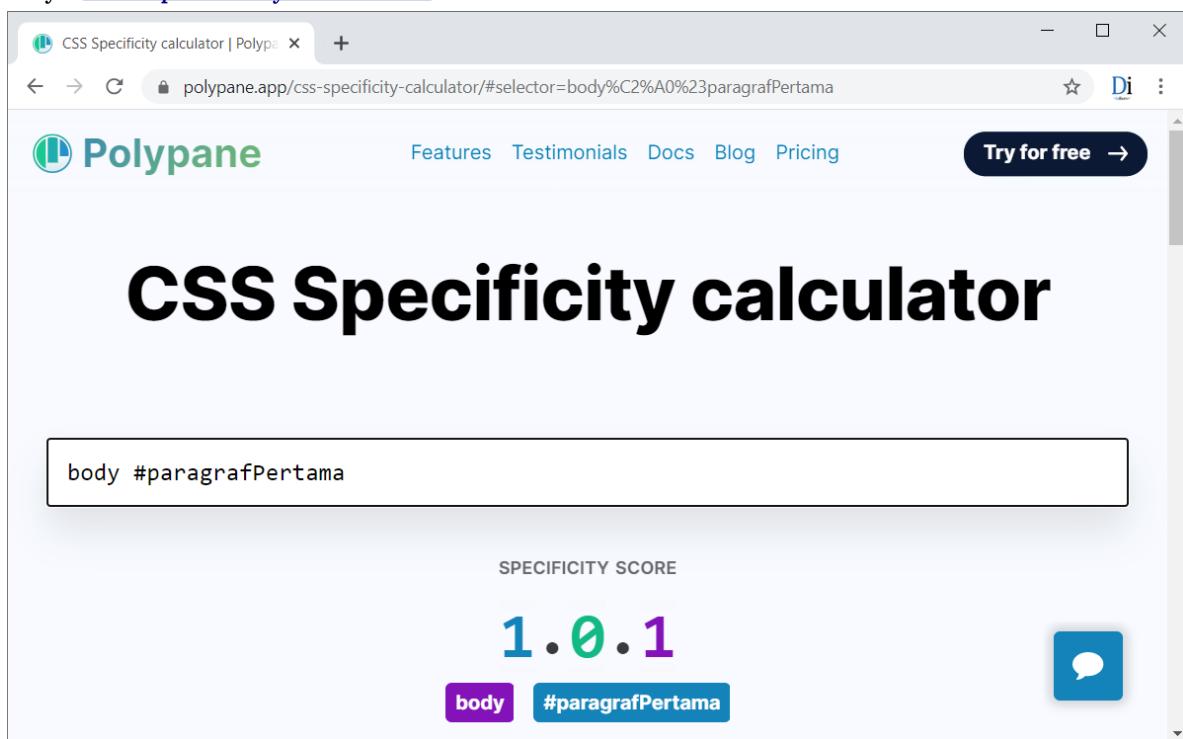
Dengan kode seperti ini, kita bisa buat selector yang spesifik untuk halaman tertentu saja.

Dibandingkan dengan cascade dan inheritance, specificity memang yang paling rumit. Dari pengalaman saya, specificity-lah yang sering menjadi akar masalah dari pertanyaan "kenapa style saya tidak berefek?". Memahami kekuatan dari masing-masing selector adalah kunci dari specificity.

Selain itu, merencanakan konsep kode CSS dari awal proses design merupakan kunci untuk memanfaatkan cascade, inheritance dan specificity.

Kita bisa mulai dengan sebuah konsep style global yang akan dipakai oleh sebagian besar tag HTML. Ini bisa dilakukan dengan element selector seperti `body`. Kemudian timpa style ketika diperlukan menggunakan class selector atau id selector. Dengan demikian, kode CSS menjadi terorganisasi dan efisien.

Specificity merupakan materi yang cukup membuat pusing namun juga sangat menarik untuk dibahas. Terdapat beberapa tools di internet yang bisa dipakai menghitung nilai specificity ini, misalnya [CSS Specificity calculator](#)¹⁴:



Gambar: CSS Specificity calculator

¹⁴ <https://polypane.app/css-specificity-calculator/>

6.4. Keyword Sakti: !important

Selain konsep cascade, inheritance dan specificity, masih ada 1 senjata yang bisa digunakan untuk menyelesaikan konflik antar selector CSS, yakni perintah **!important**.

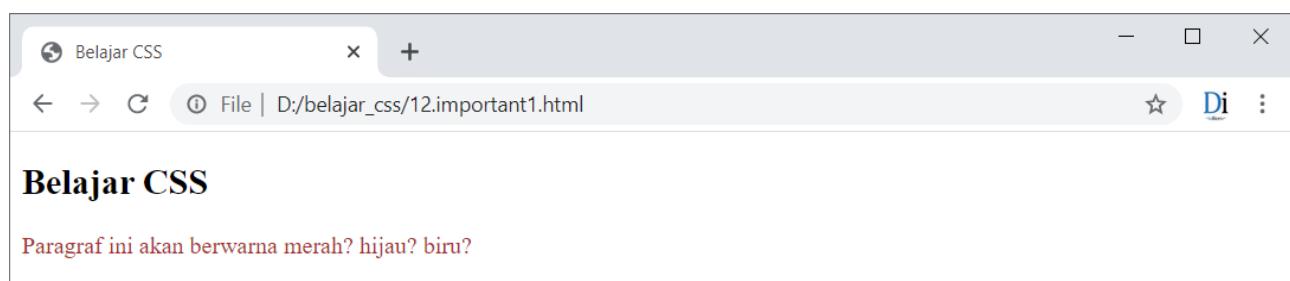
Seperi judul sub bab ini, **!important** adalah keyword sakti jika anda mengalami masalah dengan selector maupun property CSS yang tidak berjalan sebagaimana mestinya. Dengan menambah perintah **!important**, sebuah property CSS akan memiliki prioritas paling tinggi dan hanya bisa dikalahkan oleh perintah **!important** lain yang lebih spesifik.

Langsung saja kita lihat contoh penggunaannya. Perhatikan kode HTML + CSS berikut ini:

12.important1.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { color: green; }
8          body #paragrafPertama { color: brown; }
9      </style>
10 </head>
11 <body id="homepage">
12     <h2>Belajar CSS</h2>
13     <p class="paragraf" id="paragrafPertama">
14         Paragraf ini akan berwarna merah? hijau? biru?
15     </p>
16 </body>
17 </html>
```



Gambar: Efek specificity dari ID selector yang susah untuk ditimpak

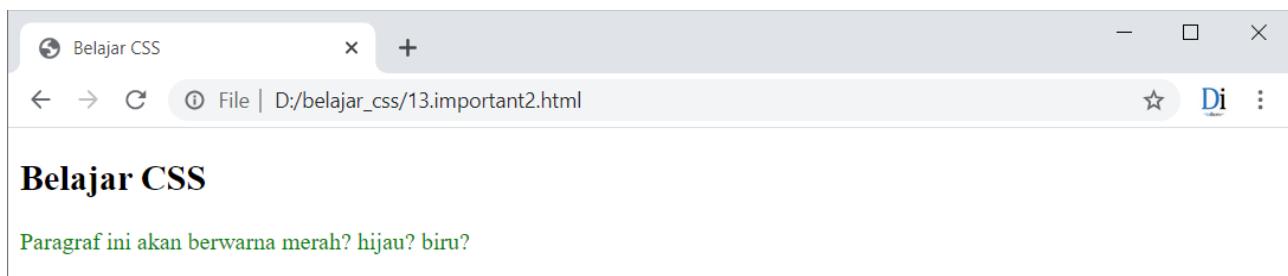
Teks paragraf akan berwarna cokelat karena selector `body #paragrafPertama` memiliki nilai specificity yang sangat tinggi. Namun kita bisa memenangkan element selector `p` dengan menambahkan perintah **important!**:

13.important2.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
```

```
4 <meta charset="UTF-8">
5 <title>Belajar CSS</title>
6 <style>
7   p { color: green !important; }
8   body #paragrafPertama { color: brown; }
9 </style>
10 </head>
11 <body id="homepage">
12   <h2>Belajar CSS</h2>
13   <p class="paragraf" id="paragrafPertama">
14     Paragraf ini akan berwarna merah? hijau? biru?
15   </p>
16 </body>
17 </html>
```



Gambar: important! is the winner...

Sekarang teks paragraf akan berwarna hijau karena terdapat penambahan perintah !important setelah property color:green di baris 7. Keyword !important akan membuat sebuah properti memiliki tingkat prioritas 10000*, yang bahkan akan mengalahkan nilai dari inline style!

Secara teknis, !important bekerja pada level *declaration* (property), bukan di dalam selector seperti specificity. Nilai prioritas 10000 ini cuma sekedar gambaran betapa tingginya efek prioritas dari !important.

Perhatikan bahwa perintah !important ditulis pada bagian property, bukan di selector. Penulisannya juga harus berada sebelum tanda titik koma penutup nilai property (dipisah dengan sebuah spasi).

Dibalik kekuatan yang besar, terdapat tanggung jawab yang besar pula. Dalam prakteknya, perintah !important sebaiknya hanya dipakai sebagai jalan terakhir. Alasannya sama seperti penggunaan id selector, yakni sekali perintah !important ditambah, sangat sulit untuk menimpanya. Perintah !important membuat konsep cascading, inheritance dan specificity menjadi 'tidak berdaya'.

Namun dalam beberapa kasus, !important juga menjadi jalan keluar yang paling praktis, terutama jika kita memodifikasi tampilan web yang sudah ada. Akan butuh waktu cukup lama untuk mempelajari seluruh struktur kode HTML dan CSS. Terlebih lagi jika kode itu tersebar ke dalam banyak file CSS. Dengan memakai perintah !important kita bisa pastikan bahwa

perintah CSS yang ditulis akan 'menimpa' style bawaan apapun.

Sebagai penutup bahasan tentang perintah `!important`, bagaimanakah cara menimpa style yang telah dikenakan perintah ini? Jawabannya adalah dengan perintah `!important` juga:

14.important3.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { color: green !important; }
8          body #paragrafPertama { color: brown !important; }
9      </style>
10 </head>
11 <body id="homepage">
12     <h2>Belajar CSS</h2>
13     <p style="color:red" id="paragrafPertama" >
14         Paragraf ini akan berwarna merah? hijau? biru?
15     </p>
16 </body>
17 </html>
```



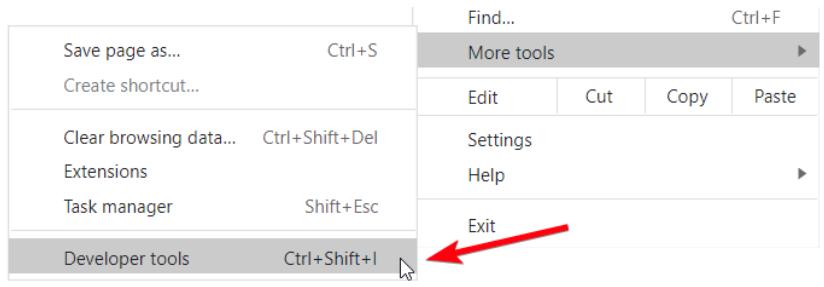
Gambar: Menimpa efek `!important` dengan `!important`

Kali ini teks akan berwarna cokelat yang berasal dari selector `body #paragrafPertama`. Di dalam tag `<p>` saya juga menambahkan inline style untuk membuktikan bahwa perintah `!important` lebih kuat daripada inline style.

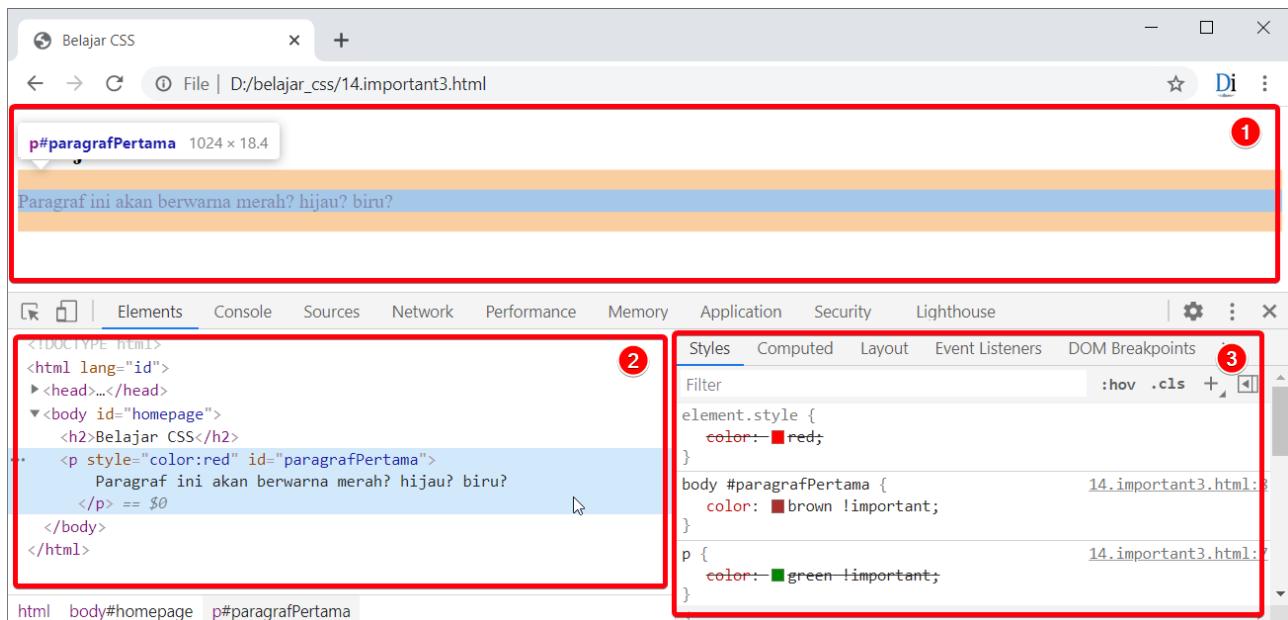
6.5. Periksa dengan Developer Tools

Masalah paling umum ketika merancang kode CSS adalah style yang ditulis tidak berefek ke element HTML. Jika web sudah cukup besar, kemungkinan bentrok antara selector dan antar property akan semakin banyak terjadi.

Developer tools bawaan web browser bisa dipakai untuk membantu kita mencari sumber masalah. Di Google Chrome, menu ini bisa diakses dari **More tools -> Developer tools**, atau dengan menekan kombinasi tombol **Crtl + Shift + I**:



Gambar: Menu developer tools di Google Chrome

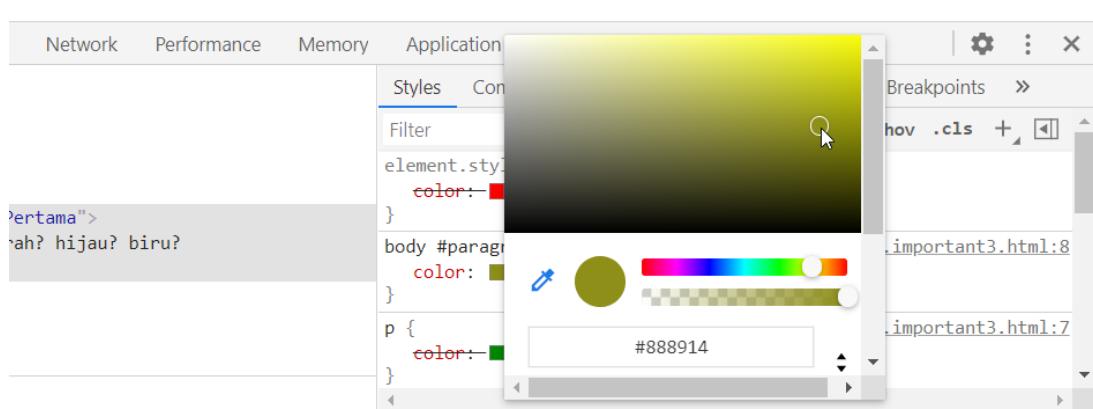


Gambar: Tampilan developer tools

Developer tools tampil dalam bentuk jendela tambahan di bagian bawah halaman. Kita bisa menelusuri kode HTML dari jendela kiri bawah (2). Hasil tampilan dari kode yang saat ini aktif akan terlihat di jendela utama (1), serta kode CSS yang dipakai oleh element tersebut di bagian kanan bawah (3).

Bagian kode CSS inilah yang akan menjadi fokus utama. Jika sebuah property CSS tampak tercoret, artinya property itu sudah tertimpa oleh property lain. Dalam contoh di atas, property color yang aktif adalah `color: brown !important` milik selector `body #paragrafPertama`, sedangkan property color dari `element.style` (inline CSS) dan milik selector `p` tampak tercoret.

Developer tools juga interaktif, kita bisa menghapus, menambah dan mengedit property CSS secara langsung. Silahkan klik kotak warna di samping property color, akan muncul color box untuk memilih warna-warna lain.



Gambar: Mengganti nilai property color dari developer tools

Dengan developer tools, proses pencarian kesalahan (debugging) menjadi lebih mudah.

6.6. Merencanakan Kode CSS

Dengan pengetahuan seputar *cascade*, *inheritance*, dan *specificity*, kita sudah memiliki dasar yang cukup untuk merencanakan kode CSS. Berikut adalah beberapa tips untuk membuat kode CSS yang lebih efisien:

Kurangi penggunaan inline style

Hampir tidak ada alasan yang bagus untuk menggunakan inline style. Selain memiliki tingkat prioritas yang tertinggi (yang membuatnya susah ditimpak), inline style juga sangat susah di deteksi. Belum lagi jika website tersebut sudah besar dan terdiri dari ribuan halaman.

Jika memungkinkan, kurangi penggunaan internal / embedded style

Walaupun masih bisa ditoleransi, penggunaan internal style/embedded style juga bisa menjadi masalah tersendiri. Internal style berada di level halaman sehingga kode CSS ini juga akan susah di kelola. Kita harus ingat halaman mana saja yang memiliki internal style.

Rencanakan struktur HTML yang akan dibangun

Menggunakan struktur HTML yang konsisten di setiap halaman akan memudahkan penulisan selector. Jika kita memakai tag `<header>` untuk bagian judul artikel, gunakan tag tersebut pada seluruh halaman website. Dengan demikian 1 selector `header` pada external style CSS bisa mengakses seluruh tag `<header>` pada setiap halaman.

Pertimbangkan jika ingin menggunakan ID Selector

Sama seperti inline style, ID selector juga memiliki tingkat prioritas yang tinggi sehingga membuatnya susah untuk ditimpak.

Batasi penggunaan group selector maksimal 3 level

Apabila anda mendapatkan penulisan selector seperti `body p+h2 #paragrafPertama`,

pertimbangkan untuk mencari cara lain. Selector yang panjang seperti ini kurang efisien dan menambah kompleks kode CSS. Jika memungkinkan, batasi penggunaan selector hanya 3 atau 2 level saja.

Manfaatkan fitur inheritance

Usahakan buat semacam "global style" untuk mengatur ukuran font dan warna teks yang akan dipakai oleh mayoritas element. Global style ini nantinya bisa ditulis dalam selector `body`.

Setiap child element dari tag `<body>` akan menurunkan ukuran font dan warna teks yang sama. Apabila child element butuh style lain, kita bisa timpa dengan selector yang lebih spesifik. Ini memanfaatkan konsep *inheritance* dari CSS.

Dalam bab ini kita telah membahas salah satu konsep terpenting di dalam CSS, yakni Cascade, Inheritance dan Specificity. Penguasaan tentang ketiga materi ini sangat diperlukan untuk membuat kode CSS yang efektif, fleksibel dan mudah di kelola.

Selanjutnya, kita akan masuk ke materi tentang property CSS, yang dimulai oleh property untuk manipulasi teks (*typography*).

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

7. CSS Typography

Typography adalah sebuah seni mengatur huruf/teks untuk membuatnya lebih mudah dibaca dan menarik secara visual.

Teks merupakan konten utama dari hampir semua website. Walaupun kita membuat web yang fokus kepada multimedia (gambar dan video), namun tetap memerlukan teks untuk bagian keterangan/deskripsi. Menu navigasi sebuah website juga hampir semuanya terdiri dari teks.

Dalam bab ini kita akan fokus membahas cara penggunaan property CSS yang berkaitan dengan typography atau teks. Selain itu, saya juga akan dibahas konsep dasar property CSS seperti nilai length (pixel, em, %, dll), nilai color(#RBG, HSL) serta CSS3 vendor prefix.

Kecuali dinyatakan lain, hampir semua property typography di dalam CSS akan diturunkan ke child element, atau dikenal sebagai inherit property.

7.1. Property font-family

Pembahasan mengenai property CSS kita mulai dengan **font-family**. Property ini digunakan untuk mengatur jenis font. Nilai dari property font-family adalah nama font yang ingin digunakan. Berikut contoh penulisannya:

```
p {  
    font-family: Arial;  
}
```

Kode CSS di atas akan men-set jenis font Arial pada semua paragraf. Akan tetapi, dari manakah font ini diambil?

Property font-family akan mencari font di dalam komputer client, yakni di dalam komputer pengunjung web browser kita, bukan di dalam komputer server tempat web berada.

Ini berarti kode CSS di atas hanya akan tampil dengan font Arial apabila di dalam komputer pengunjung terdapat font bernama "Arial". Jika tidak ditemukan, web browser akan menentukan sendiri font yang akan dipakai (pengaturan default web browser tersebut).

Bagi pengguna Windows, seluruh file font yang ada di komputer kcontita bisa dilihat dari Control Panel -> Appearance and Personalization -> Fonts.

Untuk mengantisipasi tidak tersedianya sebuah font, kita bisa menulis alternatif nama font pengganti seperti contoh berikut:

```
p {  
    font-family: Arial, Helvetica, sans-serif;  
}
```

Kode di atas akan menginstruksikan web browser untuk menampilkan semua tag <p> dalam font Arial. Apabila tidak ditemukan, gunakan font Helvetica. Apabila Helvetica juga tidak ditemukan, gunakan font generic dari tipe sans-serif.

CSS tidak membatasi banyak font alternatif yang bisa kita tulis, bisa 1, 3, bahkan 10 font:

```
body {  
    font-family: Arial, Helvetica, Verdana, Geneva, Impact, Charcoal, sans-serif;  
}
```

Di sini saya memberikan 7 alternatif font yang bisa dipilih web browser, mulai dari Arial sebagai prioritas utama, kemudian Helvetica (jika Arial tidak tersedia), lalu Verdana (jika Arial dan Helvetica tidak ditemukan), dst. Walaupun kita bisa menyiapkan banyak font alternatif, umumnya hanya perlu menulis 3 jenis font saja (2 font dan 1 font generic).

Perhatikan untuk setiap nama font dipisah dengan tanda koma ", ". Khusus untuk nama font yang terdiri dari beberapa kata atau mengandung spasi, harus ditulis dalam tanda kutip seperti "Times New Roman":

```
p {  
    font-family: "Times New Roman", Georgia, serif;  
}
```

Property `font-family` termasuk ke dalam properti yang diturunkan ke child element, atau dikenal sebagai *inherit property*. Oleh karena itu jika kita ingin memakai satu jenis font untuk seluruh halaman, tinggal menulisnya di dalam selector body:

```
body {  
    font-family: "Times New Roman", Georgia, serif;  
}
```

Kode ini akan membuat seluruh element HTML yang berada di dalam tag <body> menggunakan font "Times New Roman".

Mengenal Generic Font

Jika diperhatikan, setelah menulis nama font, pada pilihan terakhir saya menulis kata `serif` atau `sans-serif`. Ini sebenarnya bukanlah nama font, tetapi sebuah jenis font. Di dalam CSS ini dikelola dengan sebutan **generic font**.

Dalam CSS terdapat 5 jenis generic font:

- serif
- sans-serif
- monospace
- cursive
- fantasy

Generic Font: serif

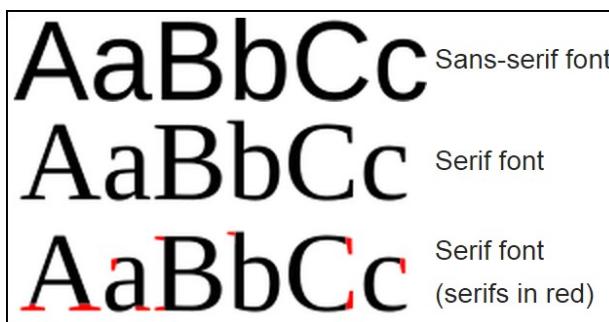
Dalam ilmu *typography*, **serif** adalah istilah untuk menyebut "kaki" atau garis tambahan pada ujung setiap huruf. Tujuannya agar memudahkan mata untuk beralih dari satu huruf ke huruf lain, terutama di media cetak seperti buku atau majalah.

Beberapa contoh font yang termasuk kategori serif adalah Times New Roman, Georgia, Lucida Bright, Lucida Fax, Palatino, Palatino Linotype dan Palladio.

Generic Font: sans-serif

Sans-serif adalah sebutan untuk kelompok font yang tidak memiliki kaki atau garis tambahan. Kata *sans* berasal dari bahasa Perancis yang berarti 'tanpa', sehingga sans-serif adalah 'tanpa-serif'. Font dengan jenis sans-serif umum dipakai sebagai tipe font utama dalam website karena dianggap lebih mudah dibaca, khususnya di media elektronik.

Contoh font sans-serif adalah Arial, Verdana, Trebuchet MS, Helvetica, dan Calibri.



Gambar: Perbedaan font serif dan sans-serif (sumber: wikipedia.org)

Generic Font: monospace

Font **monospace** adalah jenis font dengan lebar setiap karakternya sama panjang. Sebagai contoh, di dalam font reguler huruf 'i' akan mengambil tempat lebih sedikit daripada huruf 'w'. Namun di dalam font monospace, kedua huruf ini menggunakan ruang yang sama besar.

Font jenis monospace cocok dipakai untuk tulisan teknis yang butuh ketelitian. Hampir semua teks editor programming menggunakan font *monospace*. Contoh font jenis ini adalah Courier, Courier New, dan Andale Mono.



Gambar: Tampilan font monospace: Courier (sumber: wikipedia.org)

Generic Font: cursive

Font **cursive** adalah jenis font yang meniru tulisan tangan atau kaligrafi. Font jenis ini cukup beragam dengan berbagai variasi huruf. Biasanya font *cursive* dipakai untuk bagian judul, sub judul, atau teks lain yang di prioritaskan.

Contoh dari font *cursive* adalah "Comic Sans", "Brush Script MT", "Lucida Calligraphy" dan "Lucida Handwriting".

Generic Font: fantasy

Font dengan jenis **fantasy** adalah font yang bersifat visual dengan karakter font khusus seperti font disney, matrix, dll. Sama seperti *cursive*, font jenis ini relatif jarang sebagai text utama, biasanya digunakan untuk bagian teks yang butuh perhatian lebih seperti judul atau logo dari sebuah website.

Contoh font *fantasy* adalah Papyrus, Herculanum, Party LET, Curlz MT, dan Harrington.

Kelima generic font ini bisa menjadi patokan jenis font yang ingin di pakai. Sebagai contoh, jika kita memilih font Arial sebagai font utama, maka bisa menambah sans-serif sebagai generic font. Sehingga apabila di komputer client tidak terdapat font Arial, web browser akan mencari font lain dengan jenis sans-serif (yang sejenis dengan Arial).

Agar berfungsi sebagaimana mestinya, font generic harus ditempatkan di urutan terakhir **font-family**. Penulisan font generic juga harus ditulis dengan huruf kecil dan tanpa tanda kutip.

Berikut contoh kode CSS dengan berbagai variasi font:

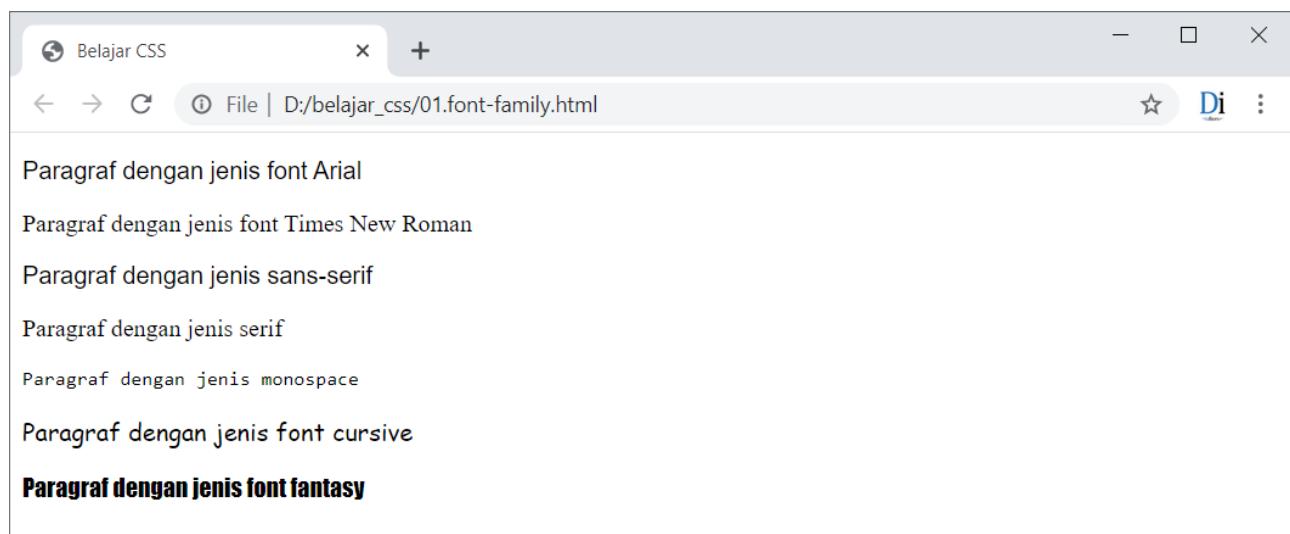
01.font-family.html

```
18 <!DOCTYPE html>
19 <html lang="id">
20 <head>
21   <meta charset="UTF-8">
```

```

22 <title>Belajar CSS</title>
23 <style>
24   p.arial      { font-family: Arial, Helvetica, sans-serif; }
25   p.times      { font-family: "Times New Roman", Georgia, serif; }
26   p.sans-serif { font-family: sans-serif; }
27   p.serif      { font-family: serif; }
28   p.monospace  { font-family: monospace; }
29   p.cursive    { font-family: cursive; }
30   p.fantasy    { font-family: fantasy; }
31 </style>
32 </head>
33 <body>
34   <p class="arial">Paragraf dengan jenis font Arial</p>
35   <p class="times">Paragraf dengan jenis font Times New Roman</p>
36   <p class="sans-serif">Paragraf dengan jenis sans-serif</p>
37   <p class="serif">Paragraf dengan jenis serif</p>
38   <p class="monospace">Paragraf dengan jenis monospace</p>
39   <p class="cursive">Paragraf dengan jenis font cursive</p>
40   <p class="fantasy">Paragraf dengan jenis font fantasy</p>
41 </body>
42 </html>

```



Gambar: Tampilan berbagai jenis font serta generic font CSS

Web Safe Font

Dari pembahasan sebelumnya telah dijelaskan bahwa property `font-family` akan mencari font di komputer lokal, yakni dari komputer pengunjung website kita.

Oleh karena itu, pilihan ini terbatas kepada font umum yang ada di kebanyakan komputer. Font-font ini dikenal juga dengan istilah **web safe font**, yakni font yang relatif aman digunakan karena biasanya tersedia di berbagai jenis sistem operasi.

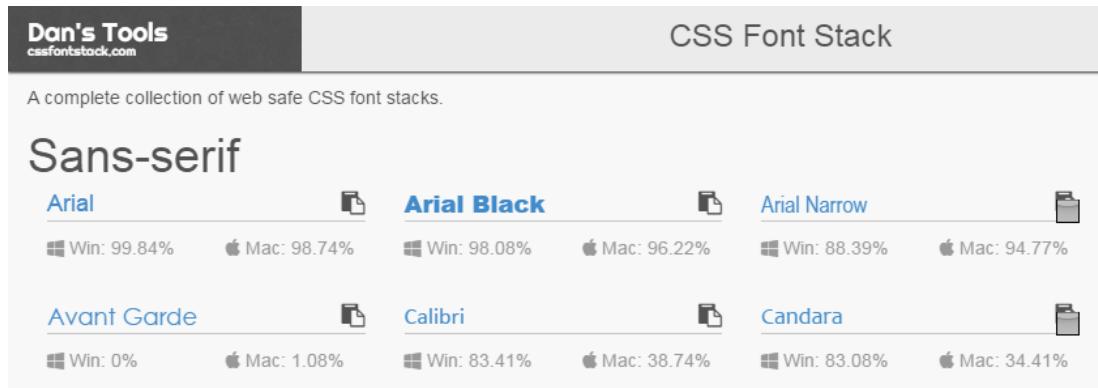
Berikut beberapa nilai `font-family` yang masuk dalam kategori web safe font:

- Georgia, serif
- "Palatino Linotype", "Book Antiqua", Palatino, serif

- "Times New Roman", Times, serif
- Arial, Helvetica, sans-serif
- "Lucida Sans Unicode", "Lucida Grande", sans-serif
- Tahoma, Geneva, sans-serif
- "Courier New", Courier, monospace
- "Lucida Console", Monaco, monospace

Font-font diatas hanya sebagai contoh, anda bebas ingin mengatur urutan sesuai dengan desain yang dirancang.

Untuk daftar yang lebih detail mengenai web safe font, bisa mengunjungi cssfontstack.com¹⁵. Di situs ini setiap font punya persentase kemungkinan tersedia. Sebagai contoh, font Arial memiliki peluang tersedia 99.84% di sistem operasi Windows dan 98.74% di sistem operasi Mac. Dengan kata lain font Arial hampir pasti ada di seluruh komputer Windows dan Mac.



Gambar: Tampilan situs www.cssfontstack.com

7.2. Property font-size

CSS menyediakan beragam satuan untuk mengatur ukuran font, yang semuanya di-set melalui property `font-size`. Berikut contoh penggunaannya:

```
p {  
    font-size: 14px;  
}
```

Kode CSS ini menginstruksikan web browser untuk mencari seluruh tag `<p>`, kemudian set ukuran font menjadi 14 pixel. Selain menggunakan satuan pixel, CSS menyediakan berbagai satuan lain yang bisa dipakai untuk mengatur ukuran font.

Jika kita tidak mendefinisikan ukuran font, sebuah teks akan memakai ukuran default web browser. Biasanya ukuran teks default ini sebesar 16 pixel.

¹⁵ [https://www.cssfontstack.com](http://www.cssfontstack.com)

Satuan Length CSS

Property `font-size` dan berbagai property lain di dalam CSS butuh nilai dalam satuan **length**, yakni satuan 'panjang' seperti pixel, cm, mm, em, atau persen.

Secara garis besar, nilai satuan *length* dikategorikan ke dalam 2 kelompok: **nilai absolut** dan **nilai relatif**.

Nilai Absolut

Nilai absolut (*absolute-size*) adalah satuan length yang nilainya tetap dan tidak terpengaruh oleh element di sekelilingnya. Contoh dari absolute-size adalah pixel (px), centimeter (cm), milimeter (mm), point (pt), inches (in) dan pica (pc).

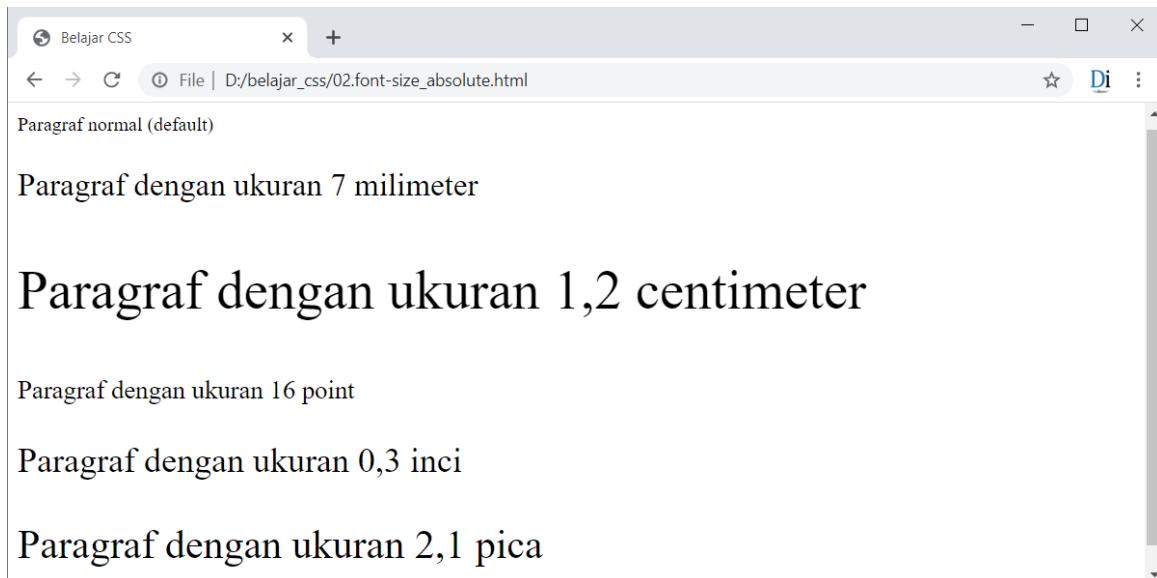
Satuan cm, mm, dan inci sudah cukup dikenal. Satuan point (pt) biasanya dipakai dalam aplikasi word processor seperti Microsoft Word, dimana 72 pt sama dengan 1 inci. Satuan pica (pc) relatif jarang terdengar, dimana 1 pc setara dengan 12 pt.

Berikut contoh penggunaan nilai absolut untuk `font-size`:

02.font-size_absolute.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p.mm { font-size: 7mm; }
8          p.cm { font-size: 1.2cm; }
9          p.pt { font-size: 16pt; }
10         p.in { font-size: .3in; }
11         p.pc { font-size: 2.1pc; }
12     </style>
13 </head>
14 <body>
15     <p>Paragraf normal (default)</p>
16     <p class="mm">Paragraf dengan ukuran 7 milimeter</p>
17     <p class="cm">Paragraf dengan ukuran 1,2 centimeter</p>
18     <p class="pt">Paragraf dengan ukuran 16 point</p>
19     <p class="in">Paragraf dengan ukuran 0,3 inci</p>
20     <p class="pc">Paragraf dengan ukuran 2,1 pica</p>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan nilai absolut untuk property font-size

Dapat terlihat bahwa kita bisa menggunakan angka desimal sebagai nilai satuan. Karena menggunakan format US, penulisan nilai desimal ini harus menggunakan karakter titik, dimana angka 1,2cm ditulis sebagai 1.2cm. Selain itu untuk nilai pecahan dibawah 1, angka 0 sebelum titik boleh diabaikan, misalnya 0.3in bisa ditulis sebagai .3in.

Diantara nilai dan satuannya juga tidak boleh terdapat spasi. Anda harus menulis '1.2cm', bukan '1.2 cm'. Penambahan tanda spasi akan menyebabkan nilai satuan tidak di proses oleh CSS. Ini juga berlaku untuk semua satuan pada seluruh property CSS.

Satuan Pixel

Pixel merupakan nilai yang termasuk ke dalam *absolute value* (walaupun beberapa referensi memasukkannya ke dalam *relative value*). Secara teori, pixel adalah satuan terkecil dari sebuah layar/monitor. Jika kita memiliki layar dengan resolusi 800×600 , maka akan terdapat 480.000 titik pixel di layar tersebut.

Satuan pixel bisa tampil berbeda-beda pada setiap monitor, tergantung resolusi dan ukuran layar. Sebagai contoh, saat ini resolusi sebuah smartphone bisa menyamai resolusi monitor 14 inci. Jika kita membuat ukuran teks sebesar 20 pixel, mungkin akan tampil setinggi 5mm di layar laptop (ukuran font normal), namun akan terlihat sangat kecil dan tidak terbaca di layar smartphone (karena resolusinya yang tinggi).

Dalam kasus seperti ini, web browser di smartphone menggunakan aturan yang disebut dengan *device pixel ratio* (DPR). Secara garis besar, device pixel ratio adalah suatu mekanisme mengalikan ukuran pixel dengan nominal tertentu. Nilai DPR 2 berarti 1 pixel akan tampil dengan ukuran 4 pixel di smartphone (2 pixel lebar dan 2 pixel tinggi). Dengan demikian, nilai satuan pixel di smarphone ber-resolusi HD tetap bisa di pakai.

Untuk menggunakan nilai pixel pada property `font-size`, cukup tambah akhiran px:

```
p {  
    font-size: 14px;  
}
```

Nilai Relatif

Selain nilai absolut, CSS juga menyediakan nilai relatif, seperti satuan persen (%), em dan rem. Sesuai dengan namanya, nilai relatif akan berubah-ubah tergantung posisi dan parent element.

Dalam spesifikasi CSS, satuan persen (%) sebenarnya bukan termasuk ke dalam nilai absolut maupun nilai relatif. Satuan persen adalah nilai satuan yang berdiri sendiri. Namun karena sifatnya sangat mirip dan untuk menyederhanakan pembahasan, saya akan mengelompokkan nilai persen ke dalam nilai relatif.

Satuan Persen (%)

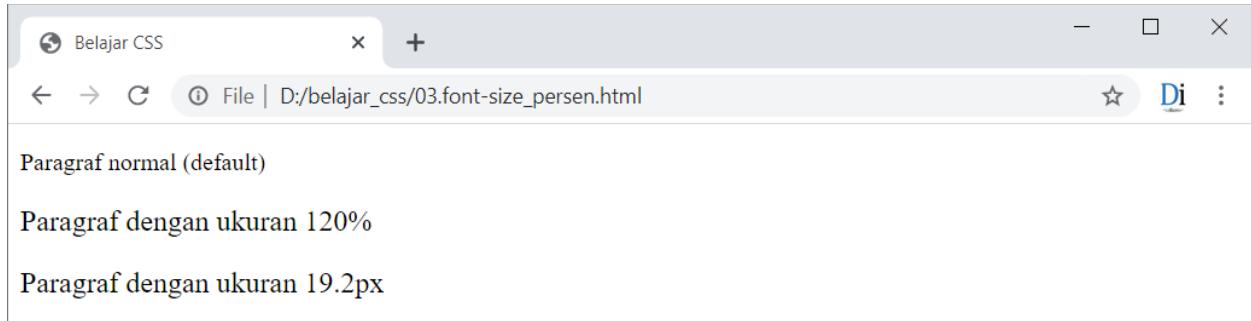
Satuan persen berfungsi untuk men-set nilai property berdasarkan persentasi dari nilai parent element. Berikut contoh penggunaan nilai % untuk font-size:

```
p {  
    font-size: 120%;  
}
```

Ini artinya kita menginstruksikan web browser untuk mencari seluruh tag `<p>`, kemudian set ukuran font sebesar 120% dari ukuran font-size parent element. Apabila saya menerapkannya dalam tag `<p>` tanpa parent element, teks tampil dengan ukuran 19.2 pixel:

03.font-size_persen.html

```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <style>  
7          p.persen { font-size: 120%; }  
8          p.pixel { font-size: 19.2px; }  
9      </style>  
10 </head>  
11 <body>  
12     <p>Paragraf normal (default)</p>  
13     <p class="persen">Paragraf dengan ukuran 120%</p>  
14     <p class="pixel">Paragraf dengan ukuran 19.2px</p>  
15 </body>  
16 </html>
```



Gambar: Hasil property font-size: 120% sama dengan font-size: 19.2 pixel

Dari manakah angka 19.2 pixel ini?

Kembali ke definisi satuan persen, kata kuncinya adalah **ukuran font-size parent element**.

Dalam kode di atas, tag `<p>` dengan `class="persen"` sebenarnya tetap memiliki parent element, yakni tag `<body>`.

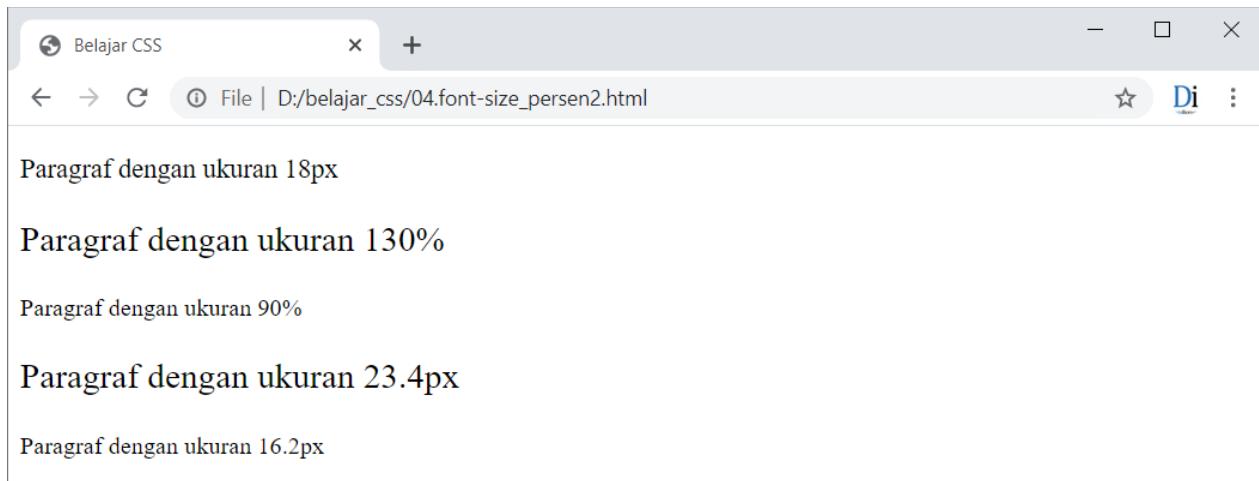
Berapa ukuran font-size tag `<body>`? Karena kita tidak mendefinisikannya (tidak ada di kode CSS), maka ukuran font yang dipakai adalah default style bawaan web browser. Pada umumnya ukuran default ini 16 pixel. Oleh karena itu $120\% * 16 \text{ pixel} = 19,2 \text{ pixel}$.

Contoh berikut ini bisa memperjelas cara penggunaan nilai persen:

04.font-size_persen2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div          { font-size: 18px; }
8          p.persen130 { font-size: 130%;   }
9          p.persen90  { font-size: 90%;    }
10         p.pixel234 { font-size: 23.4px; }
11         p.pixel162 { font-size: 16.2px; }
12     </style>
13 </head>
14 <body>
15     <div>
16         <p>Paragraf dengan ukuran 18px</p>
17         <p class="persen130">Paragraf dengan ukuran 130%</p>
18         <p class="persen90">Paragraf dengan ukuran 90%</p>
19     </div>
20     <div>
21         <p class="pixel234">Paragraf dengan ukuran 23.4px</p>
22         <p class="pixel162">Paragraf dengan ukuran 16.2px</p>
23     </div>
24 </body>
25 </html>
```

CSS Typography



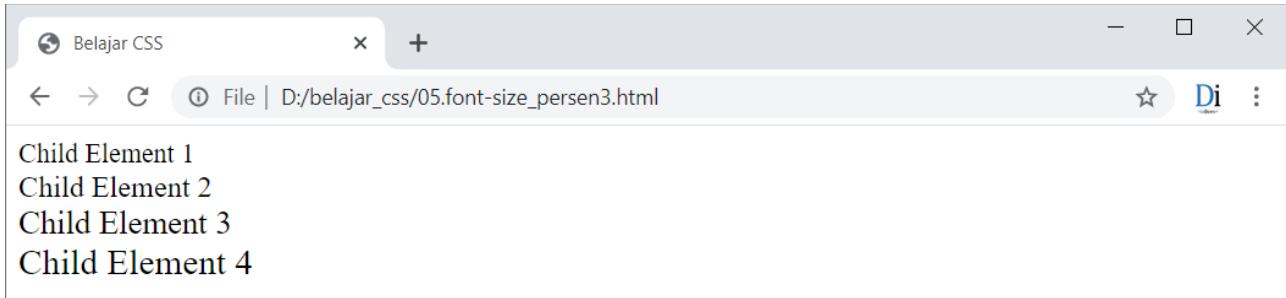
Gambar: Ukuran font-size dengan berbagai nilai satuan persen

Kali ini saya men-set ukuran font-size tag `<div>` sebesar 18px, dan semua paragraf berada di dalam tag ini. Dengan demikian, tag `<div>` berperan sebagai parent element. Ketika `<p class="persen130">` di set dengan `font-size: 130%`, maka ukuran font akhir adalah $130\% * 18\text{px} = 23.4\text{px}$.

Yang juga perlu menjadi catatan, nilai persen ini bisa terakumulasi:

05.font-size_persen3.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body { font-size: 16px; }
8          div { font-size: 110%; }
9      </style>
10 </head>
11 <body>
12     <div>Child Element 1
13         <div>Child Element 2
14             <div>Child Element 3
15                 <div>Child Element 4
16                 </div>
17             </div>
18         </div>
19     </div>
20 </body>
21 </html>
```



Gambar: Efek akumulasi font-size pada satuan persen

Dalam contoh ini saya membuat tag `<div>` di dalam tag `<div>` di dalam tag `<div>` di dalam tag `<div>`, yakni terdapat 4 div element yang saling bertumpuk. Perhatikan bahwa semakin jauh kita masuk, semakin besar ukuran font yang dihasilkan. Sehingga pada tag `<div>` terdalam ukuran fontnya adalah sebesar $110\% * 110\% * 110\% * 110\% * 16px = 23.43px$!

Ini menjadi perhatian penting agar efek akumulasi seperti ini sudah di pertimbangkan sebelum menggunakan satuan persen untuk `font-size`.

Satuan Em

Nilai relatif berikutnya adalah **em**. Satuan em berasal dari istilah typography media cetak yang merujuk kepada tinggi karakter 'm' pada suatu font. Dalam CSS, nilai em sangat mirip seperti persen (%), dimana hasilnya juga dihitung berdasarkan nilai parent element.

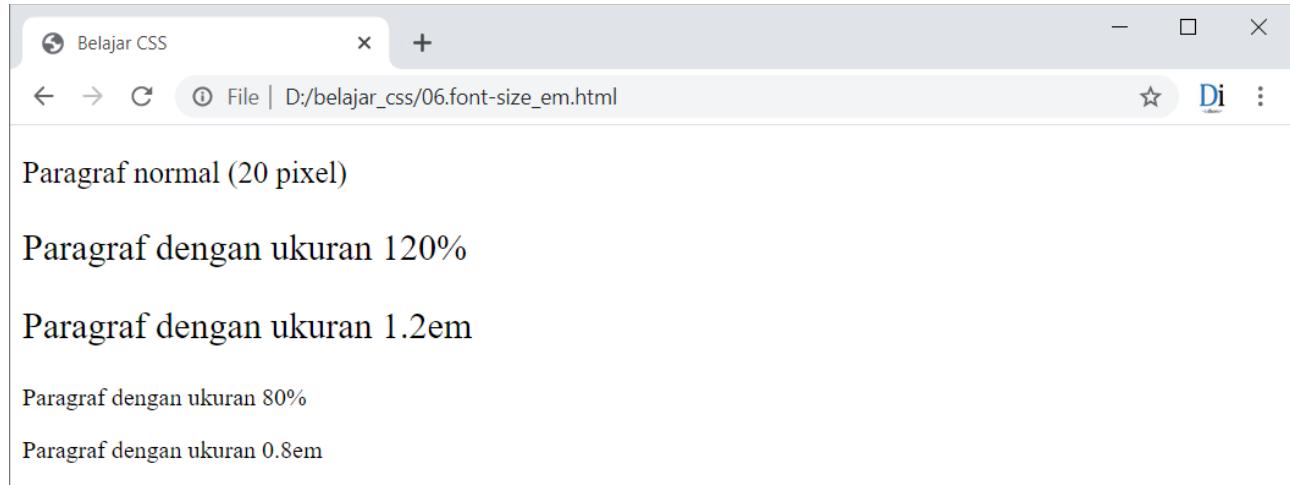
Nilai dari 1em sama dengan 100%, sehingga 1.2em sama dengan 120%, 0.8em sama dengan 80%, 2em sama dengan 200%, dst. Perhatikan contoh berikut ini:

06.font-size_em.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body { font-size: 20px; }
8          p.persen120 { font-size: 120%; }
9          p.persen80 { font-size: 80%; }
10         p.em12 { font-size: 1.2em; }
11         p.em08 { font-size: 0.8em; }
12     </style>
13 </head>
14 <body>
15     <p>Paragraf normal (20 pixel)</p>
16     <p class="persen120">Paragraf dengan ukuran 120%</p>
17     <p class="em12">Paragraf dengan ukuran 1.2em</p>
18     <p class="persen80">Paragraf dengan ukuran 80%</p>
19     <p class="em08">Paragraf dengan ukuran 0.8em</p>
20 </body>
21 </html>

```



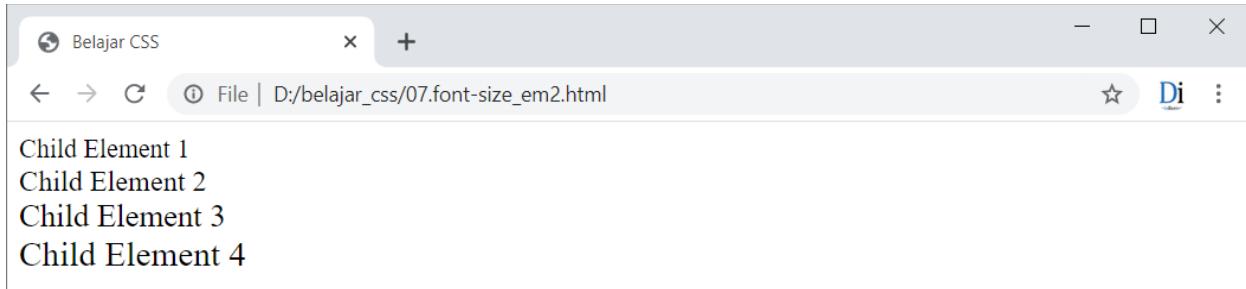
Gambar: Persamaan hasil font-size persen dan em

Dalam contoh ini saya men-set font-size tag <body> sebesar 20 pixel. Ini akan menjadi dasar perhitungan untuk menentukan nilai font-size persen dan em. Sebuah paragraf dengan font-size sebesar 1.2em akan tampil sama besar dengan paragraf berukuran 120%, begitu juga font-size 0.8em akan sama dengan font-size 80%.

Sama seperti persen, satuan em juga bersifat akumulatif::

07.font-size_em2.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body { font-size: 16px; }
8          div { font-size: 1.1em; }
9      </style>
10 </head>
11 <body>
12     <div>Child Element 1
13         <div>Child Element 2
14             <div>Child Element 3
15                 <div>Child Element 4
16                     </div>
17                 </div>
18             </div>
19         </div>
20     </body>
21 </html>
```



Gambar: Efek akumulasi font-size pada satuan em

Contoh ini sama seperti contoh pada satuan persen, hanya saja kali ini menggunakan `font-size: 1.1em`.

Jadi, apa beda antara % dan em?

Saat satuan % dan em dipakai untuk font-size, keduanya nyaris tidak berbeda. Satuan % dan em hanya akan berbeda ketika diterapkan pada property lain seperti lebar dan tinggi element (`width` dan `height`).

Ketika kita menggunakan nilai % untuk lebar sebuah element (menggunakan property `width`), lebar itu dihitung dari width parent element-nya. Sedangkan jika kita menggunakan nilai em, lebar element akan dihitung berdasarkan nilai font-size parent element-nya.

Kita akan kembali membahas ini saat masuk ke bahasan property `width` dan `height`.

Satuan Rem

Nilai relatif terakhir adalah **rem**. Rem merupakan singkatan dari "root em". Berbeda dengan satuan em yang tergantung kepada parent element, rem bergantung kepada 'root element', atau element akar. Di dalam struktur DOM HTML, root element ini adalah tag `<html>`. Seluruh tag lain berada di dalam tag ini, termasuk tag `<body>`, `<div>`, dan `<p>`.

Ketika kita membuat ukuran font-size paragraf sebesar `1.2rem`, artinya ukuran font-size paragraf itu di-set sebesar 120% dari ukuran font-size tag `<html>`, tidak peduli dimana paragraf tersebut berada. Ini artinya tidak ada efek akumulasi pada satuan rem.

Berikut contoh penggunaan rem di dalam CSS:

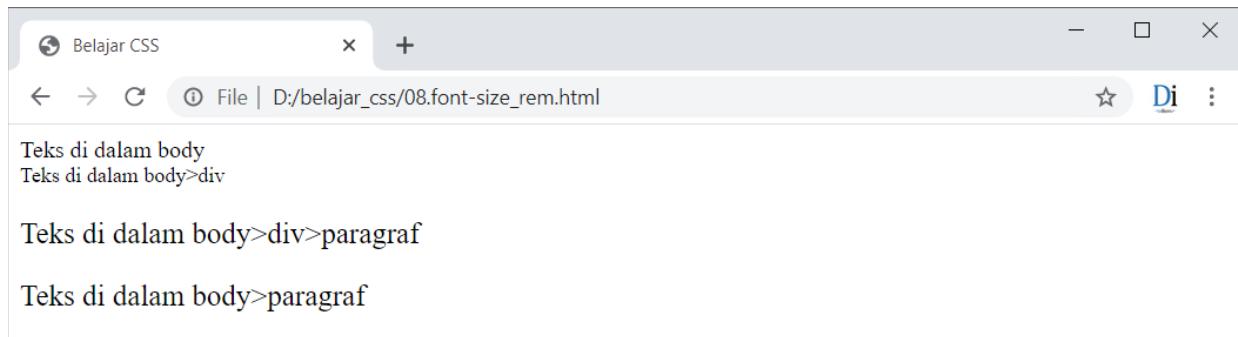
`08.font-size_rem.html`

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          html { font-size: 20px; }
```

```
8      body { font-size: 16px; }
9      div { font-size: 90%; }
10     p   { font-size: 1rem; }
11  </style>
12 </head>
13 <body>
14   Teks di dalam body
15   <div>
16     Teks di dalam body>div
17   <p>
18     Teks di dalam body>div>paragraf
19   </p>
20   </div>
21   <p>
22     Teks di dalam body>paragraf
23   </p>
24 </body>
25 </html>
```

Tanpa melihat hasilnya, dapatkah anda menebak berapa pixel ukuran font-size tag `<p>`? Yup, berdasarkan pengertian satuan rem, kedua paragraf akan berukuran 20 pixel, sesuai dengan ukuran font-size dari selector html.



Gambar: Contoh penggunaan satuan rem untuk property font-size

Selain satuan length dalam bentuk angka, font-size juga bisa diisi dengan keyword. Nilai keyword ini juga ada yang bersifat absolut (tetap) dan ada juga yang bersifat relatif.

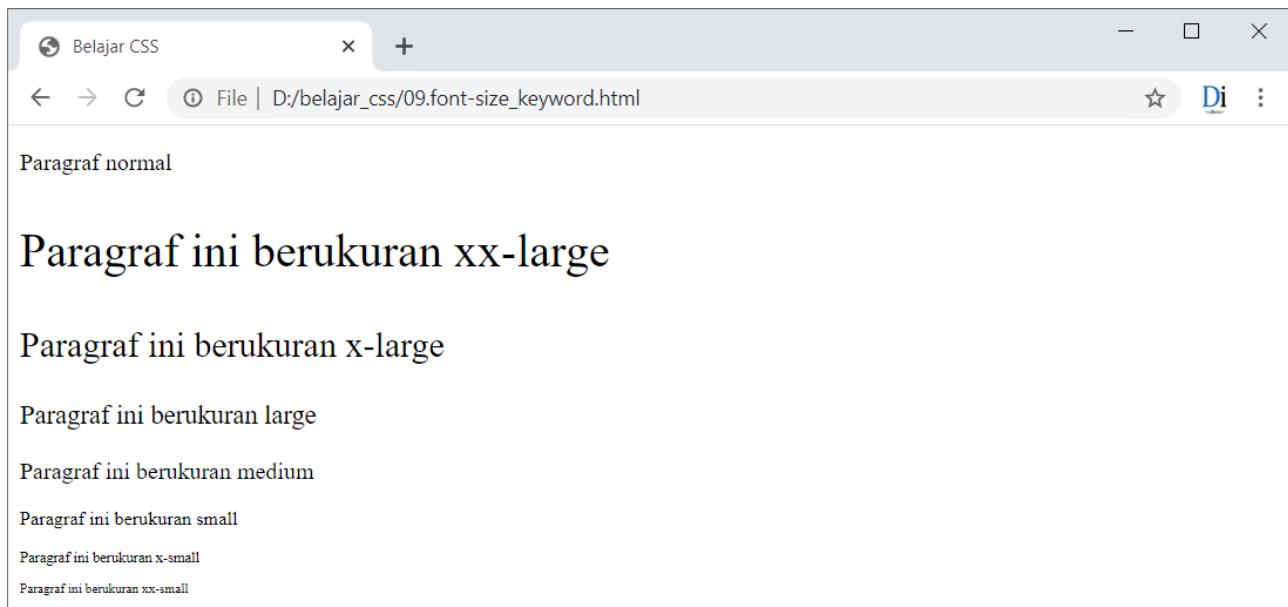
Berikut satuan keyword absolut untuk property font-size:

- xx-small
- x-small
- small
- medium
- large
- x-large
- xx-large

Dan berikut contoh penggunaannya:

09.font-size_keyword.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p.xx-large { font-size: xx-large; }
8          p.x-large { font-size: x-large; }
9          p.large { font-size: large; }
10         p.medium { font-size: medium; }
11         p.small { font-size: small; }
12         p.x-small { font-size: x-small; }
13         p.xx-small { font-size: xx-small; }
14     </style>
15 </head>
16 <body>
17     <p>Paragraf normal</p>
18     <p class="xx-large">Paragraf ini berukuran xx-large</p>
19     <p class="x-large">Paragraf ini berukuran x-large</p>
20     <p class="large">Paragraf ini berukuran large</p>
21     <p class="medium">Paragraf ini berukuran medium</p>
22     <p class="small">Paragraf ini berukuran small</p>
23     <p class="x-small">Paragraf ini berukuran x-small</p>
24     <p class="xx-small">Paragraf ini berukuran xx-small</p>
25 </body>
26 </html>
```



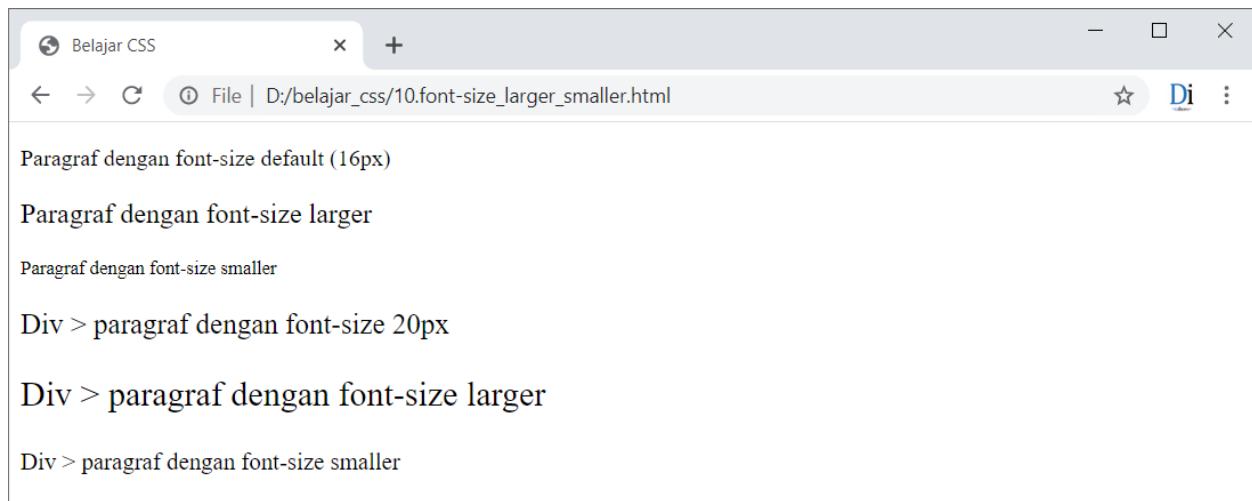
Gambar: Tampilan berbagai ukuran font menggunakan nilai keyword

Besar ukuran font dengan satuan *keyword* diatas diatur sepenuhnya oleh web browser. Biasanya ukuran font default web browser adalah 16 pixel, yakni sama dengan nilai *medium*. Ukuran *xx-small* = 9 pixel, *x-small* = 10 pixel, *small* = 13 pixel, *large* = 18 pixel, *x-large* = 24 pixel, dan *xx-large* = 32 pixel.

Untuk satuan relatif, terdapat 2 nilai: `smaller` dan `larger`. Nilai `larger` akan membuat ukuran font sedikit lebih besar dari font-size parent element, sedangkan nilai `smaller` akan sedikit memperkecil ukuran font. Kembali, jenis satuan ini disebut relatif karena bergantung kepada parent element.

10.font-size_larger_smaller.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .larger { font-size: larger; }
8          .smaller { font-size: smaller; }
9          div { font-size: 20px; }
10     </style>
11 </head>
12 <body>
13 <p>Paragraf dengan font-size default (16px)</p>
14 <p class="larger">Paragraf dengan font-size larger</p>
15 <p class="smaller">Paragraf dengan font-size smaller</p>
16 <div>
17     <p>Div > paragraf dengan font-size 20px</p>
18     <p class="larger">Div > paragraf dengan font-size larger</p>
19     <p class="smaller">Div > paragraf dengan font-size smaller</p>
20 </div>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan `font-size: larger` dan `font-size: smaller`

Walaupun terdapat 2 buah tag `<p class="larger">`, namun masing-masingnya tampil dengan ukuran font-size yang berbeda. Ini karena parent element setiap paragraf juga berbeda.

Parent element paragraf pertama, yakni tag `<body>` memiliki `font-size = 16px`, sedangkan parent element paragraf kedua yakni tag `<div>` memiliki `font-size = 20px`.

Jadi, ukuran apa yang sebaiknya digunakan untuk font-size?

Jawaban akhirnya: tergantung kepada kebutuhan dan kesukaan. Nilai absolut sebaiknya tidak dipakai karena kurang relevan dengan ukuran layar yang bermacam-macam, kecuali pixel. Pengalaman saya, satuan yang paling sering dipakai untuk font-size adalah **pixel** dan **em**.

Beberapa referensi lebih menyarankan em karena menganggap pixel tidak fleksibel untuk perangkat mobile yang memiliki layar kecil. Pada perangkat tersebut, 1 pixel tetaplah 1 pixel, sehingga teks menjadi tidak terbaca di smartphone beresolusi tinggi.

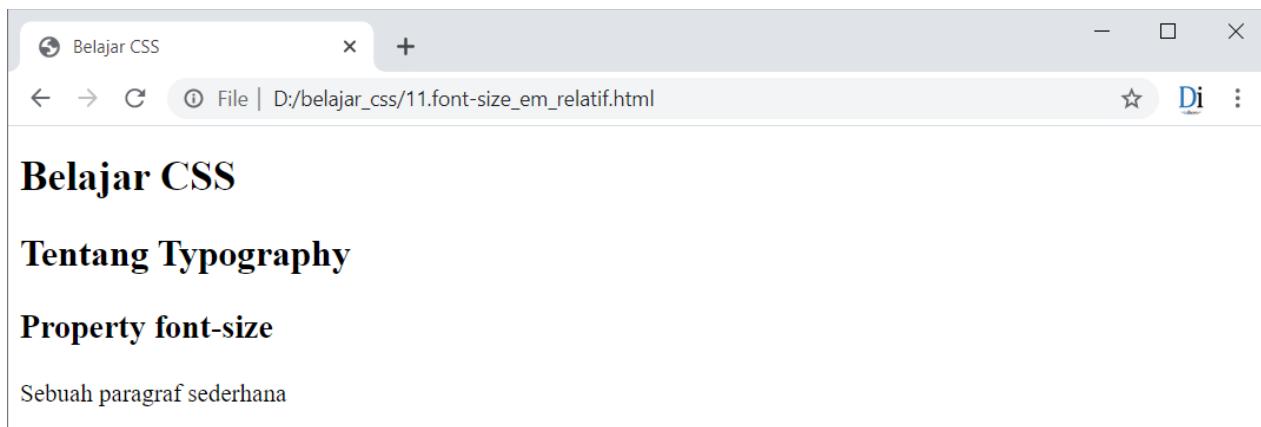
Namun seperti yang kita lihat sekarang, web browser mobile telah mengatasi keterbatasan ini dengan konsep DPR (*device pixel ratio*) sehingga satuan pixel tetap bisa dipakai.

Salah satu keuntungan menggunakan em adalah konsep 'relatif'-nya. Sebagai contoh, perhatikan kode HTML berikut:

11.font-size_em_relatif.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body { font-size: 14px; }
8          h1 { font-size: 2em; }
9          h2 { font-size: 1.8em; }
10         h3 { font-size: 1.6em; }
11         p { font-size: 1.2em; }
12     </style>
13 </head>
14 <body>
15     <h1>Belajar CSS</h1>
16     <h2>Tentang Typography</h2>
17     <h3>Property font-size</h3>
18     <p>Sebuah paragraf sederhana</p>
19 </body>
20 </html>
```



Gambar: Fleksibilitas satuan em untuk property font-size

Saya membuat ukuran font-size untuk 5 selector: body, h1, h2, h3 dan p. Bagaimana jika di kemudian hari saya berpendapat bahwa ukuran font tersebut terlalu kecil dan ingin memperbesarnya beberapa pixel?

Untuk keperluan ini, tinggal mengubah property `font-size = 20px` pada selector body, dan secara otomatis ukuran font pada semua selector akan ikut menjadi besar dengan tetap mempertahankan perbandingannya secara proporsional. Inilah keuntungan menggunakan em pada font-size.

Trik font-size : 62.5%

Salah satu trik yang sering dipakai jika anda memutuskan menggunakan satuan em adalah membuat `font-size : 62.5%` pada selector body:

```
1. body {  
2.   font-size: 62.5%;  
3. }
```

Hasilnya, 1 em = 10 pixel. Perbandingan ini di dapat dari $62.5\% * 16 = 10$ (16 pixel adalah font-size bawaan web browser).

Ini memudahkan perhitungan ukuran font yang di set sebagai em. Misalkan jika kita ingin sebuah paragraf tampil sebesar 18 pixel, maka cukup tulis `font-size: 1.8em`.

Dalam berbagai forum web desainer, akan selalu ada perdebatan tentang 'pixel vs em'. Setiap orang memiliki argumen masing-masing, tapi keputusan terakhir kembali ke kesukaan kita, jika nyaman dengan pixel silahkan menggunakaninya, atau jika ingin menggunakan em juga tidak masalah.

7.3. Property color

Mengubah warna teks merupakan salah satu aspek estetika yang cukup sering kita lakukan saat mendesain web. CSS menyediakan property **color** untuk keperluan ini.

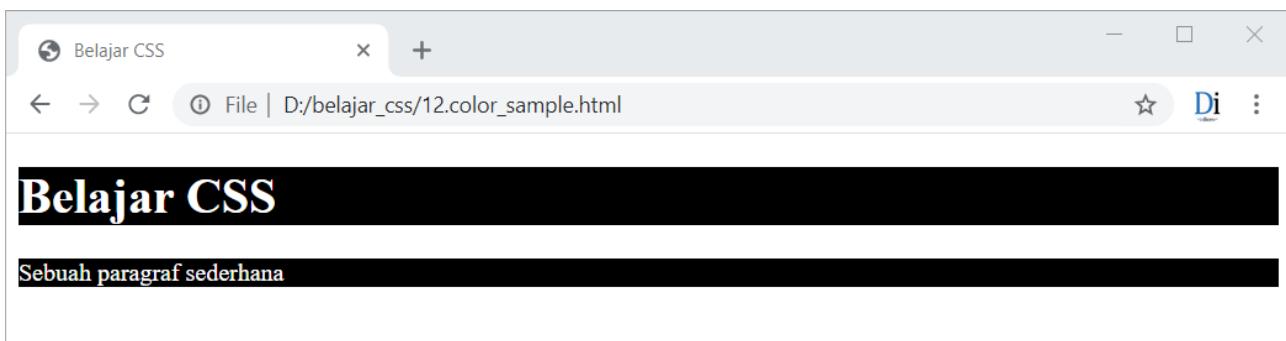
Property **color** berguna untuk mengubah warna font, sedangkan **background-color** dipakai untuk mengubah warna latar belakang dari sebuah teks (warna background). Agar lebih mudah dibedakan, property **color** sering disebut sebagai *foreground color*, dan property **background-color** sebagai... *background color*.

Penggunaan property **color** dan **background-color** sangat mudah. Cukup berikan nilai dalam satuan warna. Sebagai contoh, kode CSS berikut akan membuat warna background menjadi hitam dan warna teks menjadi putih:

```
12.color_sample.html
```

```
1  <!DOCTYPE html>
```

```
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     h1, p {
8       color:white;
9       background-color: black;
10    }
11  </style>
12 </head>
13 <body>
14   <h1>Belajar CSS</h1>
15   <p>Sebuah paragraf sederhana</p>
16 </body>
17 </html>
```



Gambar: Contoh tampilan property color dan background-color

Di sini saya menggunakan keyword `black` dan `white` sebagai nilai warna. Sama seperti satuan `length` pada property `font-size`, CSS juga menyediakan berbagai cara penulisan nilai warna.

Satuan Color (Warna) CSS

CSS mendukung setidaknya 4 notasi penulisan warna:

- keyword
- `#RRGGBB` / `#RGB`
- `rgb(rr, gg, bb)` / `rgba(rr, gg, bb, aa)`
- `hsl(hh, ss, ll)` / `hsla(hh, ss, ll, aa)`

Format Warna: Keyword

Pada beberapa contoh praktik sejak awal buku, saya sudah menggunakan keyword warna untuk property `color`. Keyword yang dimaksud adalah nama warna dalam bahasa Inggris seperti `red`, `black`, `white`, dll.

Pada awalnya CSS hanya mendukung 17 keyword warna dasar, yakni: `aqua`, `black`, `blue`, `fuchsia`, `gray`, `green`, `lime`, `maroon`, `navy`, `olive`, `orange`, `purple`, `red`, `silver`, `teal`, `white` dan `yellow`. Semua warna ini berasal dari sistem VGA Windows yang dikenal juga dengan istilah

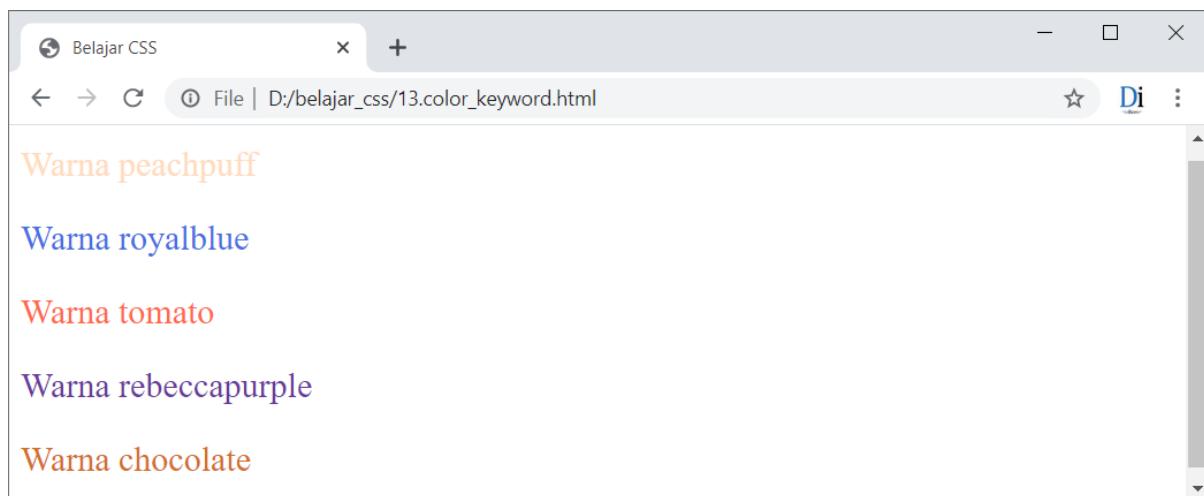
classic internet color.

Saat ini keyword warna CSS telah dikembangkan hingga 147 warna yang disebut sebagai 147 SVG colors (atau X11 colors). Nama warnanya sangat beragam dan cukup unik seperti: turquoise, whitesmoke, rebeccapurple serta peachpuff.

List lengkap untuk 147 warna ini dapat dilihat di [CSS Color Value](#)¹⁶. Berikut contoh penggunaan nilai keyword pada property color:

13.color_keyword.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p { font-size: 24px; }
8     p.peachpuff {color: peachpuff; }
9     p.royalblue {color: royalblue; }
10    p.tomato {color: tomato; }
11    p.rebeccapurple {color: rebeccapurple; }
12    p.chocolate {color: chocolate; }
13  </style>
14 </head>
15 <body>
16  <p class='peachpuff'>Warna peachpuff</p>
17  <p class='royalblue'>Warna royalblue</p>
18  <p class='tomato'>Warna tomato</p>
19  <p class='rebeccapurple'>Warna rebeccapurple</p>
20  <p class='chocolate'>Warna chocolate</p>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan nilai keyword untuk property color

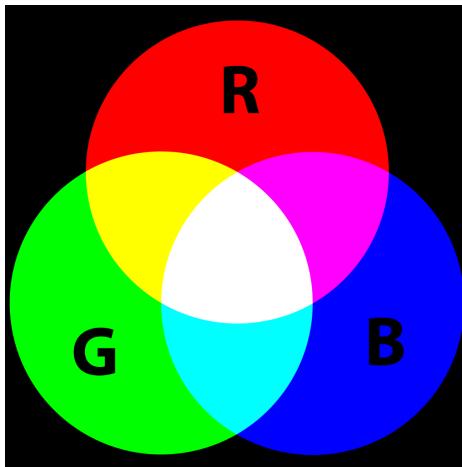
¹⁶ https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

Format Warna: #RRGGBB / #RGB

Format warna seperti #12F5A5 adalah format warna #RRGGBB. Format ini sangat sering dipakai dalam CSS sehingga amat penting untuk dipahami.

RGB adalah singkatan dari **Red Green Blue** (merah hijau biru). Dalam desain media visual seperti televisi atau komputer, model warna inilah yang umumnya dipakai. Seluruh warna lain dihasilkan dari perpaduan ketiga warna dasar RGB.

Model warna lain yang cukup populer adalah **CMYK** (Cyan Magenta Yellow Black). Model warna ini kebanyakan dipakai pada perangkat percetakan seperti printer.



Gambar: Perpaduan warna RGB, sumber: wikipedia.org

Sesuai dengan singkatannya, format #RRGGBB adalah cara penulisan warna RGB dimana **RR** mewakili warna merah (red), **GG** mewakili warna hijau (green), dan **BB** mewakili warna biru (blue). Masing masing warna ini memiliki 256 tingkat kepekatan yang diwakili oleh angka 0 hingga 255.

Misalkan untuk menghasilkan warna putih, kita harus mencampur seluruh warna dengan tingkat kepekatan tertinggi, yakni 255 red, 255 green, dan 255 blue. Sedangkan untuk warna hitam sama dengan tidak ada warna, yakni 0 red, 0 green dan 0 blue.

Contoh lain, untuk menghasilkan warna kuning bisa didapat dengan mencampur warna hijau dan merah tanpa ada warna biru. Dengan demikian, campurannya adalah 255 green, 255 red, dan 0 blue.

Pada format #RRGGBB, masing-masing nilai warna harus di konversi ke angka heksadesimal. Heksadesimal adalah sistem bilangan yang terdiri dari 16 digit, yakni angka 0 – 9 serta huruf A – F. Nilai 255 pada bilangan desimal sama dengan FF pada bilangan heksadesimal.

Untuk menghasilkan warna putih, penulisannya menjadi #FFFFFF. Dua digit FF pertama mewakili warna merah (red), dua digit FF selanjutnya mewakili warna hijau (green) dan dua digit terakhir mewakili warna biru (blue). Untuk menghasilkan warna kuning pekat,

penulisannya menjadi #FFFF00.

Berikut contoh praktik penggunaan nilai warna #RRGGBB:

14.color_rgb_heksa.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p { font-size: 24px; }
8     p.satu {color: #FFFFFF; }
9     p.dua {color: #000000; }
10    p.tiga {color: #FF0000; }
11    p.empat {color: #00FF00; }
12    p.lima {color: #FFFF00; }
13    p.enam {color: #00FFFF; }
14  </style>
15 </head>
16 <body>
17   <p class='satu'>255 Red, 255 Green, dan 255 Blue = #FFFFFF</p>
18   <p class='dua'>0 Red, 0 Green, dan 0 Blue = #000000</p>
19   <p class='tiga'>255 Red, 0 Green, dan 0 Blue = #FF0000</p>
20   <p class='empat'>0 Red, 255 Green, dan 0 Blue = #00FF00</p>
21   <p class='lima'>255 Red, 255 Green, dan 0 Blue = #FFFF00</p>
22   <p class='enam'>0 Red, 255 Green, dan 255 Blue = #00FFFF</p>
23 </body>
24 </html>
```



Gambar: Contoh penggunaan format warna #RRGGBB untuk property color

Dalam contoh di atas saya menggunakan beberapa kombinasi warna RGB. Khusus untuk paragraf pertama teksnya tidak akan terlihat karena nilai warna #FFFFFF akan menghasilkan teks berwarna putih, yang sewarna dengan warna background default web browser.

Dengan mengatur tingkat kepekatan dari 00 hingga FF, kita bisa menghasilkan warna-warna lain seperti contoh berikut:

15.color_rgb_heksa2.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p { font-size: 24px; }
8     p.satu { color: #AAAAAA; }
9     p.dua { color: #FFD700; }
10    p.tiga { color: #ADFF2F; }
11    p.empat { color: #FA4455; }
12    p.lima { color: #191970; }
13    p.enam { color: #F12268; }
14  </style>
15 </head>
16 <body>
17  <p class='satu'>Paragraf dengan warna teks #AAAAAA</p>
18  <p class='dua'>Paragraf dengan warna teks #FFD700</p>
19  <p class='tiga'>Paragraf dengan warna teks #ADFF2F</p>
20  <p class='empat'>Paragraf dengan warna teks #FA4455</p>
21  <p class='lima'>Paragraf dengan warna teks #191970</p>
22  <p class='enam'>Paragraf dengan warna teks #F12268</p>
23 </body>
24 </html>
```



Gambar: Contoh penggunaan format warna #RRGGBB untuk property color

Penulisan format #RRGGBB tidak harus dalam huruf kapital, tapi juga bisa dengan huruf kecil (case insensitive), nilai #FFC0C1 sama saja dengan #ffc0c1. Namun tidak boleh ada spasi antara tanda pagar dengan nilai warna.

Khusus untuk penulisan warna yang berulang, seperti #AABBCC, bisa disingkat menjadi #ABC:

16.color_rgb_heksa_compact.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p { font-size: 24px; }
8     p.satu { color: #AABBCC; }
9     p.dua { color: #ABC; }
10    p.tiga { color: #112233; }
11    p.empat { color: #123; }
12    p.lima { color: #FF00FF; }
13    p.enam { color: #F0F; }
14  </style>
15 </head>
16 <body>
17   <p class='satu'>Paragraf dengan warna teks #AABBCC</p>
18   <p class='dua'>Paragraf dengan warna teks #ABC</p>
19   <p class='tiga'>Paragraf dengan warna teks #112233</p>
20   <p class='empat'>Paragraf dengan warna teks #123</p>
21   <p class='lima'>Paragraf dengan warna teks #FF00FF</p>
22   <p class='enam'>Paragraf dengan warna teks #F0F</p>
23 </body>
24 </html>
```



Gambar: Contoh penulisan singkat format #RRGGBB menjadi #RGB

Penyingkatan penulisan #RGB hanya jika setiap bagian warna memiliki kode heksadesimal yang berulang, seperti #112233 menjadi #123. Jika terdapat satu pasangan angka yang tidak sama, itu tidak bisa disingkat seperti #112234.

Format Warna: **rgb(rr, gg, bb) / rgba(rr, gg, bb, aa)**

Format warna `rgb(rr, gg, bb)` dan `rgba(rr, gg, bb, aa)` sebenarnya hampir sama dengan format `#RRGGBB`, cuma kali ini kita tidak harus menggunakan angka heksadesimal.

Notasi `rgb(rr, gg, bb)` bisa ditulis menggunakan nilai desimal (basis 10) maupun dalam satuan persen. Sebagai contoh, ketiga nilai berikut akan menghasilkan warna yang sama:

- `color: #FF00FF;`
- `color: rgb(255, 0, 255);`
- `color: rgb(100%, 0%, 100%);`

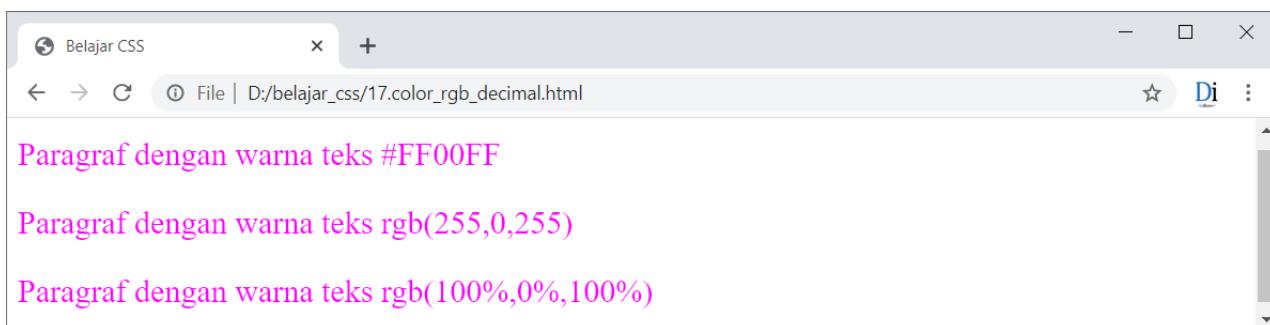
Hasilnya adalah warna pink dari perpaduan warna merah dan biru. Di sini nilai FF = 255 = 100%.

17.color_rgb_decimal.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-size: 24px; }
8          p.satu { color: #FF00FF; }
9          p.dua { color: rgb(255,0,255); }
10         p.tiga { color: rgb(100%,0%,100%); }
11     </style>
12 </head>
13 <body>
14     <p class='satu'>Paragraf dengan warna teks #FF00FF</p>
15     <p class='dua'>Paragraf dengan warna teks rgb(255,0,255)</p>
16     <p class='tiga'>Paragraf dengan warna teks rgb(100%,0%,100%)</p>
17 </body>
18 </html>

```



Gambar: Contoh penulisan format warna `rgb(rr, gg, bb)`

Dengan menggunakan notasi `rgb(rr, gg, bb)` kita bisa mencampur warna RGB dengan lebih mudah. Misalnya saya ingin mencampur 25% merah, 50% hijau dan 75% biru, maka tinggal menulis sebagai `color:rgb(25%, 50%, 75%)`.

Namun kita tidak bisa mencampur satuan nilai warna seperti `rgb(25%, 255, 75%)`. Nilai yang ada harus ditulis dalam satuan persen semua atau desimal semua, tidak bisa dicampur:

```

/* error: mencampurkan persen dengan desimal */
color: rgb(100%, 255, 20%);

```

```

/* error: nilai 0 seharusnya tetap ditulis sebagai 0% */
color: rgb(100%, 0, 20);

/* tidak boleh diisi angka pecahan */
color: rgb(75, 255, 30.5);

```

Beberapa web browser ada yang mendukung penulisan campuran seperti di atas. Tapi agar lebih standar dan tidak membuat bingung, sebaiknya tetap menggunakan salah satu jenis saja.

Format `rgba(rr, gg, bb, aa)` adalah variasi dari `rgb(rr, gg, bb)` dengan penambahan nilai *alpha channel*. Ini adalah fitur baru yang ditambahkan dalam CSS3. Alpha channel berfungsi untuk mengatur tingkat transparansi warna. Nilai yang bisa diisi berupa angka desimal (boleh berbentuk pecahan) dalam rentang 0 (transparan) hingga 1 (warna utuh).

Sebagai contoh, untuk membuat sebuah warna merah dengan tingkat transparansi 50%, bisa menggunakan `color: rgba(100%, 0%, 0%, 0.5)`.

Perhatikan nilai alpha channel harus dituliskan pada urutan keempat dan menggunakan tanda titik sebagai penanda pecahan. Berikut contoh tampilan dari beberapa nilai alpha channel:

18.color_rgba.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-size: 24px; }
8          p.satu { color: rgba(45,45,80,1);    }
9          p.dua { color: rgba(45,45,80,0.8);   }
10         p.tiga { color: rgba(45,45,80,0.6);  }
11         p.empat { color: rgba(45,45,80,0.4); }
12         p.lima { color: rgba(45,45,80,0.2);  }
13         p.enam { color: rgba(45,45,80,0);    }
14     </style>
15 </head>
16 <body>
17     <p class='satu'>Paragraf dengan alpha channel 1</p>
18     <p class='dua'>Paragraf dengan alpha channel 0.8</p>
19     <p class='tiga'>Paragraf dengan alpha channel 0.6</p>
20     <p class='empat'>Paragraf dengan alpha channel 0.4</p>
21     <p class='lima'>Paragraf dengan alpha channel 0.2</p>
22     <p class='enam'>Paragraf dengan alpha channel 0</p>
23 </body>
24 </html>

```



Gambar: Contoh penulisan format warna rgba(rr, gg, bb, aa)

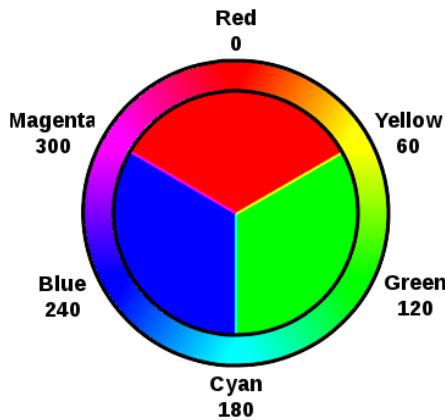
Dalam contoh ini saya membuat satu warna dengan 6 tingkat alpha channel, mulai dari 1 (tanpa transparansi) hingga 0 (full transparan). Efek transparansi ini lebih kelihatan jika kita menumpuk beberapa 'layer' atau memakai background berupa gambar.

Format Warna: **hsl(hh, ss, ll)** / **hsla(hh, ss, ll, aa)**

Format warna **hsl(hh, ss, ll)** dan pasangannya **hsla(hh, ss, ll, aa)** merupakan format penulisan warna baru dari CSS3.

HSL merupakan singkatan dari **Hue**, **Saturation** dan **Lightness** (atau kadang disebut juga dengan **Luminance**). HSL menggunakan konsep 'mencari warna' menggunakan *color wheel*.

Color wheel adalah sebuah lingkaran yang berisi berbagai jenis warna, dimana setiap warna memiliki posisi tertentu (dalam satuan derajat). Agar lebih jelas, berikut contoh dari color wheel:



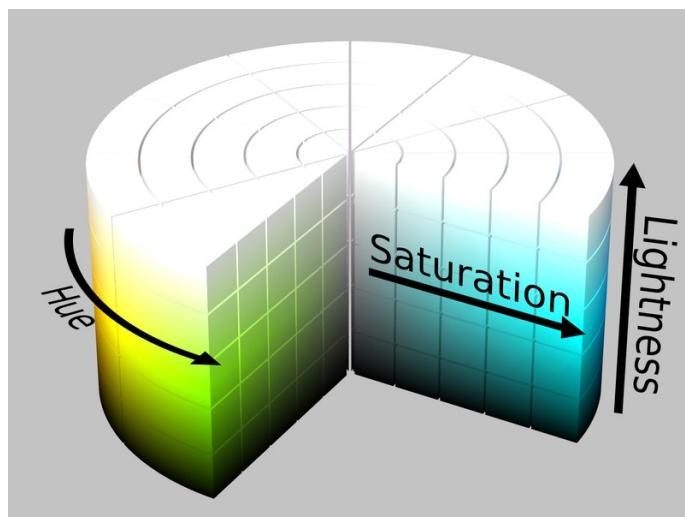
Gambar: Contoh color wheel (roda warna), sumber: wikimedia.org

Dalam color wheel, posisi setiap warna dinyatakan dalam derajat, mulai dari warna merah (0

derajat), warna kuning (60 derajat), hijau (120 derajat), biru (240 derajat), dan kembali lagi ke merah. Antara 0-360 derajat ini terdapat kombinasi warna lain. Inilah yang menjadi inputan untuk nilai **Hue** dalam format warna **HSL**.

Saturation merujuk kepada 'kekuatan' sebuah warna, mulai dari 0% yang berarti tanpa warna (grayscale), hingga 100% (warna penuh).

Lightness adalah tingkat kecerahan warna. Jika nilai lightness diinput 0%, warna apapun akan menjadi hitam, 100% akan menghasilkan putih, dan 50% adalah warna sebenarnya.



Gambar: Konsep hue, saturation dan lightness pada model warna HSL, sumber: wikipedia.org

Bagi yang belum pernah mendengar konsep warna HSL, penjelasan di atas akan terasa cukup rumit, namun sebenarnya HSL lebih mudah dipakai daripada RGB, terutama jika ingin menebak perpaduan warna (tanpa menggunakan tools).

Sebagai contoh, untuk membuat warna biru muda, bisa menggunakan kombinasi Hue= 240 derajat (dengan melihat color wheel), Saturation = 100%, dan Lightness = 70%. Di dalam CSS, ini ditulis sebagai `hsl(240, 100%, 70%)`.

Bagaimana dengan biru tua? menggunakan konsep lightness, kita tinggal menggelapkan warna tadi dengan mengurangi lightness, misalnya menjadi `hsl(240, 100%, 30%)`.

Warna pink muda? Dengan melihat color wheel, warna pink ada di kisaran 300 derajat. Agar tampak lebih "muda", bisa dengan menaikkan nilai lightness ke 80%: `hsl(300, 100%, 80%)`.

Teknik mengatur warna seperti ini akan lebih susah jika menggunakan RGB.

Tabel berikut merangkum perbandingan cara penulisan warna HSL, RBG dan keyword:

HSL	RGB	Hexadecimal	Keyword
0,0%,0%	0,0,0	#000000	black

HSL	RGB	Hexadecimal	Keyword
360,0%,100%	255,255,255	#FFFFFF	white
0,100%,50%	255,0,0	#FF0000	red
120,100%,25%	0,255,0	#00FF00	green
240,100%,50%	0,0,255	#0000FF	blue

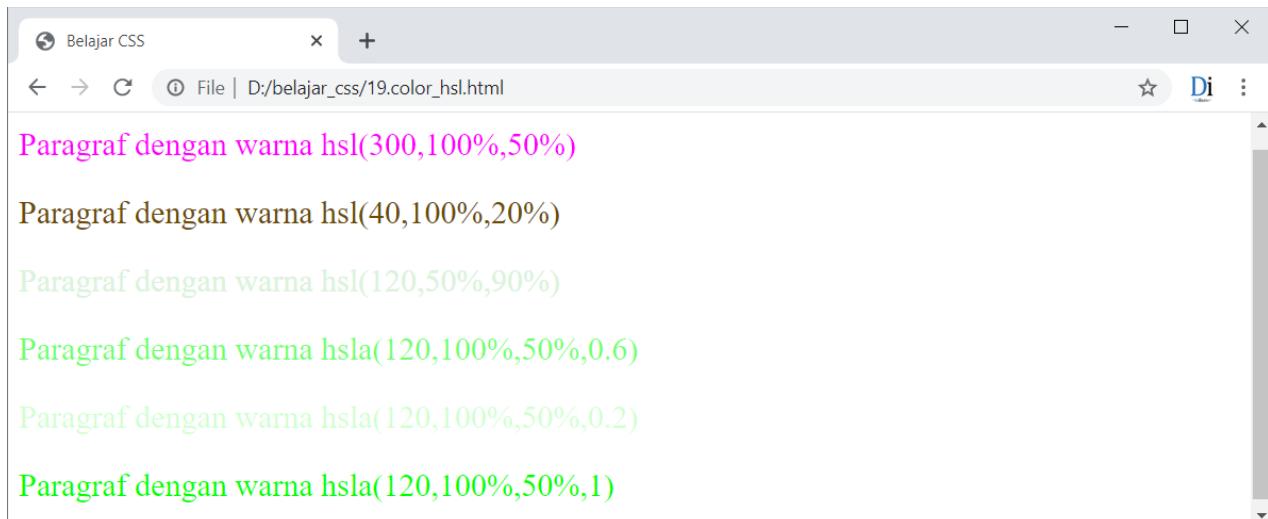
Format `hsla(hh, ss, ll, aa)` adalah penulisan warna HSL dengan penambahan *alpha channel*. Ini sama persis dengan fungsi alpha channel dalam format `rgba`, dimana nilainya berada dalam rentang 0 (transparan penuh) hingga 1 (tanpa transparansi).

Berikut contoh penggunaan berbagai nilai warna `hsl` dan `hsla`:

19.color_hsl.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-size: 24px; }
8          p.satu {color: hsl(300,100%,50%); }
9          p.dua {color: hsl(40,100%,20%); }
10         p.tiga {color: hsl(120,50%,90%); }
11         p.empat {color: hsla(120,100%,50%,0.6); }
12         p.lima {color: hsla(120,100%,50%,0.2); }
13         p.enam {color: hsla(120,100%,50%,1); }
14     </style>
15 </head>
16 <body>
17     <p class='satu'>Paragraf dengan warna hsl(300,100%,50%)</p>
18     <p class='dua'>Paragraf dengan warna hsl(40,100%,20%)</p>
19     <p class='tiga'>Paragraf dengan warna hsl(120,50%,90%)</p>
20     <p class='empat'>Paragraf dengan warna hsla(120,100%,50%,0.6)</p>
21     <p class='lima'>Paragraf dengan warna hsla(120,100%,50%,0.2)</p>
22     <p class='enam'>Paragraf dengan warna hsla(120,100%,50%,1)</p>
23 </body>
24 </html>
```



Gambar: Contoh penulisan format warna hsl(hh, ss, ll) dan hsla(hh, ss, ll, aa)

Yang cukup unik dari format warna **hsl** adalah, jika kita menggunakan nilai 0% untuk saturation, itu akan menghasilkan warna grayscale (abu-abu), terlepas dari warna apapun yang digunakan untuk nilai hue.

Hal yang sama juga terjadi jika nilai lightness di-set menjadi 0% atau 100%, warna yang dihasilkan akan menjadi hitam atau putih (tanpa memperhitungkan nilai hue).

7.4. Property font-weight

Property **font-weight** bertujuan untuk menebalkan teks (*bold*). Nilai untuk property ini bisa berupa angka 100, 200, 300, 400, 500, 600, 700, 800 dan 900 serta keyword *normal*, *bold*, *bolder* dan *lighter*.

Nilai angka 100 s/d 900 mencerminkan tingkat ketebalan huruf, mulai dari 100 untuk teks paling tipis, hingga 900 untuk teks dengan tingkat ketebalan tertinggi. Namun perlu dicatat nilai-nilai ini harus didukung oleh font yang dipakai.

Kebanyakan font hanya mendukung teks *normal* atau *bold* (tebal), dan tidak mendukung tingkat ketebalan bertahap 100 s/d 900.

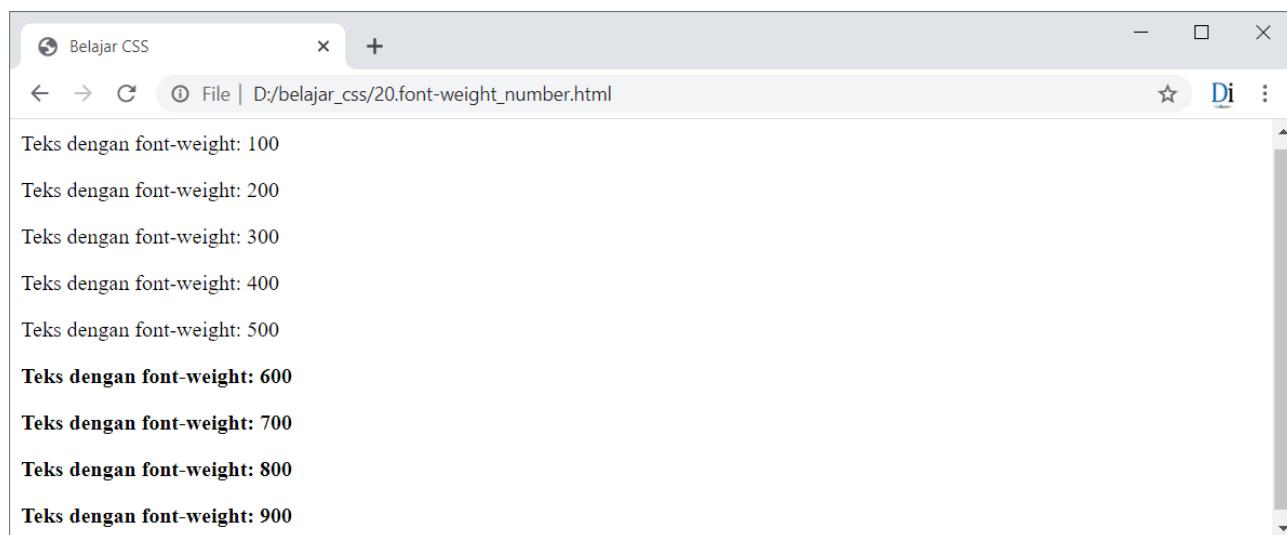
Berikut contoh praktik penggunaan property **font-weight**:

20.font-weight_number.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
```

CSS Typography

```
7     .bold100 { font-weight: 100; }
8     .bold200 { font-weight: 200; }
9     .bold300 { font-weight: 300; }
10    .bold400 { font-weight: 400; }
11    .bold500 { font-weight: 500; }
12    .bold600 { font-weight: 600; }
13    .bold700 { font-weight: 700; }
14    .bold800 { font-weight: 800; }
15    .bold900 { font-weight: 900; }
16  </style>
17 </head>
18 <body>
19  <p class="bold100">Teks dengan font-weight: 100</p>
20  <p class="bold200">Teks dengan font-weight: 200</p>
21  <p class="bold300">Teks dengan font-weight: 300</p>
22  <p class="bold400">Teks dengan font-weight: 400</p>
23  <p class="bold500">Teks dengan font-weight: 500</p>
24  <p class="bold600">Teks dengan font-weight: 600</p>
25  <p class="bold700">Teks dengan font-weight: 700</p>
26  <p class="bold800">Teks dengan font-weight: 800</p>
27  <p class="bold900">Teks dengan font-weight: 900</p>
28 </body>
29 </html>
```



Gambar: Contoh penggunaan property font-weight dengan nilai 100 s/d 900

Ternyata teks tetap tampil normal untuk nilai 100-500 dan 'bold' untuk nilai 600-900. Ini karena font default web browser tidak mendukung perbedaan ketebalan ini.

Pada prakteknya, kita hanya butuh memberikan nilai **bold** untuk property **font-weight** seperti kode berikut:

```
p {
  font-weight: bold;
}
```

Dengan ini, web browser secara otomatis menentukan font terbaik untuk menebalkan teks.

CSS juga menyediakan nilai `normal` yang bertujuan untuk mengembalikan teks tebal ke kondisi normal:

21.font-weight_normal.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h1.normal { font-weight: normal; }
8          p { font-weight: bold; }
9      </style>
10 </head>
11 <body>
12     <h1>Header h1 akan ditampilkan dengan teks tebal (default)</h1>
13     <h1 class="normal">Header h1 dengan font-weight: normal</h1>
14     <p>Paragraf ini akan ditampilkan dengan huruf tebal</p>
15 </body>
16 </html>
```



Gambar: Contoh penggunaan property `font-weight` dengan nilai `normal`

Pada style default bawaan web browser, seluruh tag header `<h1>` s/d `<h6>` akan tampil dengan teks tebal. Nilai `font-weight: normal` bisa menimpa efek ini agar tag `<h1>` tampil dengan ketebalan teks biasa.

Nilai terakhir untuk property `font-weight` adalah `lighter` dan `bolder`. Kedua nilai ini merupakan nilai relatif untuk menipiskan/menebalkan teks bergantung dari `font-weight` parent element.

Sebagai contoh, jika terdapat paragraf yang memiliki `font-weight: 500`, maka child element dengan nilai `font-weight: lighter` akan men-set nilai `font-weight` menjadi `100`, sedangkan jika child element dengan nilai `font-weight: bolder` akan men-set nilai `font-weight` menjadi `700*`.

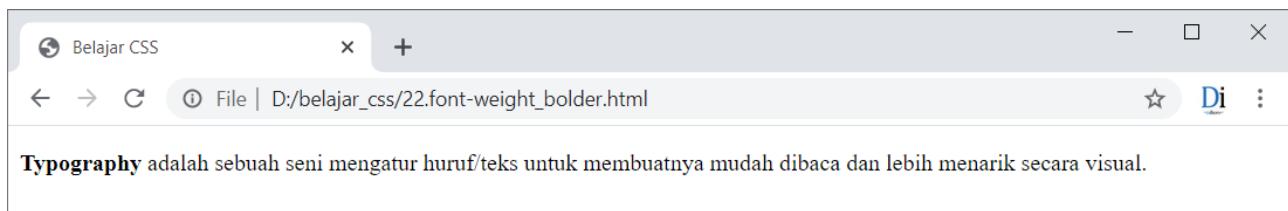
*Ini adalah aturan yang digunakan oleh Mozilla Firefox. Untuk web browser lain mungkin akan berbeda.

Berikut contoh penggunaan nilai `lighter` dan `bolder` untuk property `font-weight`:

22.font-weight_bolder.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-weight: 400; }
8          .tipis { font-weight: lighter; }
9          .tebal { font-weight: bolder; }
10     </style>
11 </head>
12 <body>
13     <p><span class="tebal">Typography</span> adalah sebuah seni mengatur
14     huruf/teks untuk membuatnya mudah dibaca dan lebih menarik
15     <span class="tipis"></span>secara visual.</p>
16 </body>
17 </html>
```



Gambar: Contoh penggunaan property `font-weight` dengan nilai `lighter` dan `bolder`

Seperti yang terlihat, efek `lighter` tidak berpengaruh apa-apa kepada teks. Sedangkan untuk nilai `bolder`, teks akan tampil dengan huruf tebal (sama seperti nilai `bold`).

Khusus untuk pengguna Windows 7 ke atas, terdapat font "Segoe UI" yang mendukung 4 jenis bold: `light`, `normal`, `semi-bold` dan `bold`. Kita bisa coba dengan memodifikasi kode sebelum ini:

23.font-weight_bold_segoe_ui.html

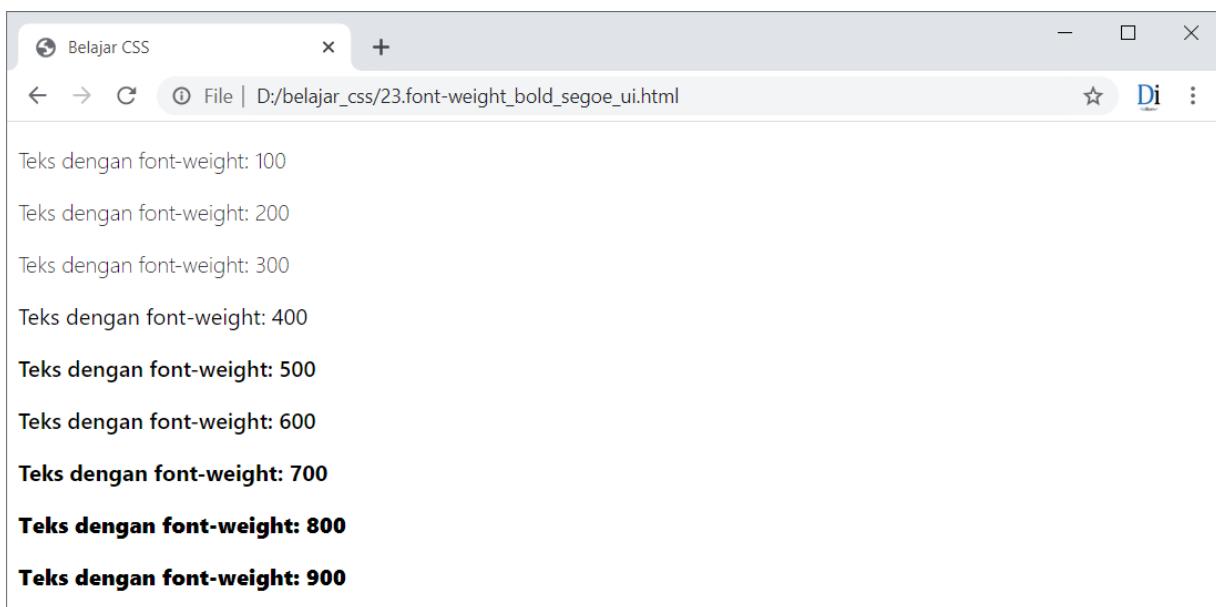
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-family: "Segoe UI"; }
8          .bold100 { font-weight: lighter; }
9          .bold200 { font-weight: 200; }
10         .bold300 { font-weight: 300; }
11         .bold400 { font-weight: 400; }
12         .bold500 { font-weight: 500; }
13         .bold600 { font-weight: 600; }
14         .bold700 { font-weight: 700; }
15         .bold800 { font-weight: 800; }
```

```

16     .bold900 { font-weight: 900; }
17   </style>
18 </head>
19 <body>
20   <p class="bold100">Teks dengan font-weight: 100</p>
21   <p class="bold200">Teks dengan font-weight: 200</p>
22   <p class="bold300">Teks dengan font-weight: 300</p>
23   <p class="bold400">Teks dengan font-weight: 400</p>
24   <p class="bold500">Teks dengan font-weight: 500</p>
25   <p class="bold600">Teks dengan font-weight: 600</p>
26   <p class="bold700">Teks dengan font-weight: 700</p>
27   <p class="bold800">Teks dengan font-weight: 800</p>
28   <p class="bold900">Teks dengan font-weight: 900</p>
29 </body>
30 </html>

```



Gambar: Perbedaan tingkatan font-weight pada font Segoe UI

Sekarang kita bisa lihat berbagai jenis tingkat ketebalan font menggunakan property `font-weight`.

7.5. Property `font-style`

Nama property ini seolah-olah memiliki banyak fungsi, namun property `font-style` hanya berfungsi untuk memiringkan text (membuat efek *italic*). Nilai yang bisa diinput adalah salah satu dari `normal`, `italic` dan `oblique`.

Pada kebanyakan font, tampilan dari `font-style: italic` dan `font-style: oblique` akan sama, karena mayoritas font hanya mendukung salah satu saja (biasanya hanya *italic*).

Dalam teori typography, *italic* dan *oblique* merupakan hal yang berbeda. Font ***italic*** adalah tipe font yang tidak hanya tampil miring, tetapi menggunakan karakter yang berbeda dari font

normal (memiliki file font sendiri). Beberapa font italic ada mirip dengan tulisan tangan, terutama pada huruf 'a' kecil, seperti contoh berikut:

The five boxing wizards jump quickly.
The five boxing wizards jump quickly.

Gambar: Perbedaan font reguler (atas) dengan font italic (bawah), sumber: wikipedia.org

Baris pertama merupakan tipe font normal dari font Garamond, sedangkan baris kedua adalah tipe italic dari font yang sama. Perhatikan beda style huruf terutama pada karakter 'a' kecil.

Font **oblique** adalah jenis font normal yang hanya dimiringkan beberapa derajat. Umumnya, tampilan yang dihasilkan tidak sebagus italic:

The five boxing wizards jump quickly.

Gambar: Contoh font oblique dari Garamond

Dalam penggunaannya di CSS, hanya sedikit font yang mendukung perbedaan antara *italic* dan *oblique*. Ditambah lagi kebanyakan web browser akan menggunakan versi italic meskipun kita memberikan nilai `font-style:oblique`.

Berikut contoh penggunaan property `font-style` dalam CSS:

24.font-style.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p { font-family: Garamond; }
8          .italic { font-style: italic; }
9          .oblique { font-style: oblique; }
10         .normal { font-style: normal; }
11     </style>
12 </head>
13 <body>
14     <p>
15         Paragraf normal tanpa property font-style
16     </p>
17     <p class="italic">
18         Paragraf ini menggunakan property font-style: italic
19     </p>
20     <p class="oblique">
21         Paragraf ini menggunakan property font-style: oblique
22     </p>
```

```

23 <p>
24   <i>Paragraf ini berada di dalam tag i yang secara default tampil italic</i>
25 </p>
26 <p>
27   <i class="normal">Paragraf ini berada di dalam tag i 'normal'</i>
28 </p>
29 </html>

```



Gambar: Contoh penggunaan property font-style

Terlihat tidak ada perbedaan antara `font-style:italic` dengan `font-style:oblique`. Pada paragraf terakhir saya menambah property `font-style:normal` untuk menghapus style italic bawaan tag `<i>`.

Pada beberapa kasus, apabila font italic tidak tersedia, web browser akan mencari solusi dengan memiringkan font normal beberapa derajat untuk mendapatkan efek 'italic', sehingga tidak lain menjadi font oblique. Kita akan lihat efek ini saat membahas font external di akhir bab nanti.

7.6. Property text-decoration

CSS menyediakan property **text-decoration** untuk membuat dekorasi teks seperti garis bawah, garis atas dan garis tercoret. Nilai untuk property ini adalah salah satu dari keyword `underline`, `overline`, `line-through` dan `none`. Kita juga bisa menggunakan 2 atau 3 efek dekorasi sekaligus.

Berikut contoh penggunaan property `text-decoration`:

25.text-decoration.html

```

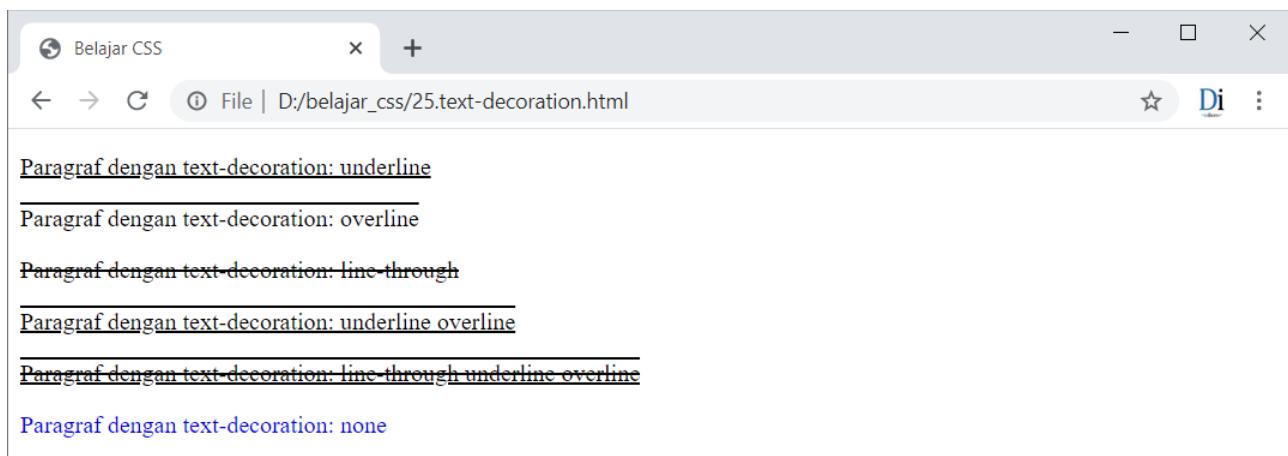
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      .satu { text-decoration: underline; }

```

```

8     .dua   { text-decoration: overline; }
9     .tiga  { text-decoration: line-through; }
10    .empat { text-decoration: underline overline; }
11    .lima  { text-decoration: line-through underline overline; }
12    .enam  { text-decoration: none; }
13  </style>
14 </head>
15 <body>
16  <p class="satu">Paragraf dengan text-decoration: underline</p>
17  <p class="dua">Paragraf dengan text-decoration: overline</p>
18  <p class="tiga">Paragraf dengan text-decoration: line-through</p>
19  <p class="empat">Paragraf dengan text-decoration: underline overline</p>
20  <p class="lima">
21    Paragraf dengan text-decoration: line-through underline overline</p>
22  <p>
23    <a class="enam" href="#">Paragraf dengan text-decoration: none</a>
24  </p>
25 </body>
26 </html>

```



Gambar: Contoh penggunaan property text-decoration

Dari tampilan ini bisa terlihat bahwa untuk membuat garis bawah (*underline*) kita memakai perintah `text-decoration:underline`. Untuk membuat garis atas (*overline*) bisa menggunakan `text-decoration:overline`, serta untuk membuat garis tercoret (*line-through/strike*) menggunakan `text-decoration:line-through`.

Selain itu, 2 atau 3 efek bisa dugabung sekaligus seperti pada paragraf ke empat dan ke lima.

Nilai property `none` bisa dipakai untuk menghapus style `text-decoration` yang sudah ada. Di paragraf ke-6, saya menggunakan `text-decoration: none` pada tag `<a>`. Perintah ini akan menghapus efek garis bawah yang secara default ada di setiap link. Trik menghapus garis bawah seperti ini sangat sering dipakai ketika membuat menu navigasi.

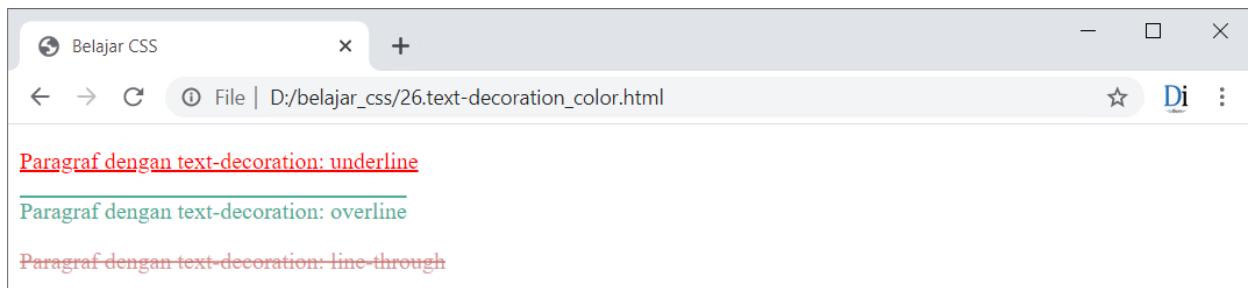
Warna garis `text-decoration` akan menyesuaikan dengan warna teks yang berasal dari property `color`:

26.text-decoration_color.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              text-decoration: underline;
9              color: red;
10         }
11         .dua {
12             text-decoration: overline;
13             color: #45AA90;
14         }
15         .tiga {
16             text-decoration: line-through;
17             color: rgba(200, 100, 100, 0.7);
18         }
19     </style>
20 </head>
21 <body>
22     <p class="satu">Paragraf dengan text-decoration: underline</p>
23     <p class="dua">Paragraf dengan text-decoration: overline</p>
24     <p class="tiga">Paragraf dengan text-decoration: line-through</p>
25 </body>
26 </html>

```



Gambar: Warna text-decoration akan menyesuaikan dengan warna dari property color

7.7. Property text-decoration CSS3

Property `text-decoration` merupakan salah satu property yang mendapat update di CSS3.

Selain mengatur posisi garis, yakni apakah di bagian bawah, atas, atau tengah, sekarang kita juga bisa mengatur warna dan jenis garis. Untuk hal ini property `text-decoration` dipecah menjadi 3 property baru:

- `text-decoration-line`
- `text-decoration-style`
- `text-decoration-color`

Property `text-decoration-line` berfungsi untuk mengatur posisi garis, yang nilainya adalah salah satu dari `underline`, `overline`, `line-through` atau `none`. Nilai ini juga bisa digabung seperti `text-decoration-line: underline overline`.

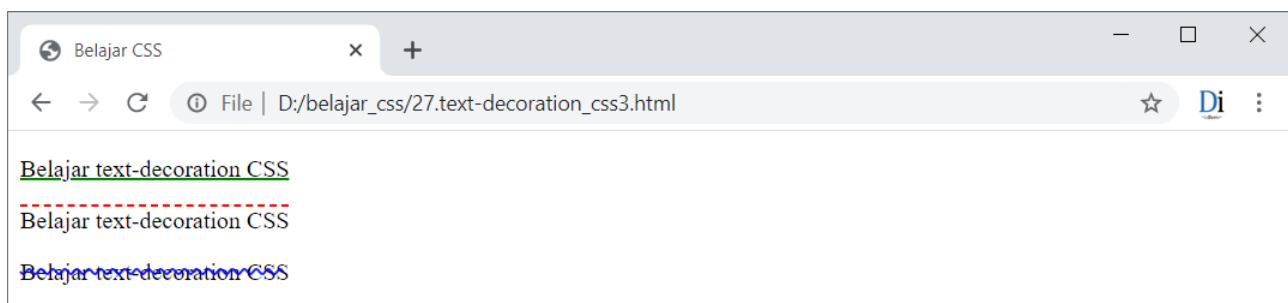
Property `text-decoration-style` berfungsi untuk mengatur jenis garis, dan bisa diisi dengan `solid`, `double`, `dotted`, `dashed` atau `wavy`.

Serta property `text-decoration-color` berfungsi untuk mengatur warna garis, nilainya bisa berupa kode warna yang kita pelajari pada property `color`, yakni keyword, warna RGB/RGBA maupun HSL/HSLA. Berikut contoh penggunaan property `text-decoration` CSS3:

27.text-decoration_css3.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              text-decoration-line: underline;
9              text-decoration-style: solid;
10             text-decoration-color: green;
11         }
12         .dua {
13             text-decoration-line: overline;
14             text-decoration-style: dashed;
15             text-decoration-color: #FF0000;
16         }
17         .tiga {
18             text-decoration-line: line-through;
19             text-decoration-style: wavy;
20             text-decoration-color: hsla(240,100%,50%,0.9);
21         }
22     </style>
23 </head>
24 <body>
25     <p class="satu">Belajar text-decoration CSS</p>
26     <p class="dua">Belajar text-decoration CSS</p>
27     <p class="tiga">Belajar text-decoration CSS</p>
28 </body>
29 </html>
```



Gambar: Contoh penggunaan property `text-decoration` CSS3

Dalam contoh ini saya membuat berbagai kombinasi kode CSS untuk `text-decoration`. Selain itu kita juga bisa menggabung penulisan ketiga property ini ke dalam satu property `text-decoration`, atau dikenal dengan *shorthand property* (penulisan singkat).

Kode CSS pada contoh diatas akan menghasilkan efek yang sama jika ditulis sebagai berikut:

```

1  .satu {
2      text-decoration: underline solid green;
3  }
4  .dua {
5      text-decoration: overline dashed #FF0000;
6  }
7  .tiga {
8      text-decoration: line-through wavy hsla(240,100%,50%,0.9);
9  }

```

Dalam spesifikasi CSS dari W3C, terdapat nilai `text-decoration:blink` yang bisa membuat teks berkedip, mirip dengan efek tag `<blink>`. Namun efek ini sudah *deprecated* dan tidak didukung lagi oleh kebanyakan web browser modern.

7.8. Property `text-transform`

Terkadang konten sebuah website bisa berasal dari sumber lain seperti kolom komentar, database, dari pengunjung, anggota forum, dll. Agar tampilan web tetap seragam, kita perlu mengatur penggunaan huruf besar/kecil pada bagian konten ini.

Sebagai contoh, judul artikel akan tampil pas jika semuanya ditulis dengan huruf besar. Atau kita bisa mengatur agar semua komentar harus tampil dalam huruf kecil, meskipun saat di input menggunakan huruf besar semua.

Untuk kasus seperti ini, CSS menyediakan property **`text-transform`**. Property `text-transform` bisa diisi dengan salah satu keyword: `capitalize`, `uppercase`, `lowercase`, atau `none`.

Hasil dari `text-transform:capitalize` akan membuat huruf pertama dari setiap kata menjadi huruf besar. Sedangkan `text-transform:uppercase` akan membuat seluruh teks menjadi huruf besar. Lalu `text-transform:lowercase` membuat seluruh teks menjadi huruf kecil, serta `text-transform:none` berguna untuk menghapus efek `text-transform` yang sudah ada sebelumnya.

Berikut contoh penggunaan property `text-transform`:

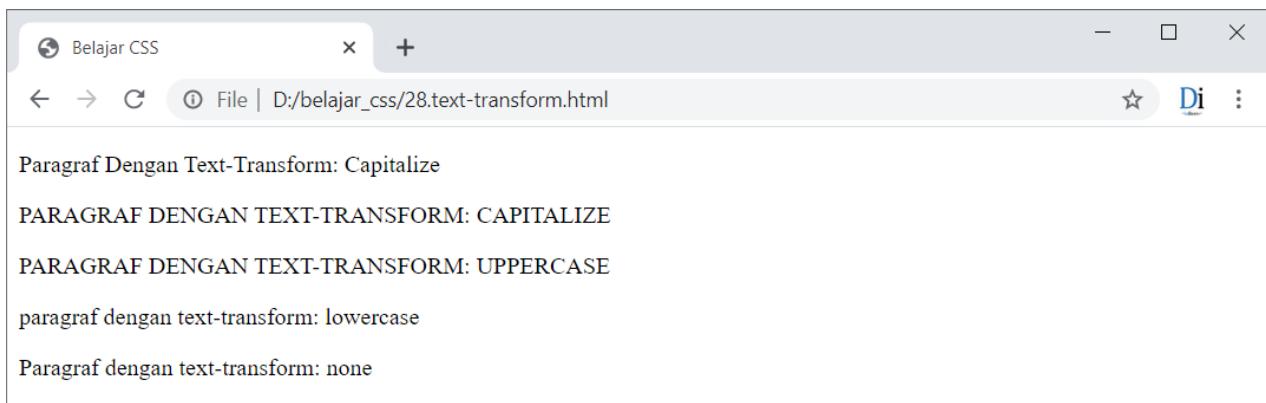
28.text-transform.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>

```

```
6  <style>
7      .satu { text-transform: capitalize; }
8      .dua { text-transform: uppercase; }
9      .tiga { text-transform: lowercase; }
10     .lima { text-transform: none; }
11  </style>
12 </head>
13 <body>
14  <p class="satu">Paragraf dengan text-transform: capitalize</p>
15  <p class="satu">PARAGRAF DENGAN TEXT-TRANSFORM: CAPITALIZE</p>
16  <p class="dua">PARAGRAF DENGAN text-transform: uppercase</p>
17  <p class="tiga">PARAGRAF DENGAN text-transform: lowercase</p>
18  <p class="lima">Paragraf dengan text-transform: none</p>
19 </body>
20 </html>
```



Gambar: Contoh penggunaan property text-transform

Paragraf pertama dan kedua menggunakan `text-transform:capitalize`, perhatikan bahwa property ini hanya peduli dengan huruf pertama dari setiap kata. Kemudian pada paragraf kedua, seluruh paragraf tetap tampil dalam huruf besar.

Untuk `text-transform:uppercase` dan `text-transform:lowercase` seluruh teks akan ditampilkan dengan huruf besar dan huruf kecil.

Paragraf terakhir yang memiliki property `text-transform:none` yang tidak berpengaruh apa-apa karena paragraf ini memang belum di set dengan `text-transform` apapun.

7.9. Property font-variant

Property **font-variant** cukup unik. Property ini dipakai untuk membuat variasi font, tetapi nilai yang didukung (saat ini) hanya 2: `small-caps` dan `normal`.

Ketika kita menggunakan `font-variant:small-caps`, teks akan diubah ke huruf kapital namun dengan ukuran yang lebih kecil. Sedangkan `font-variant:normal` berfungsi untuk menghapus efek font-variant sebelumnya, atau ini dipakai untuk mengembalikan teks ke bentuk normal.

Contoh berikut bisa menjelaskan maksud dari property `font-variant: small-caps`:

29.font-variant.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu { font-variant: small-caps; }
8      </style>
9  </head>
10 <body>
11     <h1>Judul normal, tanpa property apapun</h1>
12     <h1 class="satu">Paragraf dengan font-variant: small-caps</h1>
13 </body>
14 </html>
```



Gambar: Contoh penggunaan property font-variant

Seluruh text pada tag `<h1>` dengan `font-variant:small-caps` akan tampil dengan huruf kapital, tetapi jika karakter asal ditulis dalam huruf kecil, teks yang dihasilkan juga sedikit lebih pendek (perhatikan perbedaan antara huruf "P" dengan huruf "A" di awal judul `<h1>`).

Efek property `font-variant:small-caps` bisa dipakai untuk judul teks, atau bagian konten lain yang ingin ditampilkan berbeda.

7.10. Property text-align

Untuk teks yang panjang, kadang kita perlu mengatur rata teks apakah itu rata kiri, rata tengah, rata kanan, atau rata kiri kanan. CSS menyediakan fitur ini melalui property **text-align**. Nilai yang didukung adalah salah satu dari keyword: `left`, `center`, `right` dan `justify`.

Berikut contoh penggunaan property `text-align` di dalam CSS:

30.text-align.html

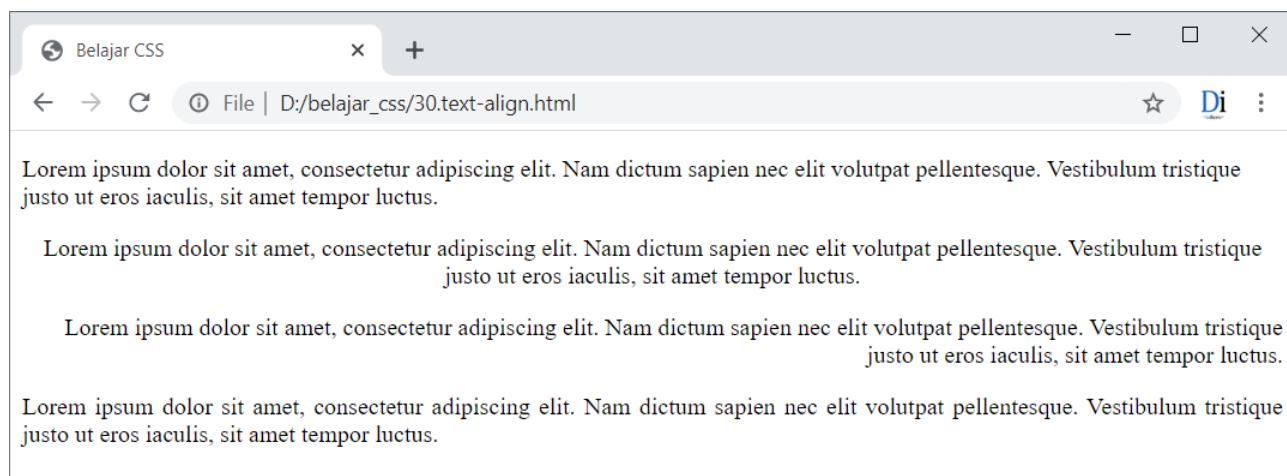
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu { text-align: left; }
```

```

8      .dua    { text-align: center; }
9      .tiga   { text-align: right; }
10     .empat  { text-align: justify; }
11  </style>
12 </head>
13 <body>
14 <p class="satu">
15 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
16 Nam dictum sapien nec elit volutpat pellentesque.
17 Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
18 </p>
19 <p class="dua">
20 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
21 Nam dictum sapien nec elit volutpat pellentesque.
22 Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
23 </p>
24 <p class="tiga">
25 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
26 Nam dictum sapien nec elit volutpat pellentesque.
27 Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
28 </p>
29 <p class="empat">
30 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
31 Nam dictum sapien nec elit volutpat pellentesque.
32 Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
33 </p>
34 </body>
35 </html>

```



Gambar: Contoh penggunaan property text-align

Selain untuk mengatur text, property `text-align` juga bisa dipakai untuk mengatur konten lain seperti gambar, atau lebih tepatnya semua element HTML yang termasuk *inline level element*.

Rata kiri vs justify

Dengan alasan agar tampak lebih rapi, kebanyakan dari kita mungkin ingin memakai

`text-align:justify` untuk seluruh konten web. Akan tetapi dari sisi user experience, teks dengan rata tengah dan kiri ini kadang lebih susah dibaca, terutama di media elektronik seperti web.

Agar sebuah paragraf bisa tampil rata tengah dan kiri sekaligus, web browser akan menyesuaikan jarak spasi antar kata. Akibatnya, teks menjadi renggang pada bagian tertentu dan rapat pada bagian lain.

Karena alasan ini pula mayoritas web tidak menggunakan paragraf `justify`, tapi tetap rata kiri. Sebagai pembuktian bisa cek web-web besar seperti kompas, detik, atau tirto. Meskipun tampak tidak rapi, tapi teks dengan rata kiri lebih mudah dibaca daripada teks `justify` (rata kiri dan kanan).

7.11. Property line-height

Salah satu faktor terpenting ketika mendesain web adalah mengatur konten agar mudah dibaca. Selain menggunakan font yang sesuai, kita juga perlu mempertimbangkan jarak antar baris supaya tidak terlalu rapat. Dalam istilah sehari-hari, ini biasa disebut dengan 'spasi paragraf' atau *line spacing*.

Untuk mengatur line spacing, CSS menyediakan property `line-height`. Nilai dari property ini bisa diisi dengan berbagai satuan length seperti pixel, em, persen, bahkan tanpa satuan (hanya angka saja).

Sebagai contoh, jika saya menggunakan property `line-height:140%`, web browser akan mengatur tinggi baris 140% dari ukuran text saat ini (diambil dari nilai property `font-size`). Jika ditulis `line-height:1.2em`, maka tinggi baris akan di-set 1,2 kali ukuran text saat ini.

Selain itu, `line-height` bisa diisi dengan nilai angka tanpa satuan apapun seperti `line-height:1.9`. Ini berarti tinggi teks akan di-set sebesar 1,9 kali ukuran font saat ini. Property `line-height` adalah salah satu dari sedikit property CSS yang bisa diisi dengan nilai tanpa satuan.

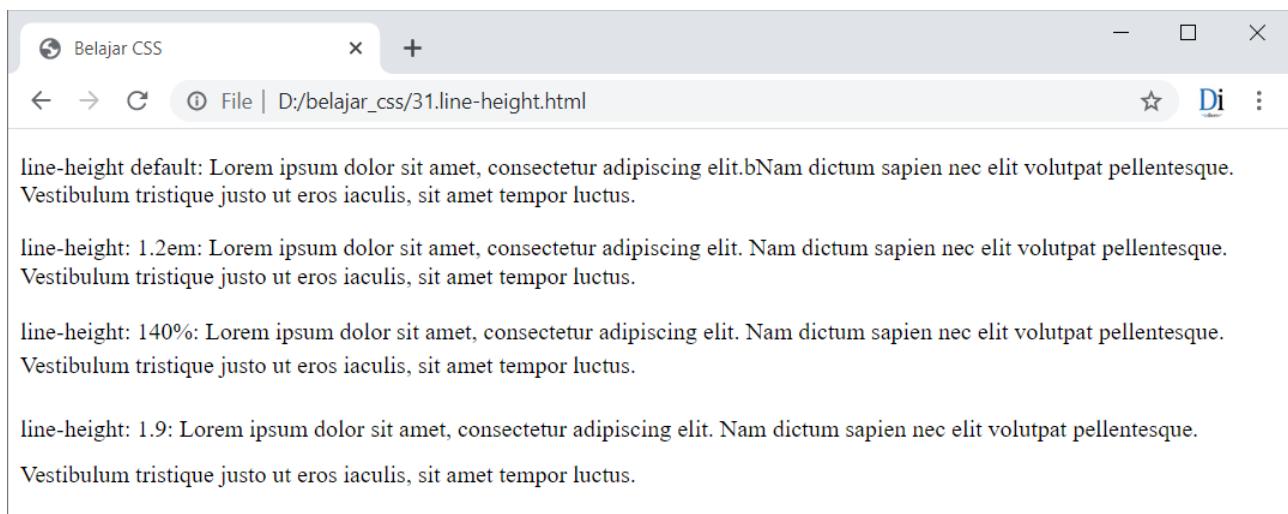
Berikut contoh tampilan dari berbagai nilai `line-height`:

31.line-height.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu { line-height: 1.2em; }
8          .dua { line-height: 140%; }
9          .tiga { line-height: 1.9; }
10     </style>
```

```
11 </head>
12 <body>
13   <p>
14     line-height default: Lorem ipsum dolor sit amet, consectetur adipiscing
15     elit. Nam dictum sapien nec elit volutpat pellentesque.
16     Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
17   </p>
18   <p class="satu">
19     line-height: 1.2em: Lorem ipsum dolor sit amet, consectetur adipiscing
20     elit. Nam dictum sapien nec elit volutpat pellentesque.
21     Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
22   </p>
23   <p class="dua">
24     line-height: 140%: Lorem ipsum dolor sit amet, consectetur adipiscing
25     elit. Nam dictum sapien nec elit volutpat pellentesque.
26     Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
27   </p>
28   <p class="tiga">
29     line-height: 1.9: Lorem ipsum dolor sit amet, consectetur adipiscing
30     elit. Nam dictum sapien nec elit volutpat pellentesque.
31     Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
32   </p>
33 </body>
34 </html>
```



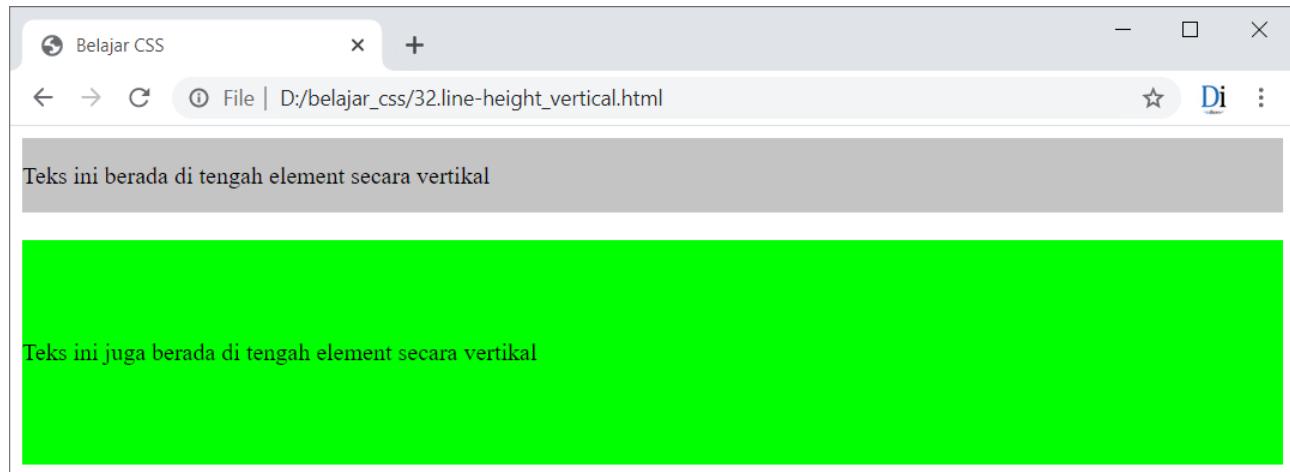
Gambar: Contoh penggunaan property line-height

Nilai default `line-height` bawaan web browser adalah sekitar 1.2em atau 1,2 kali ukuran teks saat ini. Kita bisa menggunakan property `line-height: normal` untuk mengembalikan `line-height` ke nilai default ini.

Salah satu trik dimana property `line-height` sering dipakai adalah untuk memposisikan text tepat di tengah-tengah element secara vertikal. Sebagai contoh, property `line-height: 50px` akan membuat teks tepat berada di tengah konten yang di-set setinggi 50 pixel. Syarat untuk trik ini, teks tersebut harus terdiri dari 1 baris saja. Berikut contoh penggunaannya:

32.line-height_vertical.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div.satu {
8              background-color: silver;
9              line-height: 50px;
10         }
11         div.dua {
12             background-color: lime;
13             line-height: 150px;
14         }
15     </style>
16 </head>
17 <body>
18     <div class="satu">
19         Teks ini berada di tengah element secara vertikal
20     </div>
21     <br>
22     <div class="dua">
23         Teks ini juga berada di tengah element secara vertikal
24     </div>
25 </body>
26 </html>
```



Gambar: Trik untuk mengetengahkan element dengan property line-height

Trik ini sangat sering dipakai, terutama sebelum CSS3 menghadirkan konsep **Flexbox** (akan kita bahas pada bab tersendiri).

7.12. Property margin-bottom

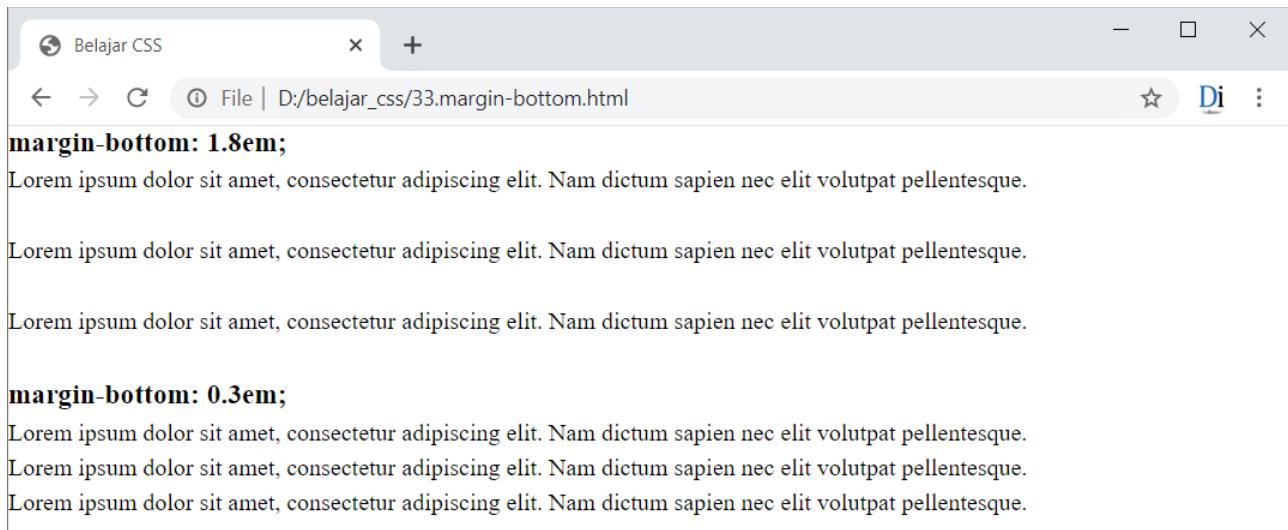
Property line-height yang kita bahas sebelumnya dipakai untuk mengatur jarak (spasi) antar baris dalam suatu paragraf. Untuk mengatur jarak (spasi) antar paragraf yang satu dengan

yang lainnya (dan juga dengan element lain) bisa menggunakan property `margin-bottom`. Nilai yang bisa diisi adalah semua satuan length seperti px, em, hingga %.

Berikut contoh penggunaan property `margin-bottom` untuk mengatur jarak antar paragraf:

33.margin-bottom.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * { margin: 0; }
8          h3 { margin-bottom: 5px; }
9          .satu { margin-bottom: 1.8em; }
10         .dua { margin-bottom: 0.3em; }
11     </style>
12 </head>
13 <body>
14     <h3>margin-bottom: 1.8em;</h3>
15     <p class="satu">
16         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
17         Nam dictum sapien nec elit volutpat pellentesque.
18     </p>
19     <p class="satu">
20         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
21         Nam dictum sapien nec elit volutpat pellentesque.
22     </p>
23     <p class="satu">
24         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
25         Nam dictum sapien nec elit volutpat pellentesque.
26     </p>
27     <h3>margin-bottom: 0.3em;</h3>
28     <p class="dua">
29         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
30         Nam dictum sapien nec elit volutpat pellentesque.
31     </p>
32     <p class="dua">
33         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
34         Nam dictum sapien nec elit volutpat pellentesque.
35     </p>
36     <p class="dua">
37         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
38         Nam dictum sapien nec elit volutpat pellentesque.
39     </p>
40 </body>
41 </html>
```



Gambar: Contoh penggunaan property margin-bottom untuk mengatur spasi antar element

Selector CSS pertama, `* { margin: 0; }` berfungsi untuk menghapus margin bawaan web browser untuk seluruh element. Mengenai konsep ini akan di bahas ketika kita mempelajari CSS Reset. Kode ini saya tambah agar efek property `margin-bottom` menjadi lebih jelas.

Pada selector CSS kedua, saya men-set `margin-bottom:5px` untuk tag `<h3>`. Dengan demikian, jarak antara tag `<h3>` dengan element lain di bawahnya adalah sebesar 5 pixel.

Selector CSS ketiga dan keempat dipakai untuk men-set `margin-bottom` sebesar `1.8em` dan `0.3em`. Hasilnya, kedua kelompok paragraf memiliki jarak yang berbeda-beda.

Property `margin-bottom` sebenarnya bagian dari konsep Box Model. Ini akan kembali kita bahas dalam bab tersendiri.

7.13. Property letter-spacing dan word-spacing

Property `letter-spacing` dan `word-spacing` digunakan untuk mengatur jarak spasi antar huruf (letter) atau antar kata (word). Kedua property ini bisa di isi satuan px, em atau rem, dan juga bisa bernilai negatif. Berikut contoh penggunaan keduanya:

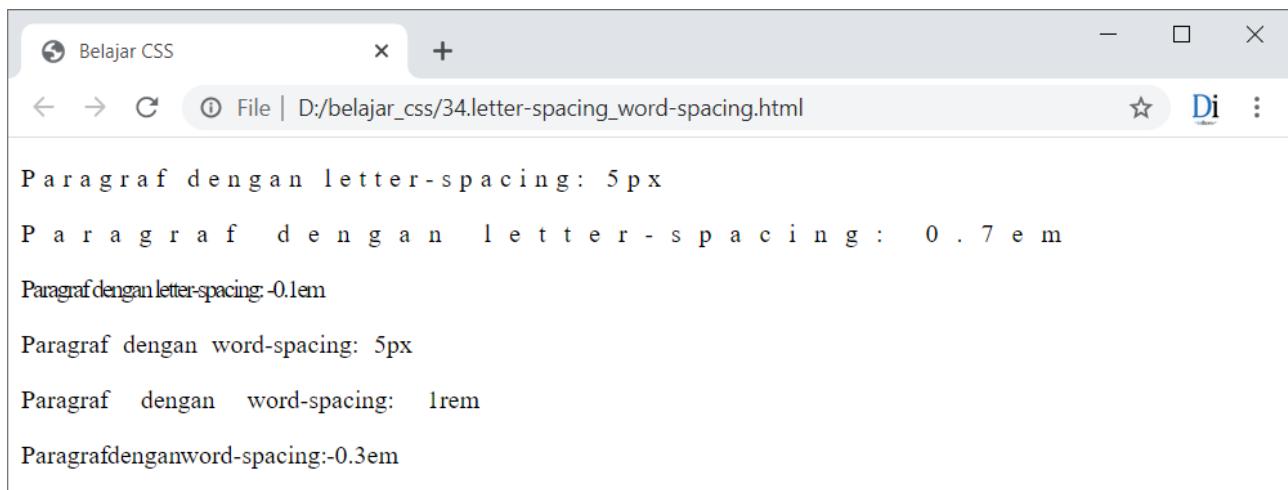
34.letter-spacing_word-spacing.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu { letter-spacing: 5px; }
8          .dua { letter-spacing: 0.7em; }
9          .tiga { letter-spacing: -0.1em; }
10         .empat { word-spacing: 5px; }
```

```

11     .lima { word-spacing: 1rem; }
12     .enam { word-spacing: -0.3em; }
13   </style>
14 </head>
15 <body>
16   <p class="satu">Paragraf dengan letter-spacing: 5px</p>
17   <p class="dua">Paragraf dengan letter-spacing: 0.7em</p>
18   <p class="tiga">Paragraf dengan letter-spacing: -0.1em</p>
19   <p class="empat">Paragraf dengan word-spacing: 5px</p>
20   <p class="lima">Paragraf dengan word-spacing: 1rem</p>
21   <p class="enam">Paragraf dengan word-spacing: -0.3em</p>
22 </body>
23 </html>

```



Gambar: Contoh penggunaan property letter-spacing dan word-spacing

Beberapa referensi menyarankan penggunaan satuan relatif (em atau rem) untuk property **letter-spacing** dan **word-spacing** karena lebih fleksibel dengan ukuran font.

Sebagai contoh, jika saya menggunakan **letter-spacing: 0.7em**, maka jarak antar huruf akan di-set sebesar $0.7 * 16px = 11.2px$, dimana 16px adalah font-size default web browser. Ketika ukuran font diubah menggunakan property **font-size: 20px**, property **letter-spacing: 0.7em** akan menghasilkan $0.7 * 20px = 14px$.

Dapat terlihat bahwa dengan menggunakan satuan em, jarak spasi antar karakter tetap mempertahankan rasio font-size. Semakin besar ukuran font, semakin besar juga jarak antar kata/huruf.

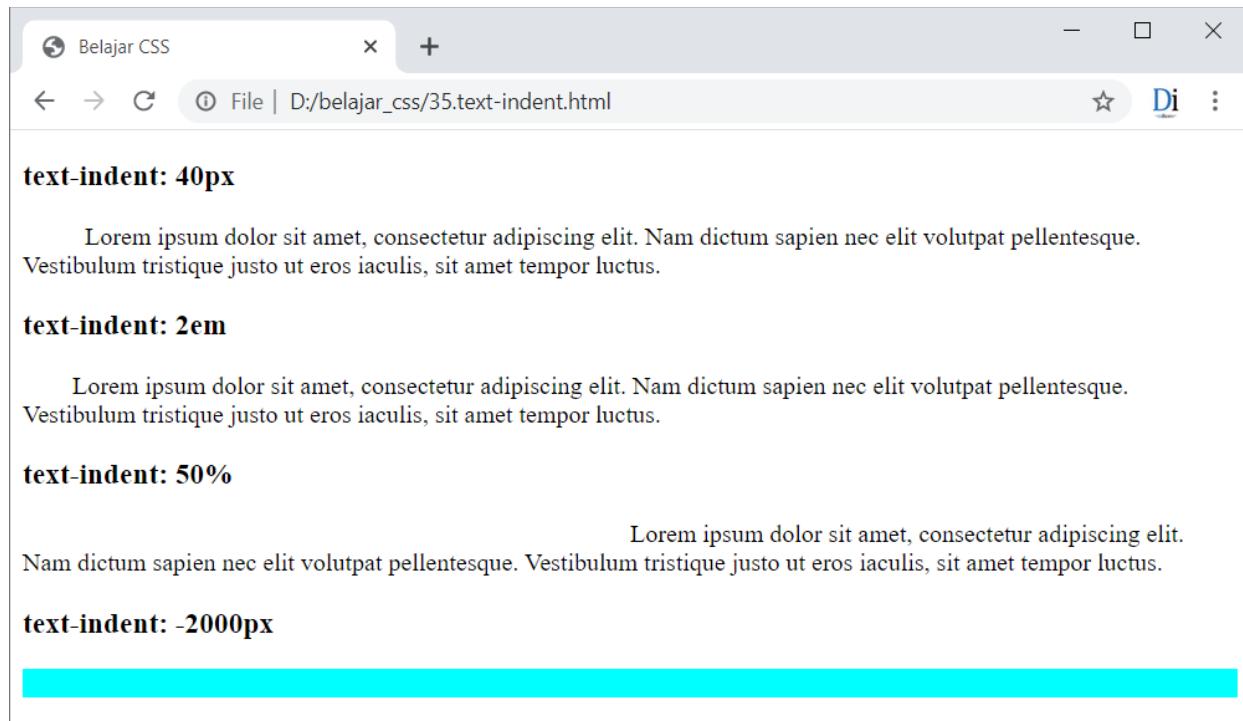
7.14. Property text-indent

Indentasi adalah sebutan untuk baris pertama paragraf yang menjorok ke arah dalam. Umumnya indentasi dipakai dalam tulisan formal seperti surat resmi. Di dalam CSS, pengaturan indentasi bisa dilakukan dengan property **text-indent**.

Property `text-indent` bisa diisi beragam satuan length seperti px, em, % atau rem. Berikut contoh penggunaannya:

35.text-indent.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu { text-indent: 40px; }
8          .dua { text-indent: 2em; }
9          .tiga { text-indent: 50%; }
10         .empat { text-indent: -2000px; background-color: aqua; }
11     </style>
12 </head>
13 <body>
14     <h3>text-indent: 40px</h3>
15     <p class="satu">
16         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
17         Nam dictum sapien nec elit volutpat pellentesque.
18         Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
19     </p>
20     <h3>text-indent: 2em</h3>
21     <p class="dua">
22         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
23         Nam dictum sapien nec elit volutpat pellentesque.
24         Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
25     </p>
26     <h3>text-indent: 50%</h3>
27     <p class="tiga">
28         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
29         Nam dictum sapien nec elit volutpat pellentesque.
30         Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
31     </p>
32     <h3>text-indent: -2000px</h3>
33     <p class="empat">
34         Sembunyikan Saya...
35         Lorem ipsum dolor sit amet, consectetur adipiscing elit.
36         Nam dictum sapien nec elit volutpat pellentesque.
37         Vestibulum tristique justo ut eros iaculis, sit amet tempor luctus.
38     </p>
39 </body>
40 </html>
```



Gambar: Contoh penggunaan property text-indent

Ketika menggunakan satuan px, nilai indentasi akan di set sebesar jumlah pixel. Untuk satuan em, nilai indentasi bergantung kepada ukuran font-size. Pada paragraf kedua saya menggunakan property `text-indent: 2em`, sehingga ukuran indentasi tampil sebesar $2 \times 16 = 32$ pixel. Nilai 16 pixel merupakan ukuran font-size default web browser.

Ketika menggunakan satuan %, nilai indentasi akan dihitung berdasarkan lebar element, bukan dari font-size. Sebagai contoh, di dalam paragraf ke-3 saya menggunakan property `text-indent: 50%`. Maka, indentasi akan di set sebesar 50% dari lebar element paragraf. Silahkan anda ubah lebar jendela web browser, maka panjang indentasinya juga akan ikut berubah.

Yang cukup unik (dan juga merupakan suatu "trik"), kita bisa memberi nilai negatif kepada `text-indent`. Pada paragraf keempat saya menggunakan nilai `-2000px`. Ini sama dengan memindahkan seluruh paragraf ke... ruangan sebelah (jika anda membaca buku ini di kamar).

Nilai sebesar `-2000px` akan menyembunyikan seluruh paragraf dari tampilan web browser. Namun ruang yang ditinggalkan masih ditempati. Saya sengaja menambah property `background-color:aqua` sebagai penanda bahwa paragraf ini masih ada.

Trik menyembunyikan teks seperti ini kadang memiliki peranan penting. Misalkan saya menggunakan gambar sebagai logo website. Pengunjung yang membuka website dengan aplikasi *screen reader* mungkin akan kesulitan untuk 'membaca' gambar.

Dengan menambah sebuah hidden teks (`text-indent:-2000px`) di judul website, saya menyediakan teks yang hanya bisa dibaca oleh screen reader, sedangkan pengunjung web yang menggunakan web browser biasa tetap melihat gambar logo website.

7.15. Font Shorthand Notation

Di dalam CSS, terdapat beberapa property yang dikenal sebagai *shorthand notation*.

Shorthand notation adalah sebutan untuk property khusus yang berisi gabungan berbagai property lain. Tujuannya sekedar menyingkat penulisan saja. Kali ini kita akan membahas cara penulisan font shorthand notation.

Property font bisa dipakai untuk menyingkat penulisan 6 property sekaligus, yakni:

- font-style
- font-variant
- font-weight
- font-size
- line-height
- font-family

Penulisan property font harus mengikuti urutan ini, namun tidak semuanya harus ditulis.

Berikut contoh cara penulisan property font:

```
/* size | family */
font: 2em "Open Sans", sans-serif;

/* style | size | family */
font: italic 2em "Open Sans", sans-serif;

/* style | variant | weight | size/line-height | family */
font: italic small-caps bolder 16px/2 cursive;
```

Cara penulisan paling sederhana adalah gabungan font-size dengan font-family seperti pada contoh pertama, yakni `font: 2em "Open Sans", sans-serif`. Sekurang-kurangnya, kedua property ini wajib ditulis untuk property font. Tanda spasi dipakai sebagai pemisah setiap nilai. Sedangkan tanda koma merupakan bagian dari nilai property font-family.

Cara penulisan selanjutnya adalah menambah satu atau dua dari property font-style, font-variant, atau font-weight di posisi awal.

Penulisan paling lengkap diperlihatkan pada contoh ketiga. Perhatikan cara penulisan property line-height, yang ditulis dengan tanda garis miring. Kode '16px/2' sama artinya dengan font-size: 16px dan line-height: 2.

Berikut contoh praktik dari font shorthand notation:

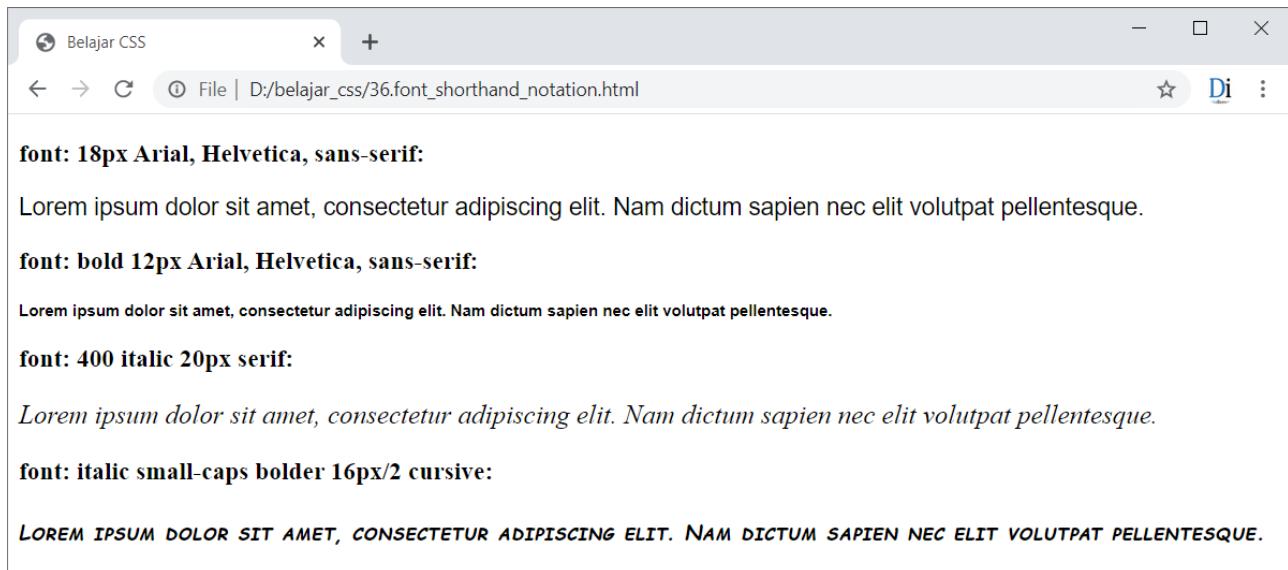
36. font_shorthand_notation.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
```

```

6  <style>
7      .satu { font: 18px Arial, Helvetica, sans-serif; }
8      .dua { font: bold 12px Arial, Helvetica, sans-serif; }
9      .tiga { font: 400 italic 20px serif; }
10     .empat { font: italic small-caps bolder 16px/2 cursive; }
11  </style>
12 </head>
13 <body>
14 <h3>font: 18px Arial, Helvetica, sans-serif:</h3>
15 <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
16 Nam dictum sapien nec elit volutpat pellentesque.</p>
17 <h3>font: bold 12px Arial, Helvetica, sans-serif:</h3>
18 <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
19 Nam dictum sapien nec elit volutpat pellentesque.</p>
20 <h3>font: 400 italic 20px serif:</h3>
21 <p class="tiga">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
22 Nam dictum sapien nec elit volutpat pellentesque.</p>
23 <h3>font: italic small-caps bolder 16px/2 cursive:</h3>
24 <p class="empat">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
25 Nam dictum sapien nec elit volutpat pellentesque.</p>
26 </body>
27 </html>

```



Gambar: Contoh penggunaan property font shorthand notation

Sebelum anda mulai menggunakan property font ini, terdapat sebuah 'jebakan' yang umum terjadi pada property *shorthand notation* (bukan hanya untuk font saja). Untuk nilai property yang tidak kita tulis, bukan berarti property itu kosong. Berikut contoh kasusnya:

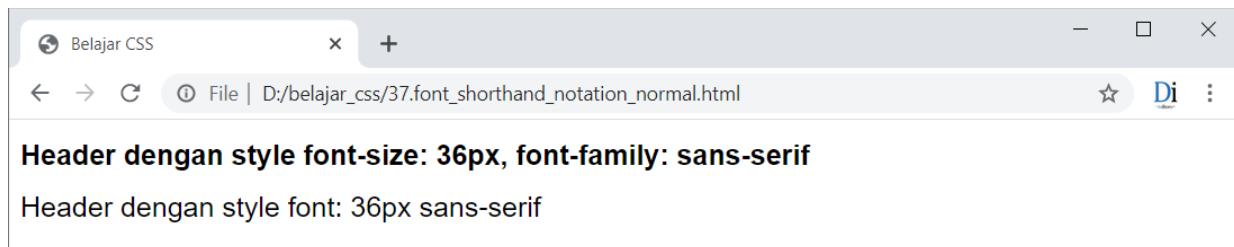
37.font_shorthand_notation_normal.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>

```

```
6  <style>
7      .satu { font-size: 20px; font-family: sans-serif; }
8      .dua { font: 20px sans-serif; }
9  </style>
10 </head>
11 <body>
12     <h1 class="satu">
13         Header dengan style font-size: 36px, font-family: sans-serif
14     </h1>
15     <h1 class="dua">
16         Header dengan style font: 36px sans-serif
17     </h1>
18 </body>
19 </html>
```



Gambar: Contoh penggunaan property font yang menghapus efek bold

Dapatkah anda tebak kenapa pada `class="dua"`, tag `<h1>` tidak tampil dengan huruf tebal? Padahal property yang kita set hanyalah `font-size` dan `font-family`.

Ini terjadi karena property `font` secara diam-diam men-set ke-4 property lainnya sebagai `normal`. Dengan kata lain, property:

`font: 20px sans-serif`

Dijalankan oleh web browser sebagai:

`font: normal normal normal 20px/normal sans-serif.`

Ini akan men-set property `font-style`, `font-variant`, `font-weight`, dan `line-height` ke nilai `normal`. Untuk property `font-weight`, nilai `normal` akan menghapus efek huruf tebal (`bold`) bawaan tag `<h1>`.

Jebakan ini bukan berarti bahwa kita tidak boleh menggunakan property `font`, namun hanya sebagai pemberitahuan supaya berhati-hati karena property ini membuka peluang hal yang tak terduga seperti kasus di atas. Untuk menghindarinya, maka lebih disarankan menulis property `font` secara terpisah (tidak menggunakan penulisan `shorthand`).

Contoh dari property `shorthand` lain adalah `background`, `border`, `margin` dan `padding`.

7.16. Property text-shadow

Membuat efek bayangan text (`text-shadow`) merupakan salah satu style CSS3 paling menarik. Melalui property `text-shadow`, kita bisa membuat berbagai efek visual yang sebelumnya hanya bisa didapat dengan bantuan aplikasi pengolah gambar seperti Photoshop.

Untuk membuat efek bayangan ini, CSS menyediakan 4 pilihan pengaturan: `offset-x`, `offset-y`, `blur-radius` dan `color`.

Nilai minimal yang harus disertakan pada property `text-shadow` adalah `offset-x` dan `offset-y`. Nilai `offset-x` berfungsi untuk mengatur seberapa jauh bayangan yang dihasilkan dari text asli ke arah sumbu x (horizontal), sedangkan `offset-y` ke arah sumbu y (vertical). Satuan yang bisa diinput berupa nilai length seperti px, em, atau rem. Nilainya juga bisa positif atau negatif, dan ini akan menentukan arah bayangan.

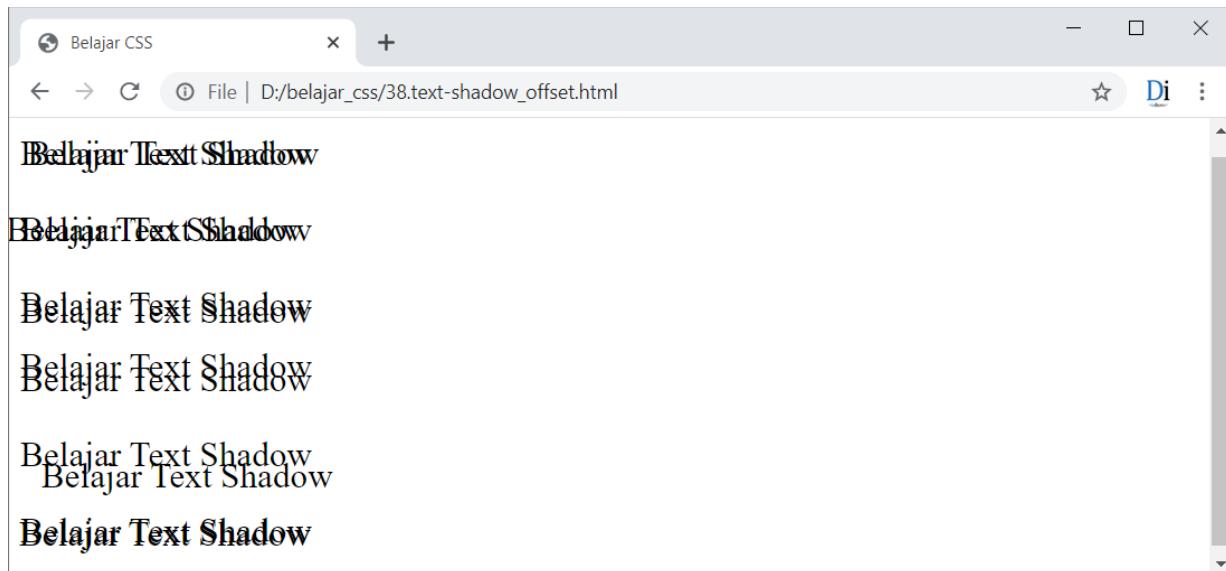
Sebagai contoh, `text-shadow: 2px 2px` akan menghasilkan bayangan sejauh 2 pixel ke arah kanan, dan 2 pixel ke arah bawah. Sedangkan `text-shadow: -10px -15px` akan membuat bayangan sejauh 10 pixel ke arah kiri dan 15 pixel ke arah atas.

Berikut contoh penggunaan berbagai settingan nilai `offset-x` dan `offset-y` pada `text-shadow`:

38.text-shadow_offset.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p {
8              font-size: 25px;
9              font-family: serif;
10         }
11         .satu { text-shadow: 5px 0px; }
12         .dua { text-shadow: -10px 0px; }
13         .tiga { text-shadow: 0px 5px; }
14         .empat { text-shadow: 0px -10px; }
15         .lima { text-shadow: 15px 15px; }
16         .enam { text-shadow: -1px 2px; }
17     </style>
18 </head>
19 <body>
20     <p class="satu">Belajar Text Shadow</p>
21     <p class="dua">Belajar Text Shadow</p>
22     <p class="tiga">Belajar Text Shadow</p>
23     <p class="empat">Belajar Text Shadow</p>
24     <p class="lima">Belajar Text Shadow</p>
25     <p class="enam">Belajar Text Shadow</p>
26 </body>
27 </html>
```



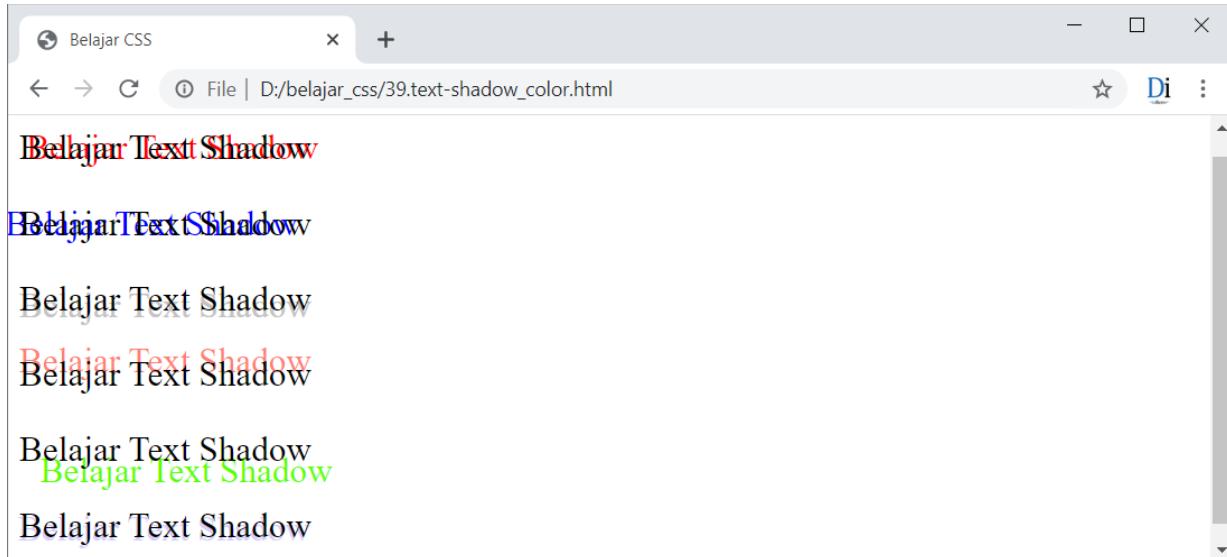
Gambar: Contoh penggunaan property text-shadow dengan berbagai nilai offset-x dan offset-y

Bayangan pada contoh ini menggunakan warna yang sama dengan warna teks. Bagaimana cara menukar warna ini? cukup dengan menambahkan nilai warna seperti contoh berikut:

39.text_shadow_color.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p {
8              font-size: 25px;
9              font-family: serif;
10         }
11         .satu { text-shadow: 5px 0px red; }
12         .dua { text-shadow: blue -10px 0px; }
13         .tiga { text-shadow: 0px 5px #C0C0C0; }
14         .empat { text-shadow: rgba(255,88,75,0.8) 0px -10px ; }
15         .lima { text-shadow: 15px 15px hsl(100,100%,50%); }
16         .enam { text-shadow: -1px 2px hsla(250,100%,80%,0.5); }
17     </style>
18 </head>
19 <body>
20     <p class="satu">Belajar Text Shadow</p>
21     <p class="dua">Belajar Text Shadow</p>
22     <p class="tiga">Belajar Text Shadow</p>
23     <p class="empat">Belajar Text Shadow</p>
24     <p class="lima">Belajar Text Shadow</p>
25     <p class="enam">Belajar Text Shadow</p>
26 </body>
27 </html>
```



Gambar: Contoh penggunaan property text-shadow dengan berbagai variasi warna

Perhatikan cara penambahan nilai warna, kita bisa meletakkannya di bagian depan (sebelum penulisan nilai offset), atau di bagian belakang (setelah penulisan nilai offset). Semua nilai warna bisa digunakan, termasuk keyword, RGB, HSL, RGBA maupun HSLA.

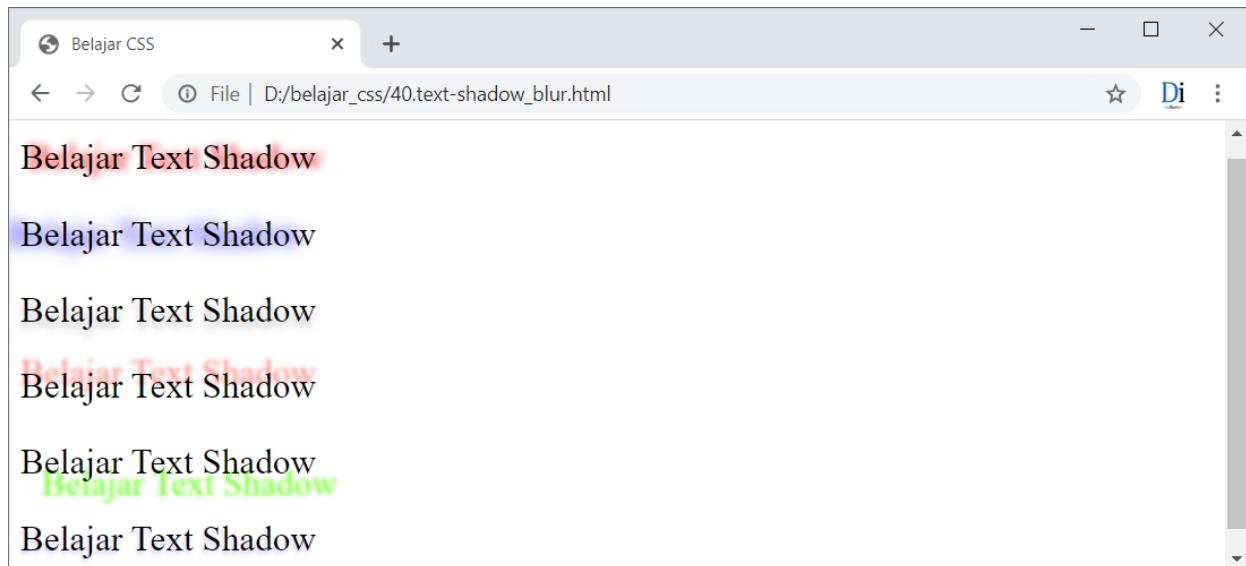
Efek terakhir yang bisa ditambah adalah *blur*. Satuannya bisa diisi dengan nilai length seperti px atau em. Nilai blur harus ditulis setelah penulisan offset seperti contoh berikut:

40.text-shadow_blur.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p {
8              font-size: 25px;
9              font-family: serif;
10         }
11         .satu { text-shadow: 5px 0px 5px red; }
12         .dua { text-shadow: blue -10px 0px 10px; }
13         .tiga { text-shadow: 0px 5px 0.2em #C0C0C0; }
14         .empat { text-shadow: rgba(255,88,75,0.8) 0px -10px 3px ; }
15         .lima { text-shadow: 15px 15px 3px hsl(100,100%,50%); }
16         .enam { text-shadow: -1px 2px 3px hsla(250,100%,80%,0.5); }
17     </style>
18 </head>
19 <body>
20     <p class="satu">Belajar Text Shadow</p>
21     <p class="dua">Belajar Text Shadow</p>
22     <p class="tiga">Belajar Text Shadow</p>
23     <p class="empat">Belajar Text Shadow</p>
24     <p class="lima">Belajar Text Shadow</p>
25     <p class="enam">Belajar Text Shadow</p>
```

```
26 </body>
27 </html>
```



Gambar: Berbagai efek blur dari property text-shadow

Efek yang dihasilkan dari nilai blur ini sangat menarik. Silahkan berkreasi dengan mencoba kombinasi efek blur dan warna lain.

Namun belum cukup dengan 1 efek, kita juga bisa menambahkan 2, 3 atau lebih efek bayangan ke dalam satu property `text-shadow`. Penulisan setiap efek dipisah dengan tanda koma seperti contoh berikut:

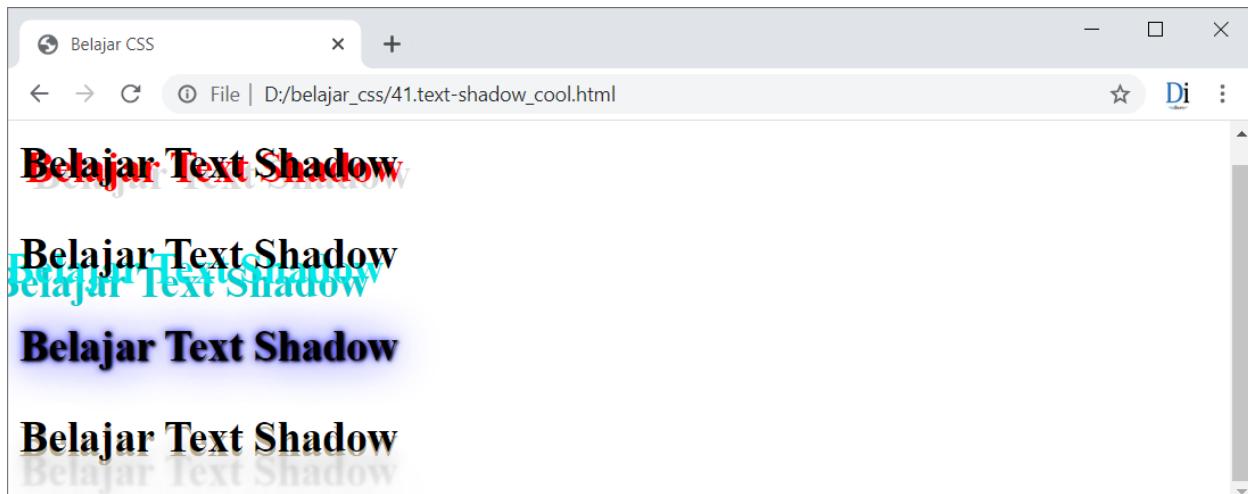
41.text-shadow_cool.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p {
8              font-size: 30px;
9              font-family: serif;
10             font-weight: bold;
11         }
12         .satu {
13             text-shadow: 4px 3px 0px red,
14                         9px 8px 0px rgba(0,0,0,0.15);
15         }
16         .dua {
17             text-shadow: -10px 10px 0px #00e6e6,
18                         -20px 20px 0px #01cccc;
19         }
20         .tiga {
21             text-shadow: 1px 1px 2px black,
22                         0 0 1em blue,
```

```

23          0 0 0.2em blue;
24      }
25  .empat {
26      text-shadow: 0px 3px 0px #b2a98f,
27                  0px 14px 10px rgba(0,0,0,0.15),
28                  0px 24px 2px rgba(0,0,0,0.1),
29                  0px 34px 30px rgba(0,0,0,0.1);
30  }
31  </style>
32 </head>
33 <body>
34  <p class="satu">Belajar Text Shadow</p>
35  <p class="dua">Belajar Text Shadow</p>
36  <p class="tiga">Belajar Text Shadow</p>
37  <p class="empat">Belajar Text Shadow</p>
38 </body>
39 </html>

```



Gambar: Berbagai efek menarik dengan penggunaan 2 buah text shadow atau lebih

7.17. Mengenal CSS Vendor Prefix

Vendor prefix adalah sebutan untuk penambahan beberapa karakter khusus di awal penulisan property, terutama untuk property CSS3 terbaru. Sebagai contoh, untuk property `text-shadow` yang kita bahas sebelum ini, jika menggunakan vendor prefix ditulis menjadi: `-webkit-text-shadow`.

Apa kegunaan dari *vendor prefix*?

Seperti yang kita bahas di bab-bab awal, CSS3 dikembangkan dengan konsep modul. Dengan demikian, property - property baru bisa hadir dengan lebih cepat. Untuk mengimbanginya, web browser modern juga berupaya mendukung property baru ini, bahkan ketika property tersebut belum resmi selesai (masih dalam tahap draft di W3C).

Property yang belum selesai ini diimplementasikan oleh web browser menggunakan vendor

prefix. Dengan tujuan, kita sebagai web desainer bisa menguji coba property tersebut. Penulisan vendor prefix sesuai dengan inisial web browser atau layout engine yang digunakan.

Berikut adalah awalan vendor prefix pada web browser:

- -webkit- (Chrome, dan versi terbaru dari Opera)
- -moz- (Firefox)
- -o- (Opera versi lama)
- -ms- (Internet Explorer)

Sebagai contoh, untuk property text-shadow, penambahan vendor prefix akan membuat penulisan property menjadi:

- -webkit-text-shadow
- -moz-text-shadow
- -o-text-shadow
- -ms-text-shadow

Pada awalnya, property dengan vendor prefix tidak ditujukan untuk website live, tapi hanya untuk uji coba. Dengan harapan ketika spesifikasi W3C masuk ke tahap rekomendasi yaitu ketika pengembangan property tersebut sudah dianggap selesai, tambahan vendor prefix juga akan dihapus.

Masalahnya, banyak web developer yang tidak sabar menunggu property ini resmi dirilis dan memilih untuk langsung menggunakannya. Oleh karena itu, jika kita ingin memakai property CSS3 yang relatif baru, harus ditulis dengan tambahan vendor prefix.

Sebagai contoh, untuk membuat efek text shadow, dulunya ditulis sebagai berikut:

```

1 .satu {
2   -webkit-text-shadow: 2px 2px 5px red;
3   -moz-text-shadow: 2px 2px 5px red;
4   -o-text-shadow: 2px 2px 5px red;
5   -ms-text-shadow: 2px 2px 5px red;
6   text-shadow: 2px 2px 5px red;
7 }
```

Walaupun terdiri dari 5 baris, property di atas hanya memiliki 1 tujuan: membuat bayangan dengan 2 pixel offset-x, 2 pixel offset-y, 5 pixel blur, dan berwarna merah.

Ketika sebuah web browser menjalankan kode ini, ia akan mulai dari baris paling atas (sesuai dengan konsep cascade CSS), kemudian secara berurutan ke bawah hingga menemukan property dengan vendor prefix yang bisa dipahami.

Sebagai contoh, ketika web browser Mozilla Firefox memproses kode CSS tersebut, ia akan mengabaikan property -webkit-text-shadow pada baris pertama, karena ini bukan vendor prefix milik Firefox.

Pada baris kedua, web browser akan membaca property `-moz-text-shadow` yang memang ditujukan untuk Firefox. Untuk property di baris ketiga dan keempat juga akan diabaikan.

Pada baris terakhir terdapat penulisan property resmi, yakni tanpa vendor prefix. Ini disiapkan agar ketika property tersebut sudah dinyatakan stabil (sudah didukung penuh), nilai dari property ini akan menimpa efek sebelumnya (yang ditulis dengan vendor prefix). Dengan demikian, hasil yang di dapat akan seragam pada setiap web browser.

Penambahan property dengan vendor prefix ini memang sedikit merepotkan. Kode CSS kita menjadi 5 kali lebih panjang, namun ini hanya digunakan untuk property CSS3 yang relatif baru. Jika property tersebut sudah resmi selesai, kita bisa membuang bagian vendor prefix dan menggunakan nama property resmi.

Property `text-shadow` yang kita praktekkan di atas tidak perlu lagi menggunakan vendor prefix karena sudah di dukung oleh mayoritas web browser.

Dalam beberapa kasus yang agak langka, nama property dan nilainya juga bisa berbeda antara satu web browser dengan web browser lain. Untuk yang seperti ini kita harus pelajari dokumentasi web browser tersebut.

7.18. Font External (@font-face)

Property `font-family` yang sudah di bahas pada awal bab membatasi penggunaan font hanya yang tersedia di komputer pengunjung saja (*web safe font*). Oleh karena itu pilihan kita terbatas kepada font-font umum seperti Arial, Times New Roman, atau Helvetica.

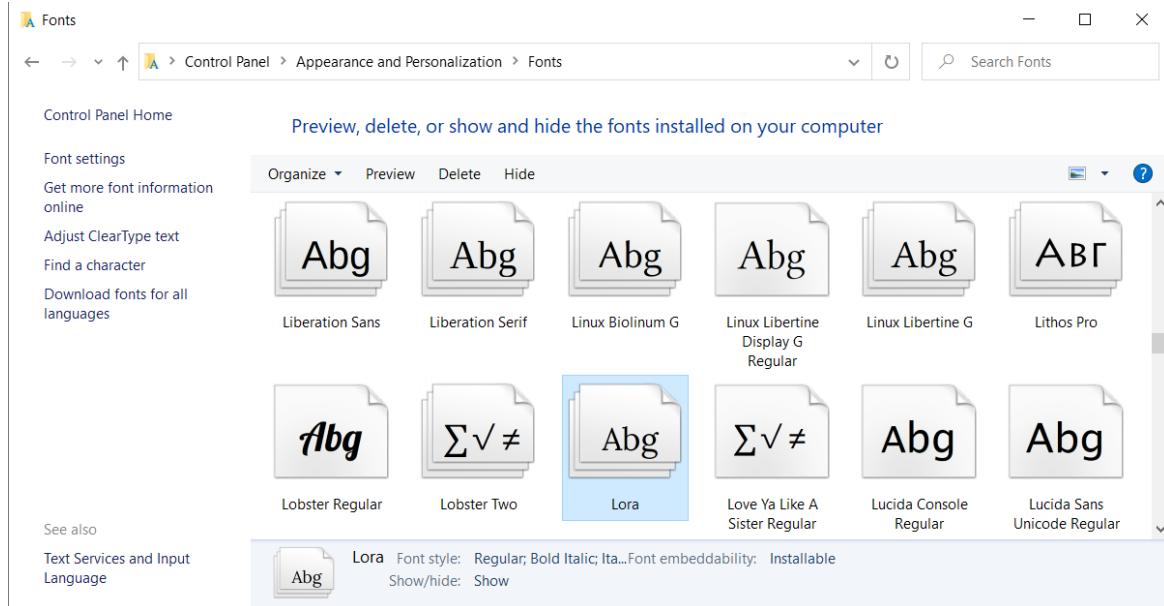
Kehadiran CSS3 serta tersedianya berbagai 'font berlisensi internet', membuka peluang penggunaan font external yang berasal dari server tempat website berada. Fitur ini sebenarnya sudah lama ada di CSS, bahkan sudah bisa digunakan di IE 5 sejak 12 tahun lalu. Namun karena keterbatasan dukungan format serta masalah lisensi, fitur ini jarang dipakai.

Lisensi font menjadi kendala tersendiri. Ketika sebuah website diakses, font external juga akan ikut ter-download ke komputer pengunjung. Oleh karena itu banyak font komersil (font berbayar) melarang penggunaannya untuk halaman web. Masalah ini bisa diatasi dengan menggunakan font khusus dengan lisensi internet.

Untuk mulai menggunakan external font, hal pertama yang harus kita lakukan adalah... mencari sebuah font sebagai bahan praktek.

Sebagai contoh, saya akan mengambil salah satu font yang ada di sistem Windows. Jika anda menggunakan Windows 10, folder font Windows berada di **Control Panel -> Appearance and Personalization -> Fonts**.

CSS Typography



Gambar: Folder fonts di Windows

Di dalam folder Fonts terdapat berbagai jenis font yang tersedia di OS Windows. Font-font ini umumnya berasal dari font bawaan Windows sendiri maupun dari aplikasi lain yang ikut menyertakan font bawaan seperti Photoshop.

Silahkan pilih salah satu yang cukup unik (bukan web safe font). Kali ini saya akan menggunakan font 'Lora'. Font ini saya tambahkan manual, sehingga mungkin tidak tersedia di Windows yang anda gunakan. File font-nya bernama: **LORA-REGULAR.TTF**.

File font ini juga tersedia di file **belajar_css.zip** yang ada di Google Drive.

Untuk bisa menggunakan font external, CSS menyediakan perintah **@font-face**. Kode CSS yang diawali dengan tanda '@' disebut juga dengan perintah '**at-rule**'. Perintah ini bukanlah sebuah property maupun selector, tetapi menjadi kelompok tersendiri.

Berikut format dasar penulisan perintah **@font-face** CSS:

```
@font-face {  
    font-family: FontName;  
    src: local('fontname'), url('/lokasi/filename.ttf') format('truetype');  
}
```

Sebagai contoh, menggunakan font 'Lora' dengan nama file **LORA-REGULAR.TTF**, kode yang diperlukan adalah sebagai berikut:

```
@font-face {  
    font-family: Loraku;  
    src: local('Lora'), url('LORA-REGULAR.TTF') format('truetype');  
}
```

Perhatikan setelah penulisan perintah `@font-face`, langsung diikuti dengan tanda kurung kurawal. Di dalam blok inilah seluruh kode CSS untuk menambah font external ditulis.

Kode `font-family` ini fungsinya berbeda dengan property `font-family` yang sudah kita pelajari. Kali ini, `font-family` berfungsi sebagai pemberi 'nama' atau identitas dari font yang akan diinput. Karena font 'Lora' cukup populer, saya menggunakan nama font 'Loraku' dengan tujuan agar font kita dianggap sebagai font baru (tidak terpengaruh cache web browser).

Kode kedua: `src` berfungsi untuk menulis alamat file font. Perintah `local('Lora')` berarti kita meminta web browser untuk mencari nama font 'Lora' di sistem komputer client. Jika tersedia, font inilah yang akan dipakai. Namun apabila tidak ditemukan, web browser akan mendownload file font yang ditulis pada bagian url.

Saya menulis `url('LORA-REGULAR.TTF')` karena file font di tempatkan dalam folder yang sama dengan file HTML. Jika anda meletakkannya di tempat lain, misalnya di dalam folder 'fonts', maka kodennya akan menjadi `url('fonts/LORA-REGULAR.TTF')`.

Bagian terakhir dari `url` adalah keterangan mengenai format font yang dipakai. Ini sepenuhnya opsional dan apabila tidak ditulis, web browser mencoba menebak format font dari extensionnya. Karena font yang saya gunakan berakhiran `.ttf`, maka format ini bernama 'truetype'. Selain format TrueType, masih terdapat format font lain seperti OpenType, Web Open File Format (WOFF), dan SVG.

Setelah deklarasi font external selesai, kita sudah bisa membuat konten dengan font tersebut. Ini dilakukan menggunakan property font family seperti berikut:

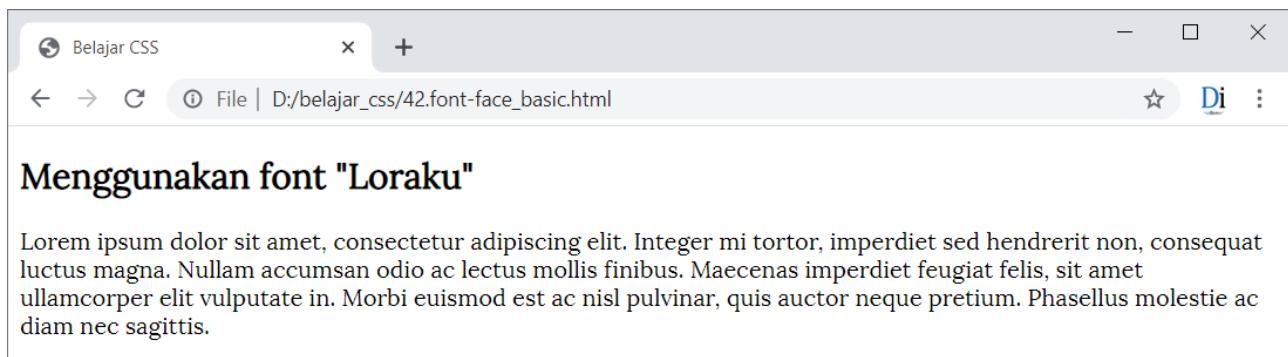
```
p {
    font-family: Loraku;
}
```

Kode ini akan membuat semua paragraf menggunakan font 'Loraku' yang baru saja definisikan. Mari lihat contoh penggunaannya di dalam kode HTML + CSS:

42.font-face_basic.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          @font-face {
8              font-family: Loraku;
9              src: local('Lora'), url('LORA-REGULAR.TTF') format('truetype');
10         }
11         h2, p {
12             font-family: "Loraku";
13         }
14     </style>
```

```
15 </head>
16 <body>
17   <h2>Menggunakan font "Loraku"</h2>
18   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
19     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
20     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
21     felis, sit amet ullamcorper elit vulputate in. Morbi euismod est ac
22     nisl pulvinar, quis auctor neque pretium. Phasellus molestie ac diam
23     nec sagittis.
24   </p>
25 </body>
26 </html>
```



Gambar: Tampilan penggunaan font external: 'Loraku'

Sekarang paragraf akan tampil dengan font yang berbeda, yakni font 'Loraku' yang sudah kita input sebelumnya. Silahkan anda coba memodifikasi kode di atas seperti menukar nama font, atau menggunakan file font lain.

Sedikit syarat tambahan, perintah @font-face harus ditulis pada bagian atas sebelum terdapat property font-family yang akan menggunakannya.

Penulisan kode local() di bagian url juga boleh tidak ditulis, sehingga perintah @font-face menjadi seperti berikut:

```
@font-face {
  font-family: Loraku;
  src: url('LORA-REGULAR.TTF') format('truetype');
}
```

Dengan perintah ini, web browser akan langsung memakai file font yang disediakan tanpa mencari font di komputer lokal.

Mengatasi Dukungan Format Font

Pada contoh sebelumnya, saya menggunakan font 'LORA-REGULAR.TTF' dengan format TrueType. Walaupun format ini cukup populer, beberapa web browser menolak menerima format font ini. Saat ini terdapat 5 format font yang umum digunakan:

EOT (Embedded Open Type)

EOT adalah format font yang hanya didukung oleh Internet Explorer, saat ini penggunaannya sudah relatif jarang. Inilah format font yang bisa dibaca oleh IE 8 ke bawah.

TTF (True Type) dan OTF (Open Type)

Jika anda membuka folder font di dalam komputer, sebagian besar diantaranya memiliki extensi .ttf (True Type) atau .otf (Open Type). Kedua format font ini merupakan format font yang paling banyak dipakai.

Format TTF dan OTF didukung oleh banyak web browser seperti: IE 9 ke atas, Firefox, Chrome, Safari, Opera, iOS Safari (versi 4.2 ke atas), Android, dan Blackberry Browser.

WOFF (Web Open Font Format)

Format WOFF merupakan format font yang relatif baru dan di desain secara khusus untuk keperluan web. WOFF pada dasarnya adalah versi kompresi dari format TTF dan OTF. WOFF memiliki ukuran lebih kecil dan akan di download dengan lebih cepat.

WOFF di dukung oleh web browser modern seperti: IE 9 ke atas, Firefox, Chrome, Safari, Opera, Blackberry browser, and iOS Safari versi 5 ke atas.

Saat ini terdapat juga variasi format WOFF2 dengan kompresi yang lebih baik.

SVG (Scalable Vector Graphic)

Format SVG bukanlah format khusus untuk font, melainkan format untuk menyimpan gambar vector (jenis gambar yang bisa di zoom tanpa merubah kualitasnya). Format font SVG tidak didukung oleh IE dan juga Firefox. Satu-satunya alasan menggunakan format SVG untuk font adalah pada iOS Safari versi 4.1 kebawah.

Seperti yang terlihat, beragam dukungan format font pada berbagai web browser menjadi masalah tersendiri. Untuk mengakali keterbatasan ini, pada awalnya web desainer terpaksa menggunakan 'hack' untuk perintah @font-face sebagai berikut:

```
@font-face {
    font-family: 'Loraku';
    src: url('lora-regular.eot');
    src: url('lora-regular.eot?#iefix') format('embedded-opentype'),
        url('lora-regular.woff2') format('woff2'),
        url('lora-regular.woff') format('woff'),
        url('lora-regular.ttf') format('truetype'),
        url('lora-regular.svg#loraregular') format('svg');
}
```

Kode ini mengakomodasi hampir semua web browser.

Urutan penulisan juga berpengaruh. Sesuai dengan konsep cascade CSS, web browser akan mencoba satu per satu perintah yang ada mulai dari yang paling atas. Format *.eot pada baris

pertama dan kedua ditujukan untuk IE 8 ke bawah. Web browser Mozilla, Chrome, dan Opera akan memilih format woff atau ttf. Sedangkan web browser iOS Safari versi lama akan mengambil format SVG.

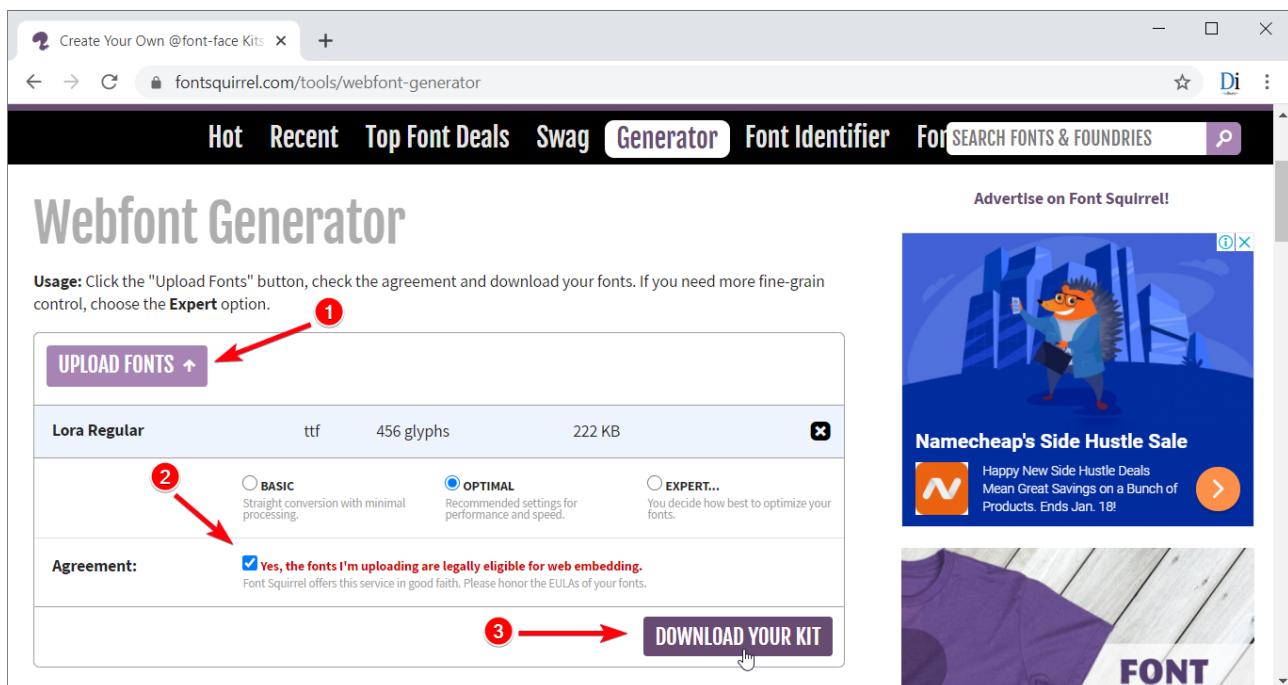
Seiring perkembangan teknologi web yang lebih matang, saat ini mayoritas web browser modern sudah mendukung format WOFF dan WOFF2. Sehingga kita bisa menulis kode di atas dengan lebih singkat:

```
@font-face {
    font-family: 'Loraku';
    src: url('lora-regular.woff2') format('woff2'),
         url('lora-regular.woff') format('woff');
}
```

Akan tetapi kode ini tidak akan berjalan di web browser yang relatif jadul seperti Internet Explorer atau iOS lama.

Masalah lain, tidak selamanya kita memiliki file font dalam format .woff dan .woff2. Untungnya tersedia tools yang bisa men-generate font dengan format ini, salah satunya adalah [Font Squirrel Webfont Generator](#)¹⁷.

Caranya, upload salah satu format font (1), lalu centang checkbox agreement (2) dan download hasil font yang sudah disatukan dalam file zip (3).

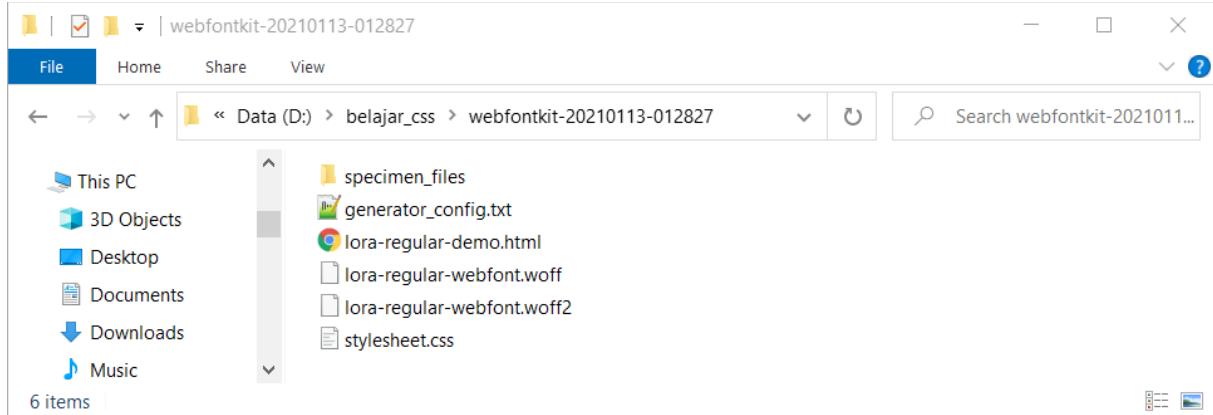


Gambar: Halaman Webfont Generator dari fontsquirrel.com

Sebagai contoh, saya akan mengupload font LORA-REGULAR.TTF dan memilih pengaturan default dari fonts quirrel (optimal), maka file hasil yang didapatkan adalah sebagai berikut:

¹⁷ <https://www.fontsquirrel.com/tools/webfont-generator>

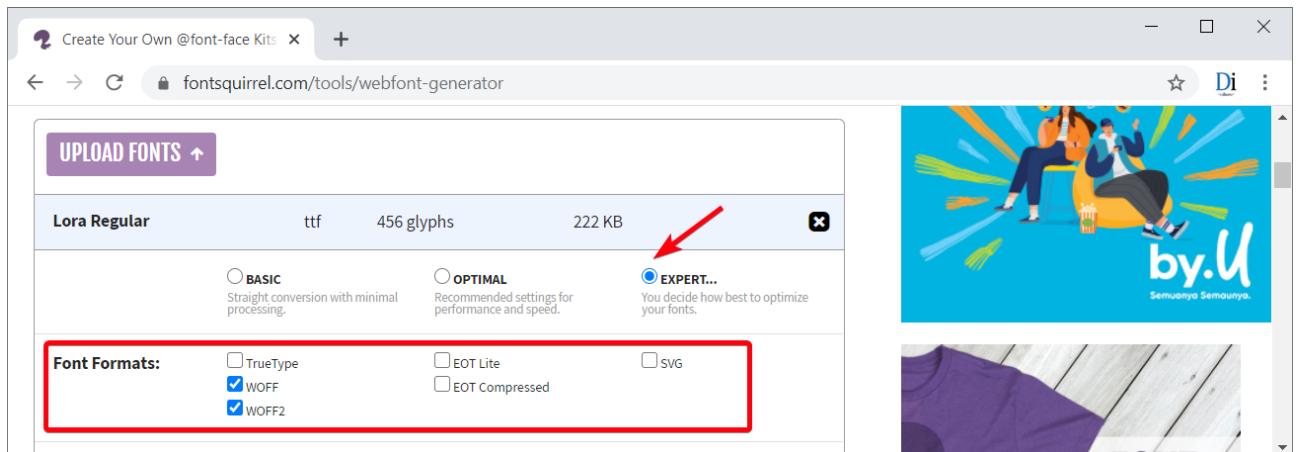
CSS Typography



Gambar: File hasil konversi font dari fontsquirrel.com

File font hasil konversi adalah `lora-regular-webfont.woff` dan `lora-regular-webfont.woff2`. Di dalam file `stylesheet.css` juga ikut disertakan kode CSS untuk penggunaan kedua format ini, yang kurang lebih sama seperti kode yang kita pakai sebelumnya.

Font Squirrel juga mendukung format font lain seperti EOT atau SVG, caranya pilih pengaturan **expert...** saat proses konversi:



Gambar: Pilihan menu expert... di Font Squirrel

File `lora-regular-webfont.woff` dan `lora-regular-webfont.woff2` hasil konversi akan saya simpan ke dalam folder fonts, dan berikut contoh penggunaannya:

43.font-face_woff.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          @font-face {
8              font-family: 'Loraku';
9              src: url('fonts/lora-regular-webfont.woff2') format('woff2'),
10                 url('fonts/lora-regular-webfont.woff') format('woff');
11      }
```

```

11      }
12      h2, p {
13          font-family: Loraku;
14      }
15  </style>
16 </head>
17 <body>
18  <h2>Menggunakan font "Loraku"</h2>
19  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
20 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
21 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
22 felis, sit amet ullamcorper elit vulputate in. Morbi euismod est ac
23 nisl pulvinar, quis auctor neque pretium. Phasellus molestie ac diam
24 nec sagittis.
25  </p>
26 </body>
27 </html>

```

Mengenal Berbagai Tipe Font

Selesai membahas format font, kita lanjut membahas tipe font. Tipe font yang saya maksud adalah **regular**, **bold**, **italic**, dan **bolditalic**. Tidak banyak yang tahu bahwa untuk setiap variasi ini, file font yang digunakan juga berbeda.

Sebagai contoh, ketika kita mengubah sebuah huruf menjadi miring (*italic*), file font yang digunakan juga khusus berisi tipe font italic. Dengan demikian, terdapat 4 tipe dari suatu font:

- Reguler
- Italic
- Bold
- Bold dan Italic

Tidak semua font memiliki ke-4 variasi di atas. Font yang termasuk ke dalam jenis *cursive* dan *fantasy* umumnya hanya memiliki 1 atau 2 variasi saja.

Apa hubungannya tipe font ini dengan CSS?

Sebelumnya kita sudah pelajari bahwa CSS menyediakan property `font-style` untuk memiringkan huruf serta property `font-weight` untuk membuat huruf tebal.

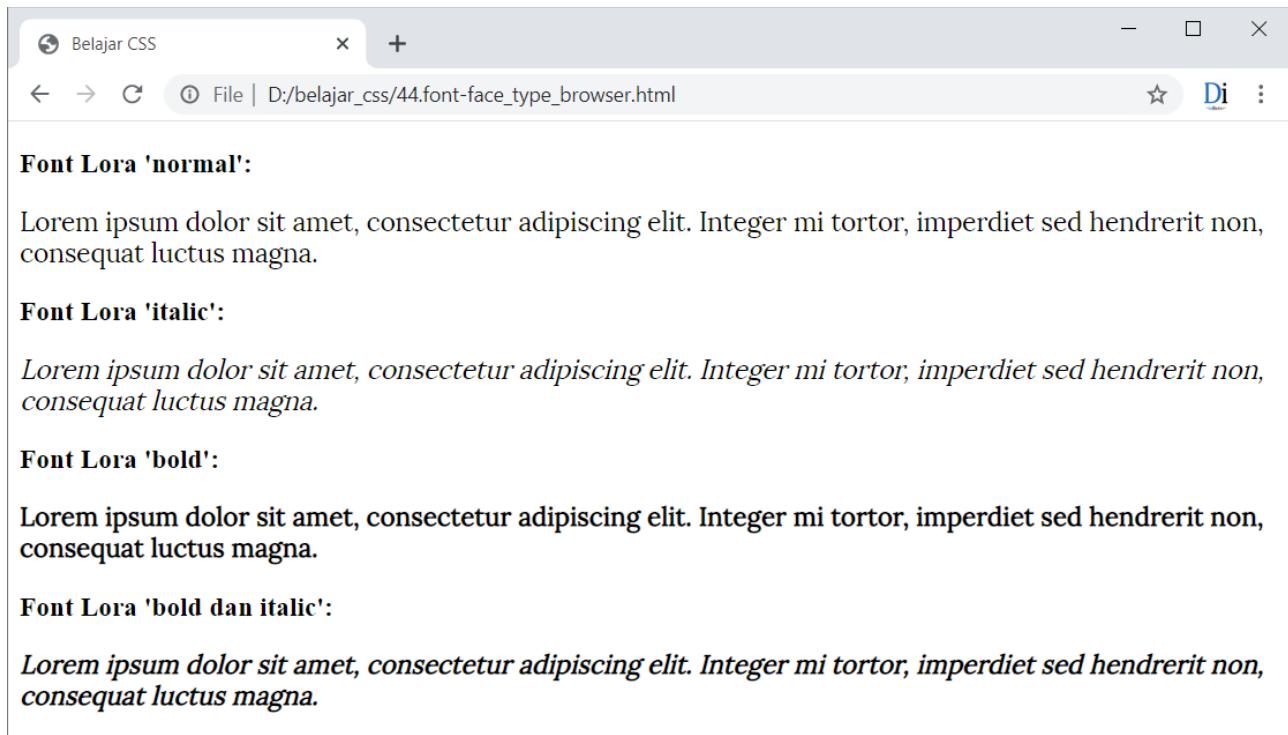
Pada saat sebuah huruf dibuat miring (melalui property `font-style:italic`), web browser akan mencari file font dengan tipe italic, jika tidak tersedia maka memproses text seolah-olah italic, hasilnya adalah huruf dengan jenis oblique!, hal yang sama juga terjadi untuk property `font-weight`.

Perbedaan antara huruf **italic** dengan **oblique** telah kita pelajari saat membahas property `font-style`.

Kode @font-rule yang kita pakai hingga saat ini hanya terdiri dari 1 tipe font, yakni font regular atau teks normal. Bagaimana hasilnya jika font tersebut ditampilkan dengan huruf miring atau tebal? Berikut prakteknya:

44. font-face_type_browser.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          @font-face {
8              font-family: Loraku;
9              src: url('LORA-REGULAR.TTF') format('truetype');
10         }
11         p {
12             font-family: "Loraku";
13             font-size: 18px;
14         }
15         .satu { font-style: italic; }
16         .dua { font-weight: bold; }
17         .tiga { font-style: italic; font-weight: bold; }
18     </style>
19 </head>
20 <body>
21     <h3>Font Lora 'normal':</h3>
22     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
23     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
24     </p>
25     <h3>Font Lora 'italic':</h3>
26     <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
27     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
28     </p>
29     <h3>Font Lora 'bold':</h3>
30     <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
31     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
32     </p>
33     <h3>Font Lora 'bold dan italic':</h3>
34     <p class="tiga">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
35     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
36     </p>
37 </body>
38 </html>
```



Gambar: Hasil berbagai tampilan type font lora dari 1 jenis font reguler

Sama seperti sebelumnya, saya kembali menggunakan font 'Loraku', kemudian membuat variasi tulisan miring (*italic*), tebal (*bold*), dan miring & tebal (*bold & italic*).

Jika dilihat sekilas tidak ada yang aneh dari tampilan di atas. Masing-masing paragraf tampil sesuai dengan efek yang diinginkan.

Namun sebenarnya seluruh efek ini adalah rekayasa web browser. Teks dengan huruf miring didapat dengan cara memiringkan font reguler beberapa derajat, sedangkan teks dengan huruf tebal didapat dengan menebalkan font reguler beberapa pixel.

Dalam kasus ideal, seharusnya ada 4 file font untuk masing-masing style. Dan inilah yang akan kita praktekkan. Saya telah menyertakan 4 file font Lora di dalam folder **fonts**. File tersebut adalah:

- LORA-REGULAR.TTF
- LORA-ITALIC.TTF
- LORA-BOLD.TTF
- LORA-BOLDITALIC.TTF

Lalu bagaimana cara menggunakan file-file ini?

Perintah `@font-face` masih menyediakan 2 property tambahan: `font-style` dan `font-weight`. Inilah perintah untuk menentukan jenis dari font tersebut. Sebagai contoh, untuk memberitahu web browser bahwa font ini adalah tipe *italic*, kita bisa menulisnya sebagai berikut:

```

@font-face {
    font-family: Loraku;
    src: url('fonts/LORA-ITALIC.TTF') format('truetype');
    font-weight: normal;
    font-style: italic;
}

```

Penulisan nama font-family masih sama, namun terdapat tambahan `font-style: italic` untuk memberitahu web browser bahwa file ini hanya untuk font bertipe italic. Hal yang sama juga dipakai untuk file font yang lain. Berikut contoh penggunaan keempat tipe font Lora:

45.font-face_type_asli.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          @font-face {
8              font-family: Loraku;
9              src: url('fonts/LORA-REGULAR.TTF') format('truetype');
10             font-weight: normal;
11             font-style: normal;
12         }
13         @font-face {
14             font-family: Loraku;
15             src: url('fonts/LORA-ITALIC.TTF') format('truetype');
16             font-weight: normal;
17             font-style: italic;
18         }
19         @font-face {
20             font-family: Loraku;
21             src: url('fonts/LORA-BOLD.TTF') format('truetype');
22             font-weight: bold;
23             font-style: normal;
24         }
25         @font-face {
26             font-family: Loraku;
27             src: url('fonts/LORA-BOLDITALIC.TTF') format('truetype');
28             font-weight: bold;
29             font-style: italic;
30         }
31         p {
32             font-family: "Loraku";
33             font-size: 18px;
34         }
35         .satu { font-style: italic; }
36         .dua { font-weight: bold; }
37         .tiga { font-style: italic; font-weight: bold; }
38     </style>
39 </head>
40 <body>
41     <h3>Font Lora 'normal':</h3>

```

```

42 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.  

43 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.  

44 </p>  

45 <h3>Font Lora 'italic':</h3>  

46 <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  

47 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.  

48 </p>  

49 <h3>Font Lora 'bold':</h3>  

50 <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  

51 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.  

52 </p>  

53 <h3>Font Lora 'bold dan italic':</h3>  

54 <p class="tiga">Lorem ipsum dolor sit amet, consectetur adipiscing elit.  

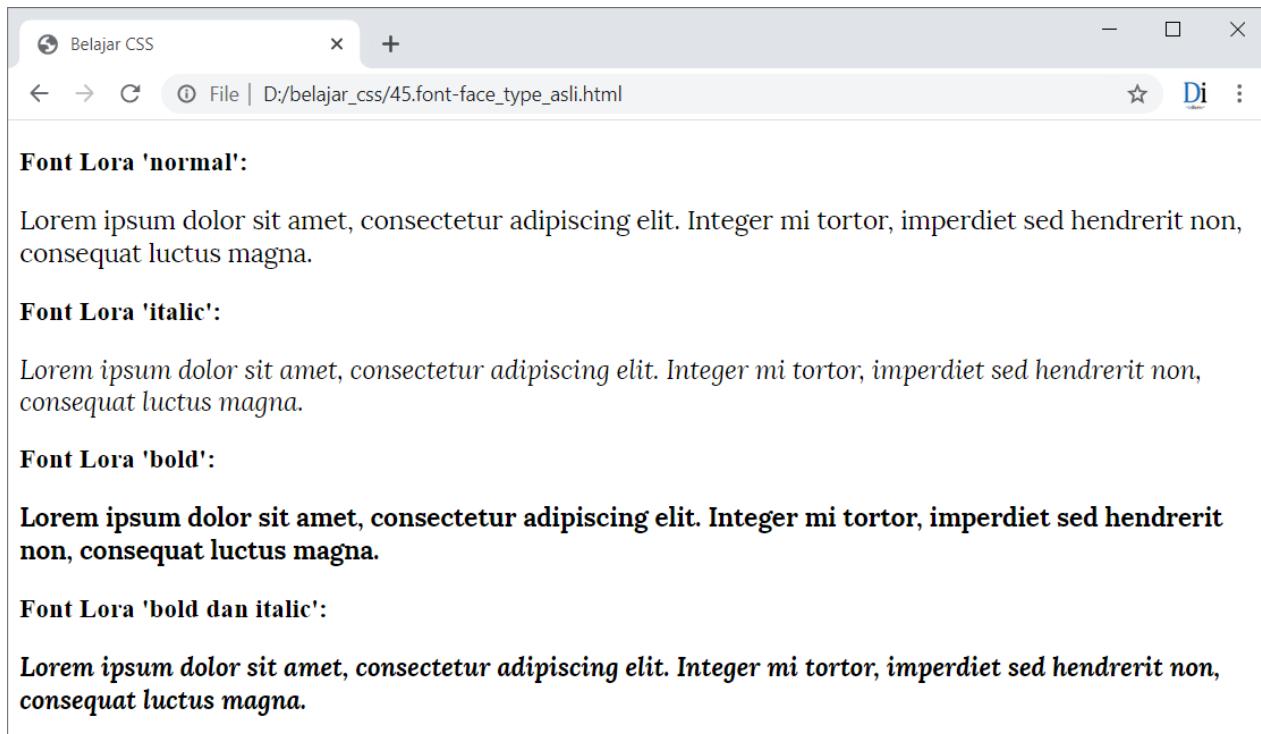
55 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.  

56 </p>  

57 </body>  

58 </html>

```



Gambar: Tampilan dari 4 jenis font lora 'asli'

Hasilnya, font Lora asli akan lebih cantik daripada font yang dihasilkan web browser.

Keempat perintah @font-face di atas nyaris sama, bedanya ada di nama file font serta nilai dari property font-style dan font-weight. Urutan penulisan font ini juga berpengaruh, dimana kita harus menggunakan urutan sebagai berikut:

1. Font normal, dengan nilai = font-weight: normal dan font-style: normal.
2. Font italic, dengan nilai = font-weight: normal dan font-style: italic.
3. Font bold, dengan nilai = font-weight: bold dan font-style: normal.
4. Font bolditalic, dengan nilai = font-weight: bold dan font-style: italic.

7.19. Menggunakan Google Font

Cara menginput font external menggunakan perintah `@font-face` memang cukup rumit, belum lagi masalah dengan perbedaan dukungan format. Kabar baiknya, terdapat alternatif lain yang bisa kita gunakan untuk menginput font external ke dalam halaman web, yakni menggunakan apa yang disebut dengan *web font services*.

Web font services adalah web online yang menyediakan layanan font external. Umumnya kita hanya perlu men-copy paste beberapa baris kode, dan font sudah siap digunakan.

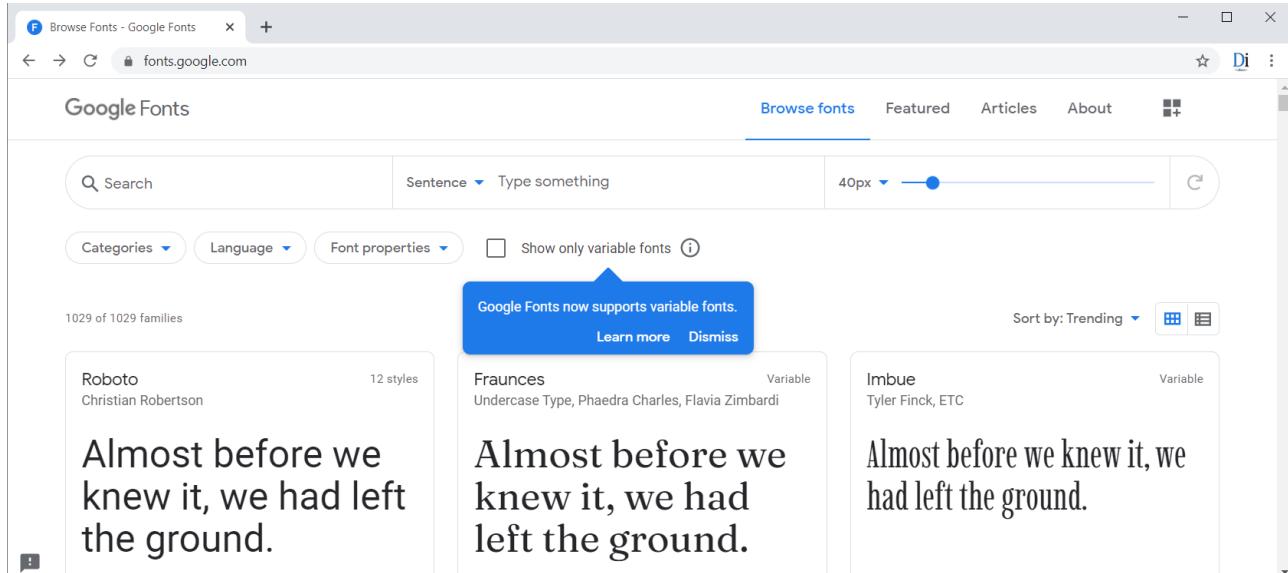
Terdapat berbagai web font services, diantaranya adalah:

- Google Fonts: fonts.google.com
- Font Squirrel: fontsquirrel.com
- Adobe Fonts: fonts.adobe.com
- Cloud Typography: typography.com

Kali ini saya hanya membahas web font services yang paling terkenal dan juga gratis, yakni **Google Fonts**. Google Fonts adalah layanan web font services dari...yup, Google. Saat buku ini di revisi pada awal 2021, Google Font menyediakan 1029 jenis font yang bisa kita pilih.

Satu hal yang perlu diingat ketika menggunakan Google Font dan layanan web font service lainnya adalah, kita harus terkoneksi ke internet. File font ini sebenarnya tetap berada di server Google, oleh karena itu jika koneksi internet terputus, web browser tidak akan bisa mengakses font ini.

Untuk web online, hal tersebut tidak jadi masalah karena web hosting akan selalu online.

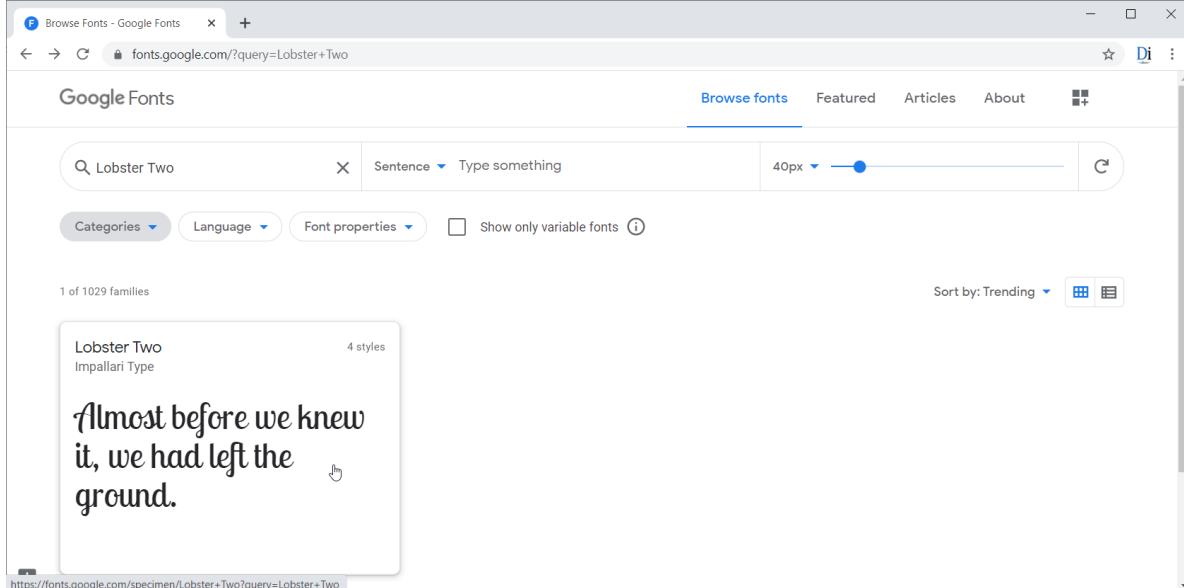


Gambar: Tampilan halaman awal Google Font

Di halaman awal Google Fonts, kita bisa menelusuri berbagai jenis font yang tersedia. Selain

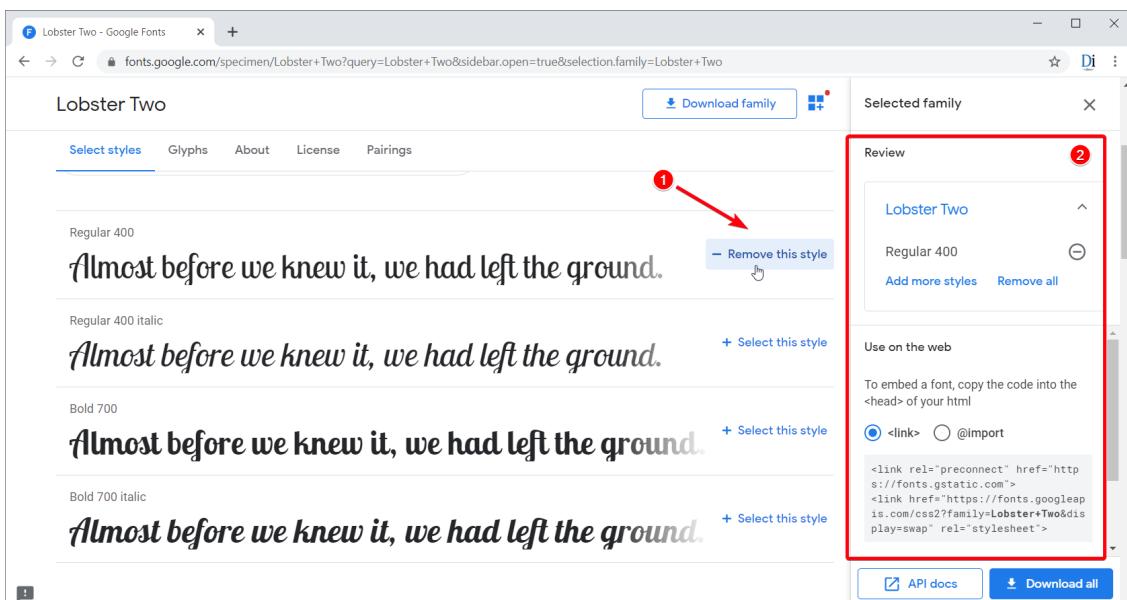
menulis langsung nama font, terdapat juga pilihan kategori font, yakni apakah serif, sans serif, display, handwriting atau monospace.

Silahkan klik salah satu font yang ingin dipakai, untuk praktek ini saya memilih "Lobster Two".



Gambar: Klik Font Lobster Two

Ketika di klik, kita akan dibawa ke halaman baru yang menampilkan semua style dari font tersebut. Style yang dimaksud adalah variasi font untuk teks reguler, italic, bold, serta bold dan italic. Tidak semua jenis font memiliki ke-4 variasi ini, beberapa ada yang hanya 1, 2, namun beberapa ada yang menyediakan hingga 10 variasi, dimana terdapat style untuk berbagai tingkatan bold.



Gambar: Klik link "+ Select this style"

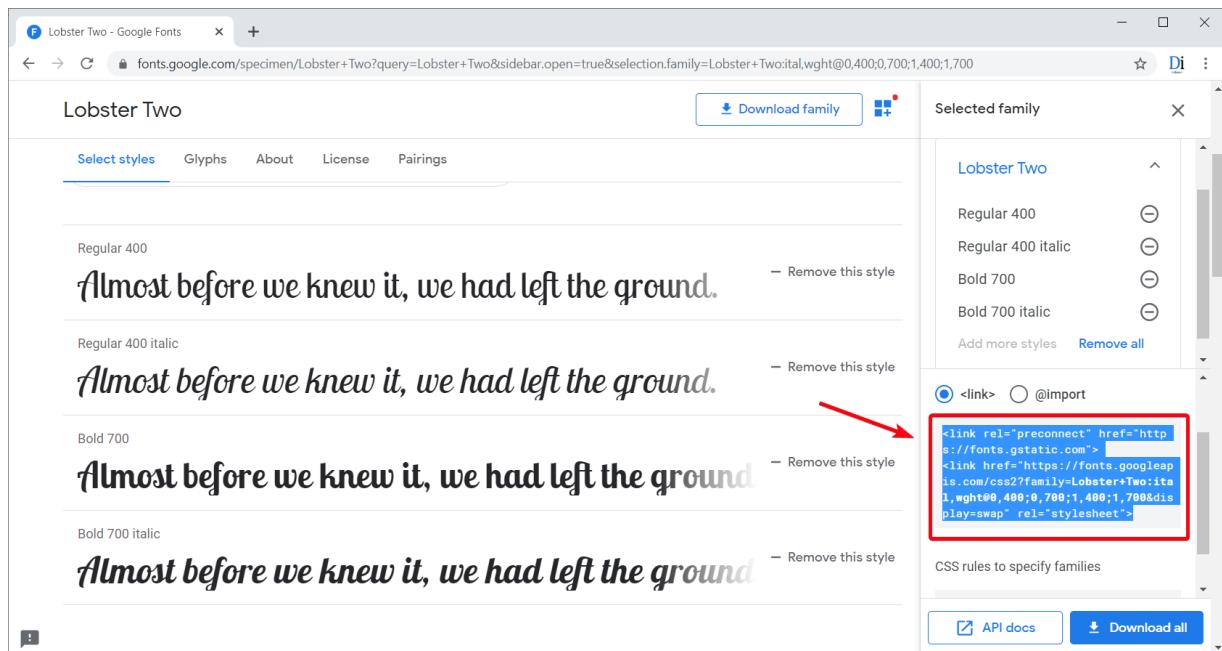
Google Font mengizinkan kita memilih jenis variasi apa saja yang ingin dipakai. Caranya, klik

link "+ Select this style" pada bagian kanan variasi font (1). Begitu teks ini di klik, akan tampil jendela sidebar baru (2):

Kita bisa memilih lebih dari satu variasi font yang diinginkan. Kali ini saya akan memilih 4 variasi yang tersedia untuk font Lobster Two. Caranya, klik kembali link "+ Select this style" untuk Regular 400 italic, Bold 700 dan Bold 700 italic. Sidebar di sisi kanan otomatis update setiap kali terdapat penambahan style baru.

Pilihan variasi font inilah yang akan dipakai oleh kode CSS nanti. Jika sebuah font tidak memiliki style untuk font italic, maka web browser mengakalinya dengan memiringkan huruf dari font reguler. Kasus ini mirip seperti yang kita praktekkan pada materi @font-face.

Setiap penambahan style, itu akan memperbesar ukuran file font yang harus di download oleh pengunjung web. Oleh karena itu batasi pilihan style yang benar-benar akan dipakai saja.



Gambar: Copy kode <link> yang disediakan Google Font

Kembali ke sidebar, di bagian bawah terdapat kolom "Use on the web", dengan 2 buah pilihan: <link> dan @import. Kedua pilihan ini merujuk ke kode yang akan kita pakai ke dalam file HTML. Kali ini saya akan pilih <link>, lalu copy kode yang disediakan:

Kode ini bersifat umum dan akan selalu sama untuk pilihan font yang sejenis. Misalnya dalam contoh ini kode yang saya dapat adalah:

```

<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css2?
family=Lobster+Two:ital,wght@0,400;0,700;1,400;1,700&display=swap"
rel="stylesheet">
    
```

Ini harus kita tempatkan ke dalam file HTML, tepatnya di bagian <head>.

Lalu scroll sidebar sedikit ke bawah, akan terlihat nama font yang diperlukan sebagai nilai dari property **font-family**:



Gambar: Nama font untuk property font-family

Dalam contoh ini kode yang diperlukan adalah **font-family: 'Lobster Two', cursive**. Dengan demikian kita sudah bisa membuat satu file HTML yang menggunakan font "Lobster Two" dari Google Font:

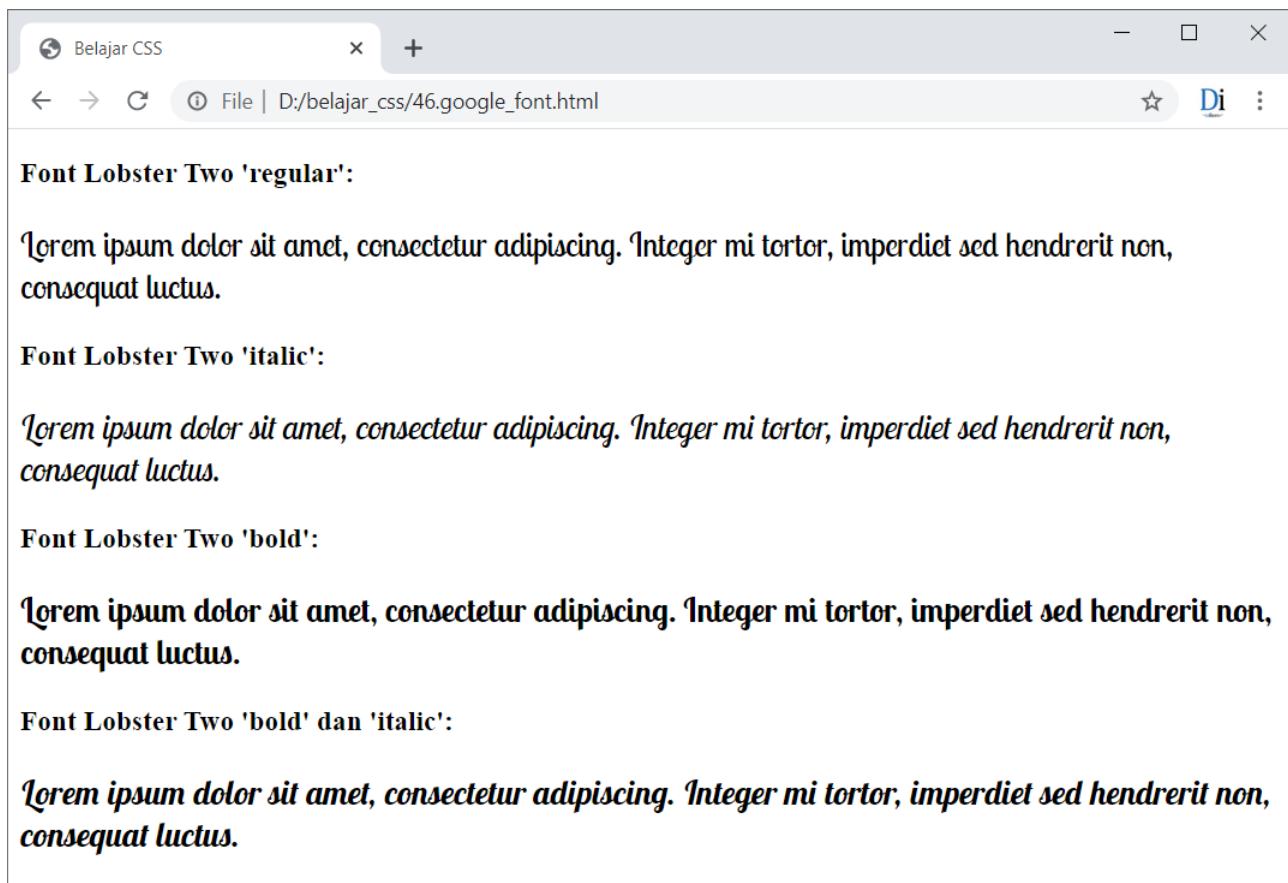
46.google_font.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <link rel="preconnect" href="https://fonts.gstatic.com">
7      <link href="https://fonts.googleapis.com/css2?
8          family=Lobster+Two:ital,wght@0,400;0,700;1,400;1,700&display=swap"
9          rel="stylesheet">
10     <style>
11         p { font-size: 22px; }
12         .satu { font-family: 'Lobster Two', cursive; }
13         .dua { font-family: 'Lobster Two', cursive; font-style: italic; }
14         .tiga { font-family: 'Lobster Two', cursive; font-weight: bold; }
15         .empat { font-family: 'Lobster Two', cursive;
16             font-style: italic; font-weight: bold; }
17     </style>
18 </head>
19 <body>
20     <h3>Font Lobster Two 'regular':</h3>
21     <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing.
22     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.
23     </p>
24     <h3>Font Lobster Two 'italic':</h3>
25     <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing.
26     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.
27     </p>
28     <h3>Font Lobster Two 'bold':</h3>
29     <p class="tiga">Lorem ipsum dolor sit amet, consectetur adipiscing.
30     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.
31     </p>
32     <h3>Font Lobster Two 'bold' dan 'italic':</h3>
33     <p class="empat">Lorem ipsum dolor sit amet, consectetur adipiscing.

```

```
34    Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.  
35    </p>  
36 </body>  
37 </html>
```



Gambar: Hasil penggunaan Google Font

Seperti yang terlihat, keempat paragraf telah menggunakan font Lobster Two dari Google Font. Khusus perintah tag `<link>` di baris 7-8 sebenarnya harus ditulis dalam 1 baris panjang. Namun karena keterbatasan lebar buku, terpaksa saya pecah jadi 2 baris.

Google Font juga mengizinkan kita mencampur berbagai jenis font, misalnya dalam contoh berikut saya menggunakan font "Pinyon Script" dan "Righteous" dalam sebuah tag `<link>`. Kode ini tinggal di copy paste dari bagian sidebar seperti sebelumnya:

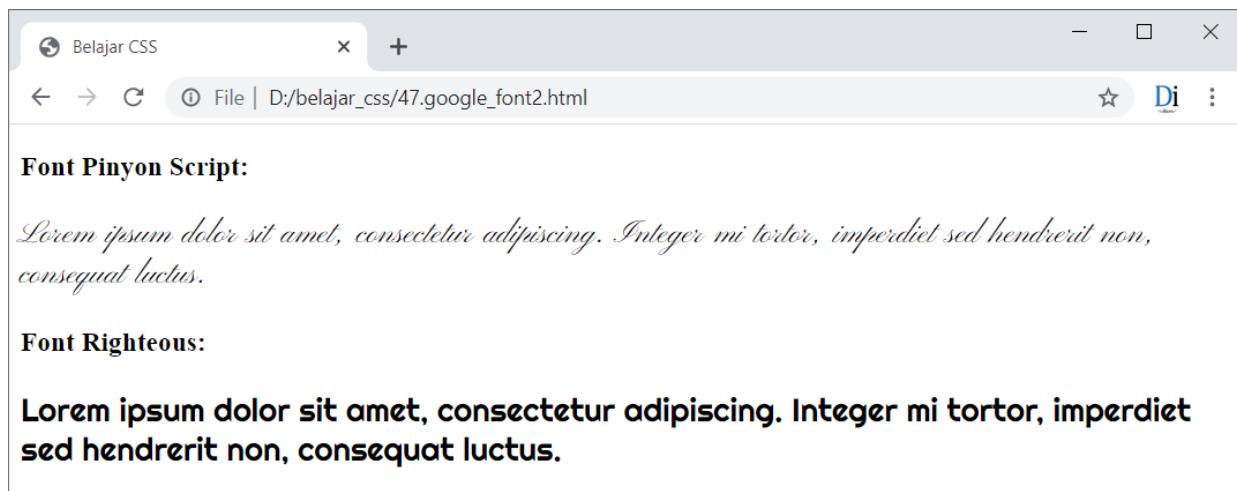
47.google_font2.html

```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <link rel="preconnect" href="https://fonts.gstatic.com">  
7      <link href="https://fonts.googleapis.com/css2?  
8          family=Pinyon+Script&family=Righteous&display=swap"  
9          rel="stylesheet">
```

```

10 <style>
11 p { font-size: 22px; }
12 .satu { font-family: 'Pinyon Script', cursive; }
13 .dua { font-family: 'Righteous', cursive; }
14 </style>
15 </head>
16 <body>
17 <h3>Font Pinyon Script:</h3>
18 <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing.
19 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.
20 </p>
21 <h3>Font Righteous:</h3>
22 <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing.
23 Integer mi tortor, imperdiet sed hendrerit non, consequat luctus.
24 </p>
25 </body>
26 </html>

```



Gambar: Hasil penggunaan Google Font untuk dua jenis font sekaligus

Jika anda menggunakan koneksi internet yang tidak terlalu cepat, anda akan melihat jeda sesaat ketika web browser mencoba mendownload font-font ini.

Web Google Font secara reguler di update waktu ke waktu, sehingga sangat mungkin ketika anda membaca buku, tampilan webnya juga sudah berubah. Akan tetapi langkah-langkah yang diperlukan tetap sama, yakni:

1. Pilih jenis font
2. Pilih variasi font
3. Copy paste tag <link> atau perintah @import yang disediakan

Dalam bab ini kita telah membahas berbagai property CSS yang berkaitan dengan typography. Termasuk juga nilai satuan CSS seperti length, kode warna, vendor prefix, hingga cara penggunaan Google Font.

Pada bab selanjutnya akan masuk ke pembahasan mengenai CSS Box Model.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

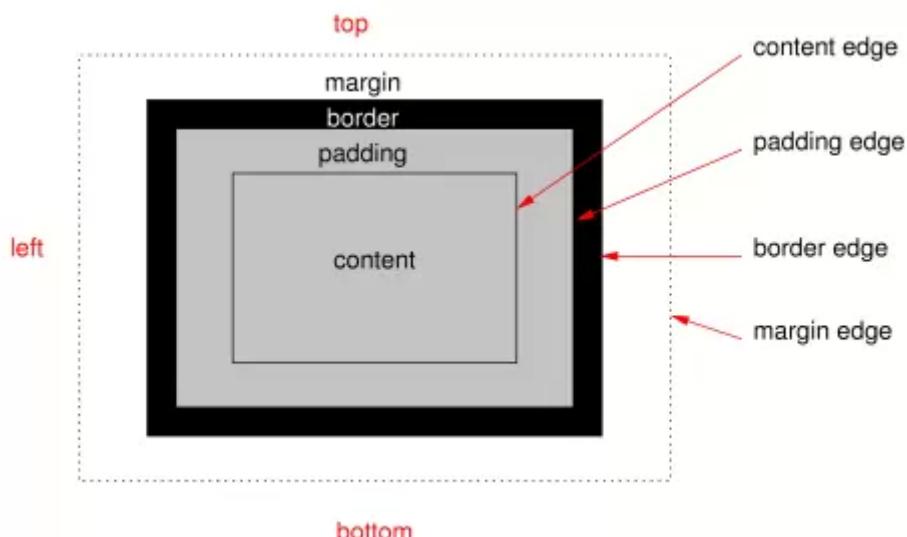
8. CSS Box Model

Salah satu konsep dasar yang wajib dipahami ketika belajar CSS adalah tentang **box model**. Dalam bab ini kita akan membahas dengan detail apa itu CSS box model, serta membahas juga tentang property overflow, rounding-corner, serta CSS Reset.

8.1. Pengertian CSS Box Model

CSS Box Model adalah sebuah konsep dimana setiap element yang terdapat pada halaman web diproses sebagai kotak (box). Mulai dari paragraf, header, form, gambar, logo hingga video sebenarnya di tampilkan oleh web browser sebagai 'box'.

Sebagaimana layaknya box atau kotak, masing-masing element HTML ini terdiri dari 4 lapis, yakni: konten (isi), padding, border dan margin. Keempat lapisan inilah yang membangun CSS Box Model:



Gambar: Konsep CSS Box Model (sumber: www.w3c.org)

Konten dari sebuah element berada di bagian tengah yang nantinya bisa diatur menggunakan property **width** dan **height**.

Selanjutnya, terdapat *padding*. **Padding** adalah jarak antara konten dengan garis tepi (border) element. Sebagai contoh, jika kita membuat sebuah teks tanpa padding, maka teks tersebut akan mulai persis setelah garis tepi. Padding biasa ditambahkan supaya teks tidak menyentuh

sisi dalam border.

Setelah padding, berikutnya **border**, yakni pembatas element. Di dalam CSS, kita bisa mengatur ketebalan, warna, dan jenis garis yang digunakan.

Di lapisan terakhir terdapat *margin*. **Margin** adalah pembatas sebuah element dengan element lain di sekelilingnya. Margin bersifat transparan dan berfungsi agar setiap element tidak saling menempel satu sama lain.

Konsep box model sebenarnya cukup mudah dipahami. Namun dalam prakteknya terdapat beberapa hal yang perlu penjelasan lebih lanjut, terutama tentang bagaimana keempat element ini saling berhubungan dengan element lain.

8.2. Property width dan height

Pembahasan mengenai Box Model akan kita mulai dari konten. Konten adalah sebutan untuk bagian utama dari sebuah element, biasanya konten ini berisi teks tapi juga bisa berbentuk gambar atau video.

Untuk mengatur lebar dan tinggi konten, tersedia property `width` dan `height`. Kedua property ini mendukung seluruh satuan nilai length yang telah kita pelajari di bab sebelumnya, seperti px, em, %, dan rem.

Berikut contoh penggunaan property `width` dan `height`:

01.width_and_height.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h3 {
8              width: 300px;
9              height: 100px;
10             background-color: yellow;
11         }
12         p {
13             width: 800px;
14             height: 150px;
15             background-color: aqua;
16         }
17     </style>
18 </head>
19 <body>
20     <h3>Belajar Box Model</h3>
21     <p>Sebuah paragraf sederhana</p>
22 </body>
23 </html>
```



Gambar: Contoh penggunaan property width dan height

Sesuai dengan property yang ditulis, tag <h3> tampil dengan lebar 300 pixel dan tinggi 100 pixel, sedangkan tag <p> tampil dengan lebar 800 pixel dan tinggi 150 pixel.

Selain property `width` dan `height`, dalam kode CSS di atas saya juga menambah property `background-color`. Property ini berguna untuk mengubah warna latar belakang (*background*) dari sebuah element. Tujuannya agar kita bisa melihat ukuran asli dari element tersebut. Jika property `background-color` tidak ditulis, secara default akan berwarna putih.

Penggunaan property `background` akan kita bahas lebih detail dalam bab berikutnya.

Warna putih di antara tag <h1> dan <p> merupakan margin bawaan web browser, kita akan membahas tentang margin beberapa saat lagi.

Satuan Persen untuk width dan height

Selain menggunakan satuan pixel, kita juga bisa mengatur lebar dan tinggi element menggunakan satuan persen (%). Berikut contohnya:

02.width_and_height_percent.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h3 {
8              width: 30%;
9              height: 5%;
10             background-color: yellow;
11         }

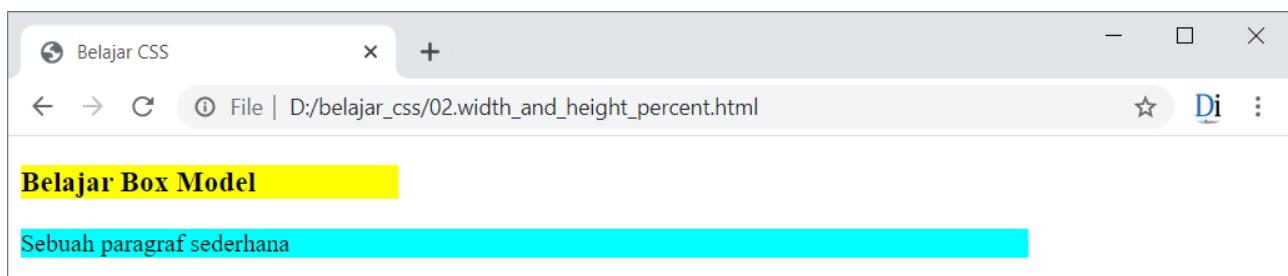
```

```

12     p {
13         width: 80%;
14         height: 15%;
15         background-color: aqua;
16     }
17 </style>
18 </head>
19 <body>
20     <h3>Belajar Box Model</h3>
21     <p>Sebuah paragraf sederhana</p>
22 </body>
23 </html>

```

Kali ini saya men-set tag `<h3>` dengan lebar 30% dan tinggi 5%, sedangkan untuk tag `<p>` diatur dengan lebar 80% dan tinggi 15%. Bagaimana hasilnya? mari kita lihat:



Gambar: Contoh penggunaan property width dan height dengan satuan persen ?

Apa yang terjadi?

Sebagaimana yang telah kita pelajari ketika membahas konsep nilai relatif, satuan persen mendapat nilainya dari parent element. Ketika lebar tag `<h3>` di set sebesar 30%, maka akan dihitung 30% dari lebar parent element. Element apa yang bertindak sebagai parent element dari tag `<h3>`?

Yup, parent element-nya adalah tag `<body>`. Tag `<body>` secara default di-set dengan lebar sesuai jendela web browser. Oleh karena itu tag `<h3>` akan mengambil tempat sekitar 30% dari ukuran lebar jendela web browser saat ini.

Silahkan anda ubah ukuran web browser, lebar tag `<h3>` juga akan mempertahankan nilai 30%. Hal yang sama juga terjadi untuk tag `<p>` yang di-set sebesar 80% dari tag `<body>`.

Namun bagaimana dengan `height` kedua element? Kenapa tidak tampil sebesar 5% dari tinggi jendela web browser (untuk tag `<h3>`) dan 15% (untuk tag `<p>`)? Ini terjadi karena parent element dari keduanya (yakni tag `<body>`) tidak memiliki tinggi yang ditulis langsung.

Efeknya, ketika saya men-set property `height` untuk tag `<h3>` sebesar 5%, itu dihitung dari tinggi parent element (tag `<body>`). Ini berbeda dengan property `width` yang tidak harus di set langsung.

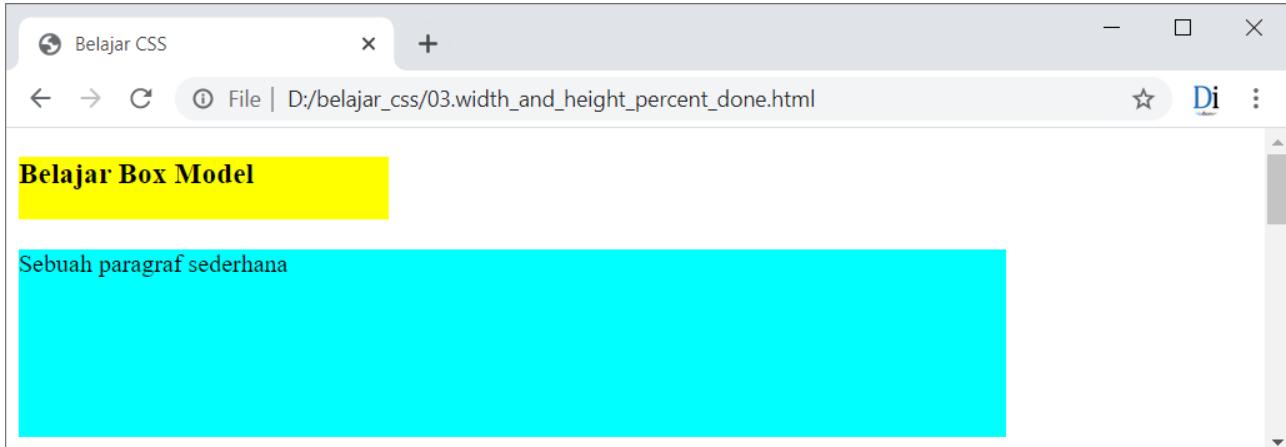
Dengan demikian agar tinggi element bisa tampil sebagaimana mestinya, kita harus tulis property `width` untuk tag `<body>` seperti contoh berikut:

03.width_and_height_percent_done.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              height: 800px;
9          }
10         h3 {
11             width: 30%;
12             height: 5%;
13             background-color: yellow;
14         }
15         p {
16             width: 80%;
17             height: 15%;
18             background-color: aqua;
19         }
20     </style>
21 </head>
22 <body>
23     <h3>Belajar Box Model</h3>
24     <p>Sebuah paragraf sederhana</p>
25 </body>
26 </html>

```



Gambar: Contoh penggunaan property width dan height dengan satuan persen

Kali ini tinggi setiap element akan tampil sebagaimana mestinya.

Satuan em untuk width dan height

Sesuai dengan definisi satuan em, nilai ini berpatokan kepada ukuran font-size element tersebut. Sebagai contoh, jika saya memiliki sebuah element dengan property `font-size: 20px`, maka nilai `width: 30em` sama dengan $30 \times 20 = 600px$. Mari kita lihat praktiknya:

04.width_and_height_em.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p.em {
8              font-size: 20px;
9              width: 30em;
10             height: 4em;
11             background-color: aqua;
12         }
13         p.pixel {
14             width: 600px;
15             height: 80px;
16             background-color: yellow;
17         }
18     </style>
19 </head>
20 <body>
21     <p class="em">Paragraf dengan lebar 5em dan tinggi 2em</p>
22     <p class="pixel">Paragraf dengan lebar 600px dan tinggi 80px</p>
23 </body>
24 </html>

```



Gambar: Contoh penggunaan property width dan height dengan satuan em

Dapat dilihat bahwa lebar dan tinggi paragraf pertama sama dengan lebar dan tinggi paragraf kedua. Jika saya mengubah ukuran font-size, lebar dan tinggi element juga akan berubah.

Sifat Default Property width dan height

Selanjutnya, bagaimana jika kita tidak menulis width dan height dari sebuah element?

Perhatikan kode berikut:

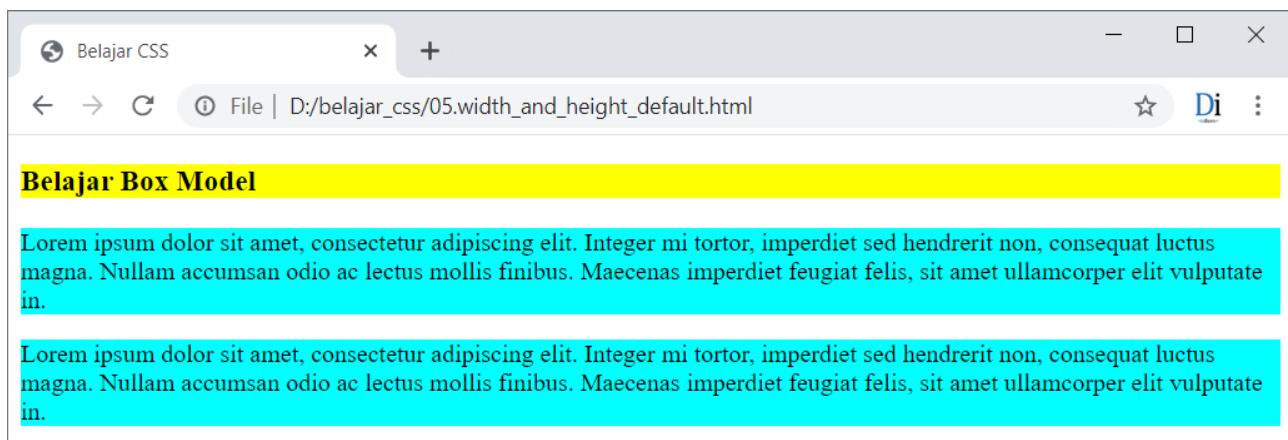
05.width_and_height_default.html

```
1  <!DOCTYPE html>
```

```

2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     h3 { background-color: yellow; }
8     p { background-color: aqua; }
9   </style>
10 </head>
11 <body>
12   <h3>Belajar Box Model</h3>
13   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
14     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
15     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
16     felis, sit amet ullamcorper elit vulputate in.</p>
17   <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
18     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
19     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
20     felis, sit amet ullamcorper elit vulputate in.</p>
21 </body>
22 </html>

```



Gambar: Tanpa menggunakan width dan height

Kode ini terdiri dari 3 buah element: sebuah judul `<h3>` dan 2 buah paragraf `<p>`. Seperti yang terlihat, apabila kita tidak men-set `width` sebuah element, element tersebut akan mengambil lebar seluruh parent element, atau sama dengan `width = 100%`.

Untuk tinggi element, apabila `height` tidak ditulis, element tersebut akan menyesuaikan diri tergantung tinggi konten yang ada di dalamnya. Contoh berikut bisa memperjelas konsep ini:

06.width_and_height_div.html

```

1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     h3 { background-color: yellow; }

```

```

8      p { background-color: aqua; }
9      div { width: 500px; }
10     </style>
11 </head>
12 <body>
13   <h3>Belajar Box Model</h3>
14   <div>
15     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
16       tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
17       accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
18       felis, sit amet ullamcorper elit vulputate in.</p>
19     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
20       tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
21       accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
22       felis, sit amet ullamcorper elit vulputate in.</p>
23   </div>
24 </body>
25 </html>

```



Gambar: Lebar element mengikuti lebar parent element

Saya memodifikasi kode sebelumnya dan menempatkan kedua paragraf di dalam tag `<div>` dengan `width:500px`. Karena lebar tag `<p>` tidak ditulis, kedua paragraf akan 'melebar' untuk memenuhi parent element-nya. Hasilnya, kedua paragraf seolah-oleh memiliki lebar 500px.

Block dan Inline element

Menutup pembahasan mengenai `width` dan `height`, perlu diketahui bahwa kita hanya bisa mengatur lebar dan tinggi element yang termasuk ke dalam **block level element**.

Block level element adalah tag-tag HTML yang ditampilkan terpisah dalam baris baru. Contoh dari block level element adalah tag `<p>`, `<h3>` dan `<div>`.

Kita tidak bisa menggunakan `width` dan `height` untuk **inline level element**. Inline level element adalah tag-tag HTML yang mengikuti tampilan yang ada saat ini (tidak pindah ke baris baru). Contoh dari inline level element adalah tag ``, `` dan ``.

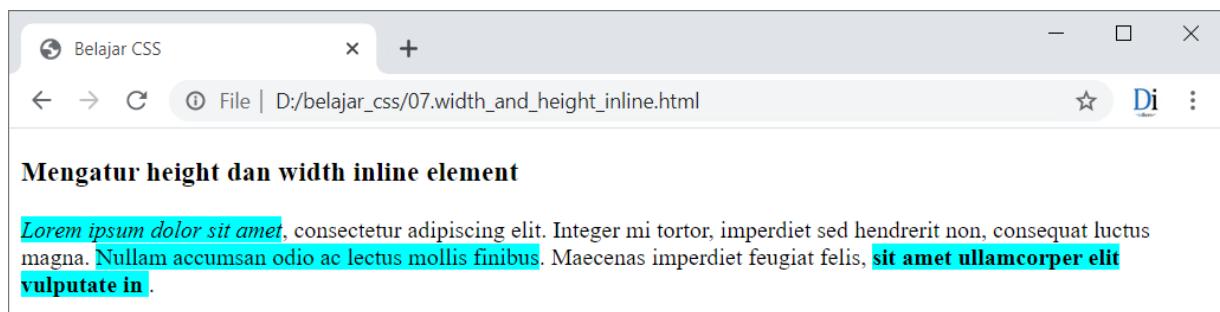
Dalam kode program berikut saya coba mengatur tinggi dan lebar dari tag ``, `<div>`, dan ``:

07.width_and_height_inline.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          em, span, strong{
8              background-color: aqua;
9              width: 100px;
10             height: 50px;
11         }
12     </style>
13 </head>
14 <body>
15     <h3>Mengatur height dan width inline element</h3>
16     <p>
17         <em>Lorem ipsum dolor sit amet</em>, consectetur adipiscing elit.
18         Integer mi tortor, imperdiet sed hendrerit non, consequat luctus
19         magna. <span>Nullam accumsan odio ac lectus mollis finibus</span>.
20         Maecenas imperdiet feugiat felis, <strong>sit amet ullamcorper
21         elit vulputate in </strong>.
22     </p>
23 </body>
24 </html>

```



Gambar: Test penggunaan property width dan height untuk inline element

Hasilnya, seluruh tag tampil seperti biasa. Warna background aqua menjadi bukti bahwa kode CSS yang saya tulis memang sudah berjalan, namun property `width` dan `height` tidak berefek.

CSS sebenarnya menyediakan property `display` untuk mengubah *inline level element* menjadi *block level element* dan sebaliknya. Kita akan bahas property ini dalam bab tentang CSS Positioning.

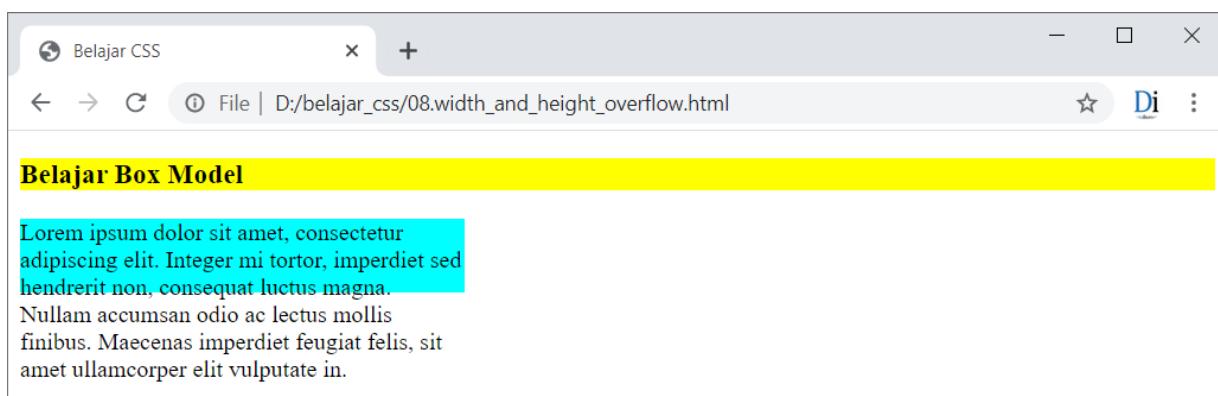
8.3. Property overflow

Ketika mengatur konten dari sebuah element, ada kalanya isi konten tersebut sangat panjang sehingga tidak muat di dalam parent element. Berikut contoh kasusnya:

08.width_and_height_overflow.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          h3 { background-color: yellow; }
8          p {
9              background-color: aqua;
10             width: 300px;
11             height: 50px;
12         }
13     </style>
14 </head>
15 <body>
16     <h3>Belajar Box Model</h3>
17     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
18         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
19         accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
20         felis, sit amet ullamcorper elit vulputate in.</p>
21 </body>
22 </html>
```



Gambar: Overflow!

Kali ini saya membuat tinggi dan lebar paragraf cukup sempit, akibatnya element tersebut tidak sanggup menampung konten yang ada. Secara default, sisa konten akan 'melimpah' keluar, yang di dalam CSS dikenal sebagai **overflow**.

Untuk mengatur bagaimana web browser menangani konten *overflow*, tersedia property... *overflow*. Property ini bisa diisi dengan nilai `visible` (default), `hidden`, `scroll`, atau `auto`. Berikut penjelasan keempat nilai ini:

overflow: visible

Nilai `visible` menginstruksikan web browser agar kelebihan konten tetap tampil semua. Hasilnya seperti contoh kita di atas, konten akan 'melimpah' keluar dari element. Nilai `overflow:visible` merupakan default bawaan web browser jika property ini tidak ditulis.

overflow: hidden

Nilai `hidden` akan menyembunyikan kelebihan konten. Dengan demikian, sisa konten yang `overflow` tidak akan terlihat di web browser.

overflow: scroll

Nilai property `scroll` akan menambah scrollbar di bagian bawah dan kanan element, sehingga kita bisa menggeser scrollbar untuk melihat konten yang melebihi tinggi dan lebar element.

overflow: auto

Nilai ini akan menambah scrollbar hanya jika diperlukan.

Berikut contoh penggunaan 3 nilai overflow selain `visible`:

09.overflow_example

```

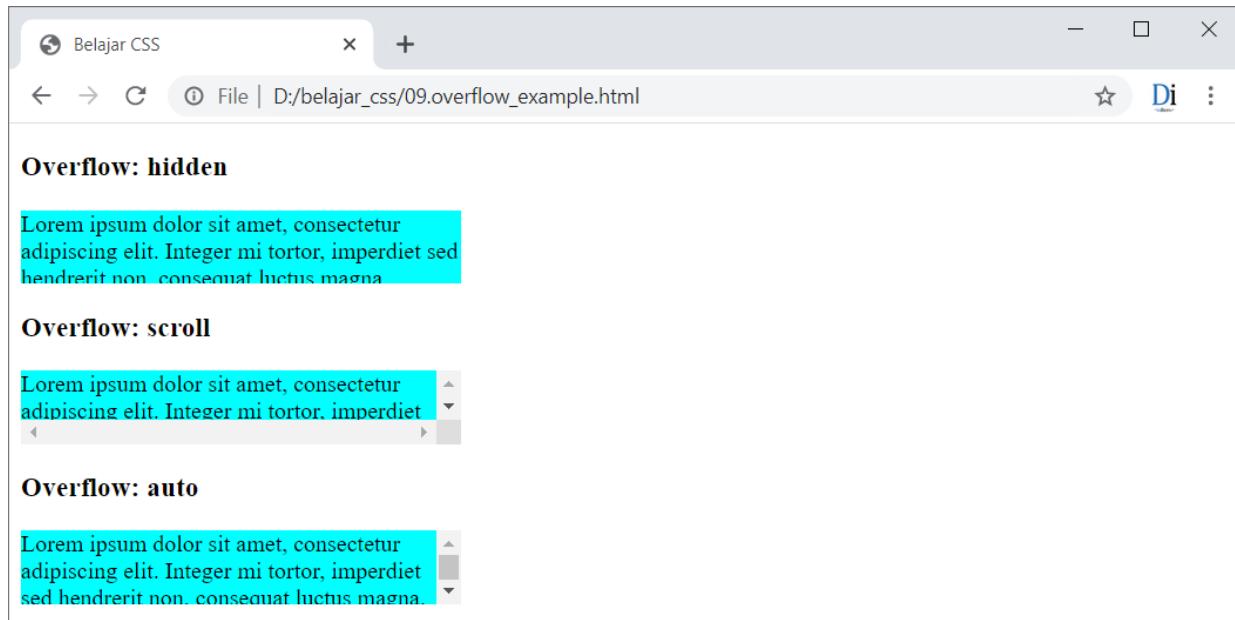
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p{
8              background-color: aqua;
9              width: 300px;
10             height: 50px;
11         }
12         .satu { overflow: hidden; }
13         .dua { overflow: scroll; }
14         .tiga { overflow: auto; }
15     </style>
16 </head>
17 <body>
18     <h3>Overflow: hidden</h3>
19     <p class="satu">
20         Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
21         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
22         accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
23         felis, sit amet ullamcorper elit vulputate in.
24     </p>
25     <h3>Overflow: scroll</h3>
26     <p class="dua">
27         Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
28         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
29         accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
30         felis, sit amet ullamcorper elit vulputate in.

```

```

31   </p>
32   <h3>Overflow: auto</h3>
33   <p class="tiga">
34     Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
35     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
36     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
37     felis, sit amet ullamcorper elit vulputate in.
38   </p>
39 </body>
40 </html>

```



Gambar: Contoh penggunaan property overflow

Contoh pertama menggunakan property `overflow: hidden`, sehingga sisa konten tidak akan terlihat di web browser (teks tampak terpotong).

Contoh kedua memakai property `overflow: scroll`, kali ini akan terlihat scrollbar di sisi bawah dan kanan element. Perhatikan bahwa walaupun teks ini hanya 'melimpah' ke arah bawah (vertikal), kita juga melihat tampilan scrollbar di kedua sisi element.

Untuk contoh ketiga, saya menggunakan property `overflow: auto`. Property ini hanya akan menampilkan scrollbar jika diperlukan. Karena konten kita hanya melimpah ke sisi bawah, scrollbar horizontal tidak tampil.

CSS juga mengizinkan kita untuk men-set scrollbar horizontal saja atau scrollbar vertikal saja. Ini dilakukan dari property `overflow-x` dan `overflow-y`. Kedua property ini bisa diisi dengan nilai yang sama dengan property `overflow`, yakni `visible` (default), `hidden`, `scroll`, atau `auto`. Berikut contoh penggunaannya:

09a.overflow_x_y_example.html

```

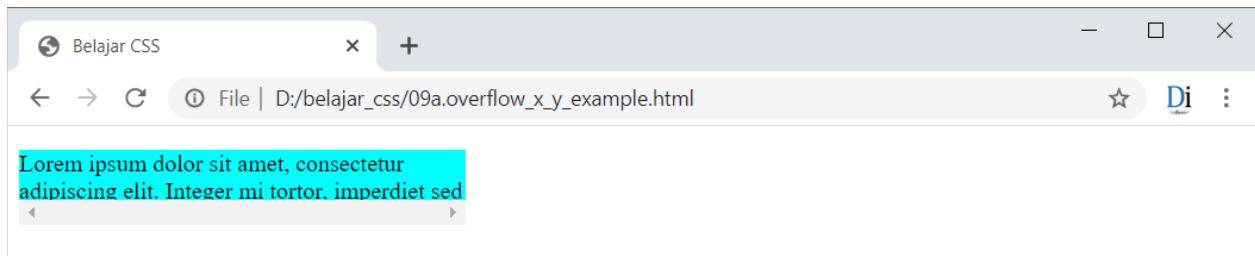
1  <!DOCTYPE html>
2  <html lang="id">

```

```

3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     p{
8       background-color: aqua;
9       width: 300px;
10      height: 50px;
11      overflow-x: scroll;
12      overflow-y: hidden;
13    }
14  </style>
15 </head>
16 <body>
17 <p>
18   Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
19   tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
20   accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
21   felis, sit amet ullamcorper elit vulputate in.
22 </p>
23 </body>
24 </html>

```



Gambar: Contoh penggunaan property overflow-x dan overflow-y

Dalam contoh ini, paragraf saya set dengan `overflow-x: scroll` dan `overflow-y: hidden`. Hasilnya, scrollbar hanya tampil di sumbu x saja.

8.4. Property padding

Lapisan kedua dari konsep box model adalah **padding**. **Padding** merupakan ruang atau 'spasi' antara konten dengan garis tepi (border) sebuah element. Padding berfungsi agar konten tidak menempel langsung dengan garis tepi.

Terdapat beberapa cara penulisan padding. Cara pertama adalah dengan menulis nilai padding untuk setiap arah secara terpisah, yakni atas, bawah, kiri dan kanan, yang dalam bahasa Inggris menjadi top, bottom, left dan right. Penulisannya menggunakan property berikut:

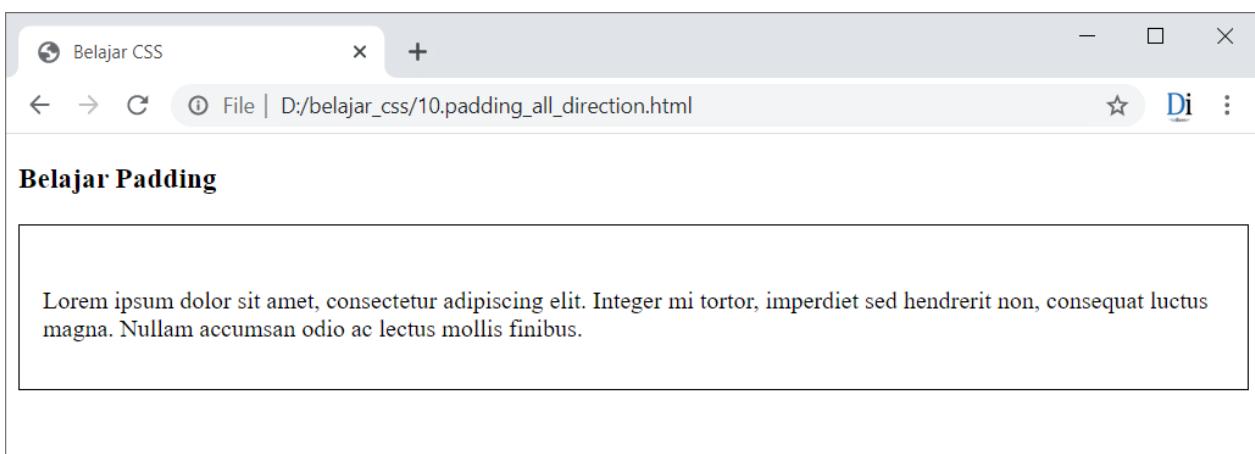
- `padding-top`
- `padding-bottom`
- `padding-left`
- `padding-right`

Property padding mendukung semua nilai length CSS, seperti px, em, atau %,. Berikut contoh penggunaannya:

10.padding_all_direction.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p{
8              padding-top: 40px;
9              padding-bottom: 30px;
10             padding-left: 15px;
11             padding-right: 25px;
12             border: 1px solid black;
13         }
14     </style>
15 </head>
16 <body>
17     <h3>Belajar Padding</h3>
18     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
19         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
20         accumsan odio ac lectus mollis finibus.</p>
21 </body>
22 </html>
```



Gambar: Contoh penggunaan keempat property padding

Dalam contoh di atas saya menulis nilai padding yang berbeda-beda untuk setiap sisi paragraf. Terlihat pada sisi atas (top) dan bawah (bottom) jarak antara paragraf dengan border lebih jauh dibandingkan sisi kiri dan kanan. Property border sengaja ditambah agar kita bisa melihat efek dari padding ini.

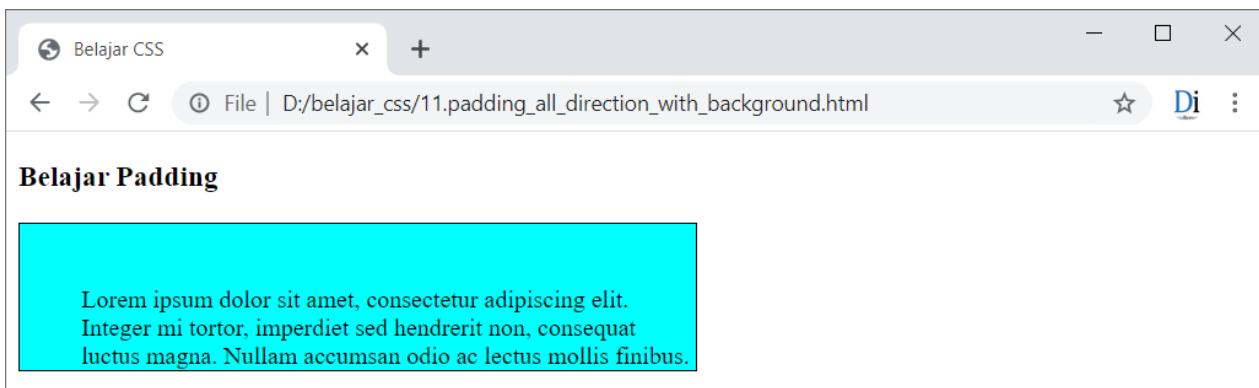
Padding juga menampilkan warna background, sehingga terlihat menyatu dengan konten yang ada di dalamnya:

11.padding_all_direction_with_background.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p{
8              width: 400px;
9              padding-top: 40px;
10             padding-bottom: 0px;
11             padding-left: 40px;
12             padding-right: 0px;
13             border: 1px solid black;
14             background-color: aqua;
15         }
16     </style>
17 </head>
18 <body>
19     <h3>Belajar Padding</h3>
20     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
21         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
22         accumsan odio ac lectus mollis finibus.</p>
23 </body>
24 </html>

```



Gambar: Warna background akan terlihat 'dibawah' padding

Terlihat ruang kosong di sisi atas dan kiri paragraf yang berasal dari `padding-top: 40px` dan `padding-left: 40px` juga akan berwarna aqua.

Padding Shorthand Notation (Penulisan Singkat Padding)

Pada contoh sebelum ini, saya menulis property padding untuk setiap sisi, yaitu `padding-top`, `padding-bottom`, `padding-left` dan `padding-right`. Agar lebih singkat, keempat property ini bisa ditulis dalam satu property saja, yakni `padding`.

Property `padding` bisa diisi dengan 1, 2, 3 atau 4 nilai, yang setiap penulisan nilai dipisah dengan karakter spasi.

Jika property padding ditulis hanya dengan 1 nilai saja, nilai itu akan dipakai untuk ke-4 sisi element. Sebagai contoh, padding: 10px dipakai untuk membuat padding 10px di atas, bawah, kiri dan kanan element.

Jika property padding ditulis dengan 2 nilai, nilai pertama dipakai untuk sisi top dan bottom, sedangkan nilai kedua untuk sisi left dan right. Sebagai contoh, padding: 10px 20px berarti padding-top dan padding-bottom sebesar 10px, serta padding-left dan padding-right masing-masing sebesar 20px.

Jika property padding ditulis dengan 3 nilai, nilai pertama digunakan untuk padding-top, nilai kedua untuk padding-left dan padding-right serta nilai ketiga untuk padding-bottom.

Sebagai contoh, padding: 10px 30px 5px artinya sama dengan:

```
padding-top: 10px
padding-bottom: 5px
padding-left: 30px
padding-right: 30px
```

Jika padding ditulis dengan 4 nilai, nilai pertama untuk padding-top, nilai kedua untuk padding-right, nilai ketiga untuk padding-bottom, dan nilai keempat untuk padding-left.

Sebagai contoh, padding: 5px 10px 15px 20px artinya sama dengan:

```
padding-top: 5px
padding-right: 10px
padding-bottom: 15px
padding-left: 20px
```

Untuk menghafal urutan ini, bisa menggunakan arah jarum jam atau menggunakan singkatan **TRouBLE** (Top Right Bottom Left). Berikut contoh penggunaan ke-4 variasi penulisan padding shorthand notation:

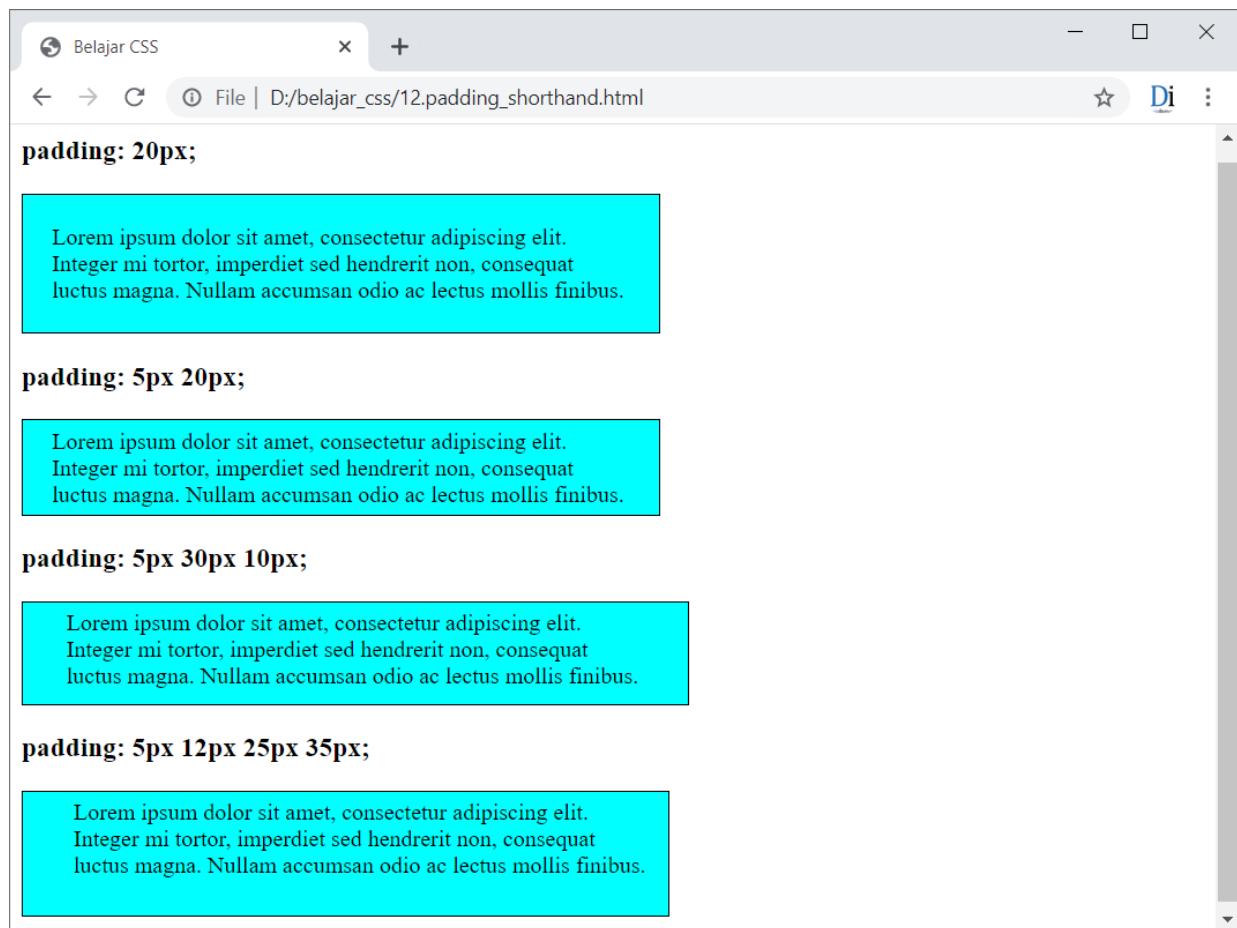
12.padding_shorthand.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p{
8              width: 400px;
9              border: 1px solid black;
10             background-color: aqua;
11         }
12         .satu { padding: 20px; }
13         .dua { padding: 5px 20px; }
14         .tiga { padding: 5px 30px 10px; }
15         .empat { padding: 5px 12px 25px 35px; }
16     </style>
```

```

17 </head>
18 <body>
19   <h3>padding: 20px;</h3>
20   <p class="satu">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
21     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
22     Nullam accumsan odio ac lectus mollis finibus.</p>
23   <h3>padding: 5px 20px;</h3>
24   <p class="dua">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
25     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
26     Nullam accumsan odio ac lectus mollis finibus.</p>
27   <h3>padding: 5px 30px 10px;</h3>
28   <p class="tiga">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
29     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
30     Nullam accumsan odio ac lectus mollis finibus.</p>
31   <h3>padding: 5px 12px 25px 35px;</h3>
32   <p class="empat">Lorem ipsum dolor sit amet, consectetur adipiscing elit.
33     Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna.
34     Nullam accumsan odio ac lectus mollis finibus.</p>
35 </body>
36 </html>

```



Gambar: Contoh penggunaan property padding shorthand notation

Pada setiap paragraf saya menulis berbagai variasi padding. Silahkan coba modifikasi nilai tersebut dan pelajari hasilnya.

Terlepas dari penulisan *shorthand notation*, perhatikan bahwa lebar setiap paragraf berbeda-beda, padahal kita menggunakan property `width: 400px`. Ini terjadi karena lebar element **berbeda** dengan lebar konten.

Property `width: 400px` adalah lebar dari konten, yang dalam contoh di atas merupakan lebar dari teks saja, sedangkan untuk lebar keseluruhan element (yang berwarna biru) didapat dari lebar konten ditambah dengan lebar padding kiri dan kanan. Perhitungan ini akan kembali kita bahas setelah kita mempelajari margin sesaat lagi.

8.5. Property border

Setelah konten (`width` dan `height`) serta `padding`, lapisan berikutnya dari box model adalah `border`. **Border** merupakan garis batas luar dari sebuah element. Kita bisa mengatur berbagai aspek tampilan dari border ini, mulai dari posisi, jenis, warna, hingga bentuk border.

Border Positioning

Sama seperti `padding`, `border` juga bisa diatur menurut arah posisinya, apakah itu `top`, `bottom`, `left` dan `right`. Berikut property yang digunakan:

- `border-top`
- `border-bottom`
- `border-left`
- `border-right`

Namun sebelum membahas cara penggunaannya, kita lihat terlebih dahulu apa yang disebut dengan *border longhand notation*.

Border Longhand Notation (Width, Style dan Color)

Longhand notation adalah kebalikan dari *shorthand notation*. Longhand notation adalah cara penulisan property yang lebih detail dan juga lebih panjang. Ini perlu kita bahas terlebih dahulu karena property `border` memiliki nilai yang bervariasi.

Menggunakan *border longhand notation*, kita bisa mengatur 3 aspek dari sebuah border, yakni **width**, **style**, dan **color**.

Border width

Border width dipakai untuk mengatur tebal border. Nilainya menggunakan satuan `length` seperti `px`, `em`, `rem` atau menggunakan nilai keyword: `thin`, `medium` dan `thick`. Jika tidak ditulis, nilai default web browser adalah `medium`.

Border Style

Border style berfungsi untuk menentukan jenis tampilan dari border, nilainya adalah salah satu dari keyword: `none`, `hidden`, `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, dan

outset. Jika tidak ditulis, nilai default web browser adalah none (tidak ditampilkan).

Border Color

Border color bertujuan untuk mengatur warna dari border. Satuannya adalah satuan color (warna) seperti RGB atau HSL. Apabila property ini tidak ditulis, nilai default diambil dari warna text (default: black).

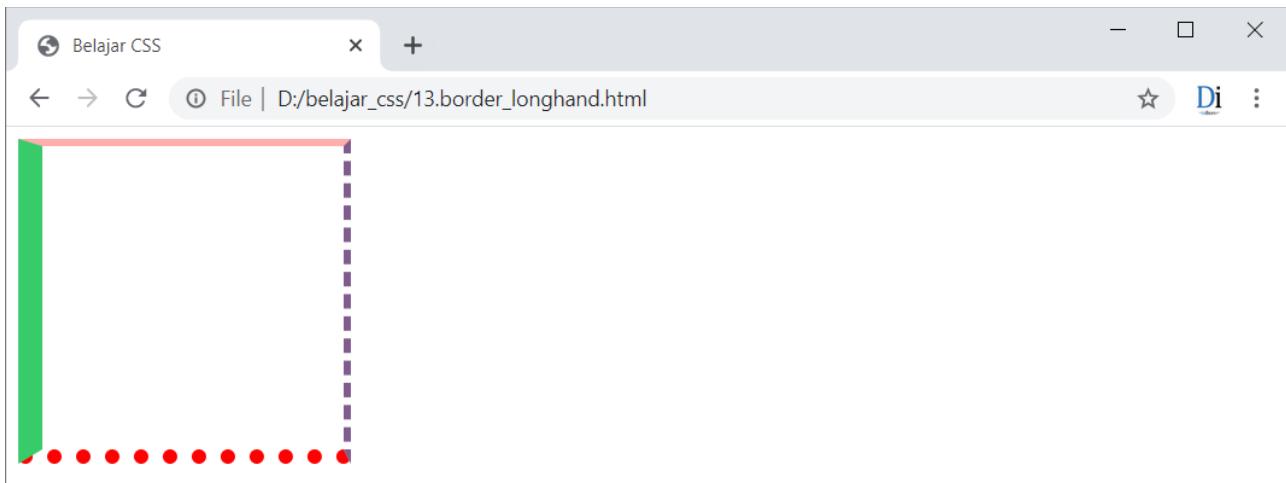
Ketiga pengaturan border ini (width, style dan color) bisa dipakai sekaligus atau terpisah. Sebagai contoh, untuk mengatur border yang berada di bagian atas sebuah element, kita membutuhkan property berikut:

```
border-top-width
border-top-style
border-top-color
```

Itu baru untuk sisi top (atas) saja. Bagaimana dengan sisi right, bottom dan left? Untuk membuat border utuh pada sebuah element kita perlu 12 property seperti contoh berikut:

13.border_longhand.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 200px;
9              height: 200px;
10
11             border-top-width: 5px;
12             border-top-style: solid;
13             border-top-color: #FAA;
14
15             border-right-width: 5px;
16             border-right-style: dashed;
17             border-right-color: #7A5587;
18
19             border-bottom-width: 10px;
20             border-bottom-style: dotted;
21             border-bottom-color: red;
22
23             border-left-width: 1em;
24             border-left-style: solid;
25             border-left-color: rgb(50,200,100);
26         }
27     </style>
28 </head>
29 <body>
30     <div></div>
31 </body>
32 </html>
```



Gambar: Contoh penggunaan property border longhand notation

Dalam kode CSS ini, saya mengatur jenis border untuk setiap arah menggunakan berbagai variasi nilai. Inilah yang dimaksud dengan penulisan panjang atau *longhand notation*.

Pada prakteknya penulisan panjang ini relatif jarang dipakai karena sebuah border akan lebih rapi jika dibuat seragam di setiap sisi.

Border Shorthand Notation

Menulis 12 property border memang cukup panjang. Untungnya, CSS menyediakan penulisan pendek (*shorthand notation*) untuk border. Tidak hanya 1, tapi 2 jenis shorthand notation.

Penulisan singkat pertama adalah dengan menyingkat 3 jenis property border untuk setiap arah. Sebagai contoh, apabila saya ingin membuat border-top sebesar 5px, solid dan berwarna merah, penulisannya adalah sebagai berikut:

```
border-top: 1px solid red;
```

Urutan dari ketiga nilai ini tidak menjadi masalah (boleh terbaik), selama dipisah dengan tanda spasi. Property di atas bisa juga ditulis dengan:

```
border-top: red 1px solid;
```

atau:

```
border-top: solid red 1px;
```

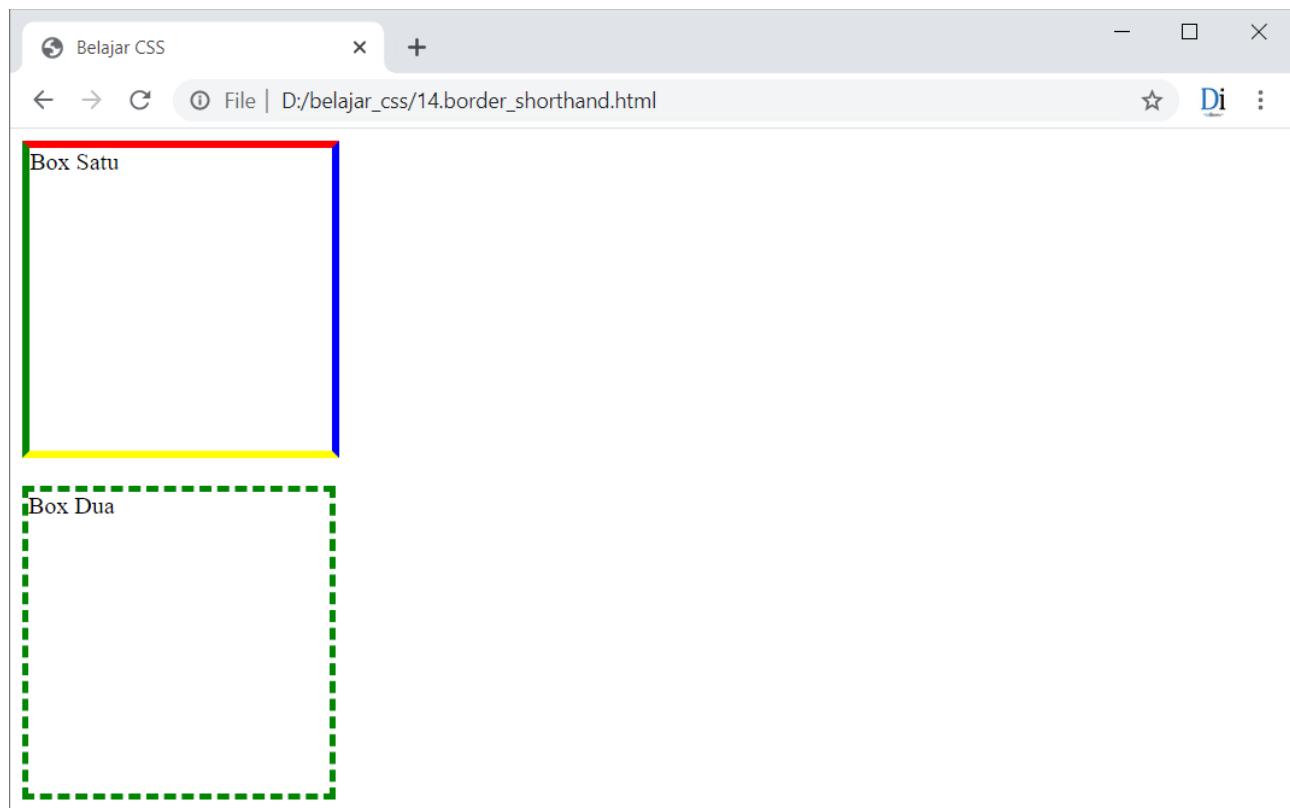
Jika kita ingin membuat border yang seragam untuk semua posisi, bisa memakai penulisan singkat kedua, yakni hanya property border saja. Sebagai contoh, untuk membuat border berukuran 5px, solid, dan berwarna hijau di semua sisi, penulisannya adalah:

```
border: 5px solid green;
```

Berikut contoh praktek dari penulisan ini:

14.border_shorthand.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div{
8       width: 200px;
9       height: 200px;
10    }
11    .satu {
12      border-top: 5px solid red;
13      border-bottom: 5px solid yellow;
14      border-left: 5px solid green;
15      border-right: 5px solid blue;
16    }
17    .dua {
18      border: 4px dashed green;
19    }
20  </style>
21 </head>
22 <body>
23   <div class="satu">Box Satu</div>
24   <br>
25   <div class="dua">Box Dua</div>
26 </body>
27 </html>
```



Gambar: Contoh penggunaan property border shorthand notation

Penulisan border shorthand notation seperti ini lebih singkat dibandingkan dengan longhand notation.

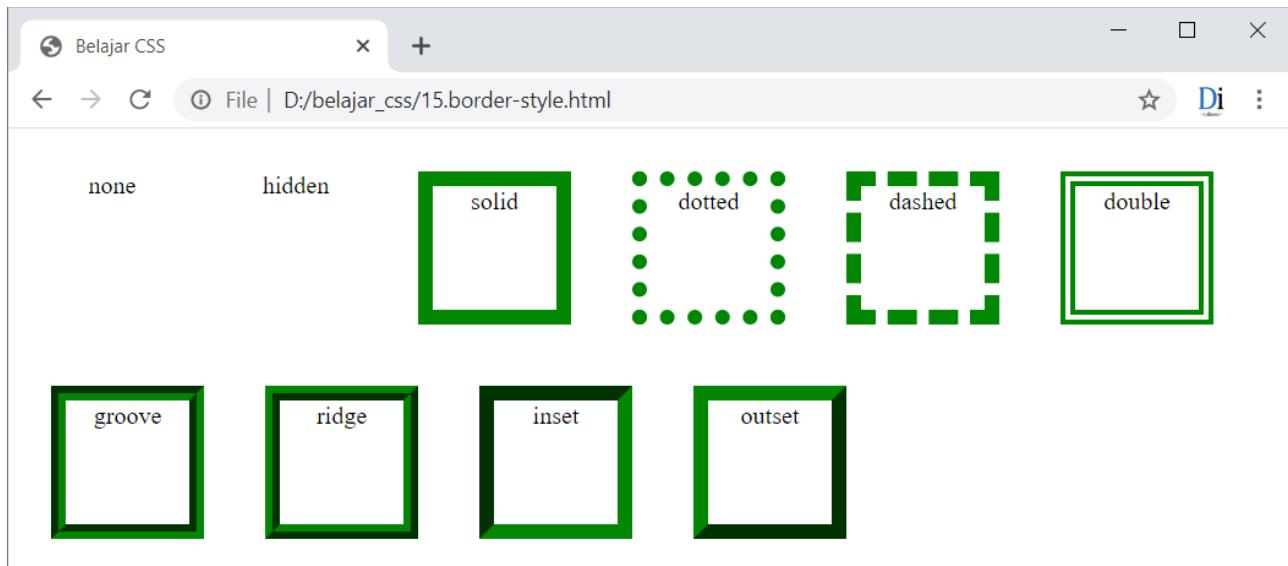
Jenis Border Style

Ketika mempelajari border longhand notation, kita belum membahas ke-10 efek dari property `border-style`. Berikut tampilan perbandingan untuk property `border-style`:

15.border-style.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 80px;
9              height: 80px;
10             float: left;
11             margin: 20px;
12             text-align: center;
13         }
14         .satu      { border: 10px none green; }
15         .dua       { border: 10px hidden green; }
16         .tiga      { border: 10px solid green; }
17         .empat     { border: 10px dotted green; }
18         .lima      { border: 10px dashed green; }
19         .enam      { border: 10px double green; }
20         .tujuh     { border: 10px groove green; }
21         .delapan   { border: 10px ridge green; }
22         .sembilan  { border: 10px inset green; }
23         .sepuluh   { border: 10px outset green; }
24     </style>
25 </head>
26 <body>
27     <div class="satu">none</div>
28     <div class="dua">hidden</div>
29     <div class="tiga">solid</div>
30     <div class="empat">dotted</div>
31     <div class="lima">dashed</div>
32     <div class="enam">double</div>
33     <div class="tujuh">groove</div>
34     <div class="delapan">ridge</div>
35     <div class="sembilan">inset</div>
36     <div class="sepuluh">outset</div>
37 </body>
38 </html>
```



Gambar: Tampilan berbagai jenis border-style

Dalam kode CSS ini terdapat tambahan property `float:left` yang nantinya akan kita bahas pada bab CSS positioning. Property `float:left` digunakan untuk mengatur posisi element menjadi ke arah kiri (left).

8.6. Property border-radius CSS3

Membuat *rounding corner* (garis tepi berbentuk bulat), sudah lama diimpikan web desainer. Berbagai teknik digunakan untuk menghasilkan efek ini, mulai dari memakai gambar hingga menggunakan JavaScript. Untungnya, efek yang sama sudah bisa dibuat dengan mudah.

CSS3 menyediakan property **border-radius** untuk menghasilkan *rounding corner*. Property ini hadir dengan penulisan singkat (*shorthand notation*) atau penulisan panjang (*longhand notation*). Kita akan membahas longhand notation terlebih dahulu.

Border-Radius Longhand Notation

Mirip dengan property lainnya dalam box model, `border-radius` memiliki 4 penulisan untuk setiap arah, yakni:

- `border-top-left-radius`
- `border-top-right-radius`
- `border-bottom-right-radius`
- `border-bottom-left-radius`

Perhatikan cara penulisan arah ini, karena kita berurusan dengan sudut (*corner*) maka arahnya adalah di bagian sudut, yakni kiri atas (top-left), kanan atas (top-right), kanan bawah (bottom-right) dan kiri bawah (bottom-left).

Nilai yang bisa diinput berupa satuan length seperti px, em, atau %. Berikut contoh penggunaan property longhand **border-radius**:

16.border-radius_longhand.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 200px;
9              height: 200px;
10             border: 5px solid red;
11             margin: 20px;
12             float: left;
13             line-height: 200px;
14             text-align: center;
15         }
16         .satu {
17             border-top-left-radius: 10px;
18             border-top-right-radius: 20px;
19             border-bottom-right-radius: 30px;
20             border-bottom-left-radius: 40px;
21         }
22         .dua {
23             border-top-left-radius: 1em;
24             border-top-right-radius: 2em;
25             border-bottom-right-radius: 3em;
26             border-bottom-left-radius: 4em;
27         }
28     </style>
29 </head>
30 <body>
31     <div class="satu">Box 1</div>
32     <div class="dua">Box 2</div>
33 </body>
34 </html>
```

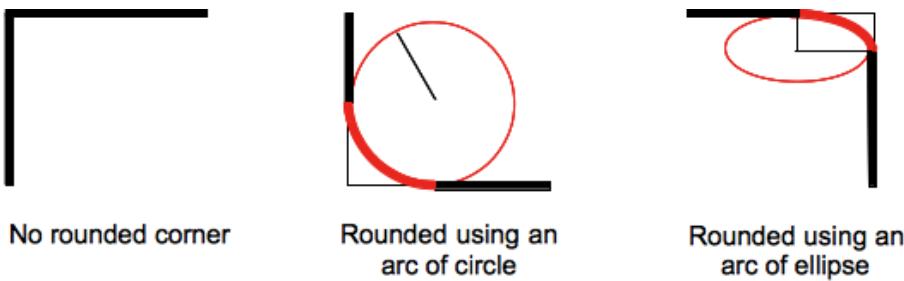


Gambar: Contoh penggunaan property border-radius longhand notation

Saya membuat 2 buah tag <div> dengan border radius. Box 1 menggunakan satuan pixel, sedangkan box 2 menggunakan satuan persen. Warna border tetap harus di set dari property border biasa, yakni border: 5px solid red yang ada di selector div.

Apa arti nilai border-radius ini?

Sekilas kita bisa lihat bahwa semakin besar nilai border-radius, semakin besar juga lingkaran di sudut box. Angka ini mencerminkan besar jari-jari lingkaran seperti yang ada di bagian tengah gambar berikut:



Gambar: Penjelasan nilai dari property border-radius (sumber: www.w3c.org)

Tidak hanya lingkaran, kita juga bisa membuat sudut elips seperti gambar paling kanan. Caranya, isi 2 buah nilai ke dalam setiap property border-radius. Nilai pertama dipakai untuk jari-jari sumbu x (horizontal) dan nilai kedua untuk jari-jari sumbu y (vertikal).

Sebagai contoh, border-top-left-radius: 40px 10px akan membuat sudut berbentuk elips yang 'besar' pada sumbu x (40px), dan 'kecil' pada sumbu y (10px). Berikut contoh prakteknya:

17.border-radius_longhand_elips.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 200px;
9              height: 200px;
10             border: 5px solid red;
11             margin: 20px;
12             float: left;
13             line-height: 200px;
14             text-align: center;
15         }
16         .satu {
17             border-top-left-radius: 50px 20px;
18             border-top-right-radius: 20px 40px;
19             border-bottom-right-radius: 30px 5px;
20             border-bottom-left-radius: 40px 10px;
21         }
22         .dua {
23             border-top-left-radius: 1em 3em;

```

```

24     border-top-right-radius: 2em 5em;
25     border-bottom-right-radius: 3em 4em;
26     border-bottom-left-radius: 4em 1em;
27   }
28 </style>
29 </head>
30 <body>
31   <div class="satu">Box 1</div>
32   <div class="dua">Box 2</div>
33 </body>
34 </html>

```



Gambar: Cara membuat border radius berbentuk elips

Border-Radius Shorthand Notation

Untuk penulisan singkat border radius (*shorthand notation*), kita hanya perlu property `border-radius`. Nilai yang bisa diinput mirip property `padding`, yakni bisa diisi dengan 1, 2, 3 atau 4 nilai.

Berikut contoh nilai `border-radius` dan sudut mana saja yang menggunakan nilai tersebut:

- `border-radius: 10px;` keempat sudut akan menggunakan nilai `border-radius` sebesar 10px.
- `border-radius: 10px 5px;` sudut top-left dan bottom-right memiliki jari-jari 10px, serta sudut top-right dan bottom-left memiliki jari-jari 5px.
- `border-radius: 5px 10px 15px;` sudut top-left memiliki jari-jari 5px, sudut top-right dan bottom-left 10px, serta bottom-right sebesar 15px.
- `border-radius: 5px 10px 15px 20px;` sudut top-left memiliki jari-jari 5px, sudut top-right 10px, sudut bottom-right 15px, dan sudut bottom-left 20px.

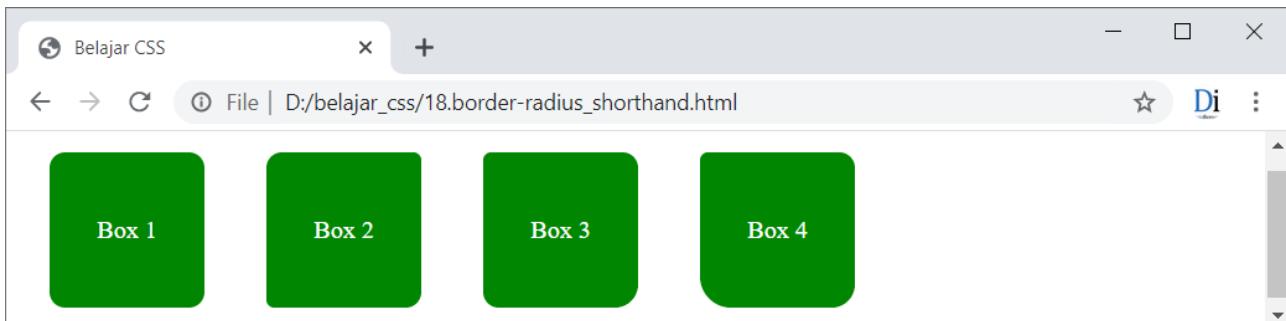
Urutan penulisan nama sudut tetap menggunakan arah jarum jam, atau singkatan **TRouBLE** (Top Right Bottom Left). Berikut contoh prakteknya:

18.border-radius_shorthand.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 100px;
9              height: 100px;
10             margin: 20px;
11             background-color: green;
12             color: white;
13             line-height: 100px;
14             text-align: center;
15             float: left;
16         }
17         .satu { border-radius: 10px; }
18         .dua { border-radius: 10px 5px; }
19         .tiga { border-radius: 5px 10px 15px; }
20         .empat { border-radius: 5px 10px 15px 20px; }
21     </style>
22 </head>
23 <body>
24     <div class="satu">Box 1</div>
25     <div class="dua">Box 2</div>
26     <div class="tiga">Box 3</div>
27     <div class="empat">Box 4</div>
28 </body>
29 </html>

```



Gambar: Contoh penggunaan property border-radius shorthand notation

Selain border-radius berwarna hijau, saya juga menulis warna background dengan warna yang sama, sehingga seolah-olah box tersebut menyatu dengan bordernya.

Bagaimana cara membuat elips menggunakan shorthand notation?

Caranya, pakai tanda '/' sebagai pemisah nilai. Nilai sebelum tanda '/' adalah untuk jari-jari sumbu x, dan nilai setelahnya untuk jari-jari sumbu y. Sebagai contoh, property border-radius: 1em/5em artinya sama dengan:

```
border-top-left-radius: 1em 5em;
border-top-right-radius: 1em 5em;
border-bottom-right-radius: 1em 5em;
border-bottom-left-radius: 1em 5em;
```

Untuk mengatur posisi setiap sudut, penulisannya menjadi lebih kompleks. Misalnya `border-radius: 4px 3px 6px / 2px 4px;`, akan sama dengan:

```
border-top-left-radius: 4px 2px;
border-top-right-radius: 3px 4px;
border-bottom-right-radius: 6px 2px;
border-bottom-left-radius: 3px 4px;
```

Penulisan di atas memang cukup buat bingung, agar lebih mudah dipahami kita harus pisah nilai-nilai di atas menjadi 2 bagian:

3 nilai sebelum tanda garis miring: "4px 3px 6px" dipakai untuk mengatur besar jari-jari sumbu x untuk ke-4 sudut. Tapi karena kita hanya menulis 3 nilai, berlaku rumus penulisan shorthand, yakni nilai pertama untuk top-left, nilai kedua untuk top-right dan bottom-left, serta nilai ketiga untuk bottom-right.

2 nilai setelah garis miring: "2px 4px" adalah untuk jari-jari sumbu y, dan karena hanya ditulis 2 nilai, maka nilai pertama untuk top-left dan bottom-right, serta nilai kedua untuk top-right dan bottom-left.

Dengan menggabung keduanya, di dapatlah hasil longhand notation seperti di atas. Berikut contoh penggunaan shorthand `border-radius` untuk membuat sudut elips:

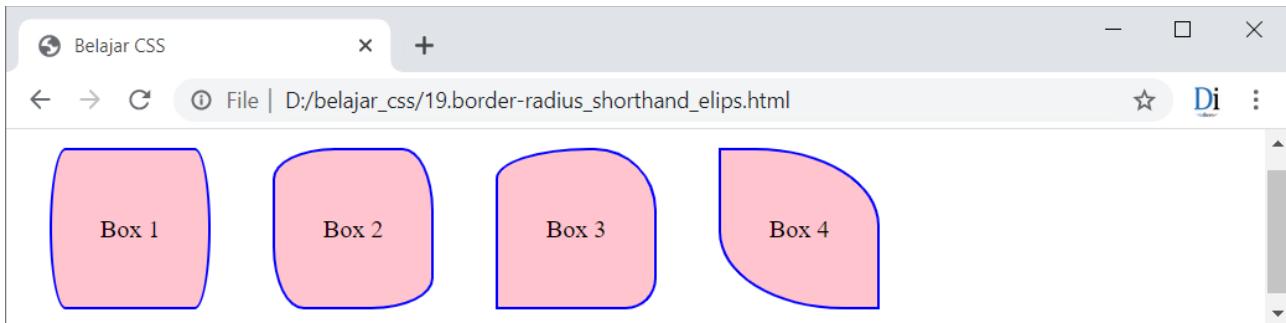
19.border-radius_shorthand_elips.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div{
8              width: 100px;
9              height: 100px;
10             border: 2px solid blue;
11             float: left;
12             margin: 20px;
13             background-color: pink;
14             line-height: 100px;
15             text-align: center;
16             float: left;
17         }
18         .satu { border-radius: 1em/5em; }
19         .dua { border-radius: 40px 20px / 20px 40px; }
20         .tiga { border-radius: 60px 40px 20px 0px / 20px 40px; }
21         .empat { border-radius: 0 80px 0 / 0 50px 0 }
22     </style>
```

```

23 </head>
24 <body>
25   <div class="satu">Box 1</div>
26   <div class="dua">Box 2</div>
27   <div class="tiga">Box 3</div>
28   <div class="empat">Box 4</div>
29 </body>
30 </html>

```



Gambar: Contoh penggunaan property border-radius shorthand notation untuk sudut elips

8.7. Property margin

Lapisan terakhir dari box model adalah **margin**. **Margin** merupakan 'spasi' atau 'ruang kosong' yang berada di sisi paling luar sebuah element. Margin bersifat transparan (tidak bisa diwarnai) dan dipakai sebagai pemisah antara satu element dengan element lain.

Kita sudah menyinggung sedikit tentang margin dalam bab typography. Dimana `margin-bottom` saya pakai untuk mengatur spasi/jarak antar paragraf. Seperti yang bisa ditebak, selain `margin-bottom`, juga terdapat `margin-top`, `margin-right`, dan `margin-left`. Satuan yang digunakan adalah satuan 'length' seperti pixel, em, rem, atau persen.

Margin mendukung penulisan *longhand notation* dan *shorthand notation*. Penggunaan keduanya mirip dengan padding, dimana longhand notation dipakai untuk mengatur setiap sisi dengan property: `margin-top`, `margin-right`, `margin-bottom` dan `margin-left`. Sedangkan untuk shorthand notation cukup dari property `margin` yang bisa diisi 1, 2, 3 atau 4 nilai.

Karena aturan penulisan margin hampir sama dengan padding (masih ingat **TRouBLE?**), langsung saja kita lihat contoh penggunaannya:

20.margin1.html

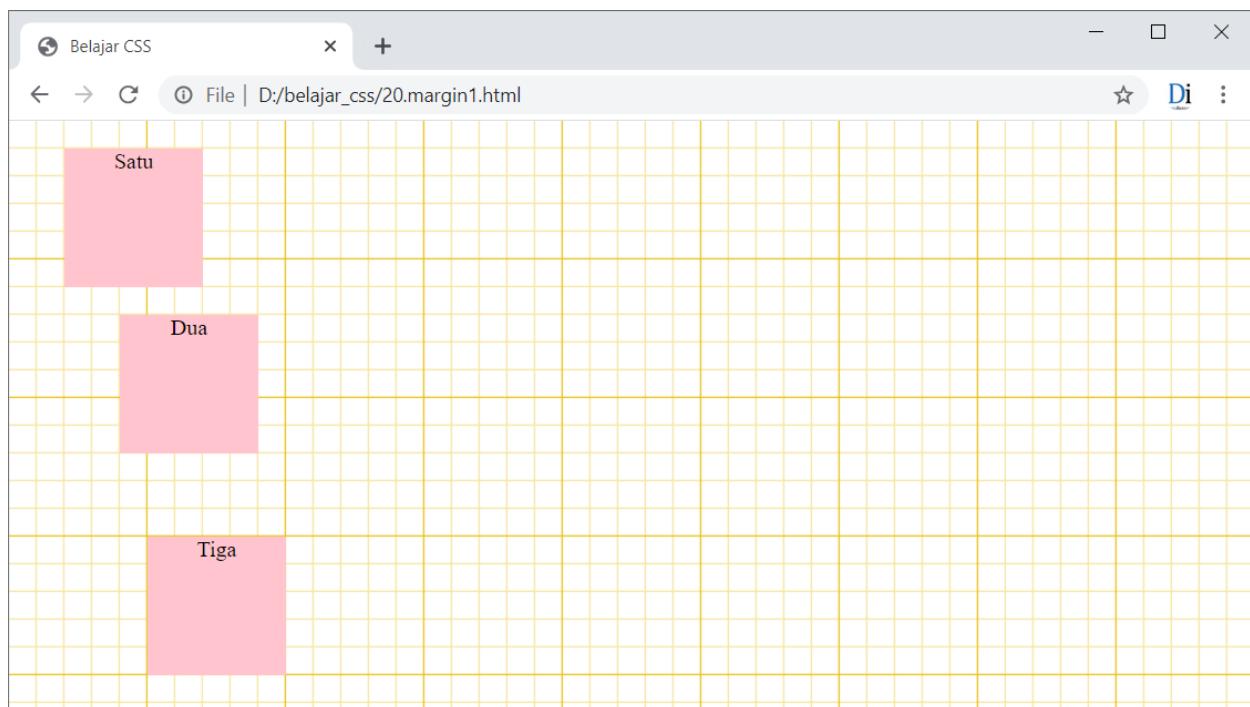
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      * {
8        margin: 0;

```

CSS Box Model

```
9      padding: 0;
10     }
11     body {
12       background: url('bg_grid.png');
13     }
14     div{
15       width: 100px;
16       height: 100px;
17       background-color: pink;
18       text-align: center;
19     }
20     .satu {
21       margin-top: 20px;
22       margin-left: 40px;
23       margin-bottom: 20px;
24     }
25     .dua { margin: 0 0 60px 80px }
26     .tiga { margin: 0 100px; }
27   </style>
28 </head>
29 <body>
30   <div class="satu">Satu</div>
31   <div class="dua">Dua</div>
32   <div class="tiga">Tiga</div>
33 </body>
34 </html>
```



Gambar: Contoh penggunaan property margin

Pada kode ini saya memakai gambar background `bg_grid.png` untuk memperjelas perhitungan ukuran margin. Ini berasal dari property `background` di baris 12. Gambar ini juga tersedia di dalam file `belajar_css.zip`.

Di baris pertama kode CSS terdapat property `margin:0` dan `padding:0` untuk seluruh element (menggunakan universal selector `*`). Ini dilakukan agar margin dan padding bawaan web browser tidak ikut mempengaruhi kode kita.

Ketiga box berukuran 100×100 pixel, sehingga akan mengambil tempat 5×5 kotak grid. Satu kotak grid dari gambar `bg_grid.png` berukuran 20×20 pixel.

Pada box `class="satu"`, saya menggunakan `margin-top: 20px`, `margin-left: 40px`, dan `margin-bottom: 20px`.

Untuk box `class="dua"` terdapat property `margin: 0 0 60px 80px`. Penulisan property singkat ini sama artinya dengan:

```
margin-top: 0  
margin-right: 0  
margin-bottom: 60px  
margin-left: 80px
```

Sedangkan dalam box `class="tiga"`, property yang digunakan adalah `margin: 0 100px`. Ini sama dengan:

```
margin-top: 0  
margin-right: 100px  
margin-bottom: 0px  
margin-left: 100px
```

Gambar `bg_grid.png` bisa membantu kita dalam mengukur setiap margin ini.

Jarak antara `<div class="satu">` dengan `<div class="dua">` adalah sebesar 20 pixel, ini berasal dari property `margin-bottom: 20px` pada `class="satu"`. Sedangkan jarak antara `<div class="dua">` dengan `<div class="tiga">` adalah sebesar 60 pixel yang berasal dari margin `class="dua"`.

Sebagai latihan, bisakah anda merancang kode program dengan tampilan berikut?



Gambar: Dapatkah anda membuat tampilan ini?

Selamat!, jika berhasil membuatnya. Berikut kode yang saya gunakan:

21.margin2.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
8              margin: 0;
9              padding: 0;
10         }
11         body {
12             background: url('bg_grid.png');
13         }
14         div{
15             width: 500px;
16             height: 40px;
17             background-color: pink;
18             text-align: center;
19         }
20         .satu { margin: 20px 0 20px 20px; }
21         .dua { margin: 0 0 20px 40px; }
22         .tiga { margin: 0 0 20px 60px; }
23         .empat { margin: 0 0 20px 80px; }
24         .lima { margin: 0 0 20px 100px; }
25     </style>
26 </head>
27 <body>
28     <div class="satu">Satu</div>
29     <div class="dua">Dua</div>
30     <div class="tiga">Tiga</div>
31     <div class="empat">Empat</div>
32     <div class="lima">Lima</div>
33 </body>
34 </html>
```

Margin Collapsing

Margin collapsing adalah sebuah konsep yang cukup sederhana namun bisa membuat pusing jika tidak dipahami. Langsung saja kita lihat contohnya:

22.margin_collapsing1.html

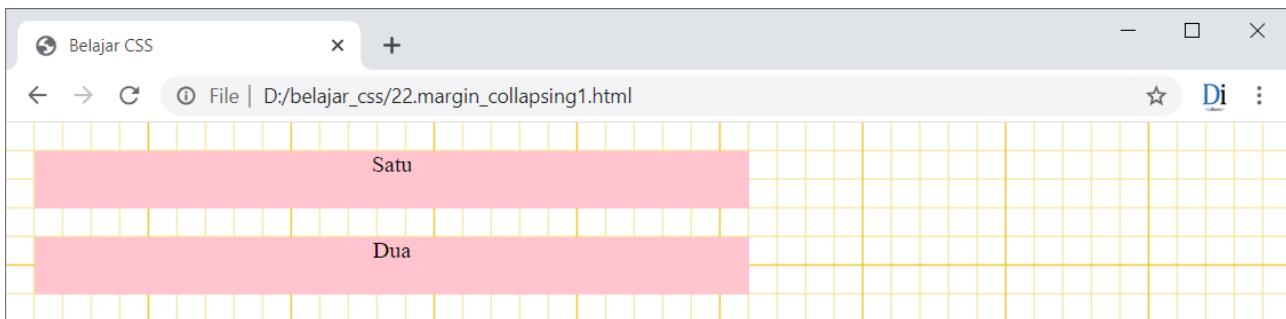
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
```

```

8      margin: 0;
9      padding: 0;
10     }
11     body {
12         background: url('bg_grid.png');
13     }
14     div{
15         width: 500px;
16         height: 40px;
17         background-color: pink;
18         text-align: center;
19     }
20     .satu { margin:20px; }
21     .dua { margin:20px; }
22   </style>
23 </head>
24 <body>
25   <div class="satu">Satu</div>
26   <div class="dua">Dua</div>
27 </body>
28 </html>

```



Gambar: Berapakah jarak (margin) antara box pertama dan kedua?

Dengan pemahaman kita tentang margin dan cara penulisan *shorthand notation*, bisakah anda melihat hal yang agak beda antara kode CSS dengan hasil tampilan di atas? Fokuskan kepada jarak (margin) antara div pertama dan kedua.

Jika melihat dari kode CSS, jarak kedua div seharusnya 40 pixel, yang berasal dari `margin-bottom` div pertama, dan `margin-top` div kedua, masing-masing property ini bernilai 20px. Dengan demikian, jarak kedua div ini harusnya 40px (20px+20px). Namun seperti yang terlihat, jarak kedua kotak hanya 20px. Kemana 20px sisanya?

Inilah yang disebut dengan *margin collapsing*. Web browser akan menggugurkan (*collapse*) salah satu margin jika keduanya saling bersentuhan. Yang perlu dicatat, ini hanya terjadi pada vertical margin, yakni antara `margin-top` dan `margin-bottom`. Untuk margin horizontal (`margin-left` dan `margin-right`), efek *margin collapsing* tidak terjadi.

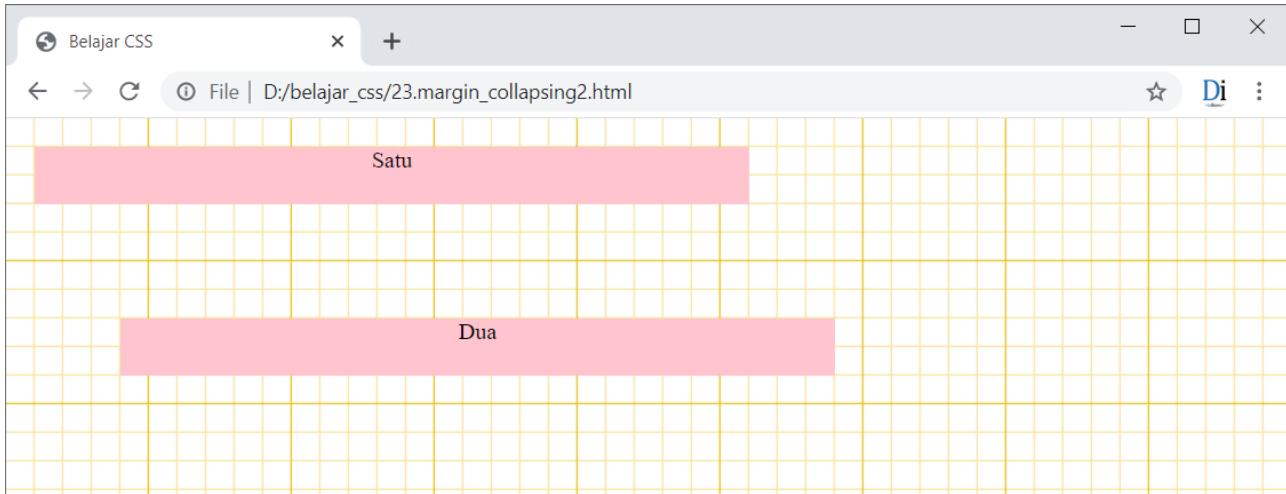
Dalam contoh ini, saya menulis nilai yang sama untuk kedua div (20px). Bagaimana jika keduanya berbeda? Berapa margin yang dihasilkan? Mari kita coba:

23.margin_collapsing2.html

```

1  ...
2      .satu { margin:20px; }
3      .dua { margin:80px; }
4  ...

```



Gambar: Contoh lain dari margin collapsing

Kali ini margin div kedua saya set dengan nilai 80px, sedangkan div pertama tetap 20px. Dengan demikian, margin yang 'bersentuhan' adalah 20px `margin-bottom` dari div pertama, dan 80px `margin-top` dari div kedua.

Ketika 2 buah vertical margin saling bersentuhan, **margin yang paling besar**-lah yang akan dipakai. Dari contoh di atas, 80px vs 20px, sehingga 80px yang akan 'menang'. Dengan demikian jarak antara box pertama dengan box kedua akan sebesar 80px.

Konsep *margin collapsing* bukanlah bug atau kesalahan program, tetapi memang fitur bawaan CSS. Ini memudahkan proses menampilkan margin dalam banyak kasus.

Contohnya, ketika kita membuat sebuah artikel yang terdiri dari beberapa paragraf, margin collapsing akan membuat jarak setiap paragraf menjadi konsisten (masing-masing tag `<p>` memiliki `margin-top` dan `margin-bottom`). Jika tidak, kita perlu me-reset salah satu margin setiap kali paragraf ini 'bersentuhan'.

Margin collapsing juga berpengaruh ke dalam nested element, yakni element yang ada di dalam element lain, seperti contoh berikut:

24.margin_collapsing3.html

```

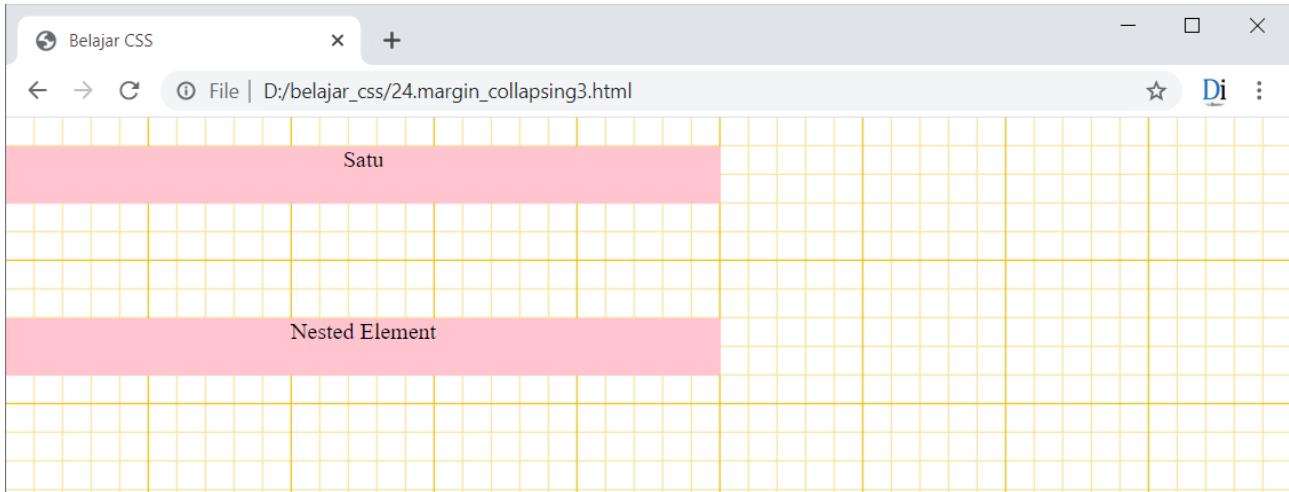
1  ...
2  <style>
3  ...
4      .satu { margin:20px 0; }
5      .dua { margin:80px 0; }
6      .tiga {margin:60px 0;}
7  </style>

```

```

8  </head>
9  <body>
10 <div class="satu">Satu</div>
11 <div class="dua">
12   <div class="tiga">Nested Element</div>
13 </div>
14 </body>
15 </html>

```



Gambar: Efek margin collapsing pada nested element (element yang saling bertumpuk)

Kali ini saya menempatkan div `class="tiga"` di dalam div `class="dua"`. Perhatikan margin-nya.

Div `class="tiga"` memiliki margin-top sebesar 60px, dan karena ia berada di dalam div `class="dua"` yang memiliki margin-top 80px, seharusnya jarak antara div `class="satu"` ke div `class="tiga"` adalah $60\text{px} + 80\text{px} + 20\text{px} = 160\text{px}$ (20px berasal dari margin-bottom div `class="satu"`).

Tapi seperti yang bisa kita lihat, margin terakhir hanya 80px. Ini terjadi karena efek margin collapsing dari ketiga div.

Margin Auto

Salah satu penggunaan margin yang hampir selalu dipakai pada setiap desain web adalah `margin:auto`. Ini berfungsi untuk membuat element berada di tengah secara horizontal (*horizontal centering*).

Nilai `auto` untuk margin berarti kita menginstruksikan web browser agar secara otomatis menghitung nilainya. Secara default ini sama dengan `margin:0`, tetapi khusus untuk element yang memiliki lebar (terdapat property `width`), nilai `auto` akan mengambil 'sisa' ruang yang tersedia.

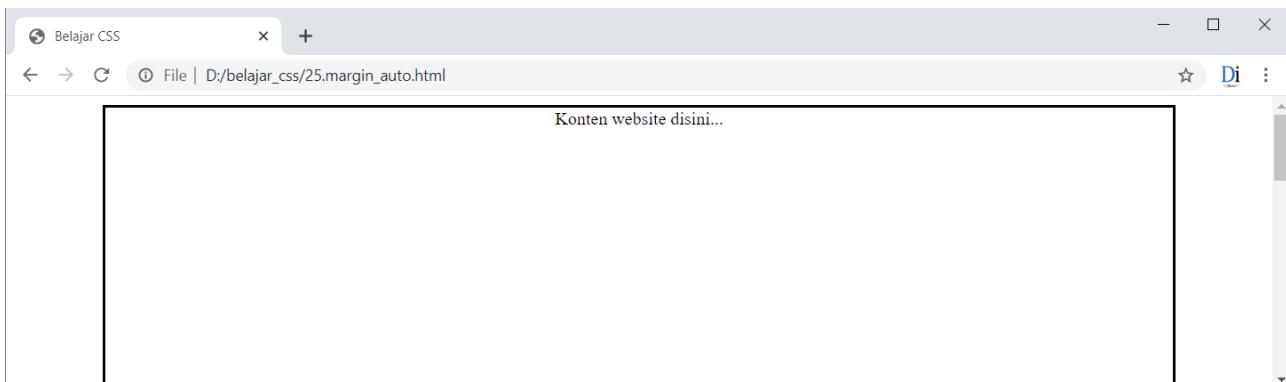
Dengan memberikan nilai `auto` ke horizontal margin (`margin-left` dan `margin-right`), sisa

ruang yang ada akan dibagi sama besar. Hasilnya, element tersebut berada tepat di tengah (center) seperti contoh berikut:

25.margin_auto.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 960px;
9              height: 1000px;
10             border: 3px solid black;
11             margin: 0 auto;
12             text-align: center;
13         }
14     </style>
15 </head>
16 <body>
17     <div>Konten website disini...</div>
18 </body>
19 </html>
```



Gambar: Contoh penggunaan nilai margin: auto

Silahkan tes besar-kecilkan web browser. Selama lebarnya lebih besar dari 960 pixel, tag <div> tersebut akan selalu berada di tengah. Teknik seperti ini banyak dipakai dalam perancangan layout web.

Property `margin: 0 auto`, berarti buat 0 margin (tanpa margin) untuk sisi atas dan bawah, serta `auto` untuk margin kiri dan kanan. Property lain seperti `width`, `height`, dan `border` ditambah sebagai efek visual dari penggunaan `margin auto` ini.

Perlu diketahui bahwa nilai `auto` hanya berefek untuk horizontal margin saja (`margin-left` dan `margin-right`). Untuk vertical margin (`margin-top` dan `margin-bottom`), nilai ini tidak berefek apa-apa dan dalam kebanyakan kasus ini sama dengan `margin = 0`.

Harap bisa membedakan antara property `margin:0 auto` dengan `text-align:center`. Walaupun sama-sama mengetengahkan sebuah element, fungsinya berbeda.

Margin auto dipakai untuk mengatur element saat ini dan hanya bisa diterapkan ke *block level element*, sedangkan `text-align` untuk mengatur *inline level element*.

8.8. Content Width vs Element Width

Sampai di sini kita telah mempelajari inti dari **box model**, yakni konten (`width & height`), `padding`, `border` dan `margin`.

Semuanya telah bahas secara detail, namun masih saling terpisah. Kali ini kita akan lihat bagaimana jika semuanya disatukan dalam sebuah element. Bahasan pertama adalah tentang perbedaan antara lebar konten (`content width`) dengan lebar element (`element width`).

Property `width` dalam bahasa Indonesia berarti lebar. Jika sebuah element memiliki `width: 200px`, bukankah lebarnya akan sebesar 200px? Bisa 'iya', namun lebih sering 'tidak'.

Perhatikan contoh kode berikut ini:

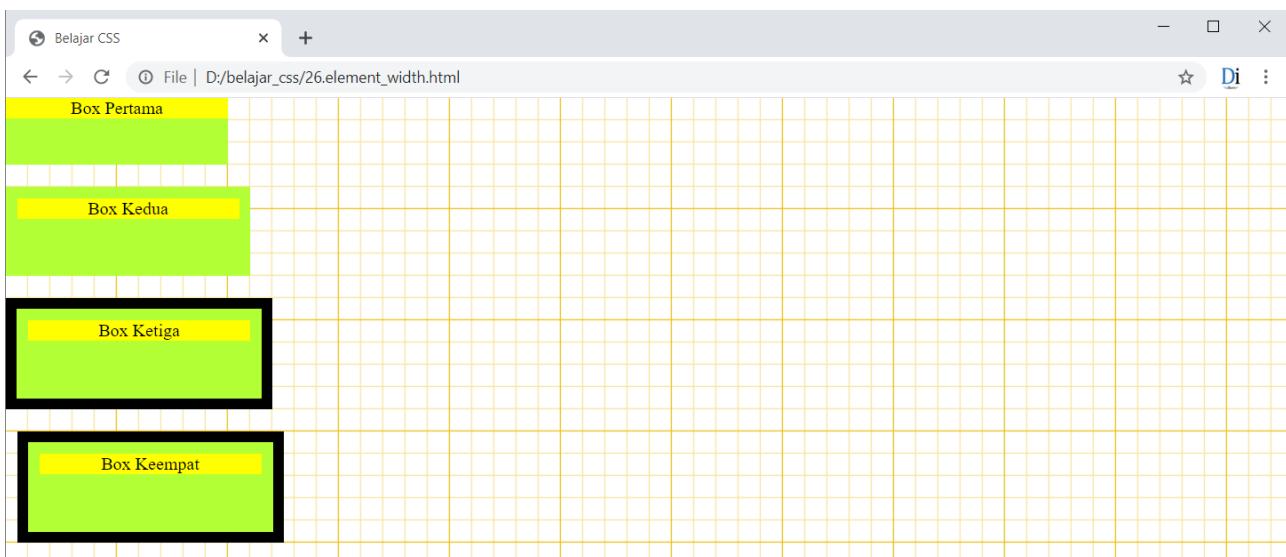
```
26.element_width.html

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
8              margin: 0;
9              padding: 0;
10         }
11     body {
12         background: url('bg_grid.png');
13     }
14     div {
15         background-color: GreenYellow;
16         text-align: center;
17         margin-bottom: 20px;
18     }
19     p {
20         background-color: yellow;
21     }
22     div.satu {
23         width: 200px;
24         height: 60px;
25     }
26     div.dua {
27         width: 200px;
28         height: 60px;
```

```

29     padding: 10px;
30 }
31 div.tiga {
32     width: 200px;
33     height: 60px;
34     padding: 10px;
35     border: 10px solid black;
36 }
37 div.empat {
38     width: 200px;
39     height: 60px;
40     padding: 10px;
41     border: 10px solid black;
42     margin: 10px;
43 }
44 </style>
45 </head>
46 <body>
47     <div class="satu"><p>Box Pertama</p></div>
48     <div class="dua"><p>Box Kedua</p></div>
49     <div class="tiga"><p>Box Ketiga</p></div>
50     <div class="empat"><p>Box Keempat</p></div>
51 </body>
52 </html>

```



Gambar: Penggunaan berbagai property box model

Saya masih menggunakan gambar background `bg_grid.png`, karena akan membantu kita mengukur lebar dan tinggi element. Selector pertama, yakni `* { margin: 0; padding: 0;}` digunakan untuk menghapus margin dan padding bawaan web browser.

Dalam kode HTML, saya membuat 4 div element yang di dalamnya terdapat sebuah paragraf. Setiap div menggunakan property CSS yang berbeda-beda. Agar mudah dibedakan, tag div menggunakan warna background hijau (GreenYellow) dan paragraf menggunakan warna background kuning (yellow).

Untuk box pertama (`class="satu"`), saya membuat property `width: 200px` dan `height: 60px`. Seperti yang terlihat, box ini tampil dengan lebar 200 pixel dan tinggi 60 pixel. Perhitungan ini sudah pas.

Pada box kedua (`class="dua"`), saya menambah property `padding: 10px`. Sekarang, kita bisa melihat efek tambahan. Paragraf di dalam box 'digeser' 10pixel dari setiap sisi dalam tag `<div>`, selain itu, ukuran total element bukan lagi 200×60pixel.

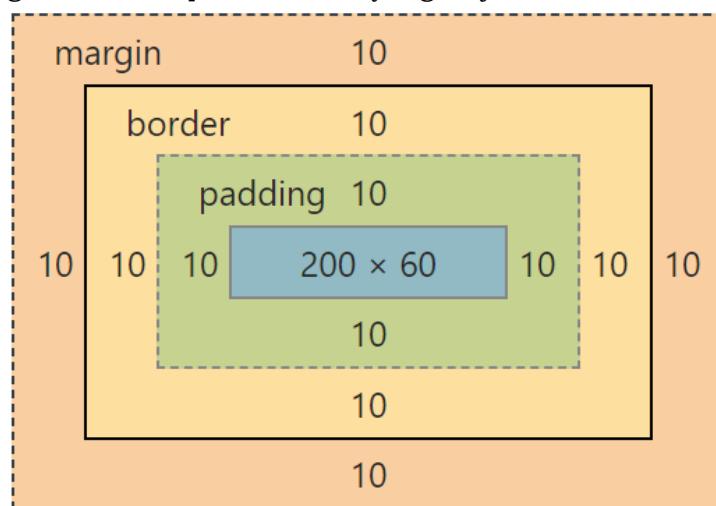
Property padding akan menambah ukuran element. Dalam box kedua, total lebar element (element width) adalah $200\text{px} + 20\text{px} = 220\text{px}$. Dari manakah angka 20pixel? Ini berasal dari `padding-left: 10px` dan `padding-right: 10px`.

Sekarang bisa terlihat walaupun saya membuat lebar konten (content width) sebesar 200px dari property `width:200px`, namun lebar element akan menjadi `width + padding` ($200\text{px} + 20\text{px}$). Hal yang sama juga berlaku untuk tinggi element, dimana akan menjadi 80px ($60\text{px}+20\text{px}$). Sehingga ukuran sebenarnya dari box ini adalah 220x80pixel.

Dalam box ketiga (`class="tiga"`), saya menambahkan lagi property `border: 10px solid black`. Hasilnya, terdapat sebuah garis hitam setebal 10px di sisi luar box. Property border ini ikut menambah lebar dan tinggi element. Sekarang lebar element adalah 240px yang berasal dari `width 200px + padding 20px` (untuk kedua sisi) + `border 20px` (untuk kedua sisi).

Terakhir, pada box keempat (`class="empat"`) saya menambah `margin: 10px`. Walaupun margin bersifat transparan, tapi ini akan mendorong box sejauh 10px dari setiap sisi. Sekarang total lebar element (+margin) menjadi 260px.

Gambar berikut mengilustrasikan penambahan yang terjadi:

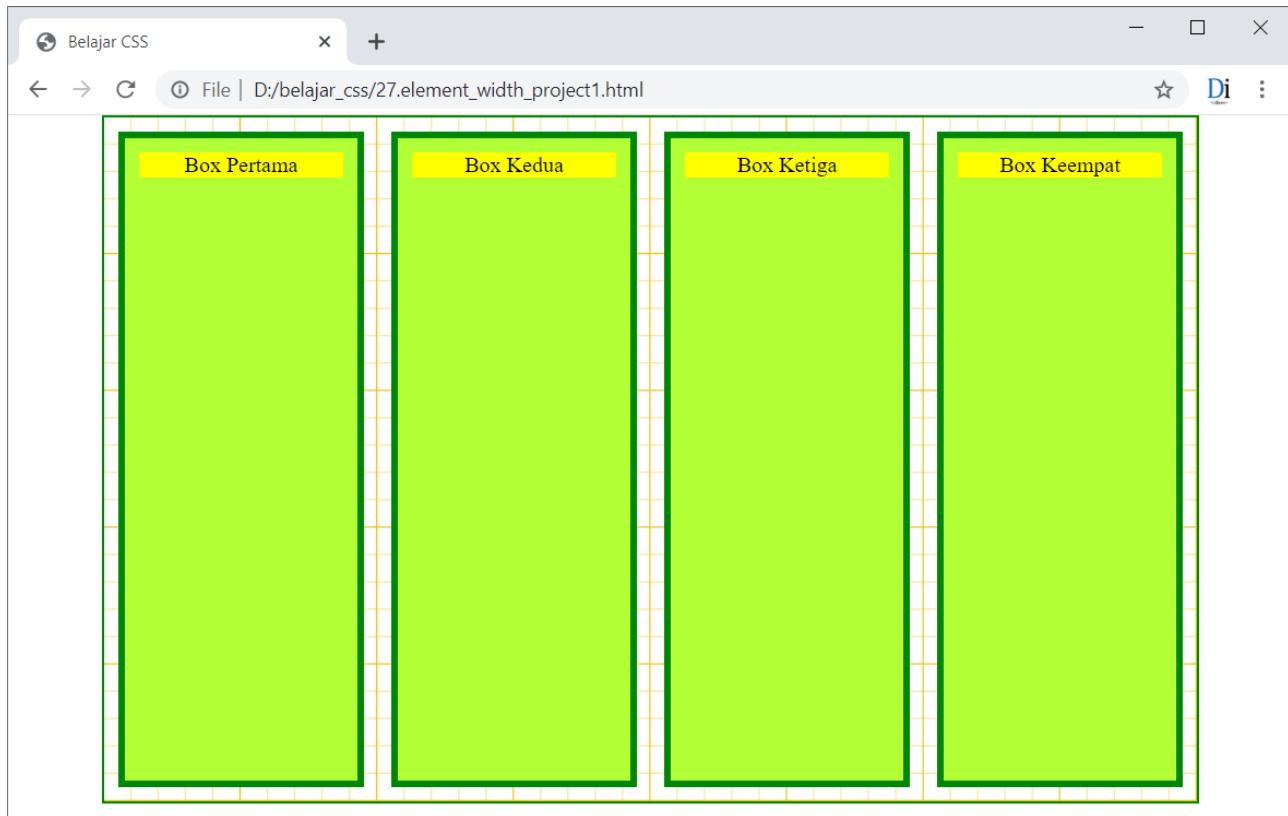


Gambar: Konten 200x60 pixel, dikelilingi dengan padding, border dan padding masing-masing 10px

Kesimpulan dari praktek ini adalah: lebar element berasal dari `width + padding + border + margin`. Begitu juga dengan tinggi element.

Agar lebih memahami konsep ini, kita akan lakukan sebuah proyek sederhana. Saya ingin

membuat 4 kotak sama besar ke dalam sebuah `<div>` dengan total lebar 800 pixel, seperti tampilan berikut:



Gambar: Tampilan akhir proyek 'element width vs konten width'

Selain tampilan box yang berderet secara horizontal, kode yang dibutuhkan masih sama, yakni kombinasi dari property `width`, `height`, `padding`, `border`, dan `margin`.

Hal pertama yang bisa ditebak: setiap box mengambil lebar 200 pixel. Ini didapat dengan membagi $800/4 = 200$ pixel, dan terbukti bahwa setiap box mengambil tempat sekitar 10 kotak grid (yang setiap kotak berukuran 20×20 pixel).

Perhatikan juga semua box terdapat padding, border dan margin. Dengan demikian, 200 pixel ini adalah lebar total keseluruhan element, bukan hanya lebar konten. Mari masuk ke hitung-hitungannya.

Dari gambar terlihat bahwa margin setiap box sekitar 10px (setengah kotak grid), border sekitar 5px (seperempat kotak grid), dan padding sekitar 10px.

Karena ketiga property ini ada di kedua sisi (kiri dan kanan), maka total lebar margin adalah $10 \times 2 = 20$ px, kemudian untuk border $5 \times 2 = 10$ px, dan untuk padding $10 \times 2 = 20$ px. Maka total butuh $20\text{px} + 10\text{px} + 20\text{px} = 50$ pixel.

Dengan mengurangi ruang element 200px dengan 50px, terdapat sisa 150px. Inilah lebar dari property `width` untuk setiap box. Untuk tinggi element, dari gambar terlihat sekitar 500 pixel (25 kotak grid). Dikurangi padding, border dan margin, terdapat sisa 450px.

Berikut property box model untuk masing-masing box :

```
.box {
    width: 150px;
    padding: 10px;
    border: 5px solid green;
    margin: 10px;
}
```

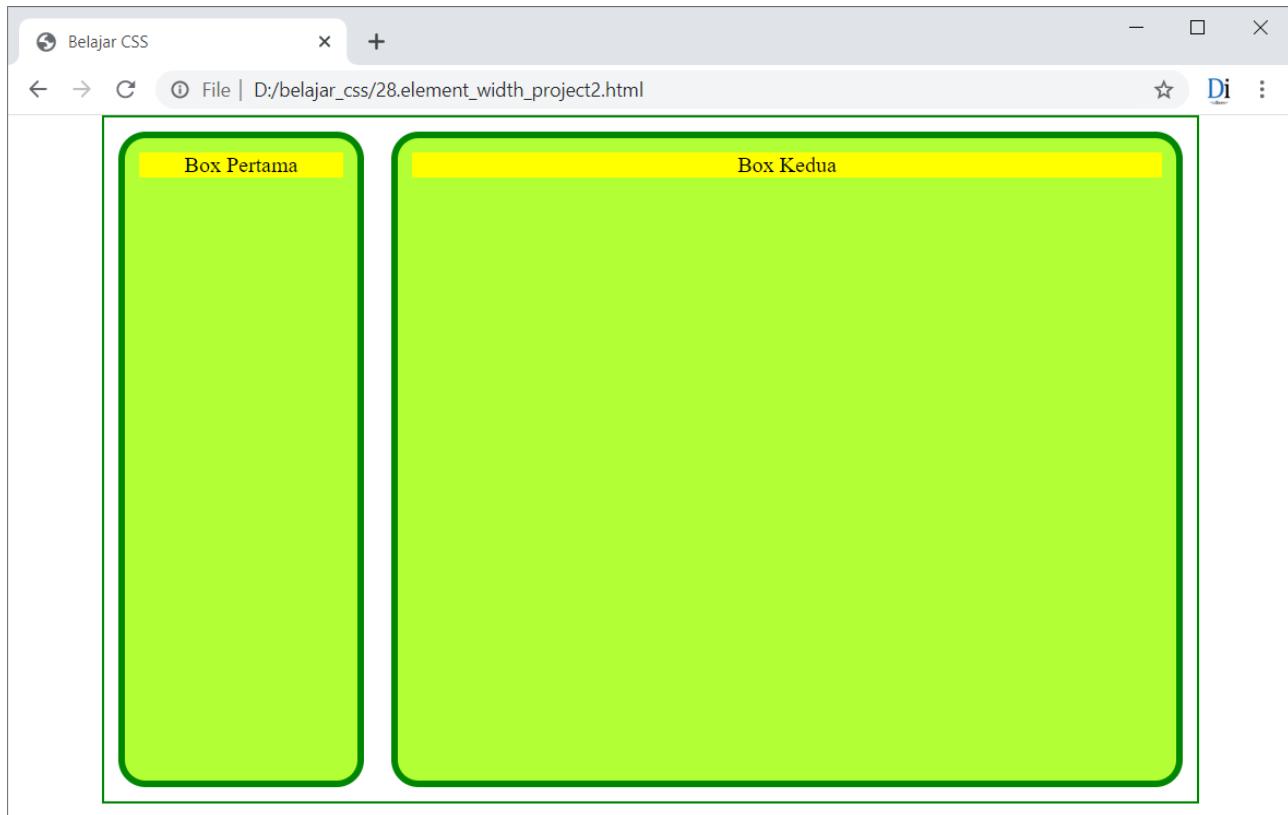
Dan, berikut adalah kode CSS + HTML lengkap untuk menghasilkan tampilan sebelumnya:

27.element_width_project1.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
8              margin: 0;
9              padding: 0;
10         }
11         div.container {
12             margin: 0 auto;
13             width: 800px;
14             height: 500px;
15             border: 2px solid green;
16             background: url('bg_grid.png');
17         }
18         p {
19             background-color: yellow;
20         }
21         .box{
22             width: 150px;
23             height:450px;
24             padding: 10px;
25             border: 5px solid green;
26             margin: 10px;
27             float: left;
28             background-color: GreenYellow;
29             text-align: center;
30         }
31     </style>
32 </head>
33 <body>
34     <div class="container">
35         <div class="box"><p>Box Pertama</p></div>
36         <div class="box"><p>Box Kedua</p></div>
37         <div class="box"><p>Box Ketiga</p></div>
38         <div class="box"><p>Box Keempat</p></div>
39     </div>
40 </body>
41 </html>
```

Silahkan pelajari sejenak kode CSS yang ada. Sebagian besar property CSS telah kita bahas (kecuali property `background` dan `float` yang akan kita pelajari pada bab selanjutnya). Saya sangat sarankan untuk mengutak-atik nilai `width`, `padding`, `border`, `margin`, dan melihat hasilnya.

Sebagai tantangan kedua, dapatkah anda mengubah tampilannya menjadi 2 kotak? Kotak pertama berukuran 200px dan kotak kedua 600px. Berikut tampilan akhir diinginkan:



Gambar: Tantangan kedua

Idenya adalah, buat class terpisah antara box pertama dan kedua. Selain itu saya juga menghapus gambar background `bg_grid.png` dan menambahkan sedikit efek rounding corner (semoga masih ingat dengan property ini).

Selamat, jika anda berhasil! Sebagai perbandingan, atau jika anda butuh bantuan berikut kode CSS yang saya pakai:

28.element_width_project2.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
8              margin: 0;

```

```

9      padding: 0;
10     }
11     div.container {
12       margin: 0 auto;
13       width: 800px;
14       height: 500px;
15       border: 2px solid green;
16     }
17     p {
18       background-color: yellow;
19     }
20     .box1{
21       width: 150px;
22       height:450px;
23       padding: 10px;
24       border: 5px solid green;
25       margin: 10px;
26       float: left;
27       background-color: GreenYellow;
28       text-align: center;
29       border-radius: 20px;
30     }
31     .box2{
32       width: 550px;
33       height:450px;
34       padding: 10px;
35       border: 5px solid green;
36       margin: 10px;
37       float: left;
38       background-color: GreenYellow;
39       text-align: center;
40       border-radius: 20px;
41     }
42   </style>
43 </head>
44 <body>
45   <div class="container">
46     <div class="box1"><p>Box Pertama</p></div>
47     <div class="box2"><p>Box Kedua</p></div>
48   </div>
49 </body>
50 </html>

```

8.9. CSS Reset

Sepanjang contoh pada bab ini, beberapa kali saya menambahkan selector berikut:

```

* {
  margin: 0;
  padding: 0;
}

```

Kode di atas berarti: cari seluruh element HTML, kemudian hapus semua margin dan padding (set menjadi 0).

Kenapa harus menggunakan ini?

Salah satu tantangan web desainer dibandingkan desainer di bidang lain adalah, kita tidak memegang kendali terhadap media desain. Media desain kita terdiri dari berbagai jenis web browser, dengan berbagai versi, yang berjalan di berbagai sistem operasi.

Seperti yang telah dijelaskan pada bab-bab awal, setiap web browser memiliki style bawaan sendiri atau *default style*. Masalahnya, default style tersebut bisa berbeda pada setiap web browser. Terkadang bedanya hanya 1 pixel di margin tertentu, atau terdapat kelebihan padding di salah satu element.

CSS Reset bertujuan men-nol-kan seluruh perbedaan ini. Dengan kata lain, kita menghapus seluruh margin, padding, border, serta beberapa property lain. Kode `* {margin: 0; padding: 0}` adalah versi paling sederhana dari CSS Reset.

Saat ini tersedia berbagai teknik terkait CSS Reset, yang paling terkenal yaitu Eric Meyer's CSS Reset. **Eric Meyer** adalah seorang web desainer Amerika Serikat dan penulis buku terkenal tentang CSS. Ia merupakan salah satu web desainer yang pertama kali menerapkan CSS reset untuk mengatasi perbedaan default style pada web browser.

Eric Meyer merilis CSS Reset-nya dengan gratis di meyerweb.com/eric/tools/css/reset/

Berikut adalah kode Eric Meyer's CSS Reset versi 2.0:

```

1  /* http://meyerweb.com/eric/tools/css/reset/
2   v2.0 | 20110126
3   License: none (public domain)
4 */
5
6  html, body, div, span, applet, object, iframe,
7  h1, h2, h3, h4, h5, h6, p, blockquote, pre,
8  a, abbr, acronym, address, big, cite, code,
9  del, dfn, em, img, ins, kbd, q, s, samp,
10 small, strike, strong, sub, sup, tt, var,
11 b, u, i, center,
12 dl, dt, dd, ol, ul, li,
13 fieldset, form, label, legend,
14 table, caption, tbody, tfoot, thead, tr, th, td,
15 article, aside, canvas, details, embed,
16 figure, figcaption, footer, header, hgroup,
17 menu, nav, output, ruby, section, summary,
18 time, mark, audio, video {
19   margin: 0;
20   padding: 0;
21   border: 0;
22   font-size: 100%;
23   font: inherit;
24   vertical-align: baseline;
25 }
```

```

26 /* HTML5 display-role reset for older browsers */
27 article, aside, details, figcaption, figure,
28 footer, header, hgroup, menu, nav, section {
29   display: block;
30 }
31 body {
32   line-height: 1;
33 }
34 ol, ul {
35   list-style: none;
36 }
37 blockquote, q {
38   quotes: none;
39 }
40 blockquote:before, blockquote:after,
41 q:before, q:after {
42   content: '';
43   content: none;
44 }
45 table {
46   border-collapse: collapse;
47   border-spacing: 0;
48 }

```

Dapat kita lihat, daripada menggunakan universal selector, Eric Meyer menulis seluruh selector yang ingin di reset. Semua selector ini kemudian diubah dengan property margin: 0, padding: 0, border: 0, font-size: 100%, font: inherit, dan vertical-align: baseline.

Pada dasarnya perintah ini akan menghapus margin, padding, dan border bawaan web browser. Dengan demikian, kita mulai dari 'lembaran kosong' yang tidak terkontaminasi oleh default style bawaan web browser.

Alternatif CSS Reset lain yang cukup terkenal adalah **normalize.css** yang bisa diakses di github.com/necolas/normalize.css.

Bagaimana Cara Menggunakan CSS Reset ini?

CSS Reset akan me-nol-kan seluruh default style bawaan web browser. Kode ini harus berada di bagian paling atas CSS, baru kemudian style yang kita rancang ditulis di bawahnya. Dengan demikian, style kita akan menimpa efek dari CSS Reset (memanfaatkan fitur cascade).

Agar lebih praktis, CSS Reset biasa ditempatkan ke dalam file external. Silahkan copy-paste kode di atas ke teks editor, kemudian save sebagai **reset.css**. Anda juga bisa mengambil file ini di blog Eric Meyer.

Tempatkan file **reset.css** ini di lokasi yang bisa diakses, misalnya di folder 'css'. Kemudian akses menggunakan external link CSS seperti biasa:

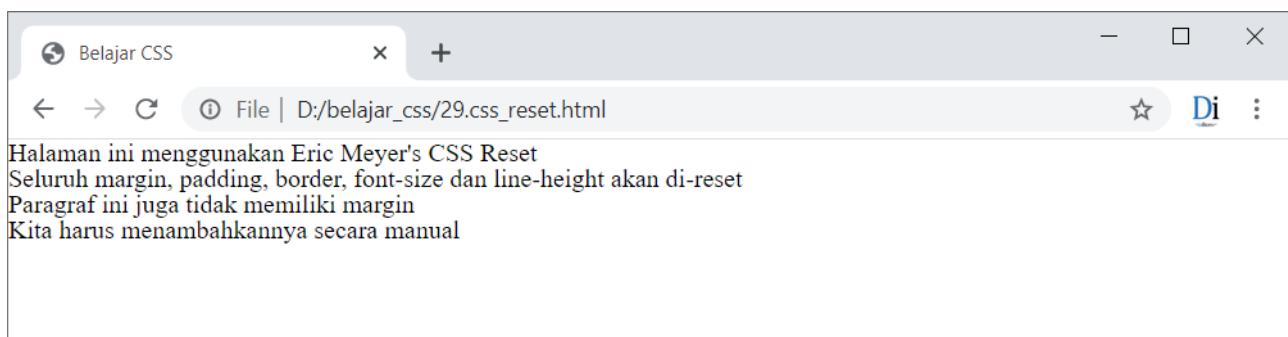
```
<link href="css/reset.css" rel="stylesheet" type="text/css" >
```

Sebagai latihan, saya telah menempatkan file `reset.css` di dalam folder `belajar_css`. Berikut kode yang dibutuhkan:

`29.css_reset.html`

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <link href="reset.css" rel="stylesheet" type="text/css" >
7      <style>
8          /* Kode CSS disini */
9      </style>
10 </head>
11 <body>
12     <h1>Halaman ini menggunakan Eric Meyer's CSS Reset </h1>
13     <h2>Seluruh margin, padding, border, font-size dan
14         line-height akan di-reset</h2>
15     <p>Paragraf ini juga tidak memiliki margin</p>
16     <p>Kita harus menambahkannya secara manual</p>
17 </body>
18 </html>
```



Gambar: Tampilan kode HTML dengan Eric Meyer's CSS Reset

Seperti yang terlihat, seluruh margin, padding, border dan font-size di-nol-kan (di-reset). Tag `<h1>` yang biasanya tampil dengan huruf besar dan tebal akan tampak sama dengan teks biasa.

Haruskah Menggunakan CSS Reset?

Selain keuntungan yang di dapat, CSS reset juga memiliki pro-kontra dari kalangan web desainer sendiri.

Kebutuhan akan CSS reset sangat berkaitan dengan proyek yang sedang kita kerjakan. Sebagai web desainer pemula, saya menyarankan untuk tidak menggunakan CSS reset terlebih dahulu. Silahkan berkreasi dan lihat apakah ada kendala dengan default style web browser. Ketika sampai ke tahap ini, barulah pertimbangkan apakah butuh CSS Reset atau tidak.

Eric Meyer sendiri mengatakan bahwa CSS Reset ini seharusnya dimodifikasi tergantung keperluan (bukan untuk dijadikan patokan mutlak). Sebagai contoh, jika kita memakai Eric

Meyer's CSS Reset, list element seperti ``, ``, dan `` tidak akan memiliki 'bullet' atau angka di depannya. Jika anda tidak menginginkan fitur ini, silahkan hapus kode tersebut dari CSS Reset.

Jika memutuskan untuk menggunakan CSS Reset, kita harus ingat untuk men-setting ulang seluruh margin, padding, font-size untuk element-element yang telah di reset. CSS Reset juga akan menambah sedikit waktu pemrosesan karena web browser melakukan 2 kali proses style CSS, yakni me-reset seluruh element, kemudian memproses kembali style untuk element tersebut.

Walaupun demikian, manfaat yang diberikan juga banyak. Dengan CSS Reset, kita bisa mulai dari "kertas kosong" sebagai dasar kode CSS. Kita juga bisa menghindari penggunaan *hack* dalam mengakali perbedaan default style dari web browser. Hampir semua CSS framework seperti Bootstrap menggunakan semacam kode yang mirip dengan CSS reset.

Jadi, wajibkah menggunakan CSS reset? Dalam banyak situasi tidak perlu, karena seperti yang dilihat dari hasil penggunaan Eric Meyer's CSS Reset, kita harus buat ulang kode CSS untuk membedakan tag `<h1>` dengan tag `<p>`, serta mengatur kembali seluruh margin dan padding yang ada. Penggunaan CSS reset lebih di lihat kasus per kasus saja.

8.10. Property box-sizing

Setelah memahami konsep box model, bisa diambil kesimpulan bahwa dimensi total sebuah element dihitung dari `width / height + padding + border + margin`. Namun ada satu property khusus yang bisa mengubah aturan ini, yakni **box-sizing**.

Property `box-sizing:content-box` merupakan nilai default web browser. Dengan nilai ini, property `width` dan `height` dipakai untuk menentukan bagian konten saja, terpisah dengan padding, border serta margin.

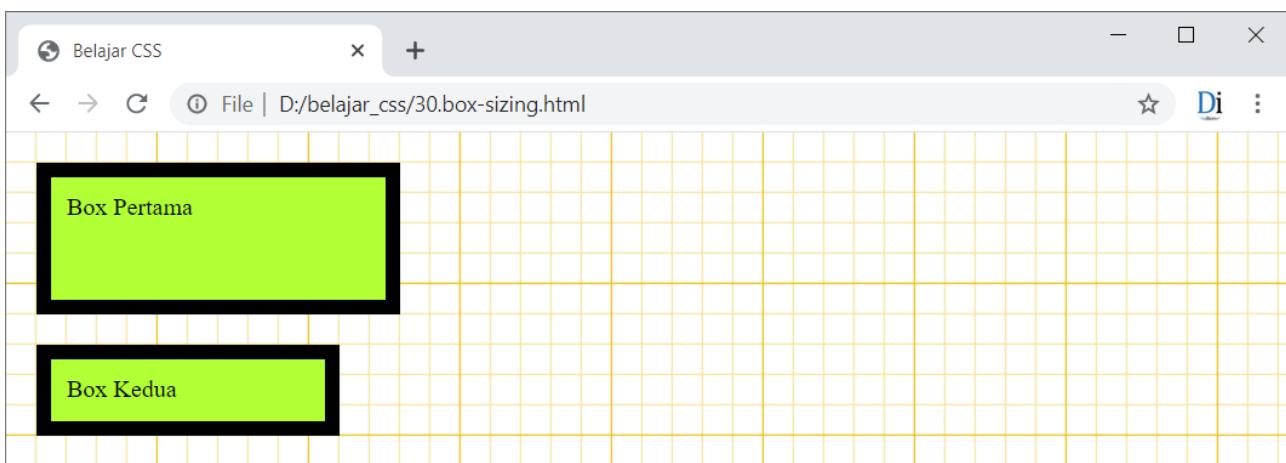
Sedangkan jika sebuah element di set dengan `box-sizing:border-box`, maka nilai `width` dan `height` sudah termasuk padding dan juga border.

Kita akan bahas dengan praktek berikut ini:

```
30.box-sizing.html
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      * {
```

CSS Box Model

```
8         margin: 0;
9         padding: 0;
10        }
11    body {
12        background: url('bg_grid.png');
13    }
14    div {
15        width: 200px;
16        height: 60px;
17        padding: 10px;
18        border: 10px solid black;
19        background-color: GreenYellow;
20        margin: 20px;
21    }
22    .satu { box-sizing: content-box; }
23    .dua { box-sizing: border-box; }
24  </style>
25 </head>
26 <body>
27   <div class="satu">Box Pertama</div>
28   <div class="dua">Box Kedua</div>
29 </body>
30 </html>
```



Gambar: Perbandingan property box-sizing

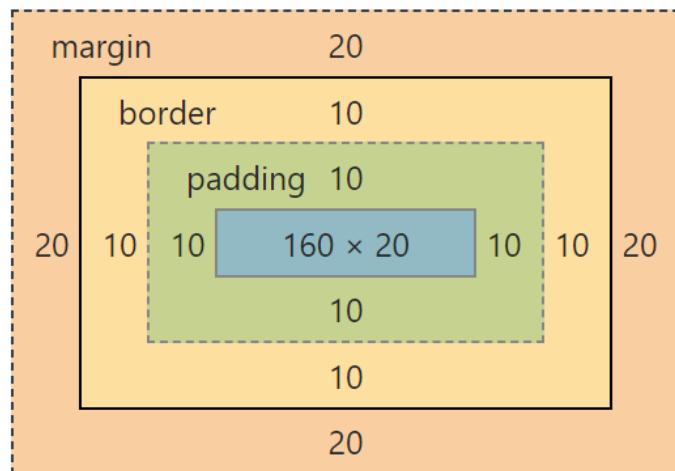
Property yang dipakai untuk kedua box sebenarnya nyaris sama, yakni `width: 200px`, `height: 60px`, `padding: 10px`, `border: 10px solid black`, serta `margin: 20px`. Yang membedakan hanya dari `box-sizing`, tapi hasil akhir terlihat berbeda jauh.

Box pertama menggunakan `box-sizing: content-box`. Maka total lebar element adalah 200px (`width`) + 20px (`padding kiri/kanan`) + 20px (`border kiri/kanan`) + 40px (`margin kiri/kanan`) = 280px . Perhitungan seperti ini sudah pernah kita bahas sebelumnya,

Box kedua menggunakan `box-sizing: border-box`, maka di sini nilai property `width: 200px` dan `height: 60px` sudah termasuk dengan padding dan border. Akibatnya, lebar ruang untuk konten hanya sisa 160px .

Hasil ini didapat dari $200\text{px} - 20\text{px} (\text{padding kiri/kanan}) - 20\text{px} (\text{border kiri/kanan})$. Dengan demikian bagian konten + padding + border tetap pas 200px sesuai dengan nilai property width. Perhitungan yang sama juga berlaku untuk height, dimana 60px itu sudah termasuk padding dan border.

Khusus untuk margin, tetap berada di luar element, sehingga lebar keseluruhan element adalah $200\text{px} + 40\text{px} (\text{margin kiri/kanan}) = 240\text{px}$.



Gambar: Perhitungan box-sizing: border-box

Secara konsep, sebenarnya `box-sizing: border-box` memudahkan kita dalam menghitung dimensi total element. Penambahan border dan padding sekarang tidak lagi berpengaruh ke perhitungan box model.

Sebagai contoh, jika saya ingin membuat 4 kotak sama besar di dalam sebuah container `<div>` dengan lebar total 800 pixel, hitungannya lebih sederhana. Kita tinggal memikirkan margin yang dipakai, dan sisanya akan menjadi nilai property `width`. Jika ingin membuat margin 10px, maka nilai untuk property `width` adalah $(800/4) - 20\text{px} = 180\text{px}$.

Berikut contoh kode yang diperlukan untuk membuat studi kasus ini:

31.box-sizing_project1.html

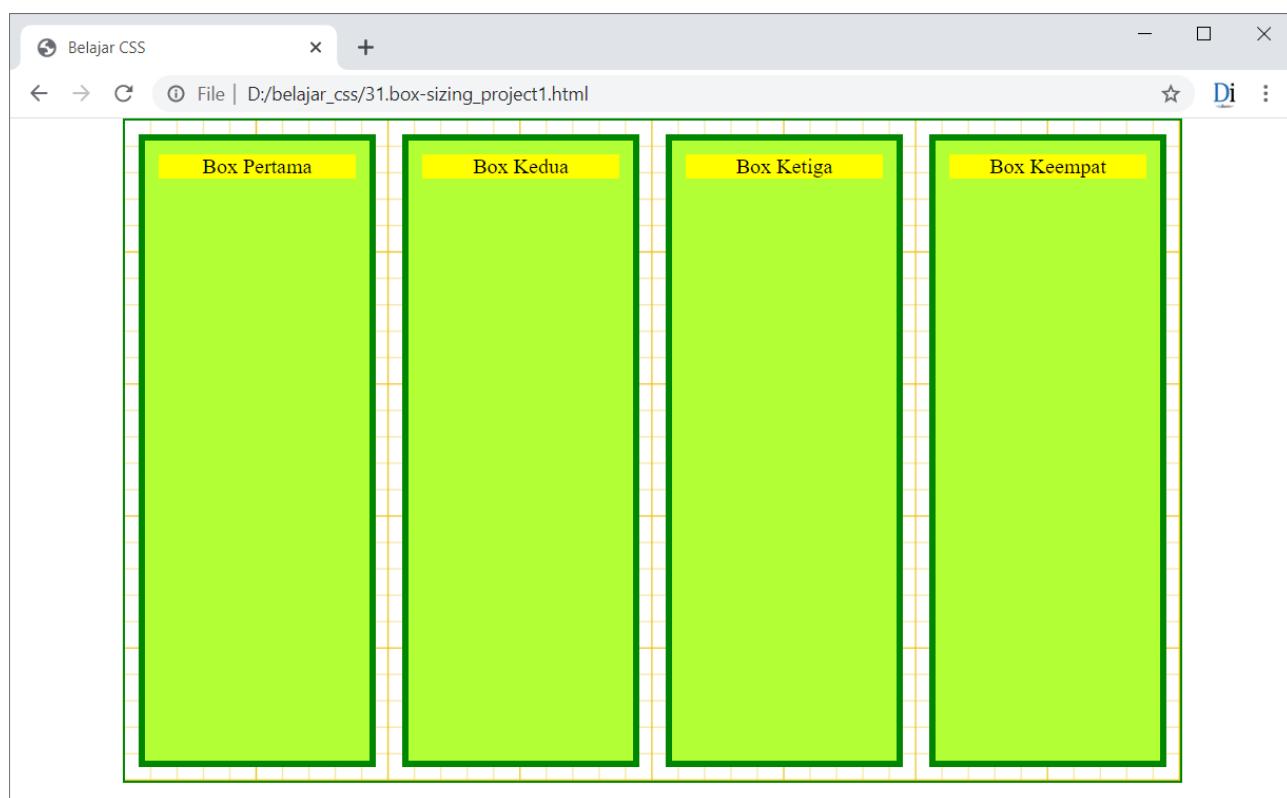
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          * {
8              margin: 0;
9              padding: 0;
10         }
11         div.container {
12             margin: 0 auto;
13             width: 800px;

```

CSS Box Model

```
14     height: 500px;
15     border: 2px solid green;
16     background: url('bg_grid.png');
17 }
18 p {
19     background-color: yellow;
20 }
21 .box{
22     width: 180px;
23     height: 480px;
24     padding: 10px;
25     border: 5px solid green;
26     margin: 10px;
27     float: left;
28     background-color: GreenYellow;
29     text-align: center;
30     box-sizing: border-box;
31 }
32 </style>
33 </head>
34 <body>
35     <div class="container">
36         <div class="box"><p>Box Pertama</p></div>
37         <div class="box"><p>Box Kedua</p></div>
38         <div class="box"><p>Box Ketiga</p></div>
39         <div class="box"><p>Box Keempat</p></div>
40     </div>
41 </body>
42 </html>
```



Gambar: Contoh pembagian width dengan box-sizing: border-box

Project ini sama seperti yang kita buat di materi Content Width vs Element Width, hanya saja sekarang menggunakan perhitungan `box-sizing: border-box`. Yang berubah ada di nilai property `width:180px` dan `height:480px` serta tambahan property `box-sizing: border-box` untuk selector `.box`.

Kemudahan lain adalah, jika kita ingin memperbesar atau mengecilkan padding dan border milik `.box`, itu tidak berpengaruh panjang element. Silahkan tes ubah border menjadi 15px, tampilan halaman tidak akan berantakan. Namun jika property `box-sizing` di hapus, tampilan layout akan berantakan karena kita harus hitung ulang nilai `width` dan `height`.

Karena keunggulan inilah property `box-sizing: border-box` sering dipakai oleh CSS reset. Framework CSS **Bootstrap** secara bawaan juga memakai `box-sizing: border-box`, sehingga perhitungan box model menjadi berbeda dari default bawaan CSS.

Meskipun perhitungan box model dengan `box-sizing: border-box` menjadi lebih mudah, namun nilai default CSS adalah `box-sizing: content-box`.

Jika anda merancang layout dengan tim atau menggunakan kode yang sudah jadi (termasuk jika menggunakan framework), pelajari terlebih dahulu apakah project tersebut memakai `box-sizing: border-box` atau tidak.

8.11. Property outline

Fungsi property **outline** sangat mirip seperti `border`, yakni membuat garis tepi pada sebuah element. Namun `outline` tidak mempengaruhi lebar elemen sehingga di anggap bukan bagian dari box model.

Property `outline` memiliki perintah longhand notation untuk 3 property berikut:

- `outline-color`, untuk menentukan warna outline (bisa diisi semua satuan warna).
- `outline-style`, untuk menentukan bentuk outline.
- `outline-width`, untuk menentukan lebar outline (bisa diisi semua satuan length).

Nilai untuk ketiga property ini sama seperti `border`, termasuk `outline-style` yang bisa diisi dengan keyword style `border` seperti `none`, `hidden`, `solid`, `dotted`, `dashed`, `double`, `groove`, `ridge`, `inset`, dan `outset`.

Selain itu kita juga bisa menggunakan shorthand notation yang juga sama seperti `border`, misalnya `outline: 5px red solid` untuk membuat outline dengan lebar 5px, berwarna merah dan style solid.

Berikut contoh praktek pembuatan outline:

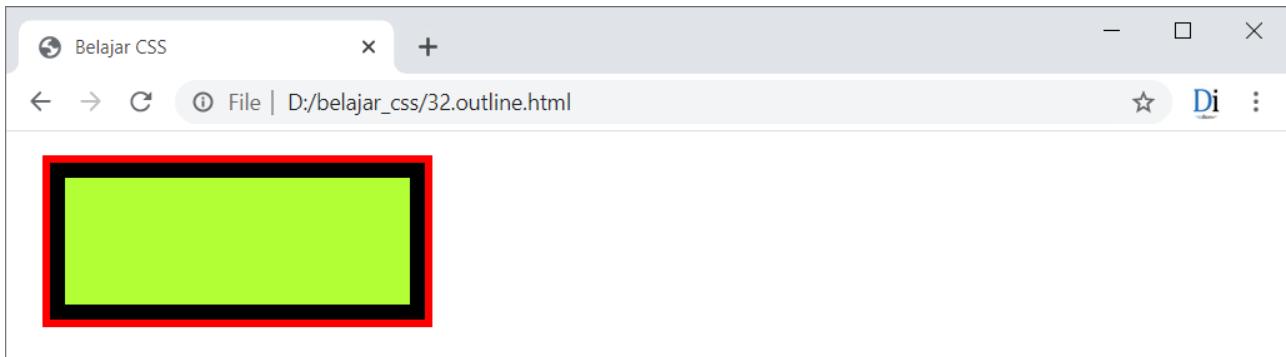
32.outline.html

```
1  <!DOCTYPE html>
```

```

2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 200px;
9       height: 60px;
10      padding: 10px;
11      border: 10px solid black;
12      background-color: GreenYellow;
13      margin: 20px;
14      outline: 5px red solid;
15    }
16  </style>
17 </head>
18 <body>
19   <div></div>
20 </body>
21 </html>

```



Gambar: Praktek dari pembuatan outline

Di sini saya membuat sebuah tag div lengkap dengan box model + outline. Garis merah setelah border itulah yang berasal dari property `outline: 5px red solid`.

Sedikit berbeda dengan border, outline tidak bisa di set secara terpisah untuk masing-masing sisi. Jadi tidak ada property `outline-left` maupun `outline-bottom`, kita hanya bisa membuat outline yang seragam untuk semua sisi.

Selain itu terdapat tambahan property `outline-offset` untuk membuat jarak antara border dengan `outline`. Nilai yang diisi berupa satuan length seperti pixel atau em. Berikut contoh penggunaannya:

33.outline_offset.html

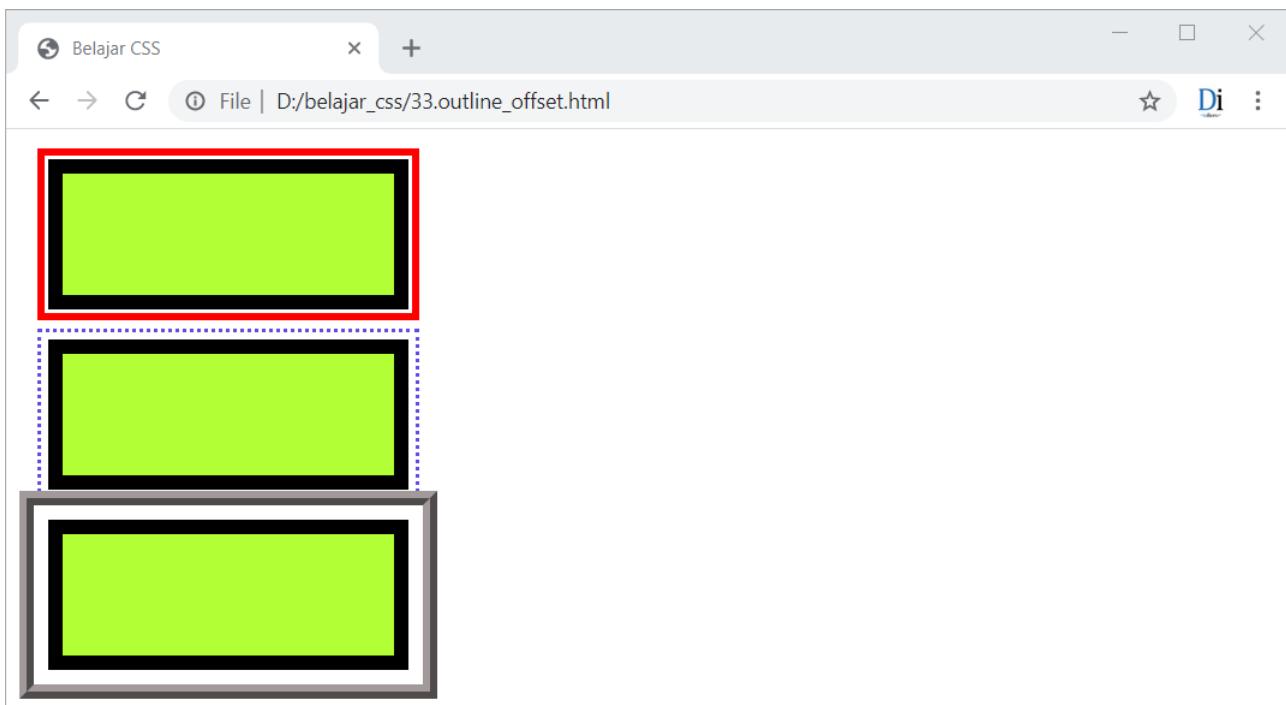
```

1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>

```

CSS Box Model

```
7   div {
8     width: 200px;
9     height: 60px;
10    padding: 10px;
11    border: 10px solid black;
12    background-color: GreenYellow;
13    margin: 20px;
14  }
15  .satu {
16    outline: 5px red solid;
17    outline-offset: 3px;
18  }
19  .dua {
20    outline: 3px #5c44e2 dotted;
21    outline-offset: 5px;
22  }
23  .tiga {
24    outline: 10px hsl(0, 4%, 60%) ridge;
25    outline-offset: 10px;
26  }
27  </style>
28 </head>
29 <body>
30  <div class="satu"></div>
31  <div class="dua"></div>
32  <div class="tiga"></div>
33 </body>
34 </html>
```



Gambar: Praktek dari penggunaan property outline-offset

Saya membuat 3 buah box dengan nilai outset yang berbeda-beda. Property `outline-offset`

akan mendorong outset menjauh dari border.

Selain itu perhatikan sisi bawah outline box kedua yang terhimpit oleh outline box ketiga. Ini bisa terjadi karena outline tidak mengambil ruang element. Jarak antar setiap box adalah 20px, yang berasal dari property `margin: 20px`. Berapa pun nilai outline yang kita tulis, tidak akan pengaruh ke dimensi total element.

Outline untuk Accessibility

Jika dilihat dari prakteknya, outline ini sangat mirip seperti border sehingga tidak jarang banyak yang bingung untuk apa W3C membuat property outline.

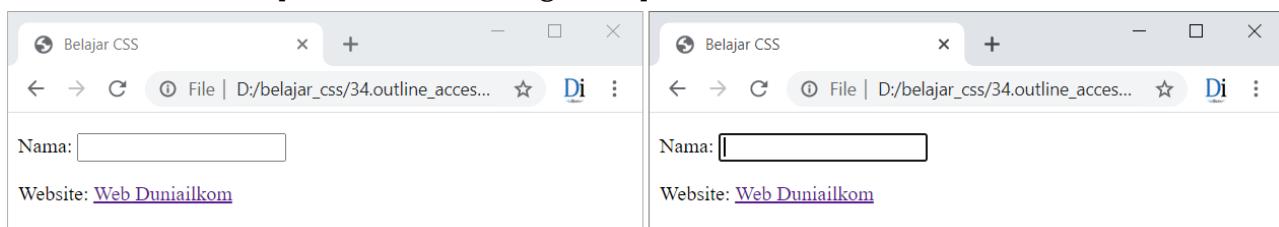
Dalam implementasi oleh web browser, outline sering dipakai untuk membantu menandai konten web (*accessibility*). Misalnya jika kita mengisi sebuah form input, akan muncul garis tepi yang berwarna tebal, atau ketika sebuah link di fokuskan (dengan cara menekan tombol tab), akan muncul garis outline di sekeliling link tersebut.

Berikut contoh prakteknya:

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <p>Nama: <input type="text"></p>
9      <p>Website: <a href="https://www.duniailkom.com">Web DuniaIlkom</a></p>
10 </body>
11 </html>
```

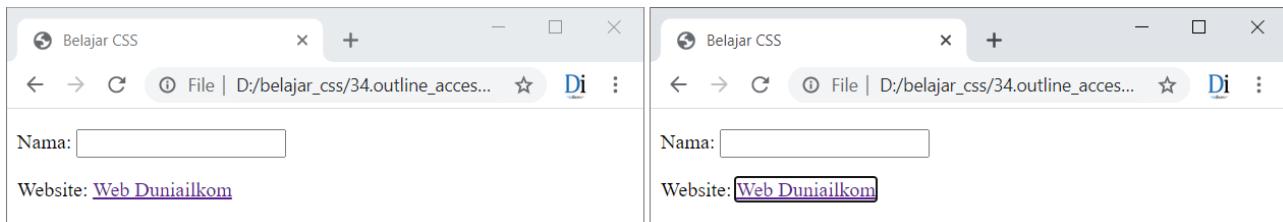
Kode ini berisi struktur HTML tag `input type="text"` serta sebuah link. Setelah halaman tampil, silahkan klik form input satu kali, maka garis tepi form akan menebal:



Gambar: Form input memiliki tambahan outline saat di fokuskan

Pada web browser Google Chrome atau Opera, garis tepi ini berasal dari outline, bukan border. Tujuannya memudahkan pengunjung menandai inputan form yang sedang aktif.

Setelah itu tekan tombol tab di keyboard untuk pindah ke link di bawahnya:



Gambar: Link memiliki tambahan outline saat di fokuskan

Sekarang giliran link "Web DuniaIlkom" yang memiliki outline hitam sebagai tanda bahwa link tersebut sedang difokuskan. Jika tombol Enter ditekan, web browser langsung membuka web duniaIlkom sesuai nilai atribut `href` dari tag `<a>`.

Inilah penggunaan umum dari outline, yakni untuk membantu *user experience* pengujung web (*accessibility*).

Implementasi outline sebagai penanda form seperti ini bisa beda-beda antar setiap web browser. Mozilla Firefox misalnya, tidak menambah garis outline praktik di atas tapi menggunakan border.

8.12. Property min-width dan max-width

Untuk mengatur lebar dan tinggi element, CSS masih menyediakan property tambahan: `min-width` dan `max-width`, serta pasangan `min-height` dan `max-height`. Keempat property ini biasa dipakai ketika membuat *responsive web design*, yakni konsep desain layout dimana setiap element web bisa membesar dan mengecil sesuai ukuran layar.

Secara bawaan, lebar suatu *block level element* sama dengan parent elementnya. Sedangkan jika property width ditulis, misal `width:600px`, maka lebar element akan selalu pas di 600 pixel. Berikut contoh yang dimaksud:

35.width-normal.html

```

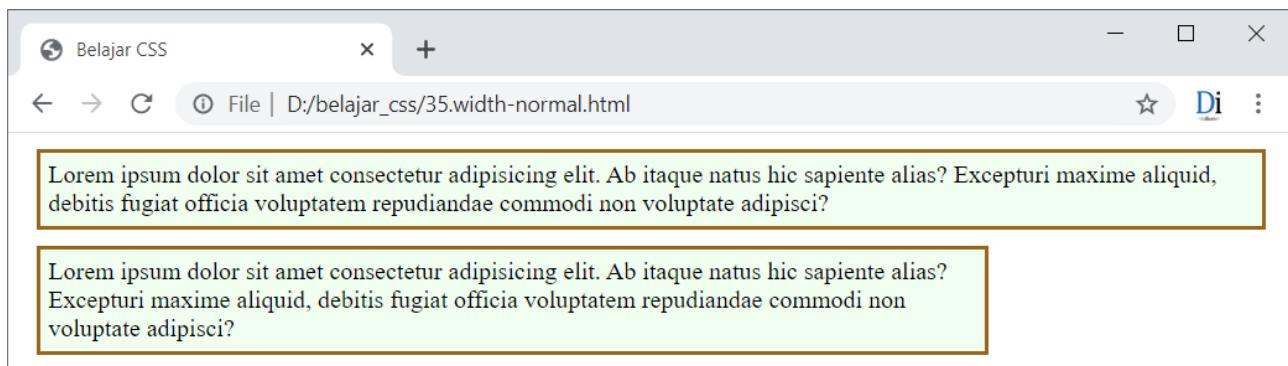
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              margin: 10px;
9              background-color: honeydew;
10             border: 3px solid #965e15;
11             padding: 0.3em;
12         }
13     </style>
14 </head>
15 <body>

```

```

16 <div>
17   Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
18   natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
19   officia voluptatem repudiandae commodi non voluptate adipisci?
20 </div>
21 <div style="width: 600px;">
22   Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
23   natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
24   officia voluptatem repudiandae commodi non voluptate adipisci?
25 </div>
26 </body>
27 </html>

```

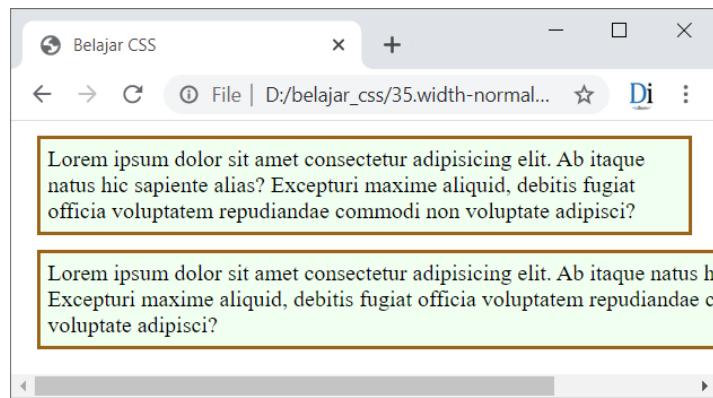


Gambar: Lebar element tanpa dan dengan property width

Box pertama tidak menggunakan property `width`. Terlihat lebar element membentang dari kiri ke kanan web browser. Ini karena tag `<div>` tersebut berada di dalam tag `<body>` yang secara default memenuhi lebar jendela web browser.

Box kedua memakai property `width:600px` yang akan membatasi lebar box sebesar 600 pixel. Konten teks akan terus membuat baris baru ke arah bawah.

Jika lebar web browser diubah, box pertama akan menyesuaikan diri karena tidak memiliki lebar yang tetap:



Gambar: Hasil tampilan saat lebar web browser diperkecil

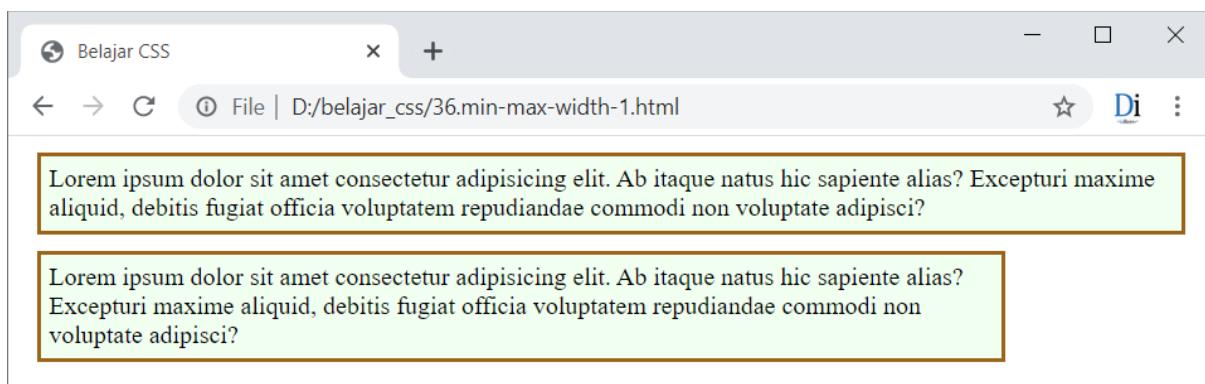
Ketika lebar web browser kurang dari 600 pixel, sebagian konten di box kedua sudah tidak terlihat (harus di geser dengan scrollbar).

Sekarang kita coba dengan `min-width` dan `max-width`:

36.min-max-width-1.html

```

1 ...
2 <body>
3   <div style="min-width: 600px;">
4     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
5     natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
6     officia voluptatem repudiandae commodi non voluptate adipisci?
7   </div>
8   <div style="max-width: 600px;">
9     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
10    natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
11    officia voluptatem repudiandae commodi non voluptate adipisci?
12  </div>
13 </body>
```

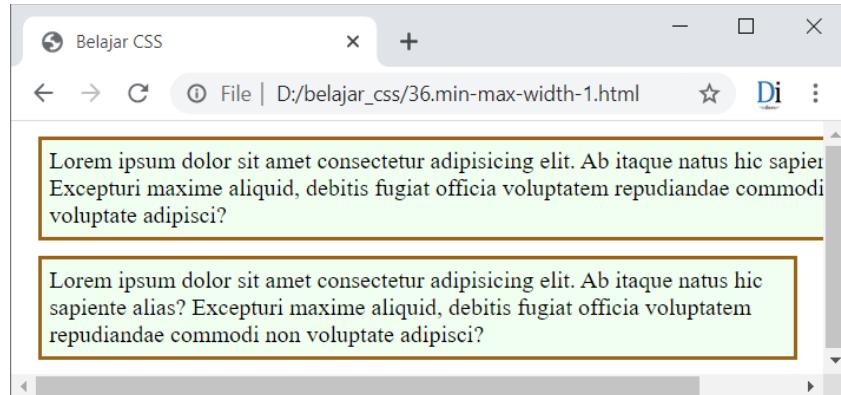


Gambar: Hasil tampilan property `min-width` dan `max-width`

Property `min-width:600px` akan membatasi **lebar minimum** element sebesar 600 pixel. Jika lebar parent element lebih dari 600px, maka element tersebut akan terus melebar seolah-olah tanpa property `width`. Inilah tampilan dari box pertama.

Box kedua menggunakan property `max-width:600px`, sehingga akan membatasi **lebar maksimum** element sebesar 600 pixel. Jika konten yang ada di dalam element butuh ruang lebih dari 600px, konten tersebut akan mengambil tempat di baris baru. Ini sama seperti penggunaan property `width:600px`.

Sekarang mari kita tes kecilkan lebar web browser:



Gambar: Hasil tampilan saat lebar web browser diperkecil

Karena box pertama terdapat property `min-width:600px`, maka element akan mempertahankan lebarnya di 600px, tidak bisa lebih kecil dari ini. Sedangkan box kedua dengan `max-width:600px` bisa terus mengecil seolah-olah tanpa batasan width.

Property `min-width` dan `max-width` juga bisa kita pakai sekaligus pada satu element:

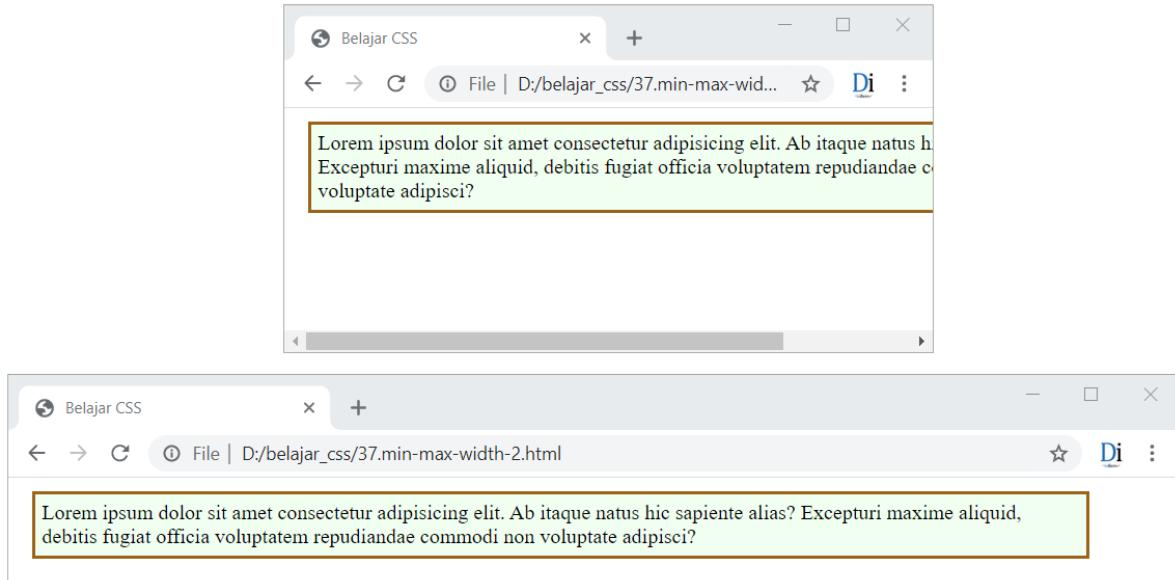
37.min-max-width-2.html

```
1 ...
2 <body>
3   <div style="min-width: 600px; max-width: 800px;">
4     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
5     natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
6     officia voluptatem repudiandae commodi non voluptate adipisci?
7   </div>
8 </body>
```

Property CSS di baris 3 artinya element ini memiliki lebar minimum 600px dan lebar maksimum 800px.

Jika jendela web browser dibesarkan lebih dari 800 pixel, lebar box akan tetap di 800px. Sedangkan jika jendela web browser dikecilkan sampai kurang dari 600 pixel, lebar box akan fix di 600px.

Penjelasan ini akan lebih mudah dipahami saat praktik langsung lalu tes ubah lebar web browser untuk melihat efek yang terjadi:



Gambar: Efek penggunaan min-width dan max-width di satu element

Bisa juga di perhatikan saat lebar web browser antara 600 hingga 800 pixel, lebar box akan sesuai dengan lebar web browser.

8.13. Property min-height dan max-height

Sudah bisa di tebak bahwa property `min-height` dan `max-height` dipakai untuk men-set tinggi minimum dan tinggi maksimum sebuah element.

Secara default, tinggi suatu *block level element* bergantung kepada **tinggi konten**. Tinggi ini otomatis akan bertambah sesuai isi konten. Sedangkan jika property `height` ditulis, misal `width:100px`, maka tinggi element akan selalu pas di 100 pixel. Jika jumlah konten melebihi tinggi element, akan terjadi *overflow*:

38.height-normal.html

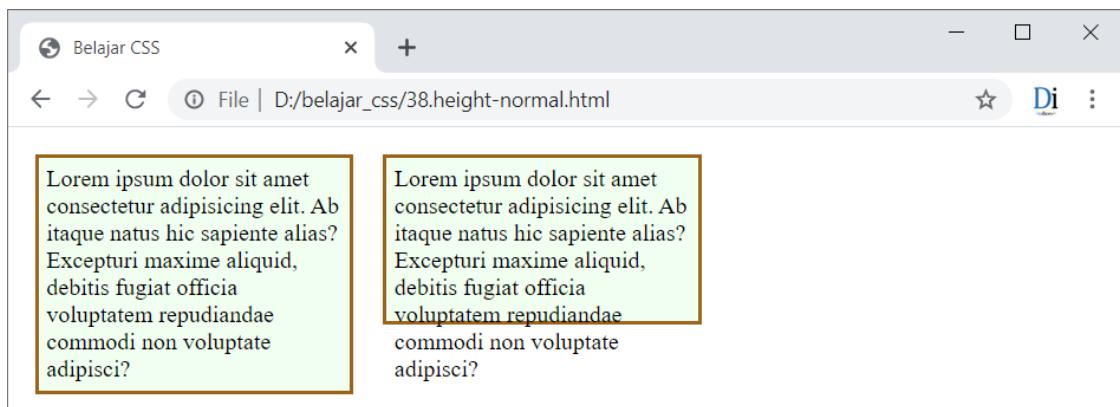
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 200px;
9              margin: 10px;
10             background-color: honeydew;
11             border: 3px solid #965e15;
12             padding: 0.3em;
13             float: left;
14         }
15     </style>
16 </head>
```

```

17 <body>
18   <div>
19     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
20     natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
21     officia voluptatem repudiandae commodi non voluptate adipisci?
22   </div>
23   <div style="height: 100px;">
24     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
25     natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
26     officia voluptatem repudiandae commodi non voluptate adipisci?
27   </div>
28 </body>
29 </html>

```



Gambar: Tinggi element tanpa dan dengan property height

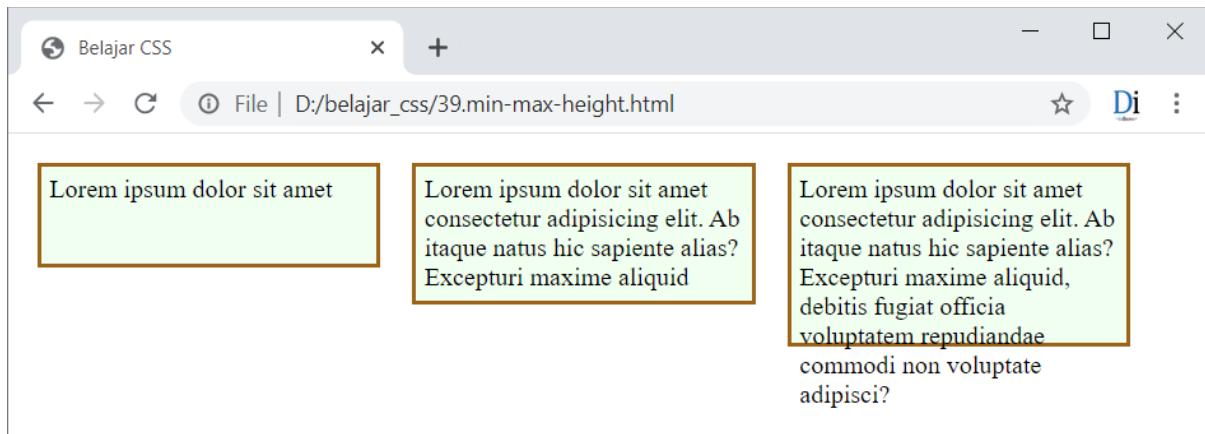
Melalui property `min-height` dan `max-height`, kita bisa batasi tinggi minimum dan maksimum yang diizinkan untuk sebuah element. Berikut contoh prakteknya:

39.min-max-height.html

```

1 ...
2 <body>
3   <div style="min-height: 50px; max-height: 100px ;">
4     Lorem ipsum dolor sit amet
5   </div>
6   <div style="min-height: 50px; max-height: 100px ;">
7     Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
8     natus hic sapiente alias? Excepturi maxime aliquid
9   </div>
10  <div style="min-height: 50px; max-height: 100px ;">
11    Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab itaque
12    natus hic sapiente alias? Excepturi maxime aliquid, debitis fugiat
13    officia voluptatem repudiandae commodi non voluptate adipisci?
14  </div>
15 </body>

```



Gambar: Hasil tampilan property min-height dan max-height

Property `min-height` dan `max-height` sebenarnya bisa dipakai satu per satu seperti praktek `min-width` dan `max-width`, tapi dalam contoh ini saya langsung menggunakan keduanya.

Ketiga box di atas memiliki property `min-height: 50px` dan `max-height: 100px`. Artinya, box memiliki tinggi minimum 50px dan tinggi maksimum 100px.

Jika tinggi konten yang ada hanya 10px (kurang dari 50px), tinggi box tetap di 50px. Sedangkan jika tinggi konten lebih dari 100px, tinggi box maksimum hanya 100px dan sisi konten akan melimpah keluar (*overflow*). Konten yang melimpah ini bisa kita set dengan property `overflow`.

Khusus jika tinggi konten antara 50 sampai 100 pixel, tinggi box akan menyesuaikan diri.

8.14. Length Value: vw dan vh

Pada bab Typography kita telah bahas sebagian besar satuan length CSS, diantaranya: `em`, `pixel`, `%`, dan `rem`. Selain itu masih ada beberapa satuan lain yang jarang dipakai untuk mengatur ukuran font, tapi cukup sering untuk men-set lebar dan tinggi element.

Satuan yang dimaksud adalah `vw` (singkatan dari `viewport width`) dan `vh` (singkatan dari `viewport height`).

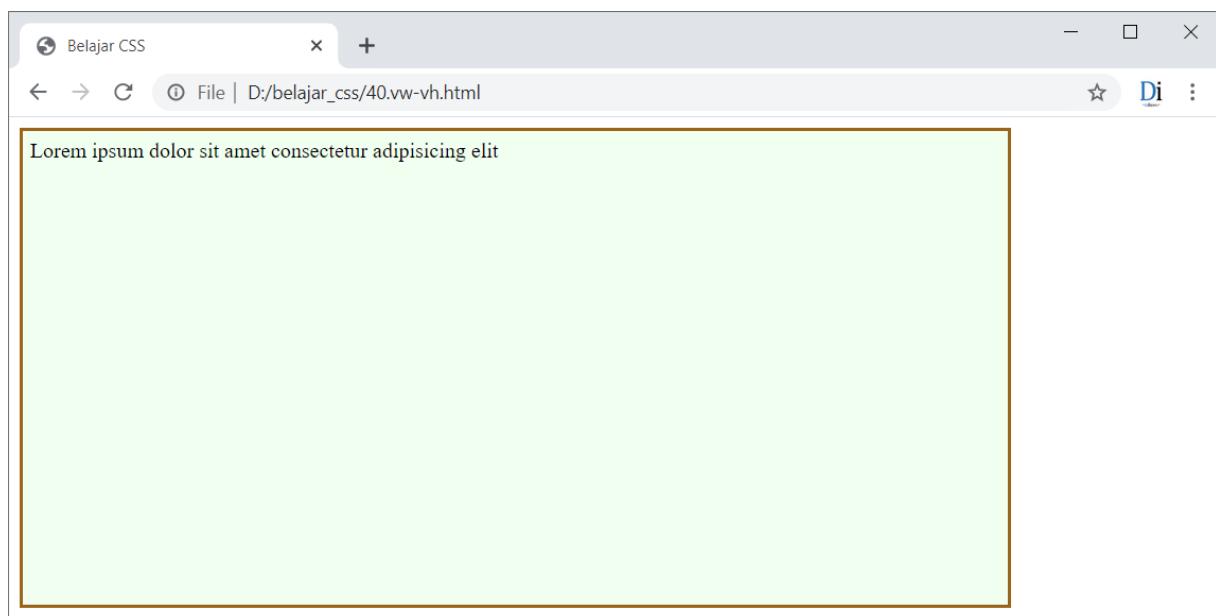
Di dalam CSS, `viewport` merujuk ke jendela tampilan web browser. Satuan `vw` dipakai untuk men-set lebar element sesuai persentase lebar jendela web browser. Jika ditulis `width: 10vw`, artinya set lebar element 10% dari lebar jendela web browser saat ini. Demikian juga jika ditulis `height: 90vh`, artinya set tinggi element sebesar 90% dari tinggi jendela web browser.

Dari pengertian ini satuan `vw` dan `vh` tampak mirip seperti `%`, akan tetapi terdapat perbedaan pada patokan pengukuran. Satuan `%` merujuk ke persentase ukuran parent element dan ini bisa berbeda-beda tergantung posisi element tersebut. Sedangkan `vw` dan `vh` secara langsung merujuk ke lebar web browser dan tidak terpengaruh parent element-nya. Ini mirip seperti beda antara satuan `em` dan `rem`.

Berikut contoh penggunaan nilai `vw` dan `vh`:

40.vw-vh.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 80vw;
9       height: 90vh;
10      background-color: honeydew;
11      border: 3px solid #965e15;
12      padding: 0.3em;
13    }
14  </style>
15 </head>
16 <body>
17   <div>Lorem ipsum dolor sit amet consectetur adipisicing elit</div>
18 </body>
19 </html>
```



9. CSS Background

Dalam bab-bab sebelumnya, beberapa kali kita telah menggunakan property `background` untuk mewarnai element HTML. Tapi tidak hanya sebatas warna saja, `background` sebuah element juga bisa di set dengan media lain seperti gambar.

Seperti namanya, property `background` digunakan untuk membuat... yah, apalagi kalau bukan `background`. Dalam bahasa indonesia, `background` bisa diartikan latar/layar belakang.

Di dalam CSS, property `background` merupakan penulisan singkat (*shorthand notation*) dari 8 property lain, yakni: `background-color`, `background-image`, `background-repeat`, `background-position`, `background-size`, `background-attachment`, `background-clip`, dan `background-origin`. Kita akan bahas setiap property ini dengan lebih detail.

9.1. Property `background-color`

Property `background-color` dipakai untuk mengubah warna latar belakang dari element HTML. Nilai default property ini adalah `transparent`, yang berarti akan melewatkannya warna parent element-nya.

Selain itu, kita bisa menggunakan berbagai nilai warna, seperti `keyword`, `#RGB`, `rgb()`, `rgba()`, `hsl()`, dan `hsla()`. Cara penggunaan warna-warna ini telah kita bahas dalam bab typography tentang property `color`.

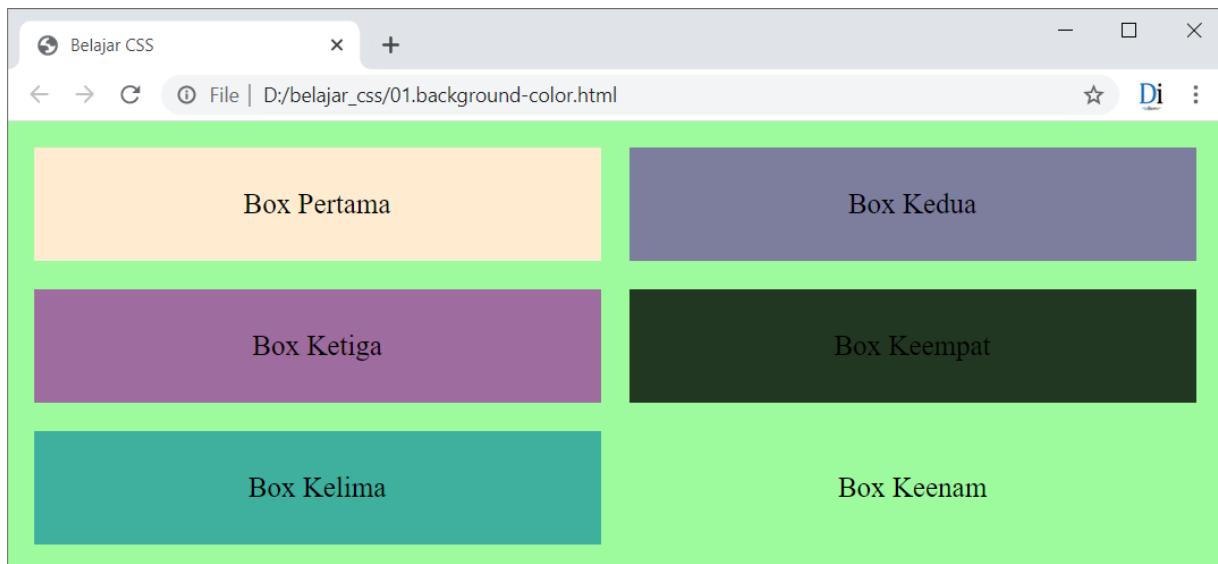
Berikut contoh penggunaan property `background-color` CSS:

01.background-color.html

```
20 <!DOCTYPE html>
21 <html lang="id">
22 <head>
23   <meta charset="UTF-8">
24   <title>Belajar CSS</title>
25   <style>
26     div {
27       width: 400px;
28       height: 80px;
29       margin: 10px;
30       text-align: center;
31       line-height: 80px;
32       font-size: 20px;
33       float: left;
```

CSS Background

```
34      }
35  body {
36      background-color: PaleGreen;
37  }
38  .satu { background-color: blanchedAlmond; }
39  .dua { background-color: #777799; }
40  .tiga { background-color: rgb(153,102,153); }
41  .empat { background-color: rgba(0, 0, 0, 0.8); }
42  .lima { background-color: hsla(170, 50%, 45%, 1); }
43  .enam { background-color: transparent; }
44 </style>
45 </head>
46 <body>
47  <div class="satu">Box Pertama</div>
48  <div class="dua">Box Kedua</div>
49  <div class="tiga">Box Ketiga</div>
50  <div class="empat">Box Keempat</div>
51  <div class="lima">Box Kelima</div>
52  <div class="enam">Box Keenam</div>
53 </body>
54 </html>
```



Gambar: Contoh penggunaan property background-color

Selector pertama: div, saya pakai untuk membuat struktur box. Box tersebut memiliki width:400px dan height:80px, yang juga dikombinasikan dengan property text-align:center dan line-height:80px untuk membuat teks tepat berada di tengah-tengah box. Semua property ini telah kita bahas dalam bab sebelumnya (kecuali property float yang akan dibahas pada bab berikutnya).

Selector kedua: body, dipakai untuk menempatkan property background-color:PaleGreen. Ini akan membuat warna background halaman menjadi hijau muda.

Pada saat property background ditulis pada sebuah element, background tersebut akan mengambil tempat seukuran lebar dan tinggi element tersebut. Dalam contoh ini, lebar dan

tinggi tag <body> adalah seukuran jendela web browser, sehingga warna PaleGreen akan menutupi seluruh jendela web browser.

Dalam 6 selector berikutnya saya membuat 6 class selector dengan nilai warna yang berbeda-beda, mulai dari class .satu hingga .enam.

Perhatikan bahwa untuk selector .enam, nilai yang dipakai adalah transparent. Ini membuat box tersebut memiliki warna transparan dan melewatkannya warna dari tag <body>.

9.2. Property background-image

Jika background dengan berbagai pilihan warna masih kurang menarik, saatnya kita gunakan gambar. Property `background-image` berfungsi untuk memasukkan gambar ke dalam background sebuah element HTML.

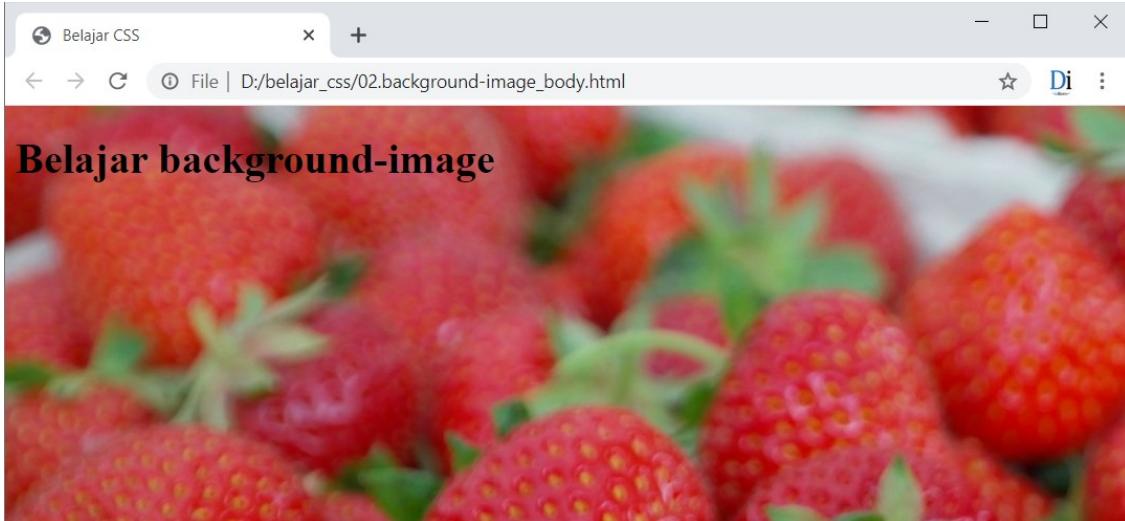
CSS mendukung berbagai format file gambar, walaupun ini sebenarnya lebih dibatasi oleh web browser. Tipe file gambar yang sering digunakan adalah JPG, PNG, GIF, dan SVG.

Nilai dari `background-image` adalah *path* gambar, yakni alamat file gambar yang akan di input. Alamat ini bisa berupa alamat relatif seperti 'image/background/nama_gambar.jpg' maupun alamat absolut seperti 'https://www.duniaIlkom.com/image/nama_gambar.jpg'. Alamat ini ditempatkan ke dalam keyword `url()` seperti contoh berikut:

```
body {
    background-image: url('image/background/gambar.jpg');
}
div {
    background-image: url('https://www.duniaIlkom.com/image/gambar.jpg');
```

Langsung saja kita lihat contohnya:

```
02.background-image_body.html
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              background-image: url('gambar/strawberries.jpg');
9          }
10     </style>
11 </head>
12 <body>
13     <h1>Belajar background-image</h1>
14 </body>
15 </html>
```



Gambar: Contoh penggunaan property background-image pada tag <body>

Gambar `strawberries.jpg` yang saya pakai di atas juga tersedia dalam file `belajar_css.zip` di Google Drive.

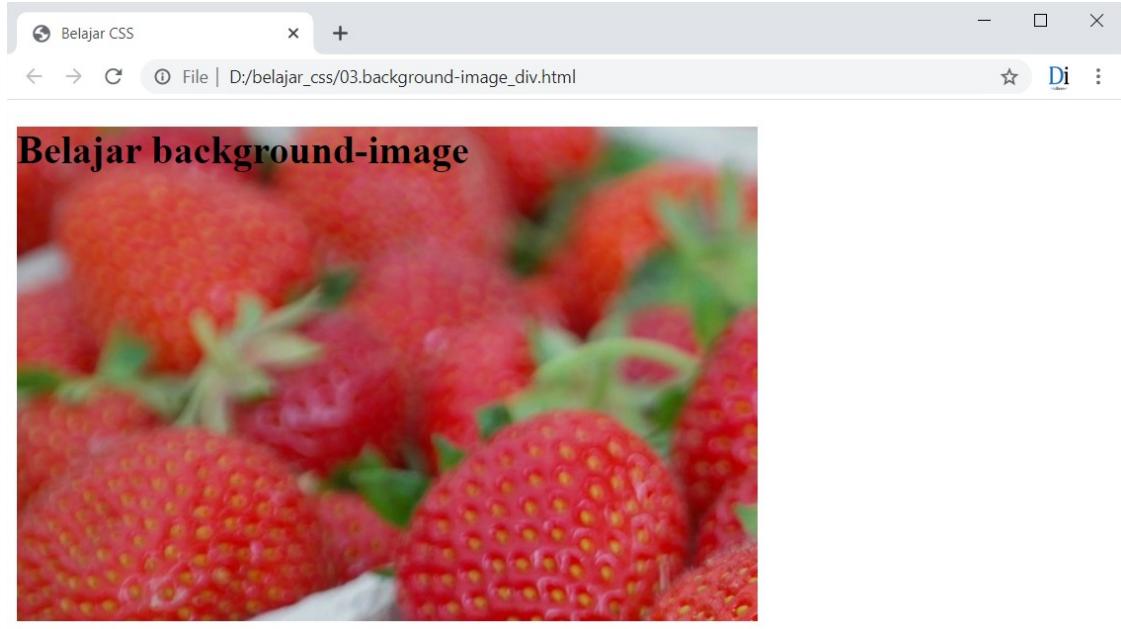
Secara default, gambar background akan dibentangkan mulai dari kiri atas element hingga kanan bawah. Karena property `background-image` ini berada di dalam tag `<body>`, maka lebar element adalah seukuran jendela web browser.

Jika ukuran gambar melebihi dimensi elemen, sisa gambar akan tersembunyi ke bagian kanan dan bagian bawah. Dalam contoh di atas, gambar `strawberries.jpg` memiliki ukuran yang lebih besar jendela web browser, sehingga bagian bawah gambar tidak terlihat.

Agar lebih memahami konsep ini, saya akan tempatkan gambar `strawberries.jpg` ke dalam element yang lebih kecil:

03.background-image_div.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 600px;
9              height: 400px;
10             background-image: url('gambar/strawberries.jpg');
11         }
12     </style>
13 </head>
14 <body>
15     <div><h1>Belajar background-image</h1></div>
16 </body>
17 </html>
```



Gambar: Tampilan background-image pada element yang lebih kecil

Kali ini gambar stroberi hanya akan tampil di dalam tag `<div>`. Tag `<div>` saya set dengan ukuran 600×400 pixel yang jauh lebih kecil daripada ukuran gambar `strawberries.jpg` sebesar 1366 x 768 pixel. Hasilnya, hanya sedikit ujung kiri atas gambar yang terlihat.

Untuk mengatur penempatan gambar ini, kita bisa menggunakan property `background-size` dan `background-position` (akan dibahas setelah ini).

Salah satu teknik yang sering dipakai ketika menggunakan `background-image` adalah, menambah property `background-color` pada selector yang sama. Ini berfungsi sebagai 'fallback' untuk jaga-jaga seandainya karena suatu hal gambar gagal tampil atau butuh waktu lama untuk tampil (misalnya karena koneksi internet yang lambat).

Berikut contoh penggunaan trik ini:

```
1 div {  
2   width:600px;  
3   height: 400px;  
4   background-color: red;  
5   background-image: url('strawberries.jpg');  
6 }
```

Kode di atas akan mewarnai tag `<div>` dengan warna merah, kemudian di atas warna inilah diletakkan gambar `strawberries.jpg`. Dengan demikian warna background akan tertutupi oleh gambar, namun jika gambar gagal tampil, warna `red` akan berpesan sebagai pengganti gambar `strawberries.jpg`.

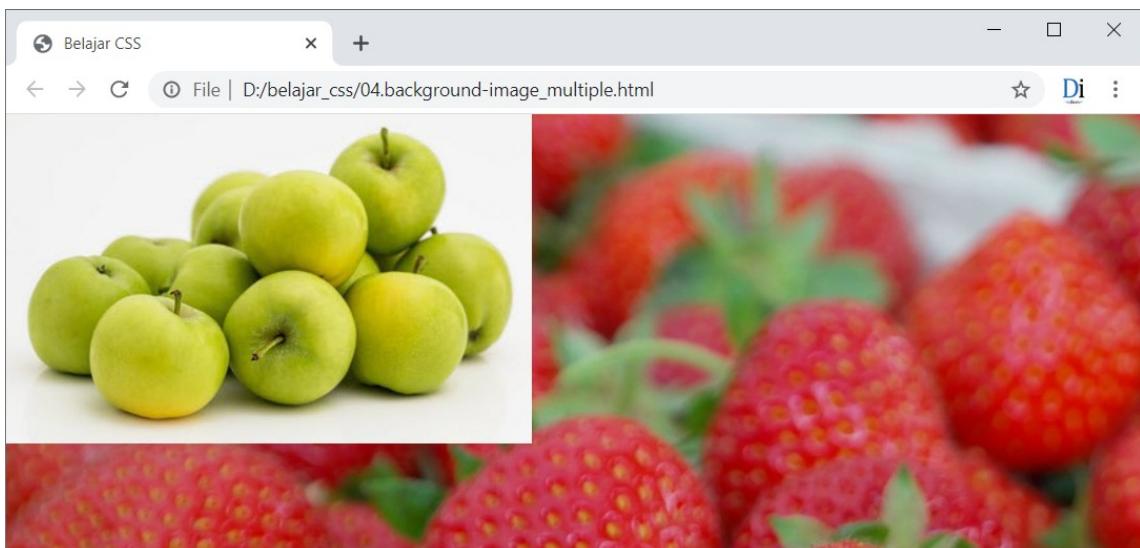
9.3. Property background-image CSS3

Di dalam modul **CSS3 background** (tepatnya: [CSS Backgrounds and Borders Module Level 3¹⁸](https://www.w3.org/TR/css-backgrounds-3/), kita bisa menginput banyak gambar background pada 1 kali penulisan property `background-image`. Teknik ini dikenal juga sebagai **CSS3 multiple backgrounds**. Setiap gambar yang ingin diinput dipisah dengan tanda koma seperti contoh berikut:

04.background-image_multiple.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              background-image: url('gambar/green_apple.jpg'),
9                  url('gambar/strawberries.jpg');
10         background-repeat: no-repeat;
11     }
12     </style>
13 </head>
14 <body>
15 </body>
16 </html>
```



Gambar: Contoh penggunaan CSS3 multiple backgrounds

Kali ini saya memakai 2 gambar sekaligus untuk `background-image`. Terlihat gambar `green_apple.jpg` tampil di sudut kiri atas gambar `strawberries.jpg`.

Urutan penulisan gambar ikut menentukan posisi layer/tumpukan gambar. Gambar yang ditulis pertama kali akan menempati posisi paling atas, gambar kedua di bawahnya, gambar ketiga di bawah gambar kedua, dst.

¹⁸ <https://www.w3.org/TR/css-backgrounds-3/>

Jika penulisan sebelumnya adalah `background-image:url('strawberries.jpg'), url('green_apple.jpg')`, maka gambar apel tidak akan terlihat karena tertutup gambar stroberi yang lebih besar.

Selain itu dalam kode sebelumnya juga terdapat tambahan property `background-repeat: no-repeat` yang berfungsi agar gambar apel tidak ditampilkan secara berulang. Kita akan bahas tentang property ini sesaat lagi.

Fitur **CSS3 multiple backgrounds** sangat praktis untuk membuat efek visual lain. Tanpa fitur ini, kita harus menumpuk 2 buah element atau lebih untuk mendapatkan hasil yang sama.

9.4. Property background-repeat

Property `background-repeat` bisa dipakai untuk mengatur apakah sebuah gambar background akan terus diulang (`repeat`) atau tidak. Nilai yang bisa diisi berupa keyword: `repeat`, `repeat-x`, `repeat-y` dan `no-repeat`.

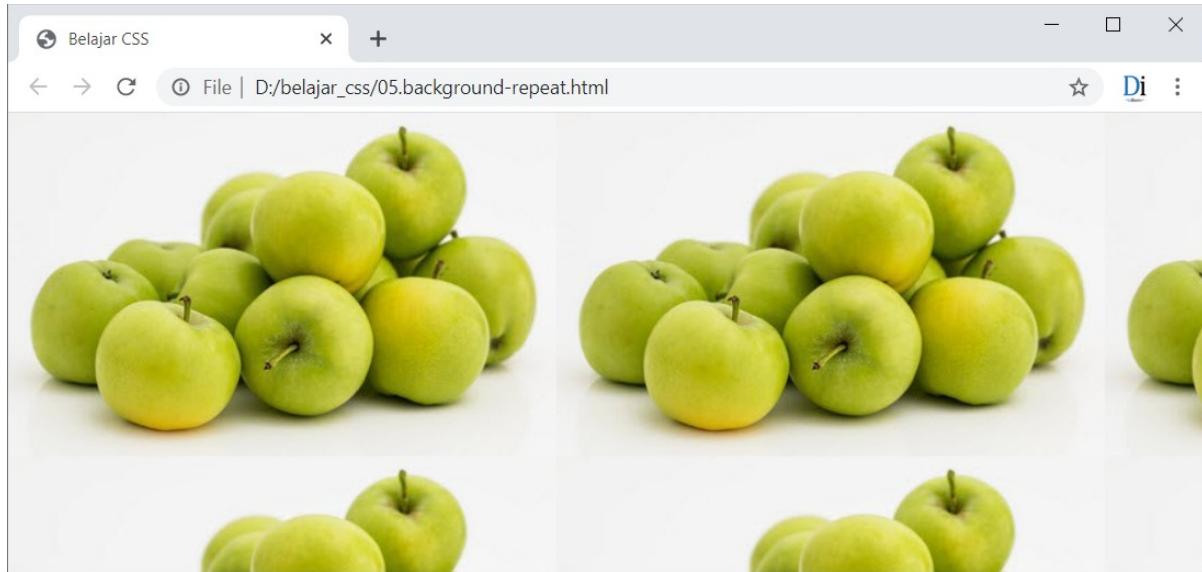
Property `background-repeat` baru terlihat efeknya ketika menggunakan gambar background yang berukuran lebih kecil daripada ukuran element saat ini.

Sebagai contoh, gambar `green_apple.jpg` dalam contoh sebelum ini hanya berukuran 400 x 250 pixel, sehingga jika dijadikan background image untuk tag `<body>` akan tampil sebagai berikut:

05.background-repeat.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     body {
8       background-image: url('gambar/green_apple.jpg');
9     }
10  </style>
11 </head>
12 <body>
13 </body>
14 </html>
```

CSS Background



Gambar: Background image akan terus diulang (repeat)

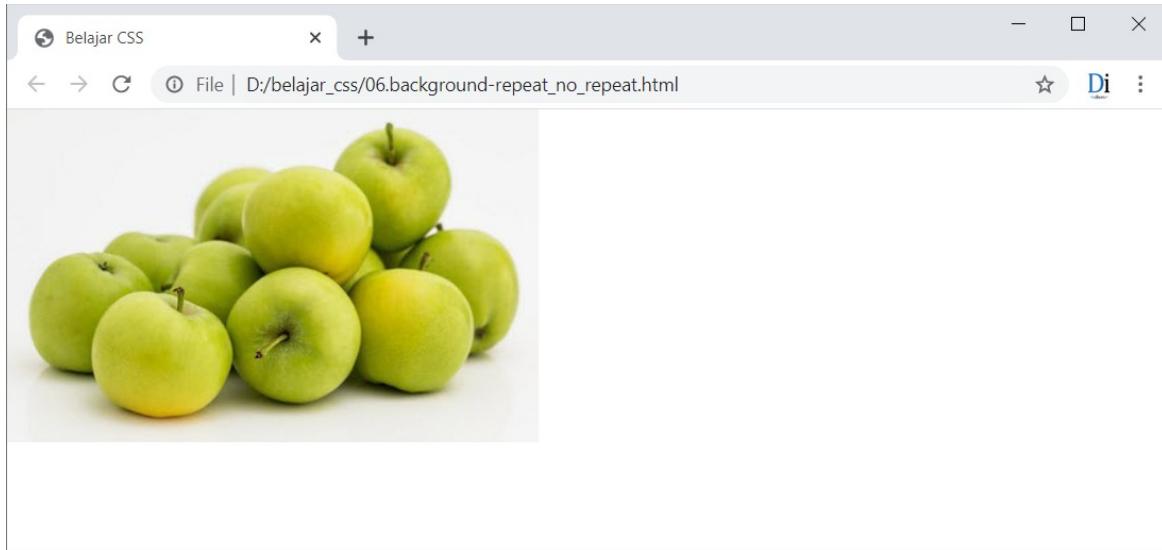
Efek ini merupakan default style bawaan web browser, dimana gambar background akan diulang sampai menutupi seluruh element. Nilai default ini sama artinya dengan menulis property `background-repeat: repeat`.

Jika kita hanya ingin gambar tersebut ditampilkan 1 kali saja, bisa gunakan nilai `background-repeat: no-repeat` seperti contoh berikut:

06.background-repeat_no_repeat.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     body {
8       background-image: url('gambar/green_apple.jpg');
9       background-repeat: no-repeat;
10    }
11  </style>
12 </head>
13 <body>
14 </body>
15 </html>
```

CSS Background



Gambar: Background image hanya akan di ulang 1 kali

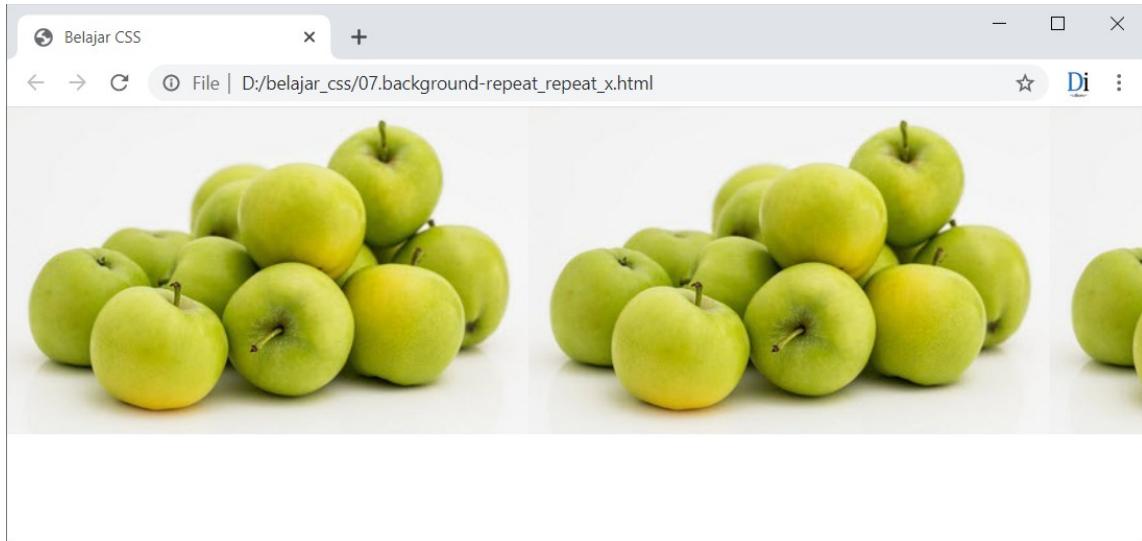
Sekarang gambar apel hanya muncul satu kali di sudut kiri atas.

Selain repeat dan no-repeat, juga terdapat nilai repeat-x jika kita ingin mengulang gambar untuk sumbu x saja (di ulang secara horizontal), sedangkan nilai repeat-y bisa dipakai jika ingin mengulang gambar pada sumbu y saja (di ulang secara vertikal).

Berikut contoh penggunaan property `background-repeat: repeat-x;`:

07.background-repeat_repeat_x.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              background-image: url('gambar/green_apple.jpg');
9              background-repeat: repeat-x;
10         }
11     </style>
12 </head>
13 <body>
14 </body>
15 </html>
```



Gambar: Background image diulang secara horizontal saja (sumbu x)

9.5. Property background-repeat CSS3

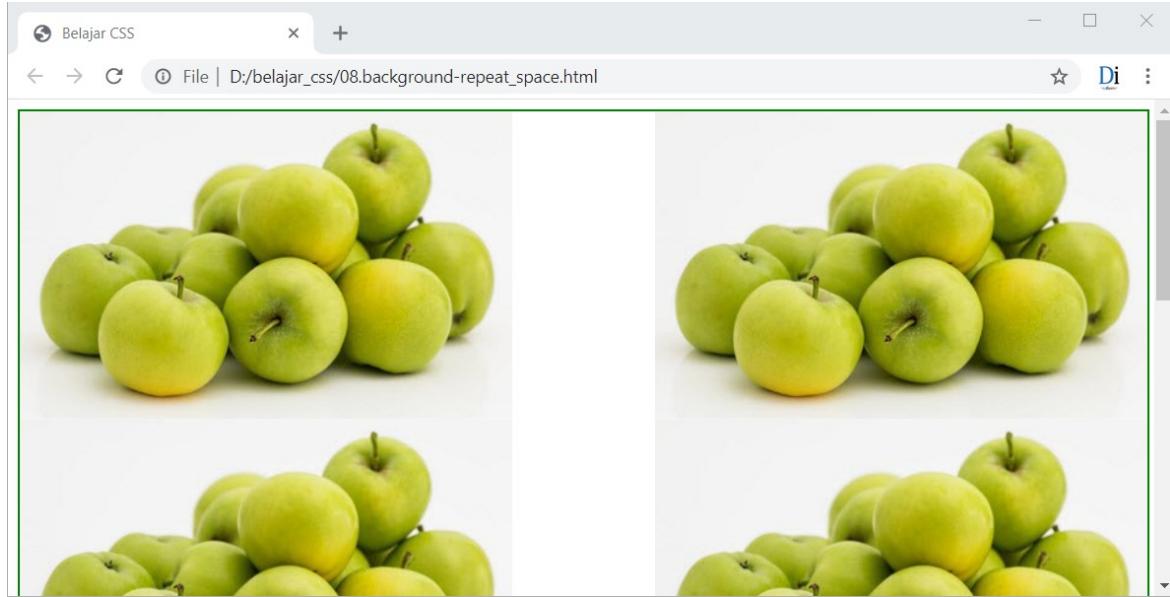
Dalam CSS3, property `background-repeat` mendapat 2 nilai tambahan yakni: `space` dan `round`.

Jika ditulis `background-repeat:space`, web browser akan berusaha mengulang gambar background tanpa memotongnya. Gambar ditempatkan di sisi kanan dan kiri terlebih dahulu, baru gambar lain akan ditambah pada bagian tengah selama masih muat. Jika tidak, ruang tengah ini akan dikosongkan.

Berikut contoh penggunaan nilai `space` untuk `background-repeat`:

08.background-repeat_space.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 100%;
9              height: 1000px;
10             border: 2px solid green;
11             margin: 0 auto;
12             background-image: url('gambar/green_apple.jpg');
13             background-repeat: space;
14         }
15     </style>
16 </head>
17 <body>
18     <div> </div>
19 </body>
20 </html>
```



Gambar: Tampilan dari nilai background-repeat: space

Kali ini saya membuat sebuah tag <div> dengan lebar 100% dan tinggi 1000 px. Dengan ukuran seperti ini, lebar tag <div> akan menyesuaikan diri dengan lebar web browser.

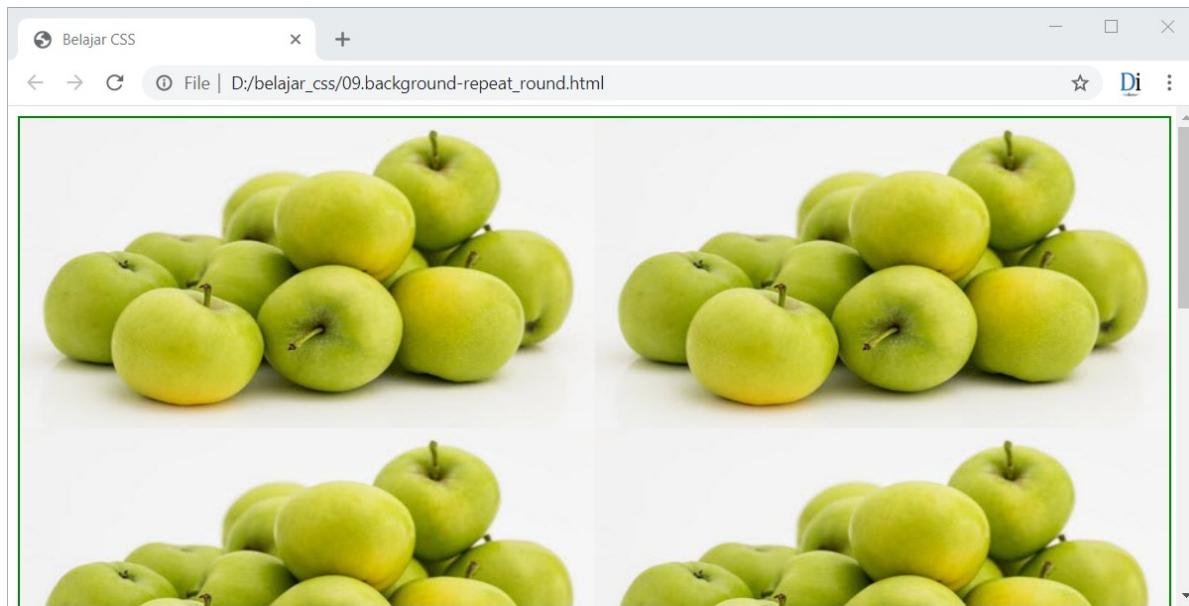
Perhatikan gambar apel tampil berulang tapi posisi awal ada di sisi kiri dan kanan. Untuk melihat efek yang lebih jelas, silahkan jalankan kode program di atas lalu ubah lebar jendela web browser, maka gambar apel ketiga akan muncul di bagian tengah jika sudah cukup ruang.

Nilai CSS3 kedua untuk property `background-repeat` adalah `round`, yang efeknya hampir sama seperti `space` namun kali ini gambar background akan dilebarkan (*expand*) untuk menutupi ruang kosong yang ada. Memakai contoh kode program sebelumnya, hasil dari `background-repeat:round` adalah sebagai berikut:

09.background-repeat_round.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 100%;
9              height: 1000px;
10             border: 2px solid green;
11             margin: 0 auto;
12             background-image: url('gambar/green_apple.jpg');
13             background-repeat:round;
14         }
15     </style>
16 </head>
17 <body>
18     <div> </div>
```

```
19 </body>  
20 </html>
```



Gambar: Tampilan dari nilai background-repeat: round

Kali ini gambar apel akan melebar karena masih belum cukup untuk gambar apel ketiga. Tapi begitu lebar element sudah mencukupi, gambar apel ketiga akan muncul di bagian tengah halaman.

Selanjutnya bagaimana jika kita ingin hanya mengulang satu sisi saja? Misalnya round hanya pada sumbu x?

Secara internal, ketika kita menulis `background-repeat: round`, yang diproses oleh web browser sebenarnya `background-repeat: round round`. Perhatikan ada 2 kali penulisan nilai `round`. Nilai pertama adalah untuk sumbu x (horizontal), sedangkan nilai kedua adalah untuk sumbu y (vertikal).

Dengan demikian, jika ingin agar gambar tampil secara `round` hanya pada satu sisi saja, bisa ditulis sebagai berikut:

```
background-repeat: round no-repeat;
```

Hasilnya, gambar akan diulang secara `round` hanya pada sumbu x (satu baris pertama saja).

9.6. Property background-position

Untuk mengatur posisi background, tersedia property `background-position`. Property ini butuh nilai dalam satuan length seperti `px`, `em`, `%` maupun keyword: `top`, `bottom`, `right`, `left`, dan `center`.

Perhitungan posisi background dimulai dari titik koordinat `0,0` pada sudut kiri atas element

hingga ke sudut kanan bawah. Nilai default untuk `background-position` adalah `0%, 0%`. Sehingga apabila property ini tidak ditulis, gambar akan dibentangkan mulai dari sudut kiri atas.

Property `background-position` bisa diisi dengan 1, 2, 3 atau 4 nilai. Yang paling mudah dipahami adalah penulisan dengan 2 nilai, dimana nilai pertama dipakai untuk posisi sumbu x (horizontal), dan nilai kedua untuk posisi sumbu y (vertikal). Format penulisannya adalah sebagai berikut:

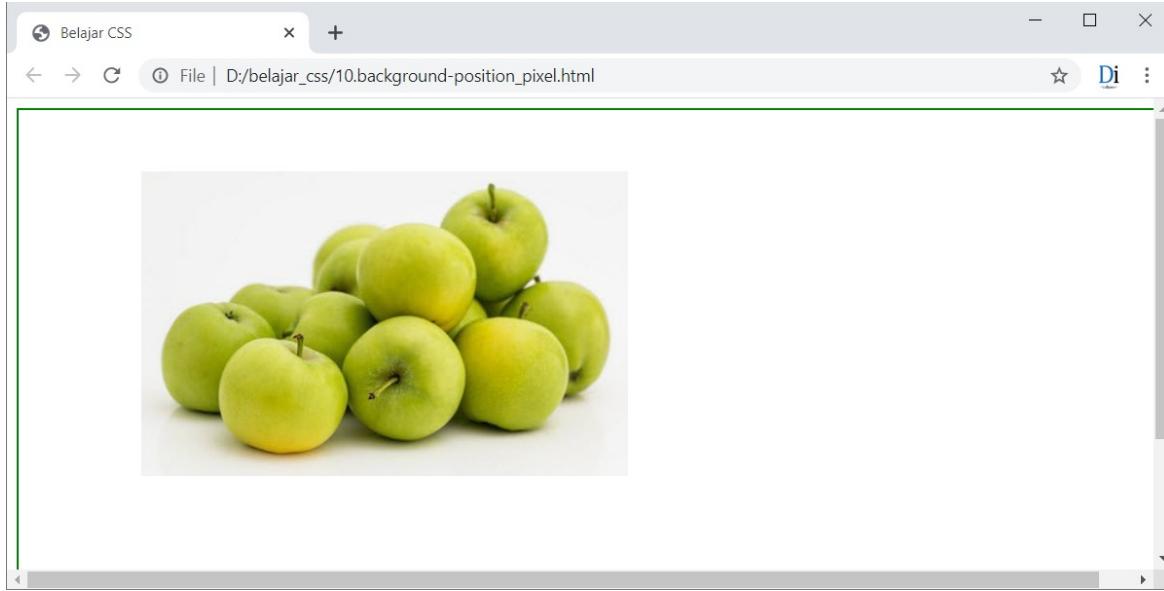
```
background-position: posisi_sumbu_x posisi_sumbu_y
```

Perhatikan bahwa di antara kedua nilai dipisah dengan karakter spasi.

Sebagai contoh, apabila saya ingin menempatkan gambar background dengan jarak `100px` dari sisi kiri element dan `50px` dari atas, bisa menggunakan kode `background-position: 100px 50px`. Berikut contoh prakteknya:

10.background-position_pixel.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 960px;
9              height: 500px;
10             border: 2px solid green;
11             margin: 0 auto;
12             background-image: url('gambar/green_apple.jpg');
13             background-repeat: no-repeat;
14             background-position: 100px 50px;
15         }
16     </style>
17 </head>
18 <body>
19     <div> </div>
20 </body>
21 </html>
```



Gambar: Gambar background berada di posisi 100px dari kiri dan 50px dari atas

Pada contoh di atas saya menyiapkan sebuah tag `<div>` berukuran 960 x 500 pixel. Tambahan property `margin: 0 auto` akan memposisikan element ini di tengah-tengah web browser. Serta property `border: 2px solid green` dipakai untuk membuat bingkai berwarna hijau.

Dengan memakai `background-repeat: no-repeat`, gambar `green_apple.jpg` hanya akan tampil 1 kali saja.

Perhatikan bahwa sudut kiri atas gambar background berada 100 pixel dari border kiri element, dan 50 pixel dari border atas element.

Bagaimana jika menggunakan nilai keyword?

Nilai keyword untuk `background-position` berfungsi menempatkan gambar di posisi tertentu, yakni `top` (atas), `bottom` (bawah), `right` (kanan), `left` (kiri) dan `center` (tengah).

Sebagai contoh, untuk membuat gambar background berada di sudut kanan bawah, bisa menggunakan property `background-position: right bottom`.

Sedangkan untuk membuat background tepat berada di tengah-tengah element, bisa menggunakan `background-position: center center`.

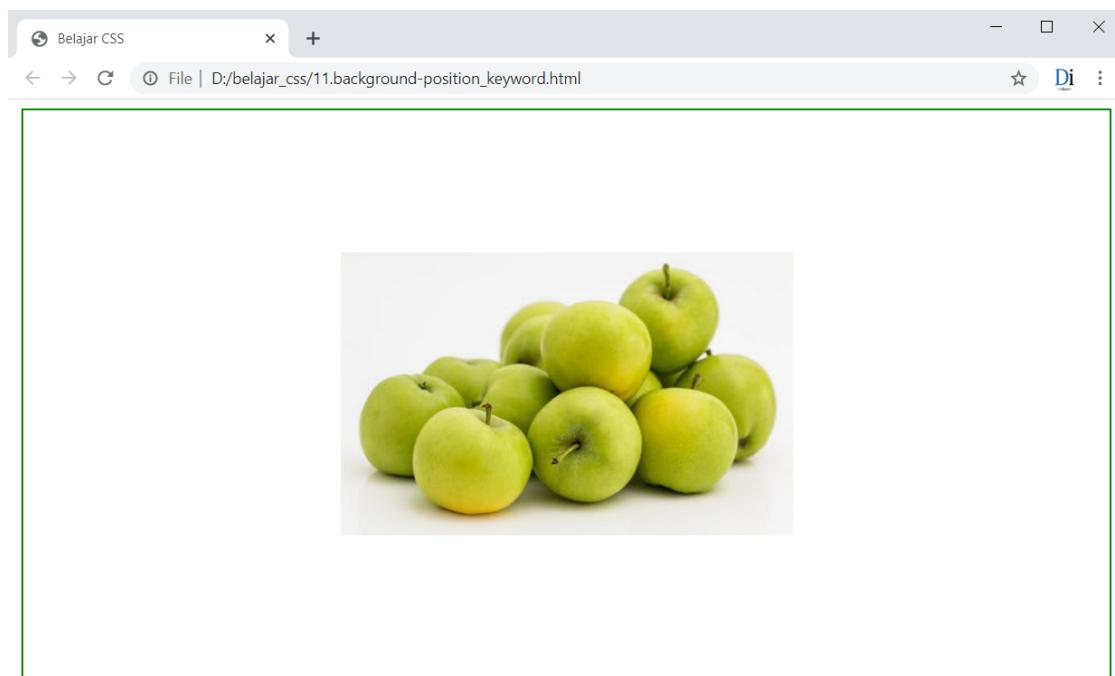
Berikut contohnya:

11.background-position_keyword.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
```

CSS Background

```
8     width: 960px;
9     height: 500px;
10    border: 2px solid green;
11    margin: 0 auto;
12    background-image: url('gambar/green_apple.jpg');
13    background-repeat: no-repeat;
14    background-position: center center;
15  }
16 </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Hasil tampilan property background-position: center center

Dalam kode program ini saya menggunakan property `background-position: center center`, sehingga gambar apel akan tampil di tengah-tengah tag `<div>`.

Nilai property `background-position` ini bisa dicampur seperti: `background-position: right 20px`, yang berarti gambar akan berada di kanan element dan berjarak 20 pixel dari sisi atas.

Khusus untuk nilai keyword, kita boleh membalik urutan posisi nilainya. Property `background-position: right bottom` bisa juga ditulis menjadi `background-position: bottom right`.

Nilai persen untuk `background-position`

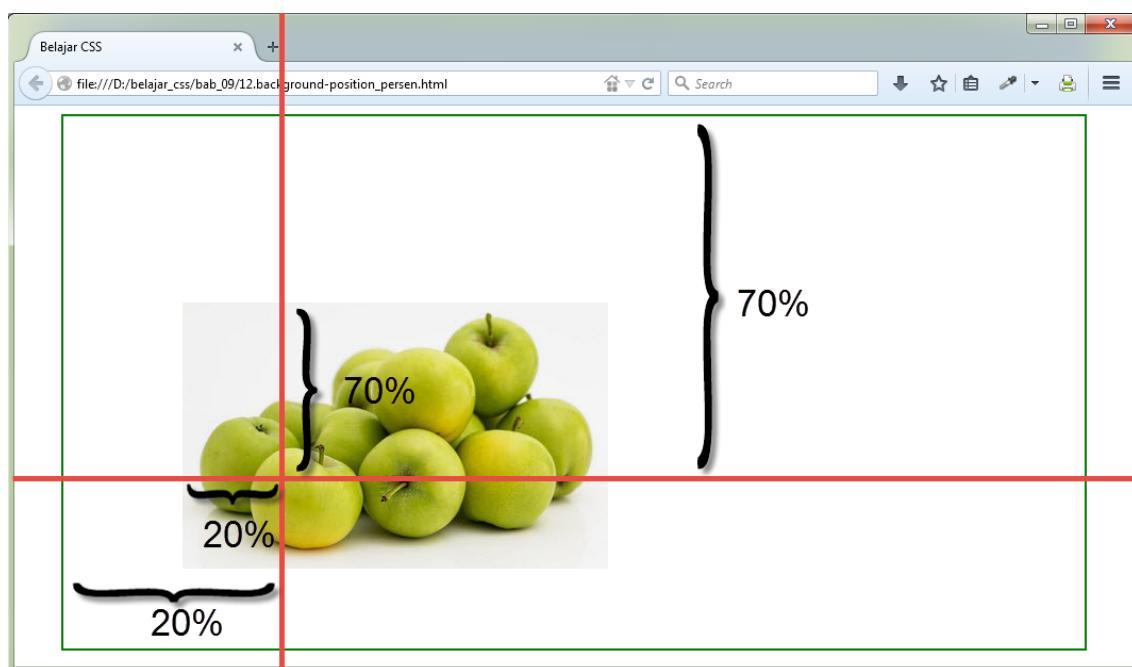
Jika kita memakai satuan % untuk `background-position`, perhitungannya menjadi lebih rumit.

Property `background-position: 20% 70%` berarti titik koordinat gambar di posisi 20% pada sumbu x dan 70% pada sumbu y akan bertepatan dengan koordinat element di 20% pada sumbu x dan 70% pada sumbu y.

Untuk lebih jelasnya, perhatikan gambar berikut:

12.background-position_persen.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 500px;
10      border: 2px solid green;
11      margin: 0 auto;
12      background-image: url('gambar/green_apple.jpg');
13      background-repeat: no-repeat;
14      background-position: 20% 70%;
15    }
16  </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



tengah-tengah element, dimana koordinat 50% sumbu x dan y gambar berhimpit dengan koordinat 50% pada sumbu x dan y dari element. Nilai ini sama artinya dengan `background-position: center center`.

Satu, tiga atau empat nilai untuk `background-position`

Dalam beberapa contoh `background-position` sebelum ini, kita memakai 2 buah nilai untuk posisi background. Selain itu, property `background-position` juga bisa ditulis dengan 1 nilai saja.

Jika nilai property `background-position` ditulis hanya satu, nilai tersebut dihitung sebagai koordinat sumbu x, sedangkan untuk koordinat sumbu y akan di set menjadi `center`.

Ini berarti property `background-position: 100px` sama artinya dengan `background-position: 100px center`. Dan property `background-position: right` sama artinya dengan `background-position: right center`.

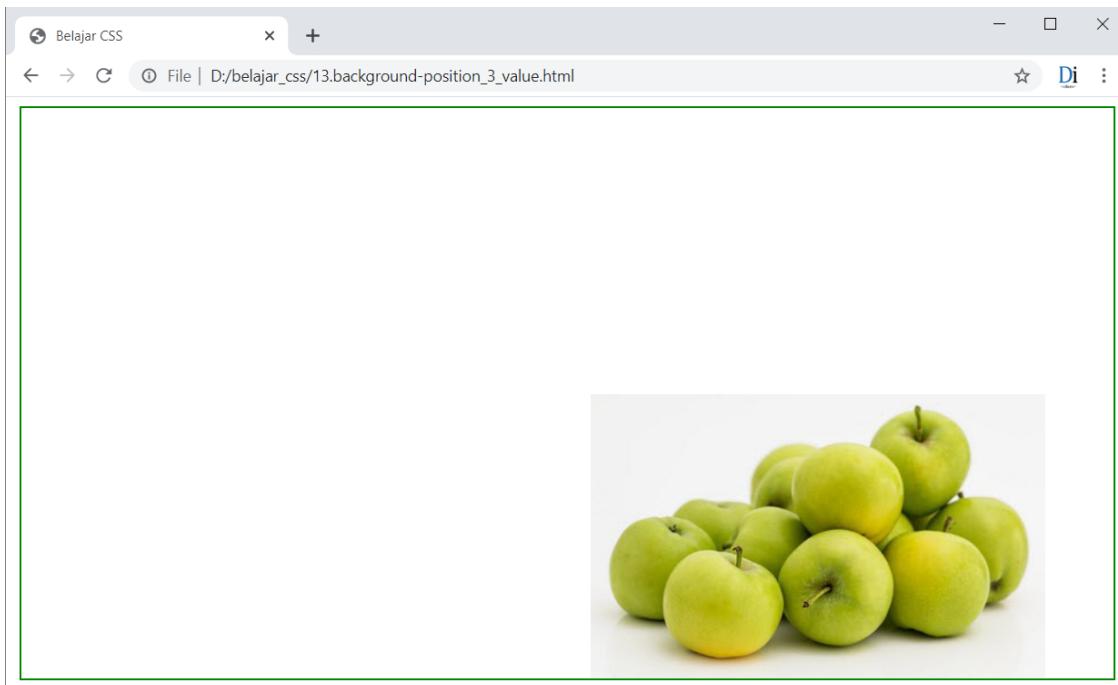
Property `background-position` juga bisa ditulis dengan 3 atau 4 nilai, namun fungsinya akan berbeda.

Sebagai contoh, `background-position: right 20px top 50px` berarti: tempatkan gambar background 20 pixel dari kanan (right), dan 50 pixel dari atas.

Atau `background-position: right 60px bottom` berarti: tempatkan gambar background 60 pixel dari kanan (right), dan berada di bagian bawah. Berikut contohnya:

13.background-position_3_value.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 500px;
10      border: 2px solid green;
11      margin: 0 auto;
12      background-image: url('gambar/green_apple.jpg');
13      background-repeat: no-repeat;
14      background-position: right 60px bottom;
15    }
16  </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Hasil dari property background-position: right 60px bottom

9.7. Property background-position CSS3

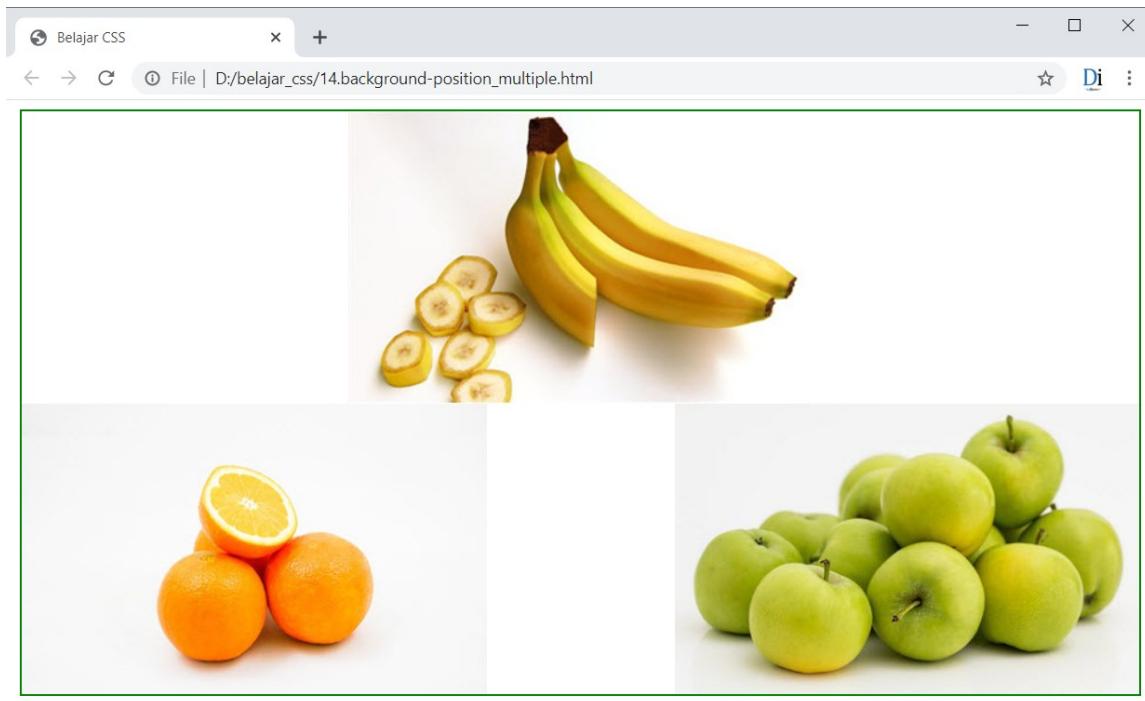
Dalam pembahasan mengenai property `background-image`, kita bisa menempatkan lebih dari 1 gambar untuk background. Dengan tambahan `background-position`, posisi setiap gambar tersebut juga bisa diatur secara terpisah.

Berikut contohnya:

14.background-position_multiple.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 960px;
9              height: 500px;
10             border: 2px solid green;
11             margin: 0 auto;
12             background-image: url('gambar/green_apple.jpg'),
13                             url('gambar/banana.jpg'),
14                             url('gambar/orange.jpg');
15             background-position: right bottom, 50% 0%, left bottom;
16             background-repeat: no-repeat;
17         }
18     </style>
19 </head>
```

```
20 <body>
21   <div> </div>
22 </body>
23 </html>
```



Gambar: Multiple background-position

Dalam kode ini saya menggunakan 3 gambar background, perhatikan cara penulisan property `background-image` yang diikuti dengan `background-position`. Setiap gambar ini berpasangan dengan posisinya masing-masing.

Posisi gambar pertama dari `background-image` ditentukan oleh nilai pertama dari `background-position`, begitu seterusnya untuk gambar kedua dan ketiga.

9.8. Property `background-size` CSS3

Property `background-size` merupakan salah satu property baru dari CSS3. Property ini berfungsi untuk mengatur ukuran gambar background. Kita bisa membesarkan atau mengecilkan gambar tergantung nilai yang diberikan.

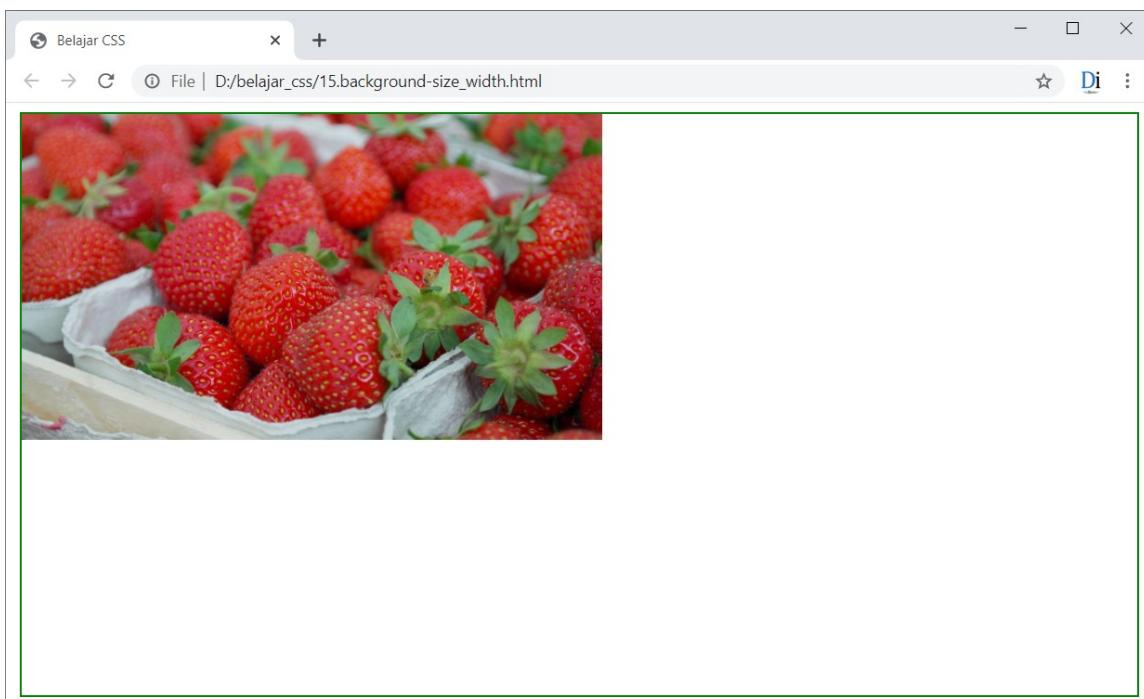
Nilai property `background-size` bisa diisi dengan satuan length seperti `px`, `em`, `%` atau bisa juga berupa keyword `auto`, `contain` dan `cover`.

Untuk satuan `length`, property `background-size` bisa diisi dengan 1 atau 2 nilai. Jika menggunakan 1 nilai, itu berfungsi untuk mengatur lebar gambar (`width`), sedangkan untuk tinggi (`height`) akan menyesuaikan secara proporsional. Berikut contohnya:

CSS Background

15.background-size_width.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 500px;
10      border: 2px solid green;
11      margin: 0 auto;
12      background-image: url('gambar/strawberries.jpg');
13      background-repeat: no-repeat;
14      background-size: 500px;
15    }
16  </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Contoh penggunaan property background-size dengan 1 nilai

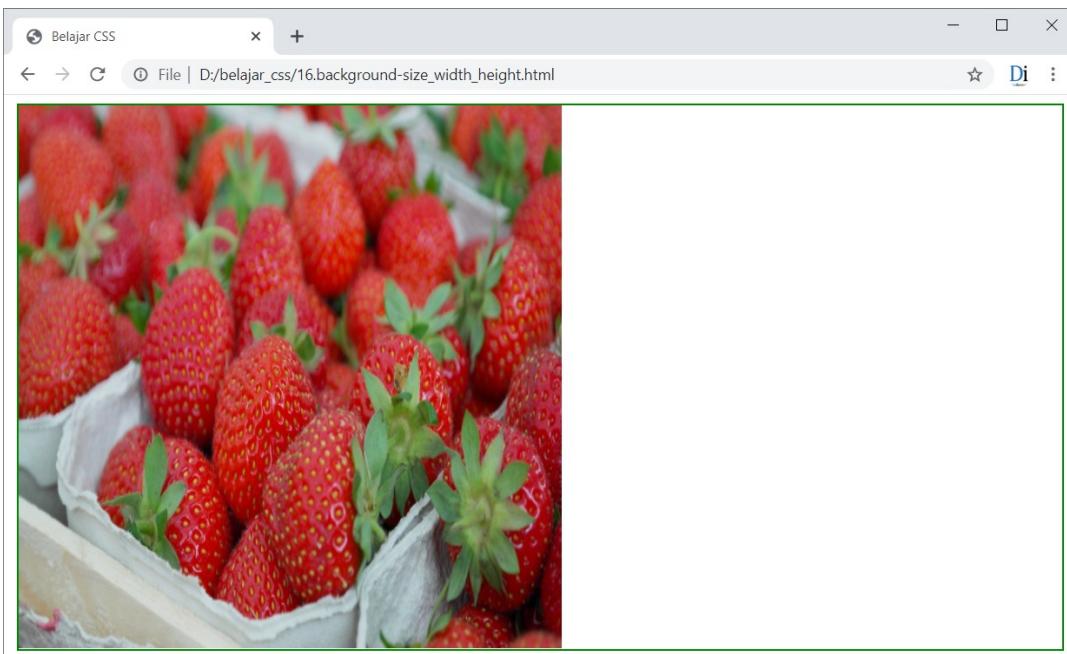
Ukuran asli gambar `strawberries.jpg` adalah 1366 x 768 pixel. Ketika di set dengan property `background-size: 500px`, gambar tersebut akan diperkecil. Property `background-repeat: no-repeat` sengaja juga saya tambah agar gambar tidak tampil secara berulang.

Jika ditulis dengan 2 buah nilai, nilai kedua dipakai untuk mengatur tinggi (height) dari gambar seperti contoh berikut:

CSS Background

16.background-size_width_height.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 500px;
10      border: 2px solid green;
11      margin: 0 auto;
12      background-image: url('gambar/strawberries.jpg');
13      background-repeat: no-repeat;
14      background-size: 500px 500px;
15    }
16  </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Contoh penggunaan property background-size dengan 2 nilai

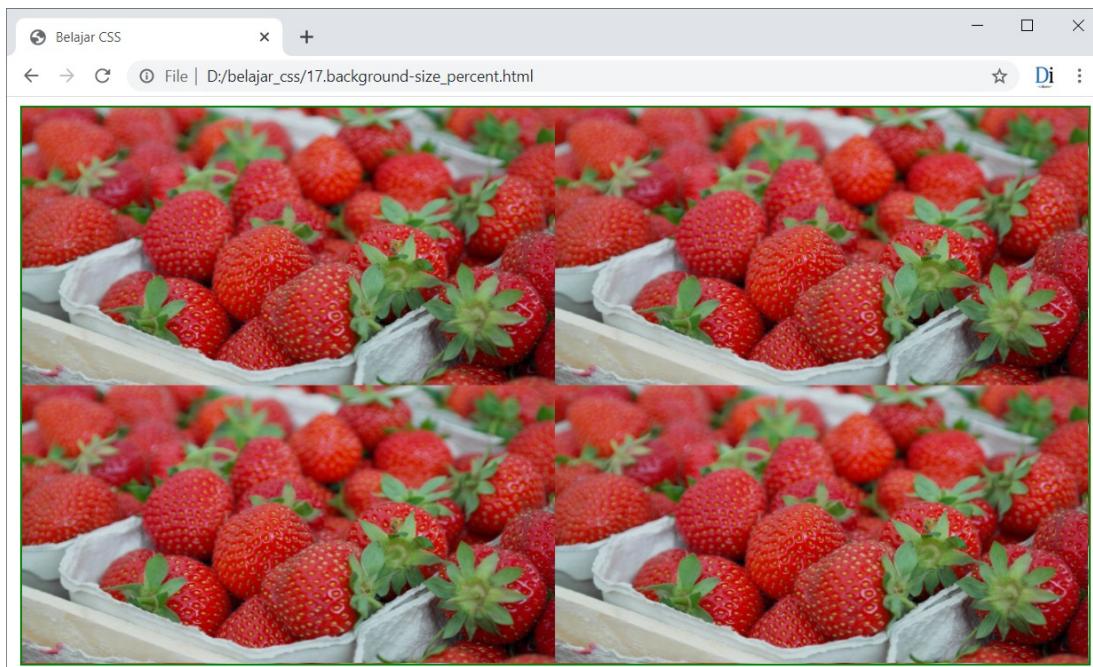
Kali ini gambar `strawberries.jpg` akan "dipaksa" tampil dengan ukuran 500×500 pixel.

Menggunakan satuan persen untuk `background-size` sedikit berbeda dibandingkan dengan pixel. Nilai persen dihitung berdasarkan lebar dan tinggi element, bukan dari lebar dan tinggi gambar.

Sebagai contoh, `background-size: 50% 50%` akan menghasilkan gambar yang persis seukuran $\frac{1}{4}$ element:

17.background-size_percent.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 500px;
10      border: 2px solid green;
11      margin: 0 auto;
12      background-image: url('gambar/strawberries.jpg');
13      background-size: 50% 50%;
14    }
15  </style>
16 </head>
17 <body>
18   <div> </div>
19 </body>
20 </html>
```



Gambar: Nilai persen untuk background-size

Gambar strawberries.jpg akan tampil pas sebanyak 4 kali karena saya tidak mencantumkan background-repeat: no-repeat.

Efek yang menarik ada di 2 nilai keyword untuk background-size, yakni contain dan cover.

Jika menggunakan background-size:contain, web browser berusaha menampilkan 1 gambar background utuh, namun tetap mempertahankan rasio gambar. Maka jika dimensi gambar tidak pas dengan ukuran element, akan terdapat sisa ruang kosong.

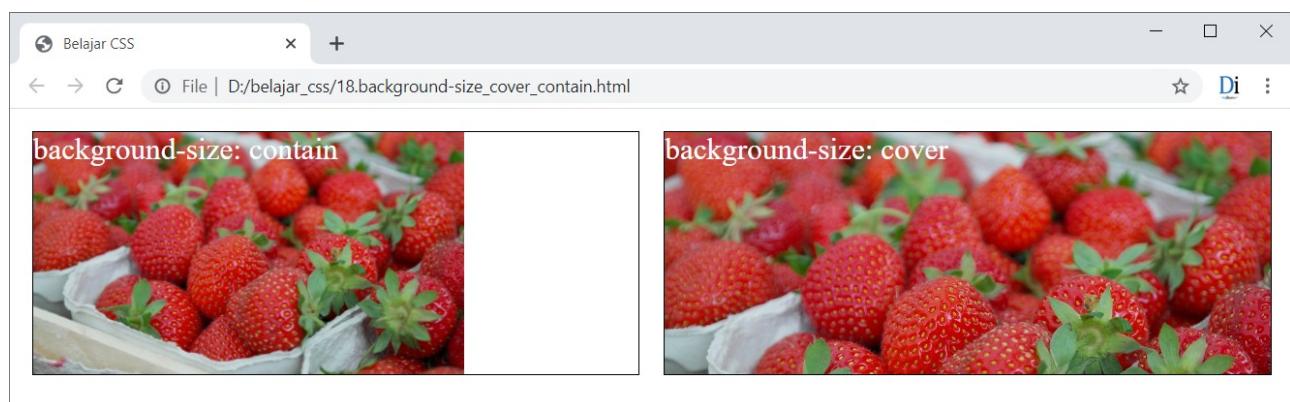
Sedangkan jika memakai `background-size:cover`, gambar akan disesuaikan oleh web browser agar 1 gambar dapat menutupi seluruh element. Ini dilakukan dengan memperbesar gambar background dengan tetap mempertahankan rasio gambar. Efeknya, bisa jadi ada potongan gambar yang tidak terlihat.

Berikut contoh tampilan nilai `contain` dan `cover`:

18.background-size_cover_contain.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 500px;
9              height: 200px;
10             margin: 10px;
11             border: 1px solid black;
12             color: white;
13             font-size: 25px;
14             background-image: url('gambar/strawberries.jpg');
15             background-repeat: no-repeat;
16             float: left;
17         }
18         .satu {
19             background-size: contain;
20         }
21         .dua {
22             background-size: cover;
23         }
24     </style>
25 </head>
26 <body>
27     <div class="satu">background-size: contain</div>
28     <div class="dua">background-size: cover</div>
29 </body>
30 </html>
```



Gambar: Perbedaan nilai `background-size contain` (kiri) dan `cover` (kanan)

Gambar pertama menggunakan `background-size:contain`, perhatikan terdapat ruang kosong di sisi kanan element.

Gambar kedua menggunakan `background-size:cover`. Kali ini gambar tampil menutupi seluruh element, namun karena rasio element tidak cocok dengan gambar, bagian gambar kanan bawah jadi tidak terlihat.

Nilai `background-size:cover` sangat sesuai jika kita ingin memakai gambar besar sebagai background, karena web browser secara otomatis akan mengatur ukuran gambar agar pas menutupi seluruh element.

Nilai keyword terakhir untuk `background-size` adalah `auto` yang merupakan nilai default. Ketika kita menggunakan 1 nilai seperti `background-size: 500px`, sebenarnya diproses sebagai `background-size: 500px auto`. Nilai `auto` berarti akan mempertahankan rasio gambar.

9.9. Property `background-attachment`

Property `background-attachment` dipakai untuk mengatur apakah gambar melekat (*attach*) kepada halaman atau ke tampilan jendela (*viewport*). Nilai yang tersedia adalah `scroll` (nilai default), `fixed`, dan `local` (baru ditambahkan pada CSS3).

Jika kita menggunakan nilai `background-attachment:scroll`, gambar background akan menempel ke halaman saat ini. Ini merupakan default style bawaan web browser jika property `background-attachment` tidak ditulis. Berikut contoh penggunaannya:

19. `background-attachment_scroll.html`

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 960px;
9              height: 1500px;
10             border: 2px solid green;
11             background-image: url('gambar/banana.jpg');
12             background-repeat: no-repeat;
13             background-position: 100px 100px;
14             background-attachment: scroll;
15         }
16     </style>
17 </head>
18 <body>
19     <div> </div>
20 </body>
21 </html>
```

CSS Background



Gambar: Penggunaan background-attachment: scroll, gambar akan melekat ke halaman

Ketika kita men-scroll halaman ke bagian bawah, gambar `banana.jpg` tetap tinggal di atas. Gambar ini bisa saja tidak terlihat lagi ketika sudah sampai ke bagian paling bawah halaman.

Si sisi lain, nilai `background-attachment:fixed` menempatkan gambar background melekat ke jendela tampilan yang disebut juga sebagai viewport:

20.background-attachment_fixed.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div {
8       width: 960px;
9       height: 1500px;
10      border: 2px solid green;
11      background-image: url('gambar/banana.jpg');
12      background-repeat: no-repeat;
13      background-position: 100px 100px;
14      background-attachment: fixed;
15    }
16  </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Penggunaan background-attachment: fixed, gambar akan melekat ke viewport

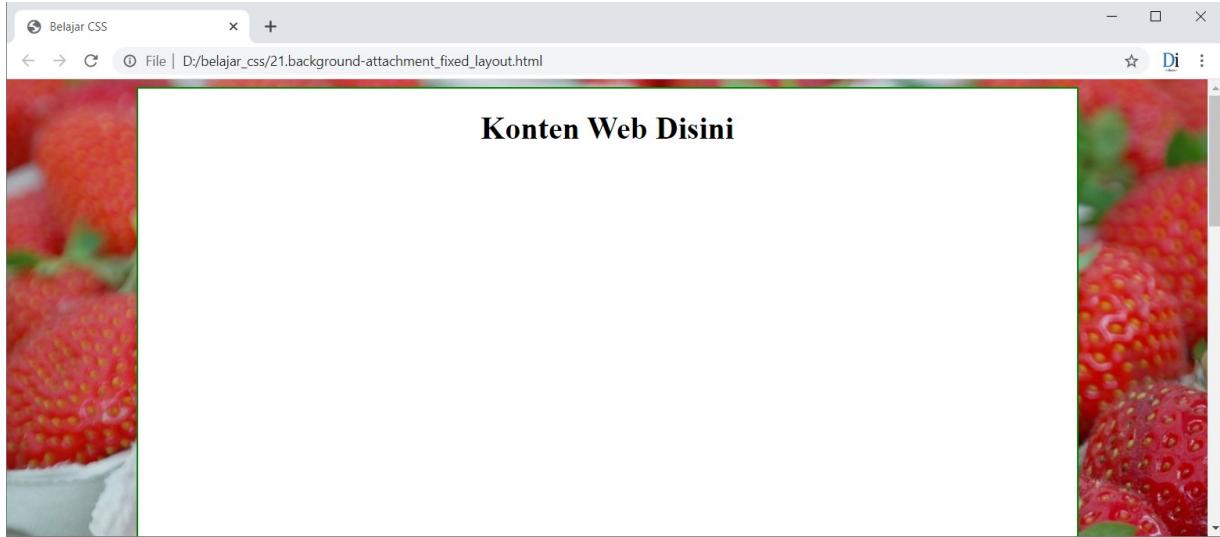
Efek ini baru akan terlihat saat kode di atas dijalankan langsung, kemudian scroll halaman web ke bawah dan ke atas. Posisi gambar banana.jpg akan tetap dan tidak berubah. Efek seperti ini cocok dipakai untuk gambar background besar di latar belakang halaman seperti contoh berikut:

21.background-attachment_fixed_layout.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              background-image: url('gambar/strawberries.jpg');
9              background-size: cover;
10             background-attachment: fixed;
11         }
12         div {
13             width: 960px;
14             height: 1500px;
15             border: 2px solid green;
16             margin: 0 auto;
17             text-align: center;
18             background-color: white;
19         }
20     </style>
21 </head>
22 <body>
23     <div>
24         <h1>Konten Web Disini</h1>
25     </div>
26 </body>
27 </html>
```

CSS Background



Gambar: Penggunaan background-attachment: fixed pada latar belakang halaman

Dalam contoh ini saya memakai gambar `strawberries.jpg` sebagai background. Dengan `background-attachment: fixed`, gambar akan tetap berada di latar belakang halaman dan pas untuk ukuran 1 gambar (menggunakan property `background-size: cover`).

Nilai terakhir untuk `background-attachment: local` hampir mirip dengan `background-attachment: scroll`, namun gambar background melekat ke element itu sendiri, bukan ke viewport. Property ini baru ditambahkan ke CSS3.

Namun efek `local` ini agak susah untuk dibuat karena melibatkan element yang `overflow` (berisi konten yang melebihi height element tersebut). Anda bisa kunjungi halaman [codepen](#)¹⁹ yang membandingkan ketiga nilai `scroll`, `fixed`, dan `local`.

A screenshot of the CodePen interface comparing three examples of `background-attachment`.

- Scroll:** Shows a background image of a person's face with a scrollable content area below it. The background image moves as the content is scrolled.
- Fixed:** Shows a background image of a person's face with a fixed content area. The background image remains static while the content scrolls over it.
- Local:** Shows a background image of a person's face with a fixed content area. The background image is attached to the content area, moving along with it.

The left sidebar shows the HTML and CSS code for each example. The CSS for the `body` includes:

```
1 body {  
2   text-align: center;  
3   display: grid;  
4   grid-gap: 3em;  
5   grid-template-columns: repeat(3, 1fr);  
6 }
```

Gambar: Perbandingan antara background-attachment scroll, fixed, dan local

19 <https://codepen.io/kevinpowell/pen/bzMvzN>

9.10. Property background-origin dan background-clip CSS3

Kedua property ini merupakan property baru di dalam modul background CSS3. Saya menyatukan keduanya karena berfungsi mirip.

Property `background-origin` dan `background-clip` dipakai untuk menentukan dari mana gambar/warna background mulai dibentangkan, apakah dari border, dari padding, atau hanya untuk konten saja.

Nilai yang bisa diisi adalah salah satu dari keyword `border-box`, `padding-box`, dan `content-box`. Berikut penjelasannya:

- **`border-box`**: background mulai dari border, sehingga akan tampil di bagian border, padding dan konten.
- **`padding-box`**: background mulai dari padding, dan akan tampil di bagian padding dan konten saja, sedangkan di bagian border dikosongkan.
- **`content-box`**: background hanya pada bagian konten saja, sehingga tidak akan tampil di border dan padding.

Jadi jika menggunakan nilai yang sama, apa perbedaan dari `background-origin` dengan `background-clip`?

Sederhananya, `background-origin` dipakai untuk mengatur gambar background, sedangkan `background-clip` untuk mengatur warna background. Mari kita lihat contohnya:

22.background-clip.html

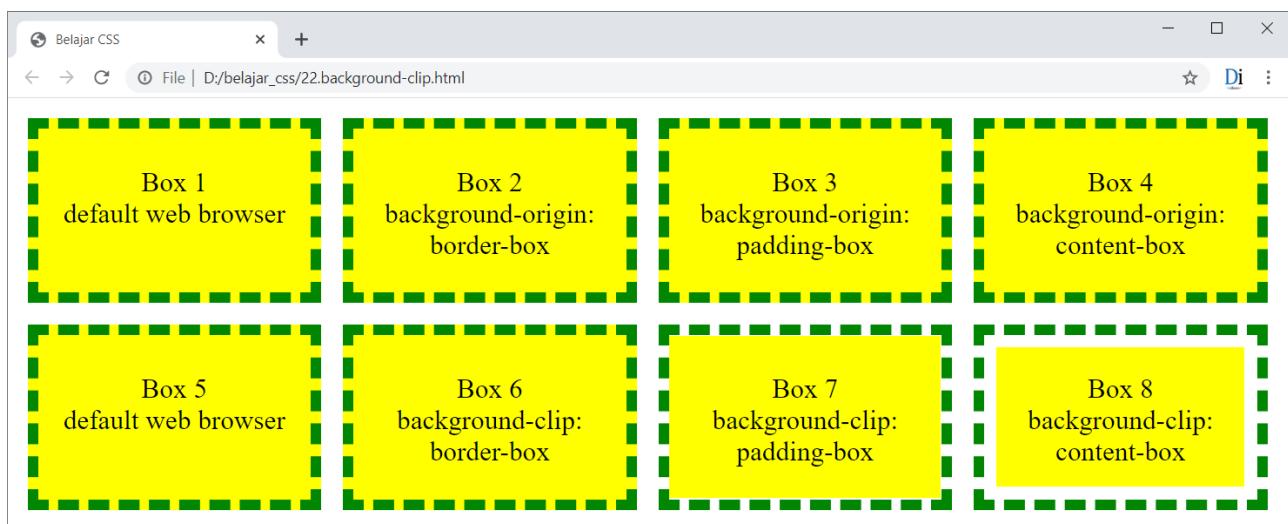
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 230px;
9              height: 130px;
10             text-align: center;
11             float: left;
12             color: black;
13             font-size: 25px;
14             border: 10px dashed green;
15             padding: 10px;
16             margin: 10px;
17             background-color: yellow;
18         }
19         .satu { background-origin: border-box; }
20         .dua { background-origin: padding-box; }
21         .tiga { background-origin: content-box; }
22         .empat { background-clip: border-box; }
```

```

23     .lima { background-clip: padding-box; }
24     .enam { background-clip: content-box; }
25   </style>
26 </head>
27 <body>
28   <div><p>Box 1 <br>default web browser</p></div>
29   <div class="satu"><p>Box 2 <br> background-origin: border-box</p></div>
30   <div class="dua"><p>Box 3 <br> background-origin: padding-box</p></div>
31   <div class="tiga"><p>Box 4 <br> background-origin: content-box</p></div>
32   <div><p>Box 5 <br> default web browser</p></div>
33   <div class="empat"><p>Box 6 <br> background-clip: border-box</p></div>
34   <div class="lima"><p>Box 7 <br> background-clip: padding-box</p></div>
35   <div class="enam"><p>Box 8 <br> background-clip: content-box</p></div>
36 </body>
37 </html>

```



Gambar: Contoh penggunaan background-origin dan background-clip pada background-color

Dalam kode program di atas saya membuat 8 box untuk masing-masing nilai. Setiap box berukuran 230 x 130 pixel, dengan border 10px, padding 10px, dan margin 10pixel. Saya juga menggunakan beberapa property lain yang sudah kita pelajari sejauh ini (kecuali property `float:left` yang akan dipelajari dalam bab berikutnya).

Box 1 dan Box 5 tidak menggunakan property `background-origin` maupun `background-clip`. Ini hanya sebagai patokan bagaimana style default web browser saat menampilkan warna background.

Pada box 2, 3 dan 4, saya memakai property `background-origin` dengan berbagai nilai. Terlihat tidak ada efek apapun ke dalam box kita, dimana tampil sama default style.

Pada box 6, 7 dan 8, saya menggunakan property `background-clip`. Kali ini kita bisa melihat efeknya.

Untuk box 6, property `background-clip:border-box` akan menampilkan warna background pada seluruh element, termasuk bagian border, padding dan konten. Perhatikan warna kuning background ikut masuk di bawah border. Nilai `background-clip: border-box` ini merupakan

style default dari web browser.

Untuk box 7, property `background-clip: border-box` akan menampilkan warna background hanya pada bagian padding dan konten saja. Kali ini warna background tidak muncul di sisi border (perhatikan warna putih di bawah border).

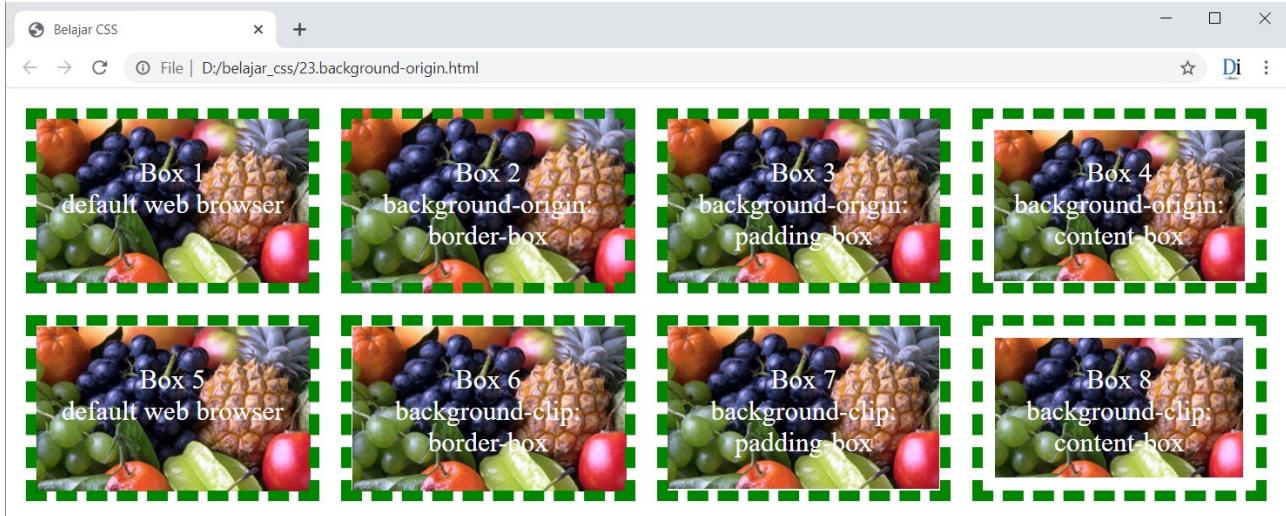
Terakhir, untuk box 8 saya menggunakan `background-clip: content-box`. Sekarang warna background hanya tampil di dalam konten saja. Pada bagian padding dan border, tidak terlihat warna kuning.

Bagaimana jika kita menggunakan gambar background? Berikut contoh prakteknya:

23.background-origin.html

```
<!DOCTYPE html>
<html lang="id">
<head>
<meta charset="UTF-8">
<title>Belajar CSS</title>
<style>
div {
    width: 230px;
    height: 130px;
    text-align: center;
    float: left;
    color: white;
    font-size: 25px;
    border: 10px dashed green;
    padding: 10px;
    margin: 10px;
    background-image: url('gambar/fruit.jpg');
    background-size: cover;
    background-repeat: no-repeat;
}
.satu { background-origin: border-box; }
.dua { background-origin: padding-box; }
.tiga { background-origin: content-box; }
.empat { background-clip: border-box; }
.lima { background-clip: padding-box; }
.enam { background-clip: content-box; }
</style>
</head>
<body>
<div><p>Box 1 <br>default web browser</p></div>
<div class="satu"><p>Box 2 <br> background-origin: border-box</p></div>
<div class="dua"><p>Box 3 <br> background-origin: padding-box</p></div>
<div class="tiga"><p>Box 4 <br> background-origin: content-box</p></div>
<div><p>Box 5 <br> default web browser</p></div>
<div class="empat"><p>Box 6 <br> background-clip: border-box</p></div>
<div class="lima"><p>Box 7 <br> background-clip: padding-box</p></div>
<div class="enam"><p>Box 8 <br> background-clip: content-box</p></div>
</body>
</html>
```

CSS Background



Gambar: Contoh penggunaan background-origin dan background-clip pada background-image

Kode HTML dan CSS di atas sama persis seperti sebelumnya. Namun kali ini saya menggunakan gambar background melalui property `background-image`. Gambar `fruit.jpg` berukuran 250 x 150 pixel, di set dengan `background-size: cover` dan `background-repeat: no-repeat`.

Kali ini perhatikan box 2, 3 dan 4. Property `background-origin` akan membuat ruang gambar tampil berbeda-beda. Efek ini sama dengan penjelasan pada `background-clip`, hanya kali ini kita menggunakan gambar.

Khusus pada box 8, property `background-clip: content-box` juga berefek kepada gambar, dimana hanya tampil di bagian konten saja. Tapi jika diperhatikan dengan teliti, gambar kita sedikit terpotong di bagian sisi paling luar (bandingkan dengan box 4).

Ini juga jadi pembeda antara `background-clip` dengan `background-origin`, dimana pada `background-origin`, gambar akan diperbesar/diperkecil sesuai lebar element. Sedangkan pada `background-clip`, gambar akan dipotong.

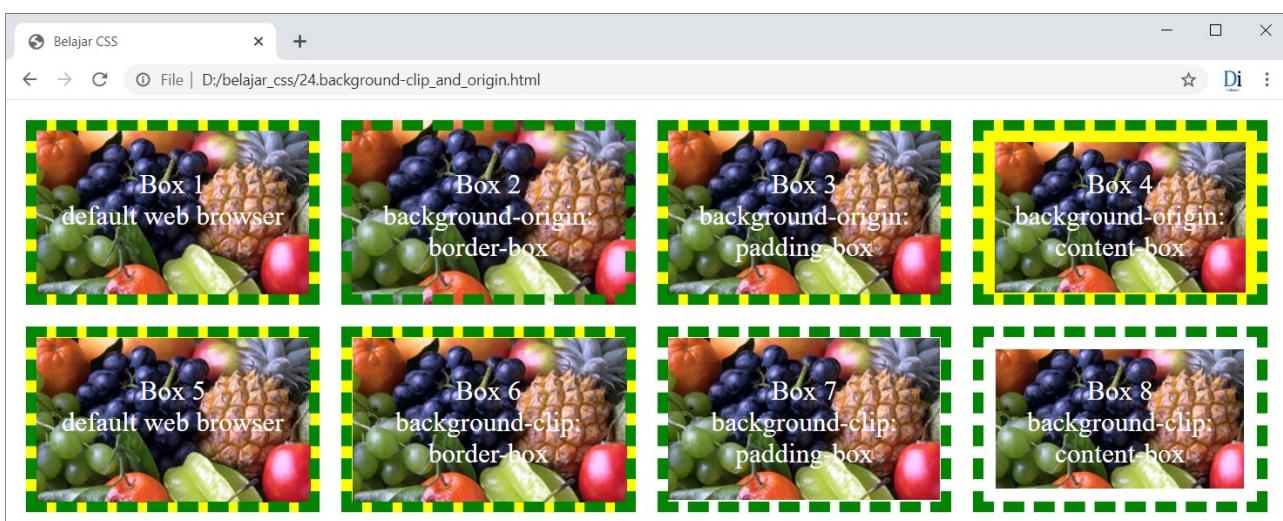
Kita juga bisa menggabungkan warna background dan gambar background seperti contoh berikut:

24.background-clip_and_origin.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 230px;
9              height: 130px;
10             text-align: center;
11             float: left;
```

CSS Background

```
12     color: white;
13     font-size: 25px;
14     border: 10px dashed green;
15     padding: 10px;
16     margin: 10px;
17     background-image: url('gambar/fruit.jpg');
18     background-size: cover;
19     background-repeat: no-repeat;
20     background-color: yellow;
21 }
22 .satu { background-origin: border-box; }
23 .dua { background-origin: padding-box; }
24 .tiga { background-origin: content-box; }
25 .empat { background-clip: border-box; }
26 .lima { background-clip: padding-box; }
27 .enam { background-clip: content-box; }
28 </style>
29 </head>
30 <body>
31 <div><p>Box 1 <br>default web browser</p></div>
32 <div class="satu"><p>Box 2 <br> background-origin: border-box</p></div>
33 <div class="dua"><p>Box 3 <br> background-origin: padding-box</p></div>
34 <div class="tiga"><p>Box 4 <br> background-origin: content-box</p></div>
35 <div><p>Box 5 <br> default web browser</p></div>
36 <div class="empat"><p>Box 6 <br> background-clip: border-box</p></div>
37 <div class="lima"><p>Box 7 <br> background-clip: padding-box</p></div>
38 <div class="enam"><p>Box 8 <br> background-clip: content-box</p></div>
39 </body>
40 </html>
```



Gambar: Efek property property background-clip dan background-origin pada gambar dan warna background

Sekarang kita bisa lihat dengan lebih jelas perbedaan masing-masing tampilan untuk property background-clip dan background-origin.

9.11. Property background-blend-mode CSS3

Property background-blend-mode berfungsi untuk mendapatkan efek gabungan dari gambar background dengan warna background. Jika anda pernah menggunakan aplikasi pengolah gambar seperti Adobe Photoshop, tentunya sudah tidak asing dengan fitur **blend-mode**.

Property ini baru ditambah pada CSS3, namun bukan pada CSS3 background, tetapi pada modul terpisah: **Compositing and Blending Level 2**.

Nilai yang bisa diinput adalah salah satu dari 16 keyword. Keyword inilah yang akan menentukan bagaimana gambar background 'bergabung' dengan warna background. Ke-16 keyword itu adalah:

- ◆ normal
- ◆ multiply
- ◆ overlay
- ◆ screen
- ◆ darken
- ◆ lighten
- ◆ color-dodge
- ◆ color-burn
- ◆ hard-light
- ◆ soft-light
- ◆ difference
- ◆ exclusion
- ◆ hue
- ◆ saturation
- ◆ color
- ◆ luminosity

Langsung saja kita lihat efek dari setiap nilai ini:

25.background-blend-mode.html

```

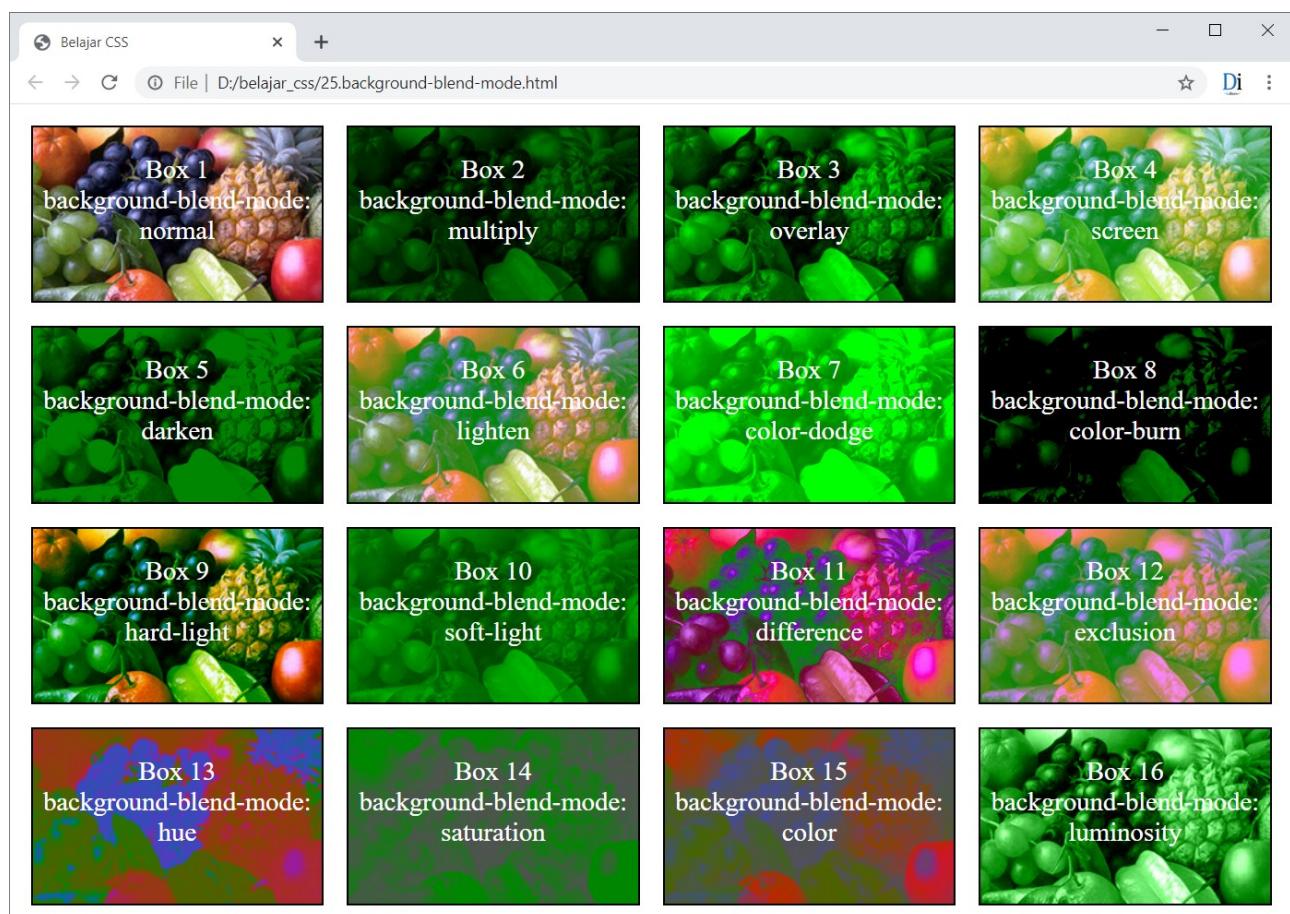
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 250px;
9              height: 150px;
10             text-align: center;
11             float: left;
12             color: white;

```

```
13     font-size: 23px;
14     border: 2px solid black;
15     margin: 10px;
16     background-image: url('gambar/fruit.jpg');
17     background-size: cover;
18     background-repeat: no-repeat;
19     background-color: green;
20 }
21 .satu { background-blend-mode: normal; }
22 .dua { background-blend-mode: multiply; }
23 .tiga { background-blend-mode: overlay; }
24 .empat { background-blend-mode: screen; }
25 .lima { background-blend-mode: darken; }
26 .enam { background-blend-mode: lighten; }
27 .tujuh { background-blend-mode: color-dodge; }
28 .delapan { background-blend-mode: color-burn; }
29 .sembilan { background-blend-mode: hard-light; }
30 .sepuluh { background-blend-mode: soft-light; }
31 .sebelas { background-blend-mode: difference; }
32 .duabelas { background-blend-mode: exclusion; }
33 .tigabelas { background-blend-mode: hue; }
34 .empatbelas { background-blend-mode: saturation; }
35 .limabelas { background-blend-mode: color; }
36 .enambelas { background-blend-mode: luminosity; }
37 </style>
38 </head>
39 <body>
40 <div class="satu">
41   <p>Box 1 <br> background-blend-mode: normal</p>
42 </div>
43 <div class="dua">
44   <p>Box 2 <br> background-blend-mode: multiply</p>
45 </div>
46 <div class="tiga">
47   <p>Box 3 <br> background-blend-mode: overlay</p>
48 </div>
49 <div class="empat">
50   <p>Box 4 <br> background-blend-mode: screen</p>
51 </div>
52 <div class="lima">
53   <p>Box 5 <br> background-blend-mode: darken</p>
54 </div>
55 <div class="enam">
56   <p>Box 6 <br> background-blend-mode: lighten</p>
57 </div>
58 <div class="tujuh">
59   <p>Box 7 <br> background-blend-mode: color-dodge</p>
60 </div>
61 <div class="delapan">
62   <p>Box 8 <br> background-blend-mode: color-burn</p>
63 </div>
64 <div class="sembilan">
65   <p>Box 9 <br> background-blend-mode: hard-light</p>
66 </div>
67 <div class="sepuluh">
```

CSS Background

```
68      <p>Box 10 <br> background-blend-mode: soft-light</p>
69  </div>
70  <div class="sebelas">
71    <p>Box 11 <br> background-blend-mode: difference</p>
72  </div>
73  <div class="duabelas">
74    <p>Box 12 <br> background-blend-mode: exclusion</p>
75  </div>
76  <div class="tigabelas">
77    <p>Box 13 <br> background-blend-mode: hue</p>
78  </div>
79  <div class="empatbelas">
80    <p>Box 14 <br> background-blend-mode: saturation</p>
81  </div>
82  <div class="limabelas">
83    <p>Box 15 <br> background-blend-mode: color</p>
84  </div>
85  <div class="enambelas">
86    <p>Box 16 <br> background-blend-mode: luminosity</p>
87  </div>
88 </body>
89 </html>
```



Gambar: Tampilan ke-16 efek background-blend-mode

Pada contoh ini terdapat 16 tag `<div>` dengan berbagai nilai `background-blend-mode`. Silahkan

anda ganti warna `background-color`: green dengan warna lain seperti red, yellow, white, dan lihat bagaimana hasilnya. Efek ini sangat menarik untuk membuat design foto.

9.12. Property background (Shorthand Notation)

Dari awal bab hingga sekarang setidaknya kita telah membahas 9 property background. Delapan diantaranya bisa disingkat melalui shorthand notation `background`. Ke-8 property tersebut adalah:

- ◆ `background-image`
- ◆ `background-position`
- ◆ `background-size`
- ◆ `background-repeat`
- ◆ `background-attachment`
- ◆ `background-origin`
- ◆ `background-clip`
- ◆ `background-color`

Berikut contoh urutan penulisan singkat property background:

```
div {
  background:
    url('gambar/strawberries.jpg') /* image */
    bottom right / 200px 200px      /* position / size */
    no-repeat                      /* repeat */
    scroll                         /* attachment */
    padding-box                    /* origin */
    content-box                    /* clip */
    red;                          /* color */
}
```

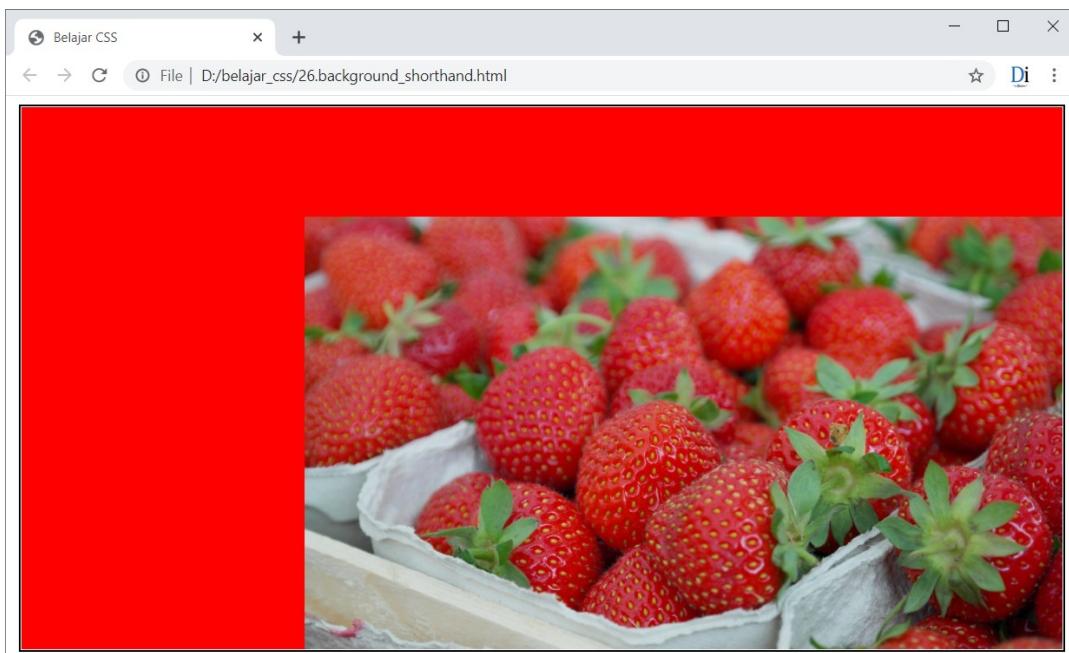
Setiap nilai dipisah dengan spasi, kecuali untuk `position` dan `size` yang dipisah dengan tanda garis miring (/). Berikut contoh penggunaannya di dalam kode CSS dan HTML:

26.background_shorthand.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7    div {
8      width: 960px;
9      height: 500px;
10     border: 2px solid black;
11     margin: 0 auto;
12     background: url('gambar/strawberries.jpg') bottom right / 700px 400px
13     no-repeat scroll padding-box content-box red;
```

CSS Background

```
14    }
15  </style>
16 </head>
17 <body>
18  <div> </div>
19 </body>
20 </html>
```



Gambar: Contoh hasil dari penulisan singkat property background

Property background di baris 11 – 12 sama artinya dengan 8 property berikut:

```
background-image: url('gambar/strawberries.jpg');
background-position: bottom right;
background-size: 700px 400px;
background-repeat: no-repeat;
background-attachment: scroll;
background-origin: padding-box;
background-clip: content-box;
background-color: red;
```

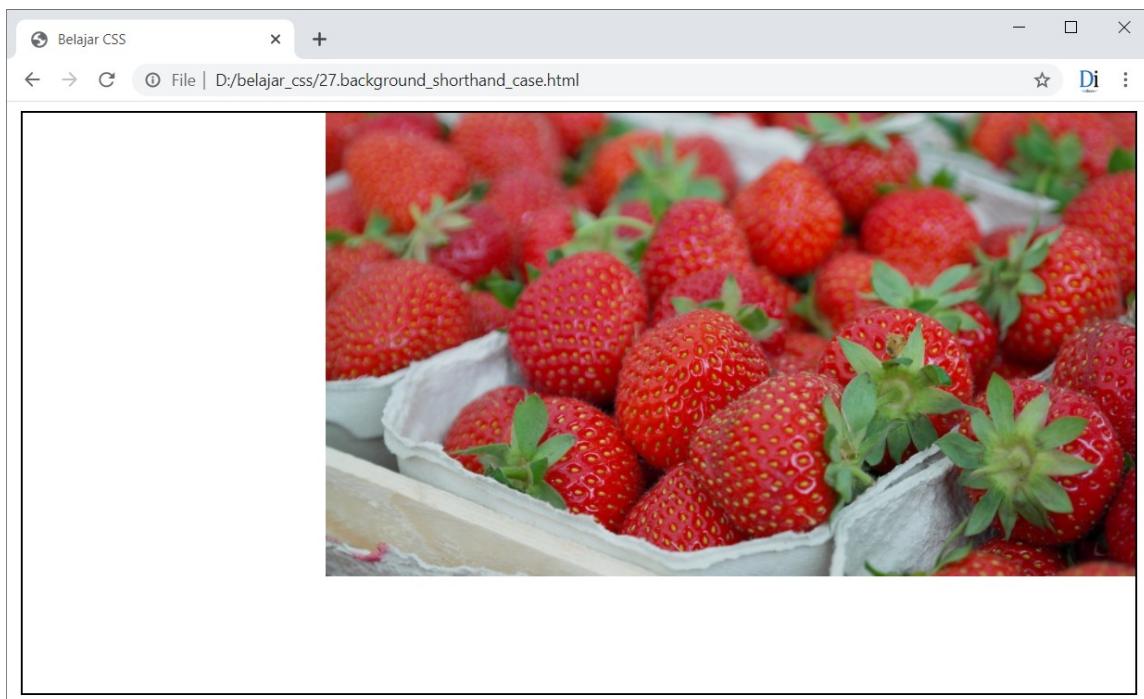
Walaupun praktis, kita juga harus hati-hati menggunakan background shorthand notation, berikut salah satu contoh kasusnya:

27.background_shorthand_case.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      div {
8        width: 960px;
```

CSS Background

```
9      height: 500px;
10     border: 2px solid black;
11     margin: 0 auto;
12     background-color: red ;
13     background: url('gambar/strawberries.jpg')
14                     top right / 700px 400px no-repeat;
15   }
16 </style>
17 </head>
18 <body>
19   <div> </div>
20 </body>
21 </html>
```



Gambar: Warna background tidak tampil!

Di baris 12 terdapat property `background-color: red`, namun seperti yang terlihat warna background tampil dengan warna putih. Apa yang terjadi?

Ini sama seperti kasus pada property `font shorthand notation`. Dimana `background shorthand notation` juga membawa nilai default yang akan menimpa style sebelumnya. Nilai-nilai tersebut adalah sebagai berikut:

```
background-image: none;
background-position: 0% 0%;
background-size: auto auto;
background-repeat: repeat;
background-origin: padding-box;
background-clip: border-box;
background-attachment: scroll;
background-color: transparent;
```

Jika kita tidak menulis salah satu nilai untuk property ini, nilai default-lah yang akan dipakai. Dalam contoh kasus di atas, property `background-color: red` akan ditimpakan oleh `background-color: transparent` yang berasal dari property background. Untuk mengatasi hal ini, kita bisa tulis `background-color: red` setelah property background:

```
background: url('strawberries.jpg') top right / 700px 400px no-repeat;  
background-color: red;
```

Dengan demikian efek cascade CSS akan menimpa warna transparent milik property background.

Saya pribadi cenderung menghindari menulis property background shorthand notation. Selain masalah di atas, cukup susah menghafal urutan penulisan property. Lebih baik tulis property background secara terpisah saja. Meskipun lebih panjang, tapi kodennya lebih mudah dipahami bagi kita dan juga bagi programmer lain.

Dalam bab ini kita telah membahas detail apa saja property yang berhubungan dengan background. Gambar background menjadi suatu efek visual yang sangat penting dalam sebuah web modern.

Berikutnya akan lanjut membahas **CSS Positioning**, yakni property CSS yang berkaitan dengan posisi sebuah element di dalam halaman website.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

10. CSS Positioning

Setelah memahami **box model** dan property **background**, kali ini kita masuk ke pembahasan tentang **CSS Positioning**. Dalam bab ini akan dibahas berbagai aspek terkait bagaimana cara mengatur penempatan element di dalam halaman HTML. Ini juga sebagai dasar bagi kita untuk merancang layout website.

10.1. Mengenal Normal Document Flow

Normal Document Flow adalah istilah keren untuk menyebut bagaimana element HTML diproses dan ditampilkan oleh web browser secara normal tanpa campur tangan CSS. Atau ini juga bisa dipahami sebagai *default style* bawaan web browser.

Sebagaimana yang kita lihat, web browser memproses setiap element berdasarkan urutannya, mulai dari atas ke bawah dan saling bertumpuk mulai dari element pertama, kedua, hingga element paling akhir.

Untuk element HTML yang termasuk ke dalam **block level element**, seperti tag `<h1>`, `<p>`, `<div>` atau ``, akan ditampilkan di baris tersendiri. Setiap element mengambil baris terpisah dan berjarak beberapa pixel (margin) satu sama lain.

Sedangkan kelompok **inline level element** seperti tag ``, `<i>`, `` atau `` akan diproses secara berdampingan dari kiri ke kanan dalam baris yang sama. Dengan catatan, sepanjang masih muat di dalam parent element-nya. Jika sudah tidak muat lagi, inline level element akan mengambil tempat di baris berikutnya.

Sebagai contoh, bisakah anda menebak' bagaimana web browser akan menampilkan kode HTML berikut?

01.normal_document_flow.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <h1>Normal Document Flow</h1>
9      <div>
10         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
```

```

11 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
12 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
13 sit amet ullamcorper elit vulputate in.</p>
14 <div>
15   
16   
17   
18   
19 </div>
20 <h2>Sub judul disini</h2>
21 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
22 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
23 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
24 sit amet ullamcorper elit vulputate in.</p>
25 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
26 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
27 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
28 sit amet ullamcorper elit vulputate in.</p>
29 </div>
30 </body>
31 </html>

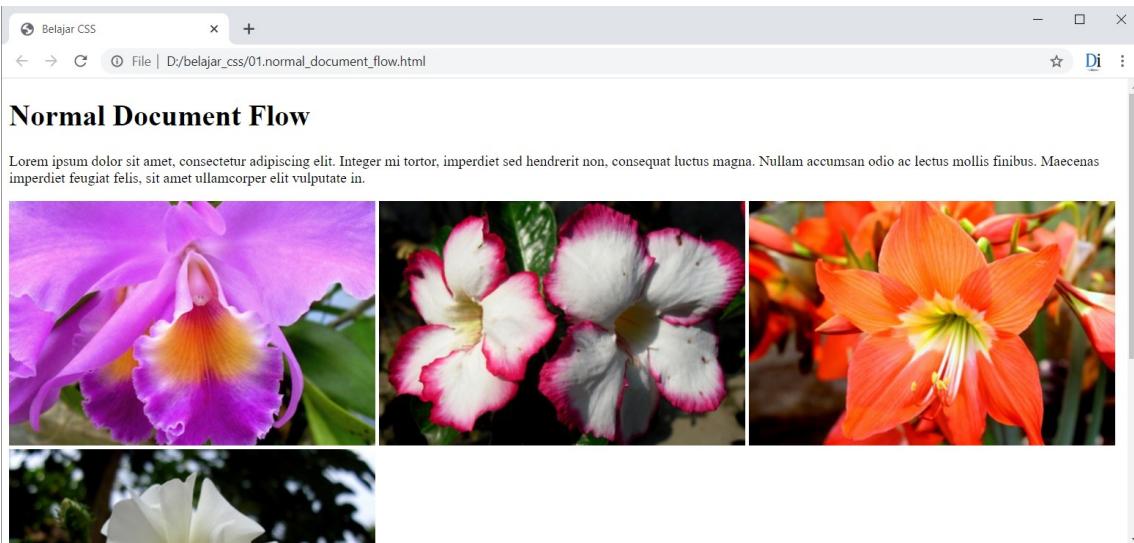
```

Kode HTML di atas saya tulis tanpa CSS sehingga yang berlaku adalah *normal document flow*.

Halaman dimulai dengan sebuah judul yang dibuat dari tag `<h1>`. Tag `<h1>` merupakan block level element sehingga akan mengambil 1 baris dari kiri ke kanan. Sisa konten di bawah tag `<h1>` dibungkus ke dalam tag `<div>`. Di dalamnya terdapat paragraf, serta diikuti dengan 4 gambar.

Di dalam tag `<div>` ini terdapat tag `` yang merupakan *inline level element* sehingga gambar akan tampil berjejer mulai dari sisi kiri ke kanan. Ketika ruang untuk gambar tidak cukup lagi (di sisi kanan), maka akan pindah ke baris dibawahnya.

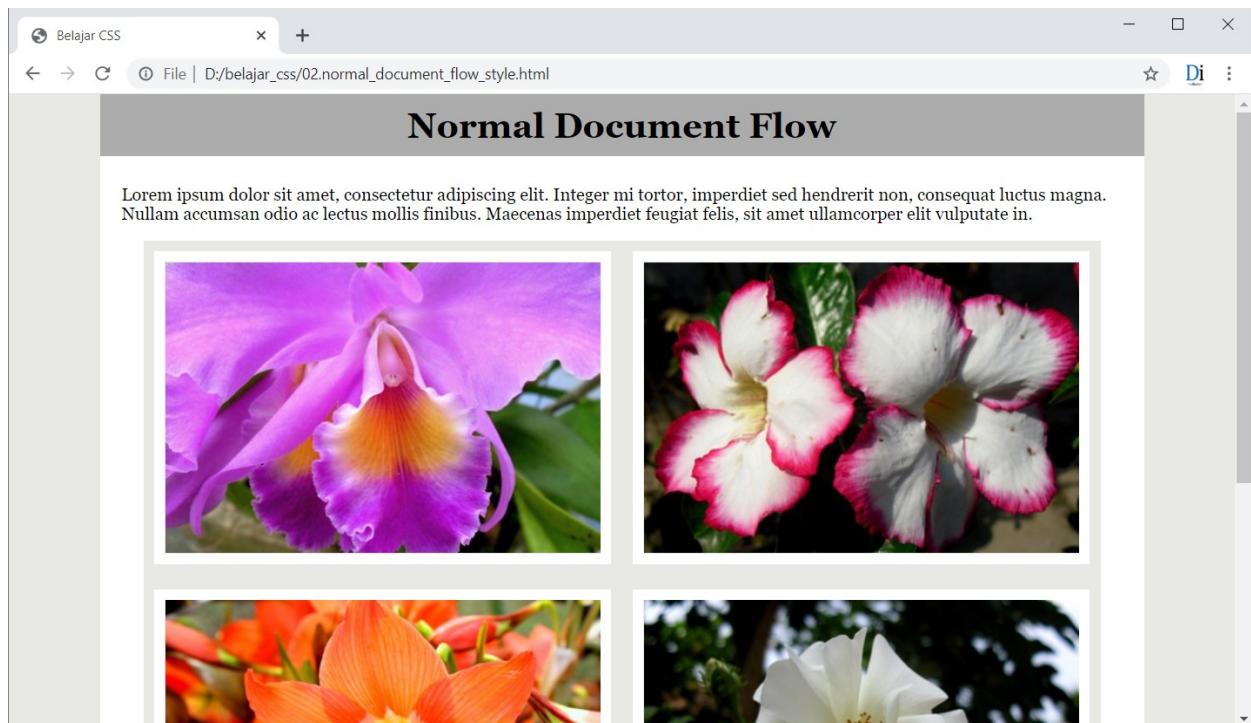
Halaman HTML ditutup dengan sebuah sub judul `<h2>` serta beberapa paragraf. Berikut tampilan dari kode HTML tersebut:



Gambar: Contoh tampilan Normal Document Flow

Jika lebar jendela web browser berubah, 4 gambar yang ada akan menyesuaikan diri. Selama masih cukup ruang, gambar akan terus di tempatkan di baris yang sama, namun jika tidak muat lagi, ia akan pindah ke baris berikutnya.

Dengan memanfaatkan *normal document flow*, kita sudah bisa merancang layout halaman sederhana. Berikut hasil yang bisa saya dapat dengan menambah beberapa property CSS yang sudah kita pelajari sejauh ini:



Gambar: Hasil style kode CSS untuk Normal Document Flow

Berikut kode lengkapnya:

02.normal_document_flow_style.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          html {
8              padding:0;
9              margin:0;
10             background: rgb(231, 232, 226);
11         }
12         body{
13             background: white;
14             width: 960px;
15             margin: 0 auto;
16             font-family: Georgia, "Times new Roman", serif;
17     }

```

```

18  h1,h2 {
19      background: rgb(169, 167, 167);
20      margin: 0;
21      padding: 10px;
22      text-align: center;
23  }
24  .content {
25      padding: 10px 20px;
26  }
27  .gallery{
28      background: rgb(231, 232, 226);
29      width: 880px;
30      margin: 10px auto;
31  }
32  img {
33      border: 10px solid white;
34      margin: 10px;
35  }
36  </style>
37 </head>
38 <body>
39 <h1>Normal Document Flow</h1>
40 <div class="content">
41 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
42 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
43 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
44 sit amet ullamcorper elit vulputate in.</p>
45 <div class="gallery">
46     
47     
48 </div>
49 <h2>Sub judul disini</h2>
50 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
51 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
52 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
53 sit amet ullamcorper elit vulputate in.</p>
54 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
55 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
56 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
57 sit amet ullamcorper elit vulputate in.</p>
58 </div>
59 </body>
60 </html>

```

Khusus untuk penulisan 4 tag `` di baris 46 dan 47, harus ditulis dalam 1 baris panjang (tanpa ada spasi). Karena keterbatasan lebar buku, terpaksa saya pecah menjadi 2 baris, seharusnya ditulis seperti ini:

```

```

Alasannya karena karakter enter dan spasi di antara tag `` akan mengambil tempat beberapa pixel.

Ini merupakan sedikit keanehan cara web browser memproses kode HTML. Penjelasan dan contoh kasusnya bisa ke sini: [Removing the Space Between Images using Only CSS²⁰](#). Kasus ini terjadi hanya jika tag berada di dalam sebuah container yang dalam contoh di atas berada di dalam tag <div>.

Mari kita bahas sekilas kode CSS di atas.

Selector pertama:

```
1 html {
2   padding:0;
3   margin:0;
4   background: rgb(231, 232, 226);
5 }
```

Dipakai untuk me-reset padding dan margin milik tag <html>. Jika ini tidak dilakukan, pada sisi atas tag <h1> akan terlihat garis putih yang berasal dari margin bawaan web browser (*web browser default style*). Kemudian property `background: rgb(231, 232, 226)` akan memberi warna background abu-abu pada halaman web.

Selector kedua:

```
1 body{
2   background: white;
3   width: 960px;
4   margin: 0 auto;
5   font-family: Georgia, "Times new Roman", serif;
6 }
```

Digunakan untuk men-style bagian konten. Saya menetapkan lebar tag <body> sebesar 960 pixel, kemudian mengurnya agar pas berada di tengah-tengah halaman melalui property `margin: 0 auto`. Setelah itu jenis font di set menjadi *Georgia*.

Selector berikutnya:

```
1 h1,h2 {
2   background: rgb(169, 167, 167);
3   margin: 0;
4   padding: 10px;
5   text-align: center;
6 }
```

Berfungsi untuk men-style bagian judul h1 dan h2 agar memiliki latar belakang abu-abu gelap: `rgb(169, 167, 167)`. Kemudian saya menghapus margin bawaan web browser, serta menambah padding sebesar 10 pixel untuk semua sisi element (top, right, bottom, left). Terakhir property `text-align: center` dipakai agar text tampil dengan rata tengah.

²⁰ https://www.kirupa.com/html5/removing_space_between_images_using_css.htm

Selector selanjutnya:

```
1 .content {
2   padding: 10px 20px;
3 }
```

Digunakan agar seluruh konten memiliki padding di sisi atas, bawah, kiri dan kanan. Tujuannya supaya teks tidak terlalu dekat dengan sisi tepi tag <body>.

Dua selector terakhir berfungsi untuk men-style gambar di dalam tag :

```
1 .gallery{
2   background: rgb(231, 232, 226);
3   width: 880px;
4   margin: 10px auto;
5 }
6 img {
7   border: 10px solid white;
8   margin: 10px;
9 }
```

Berdasarkan kode HTML, terlihat bahwa semua gambar berada di dalam tag <div> dengan class="gallery". Dengan demikian tag ini berperan sebagai parent element dari semua tag .

Saya menggunakan property background: `rgb(231, 232, 226)` untuk membuat warna background abu-abu, kemudian men-set ukuran tag ini sebesar 880 pixel. Darimanakah angka 880 pixel ini berasal?

Mari ingat kembali pembahasan tentang box model. Panjang sebuah element adalah gabungan dari width + padding + border + margin. Saya ingin agar 4 gambar tersebut tampil dengan 10px border dan 10px margin.

Dimensi dari setiap gambar adalah 400 x 267 pixel. Jika saya ingin menambah padding dan margin 10 pixel, maka lebar element gambar akan menjadi: $400 + (10 * 2) + (10 * 2)$. Padding dan margin perlu di kali 2 karena mengambil tempat di dua sisi gambar: kiri dan kanan, sehingga lebar total menjadi 440 pixel.

Karena saya ingin 2 gambar berdampingan, maka total lebar yang dibutuhkan adalah $440 * 2 = 880$ pixel. Hitung-hitungan ini cukup penting dipahami. Jika salah, misalnya lebar gallery di set menjadi 800 pixel, maka gambar kedua akan pindah ke bawah karena tidak cukup lebar.

Selector terakhir: `img`, dipakai untuk membuat lebar padding dan margin 10 px.

Contoh proses pembuatan layout di atas cocok sebagai latihan untuk memahami bagaimana perhitungan pixel demi pixel untuk membuat layout halaman web.

10.2. Property position

Agar bisa membawa sebuah element "keluar" dari normal document flow, CSS menyediakan berbagai property. Kita akan mulai dengan property **position**.

Sesuai dengan namanya, property **position** berfungsi untuk mengatur penempatan posisi dari sebuah element. Nilai untuk property ini adalah salah satu dari keyword: **static**, **relative**, **absolute**, **fixed** dan **sticky**.

Static Positioning

Nilai **static** untuk property **position** merupakan nilai bawaan (*default*). Tanpa ditulispun secara tidak langsung sebuah element HTML menggunakan property **position:static**. Penulisan nilai ini juga berarti membuat sebuah element kembali ke *normal document flow*. Berikut contoh penggunaannya:

03.positioning_static.html

```

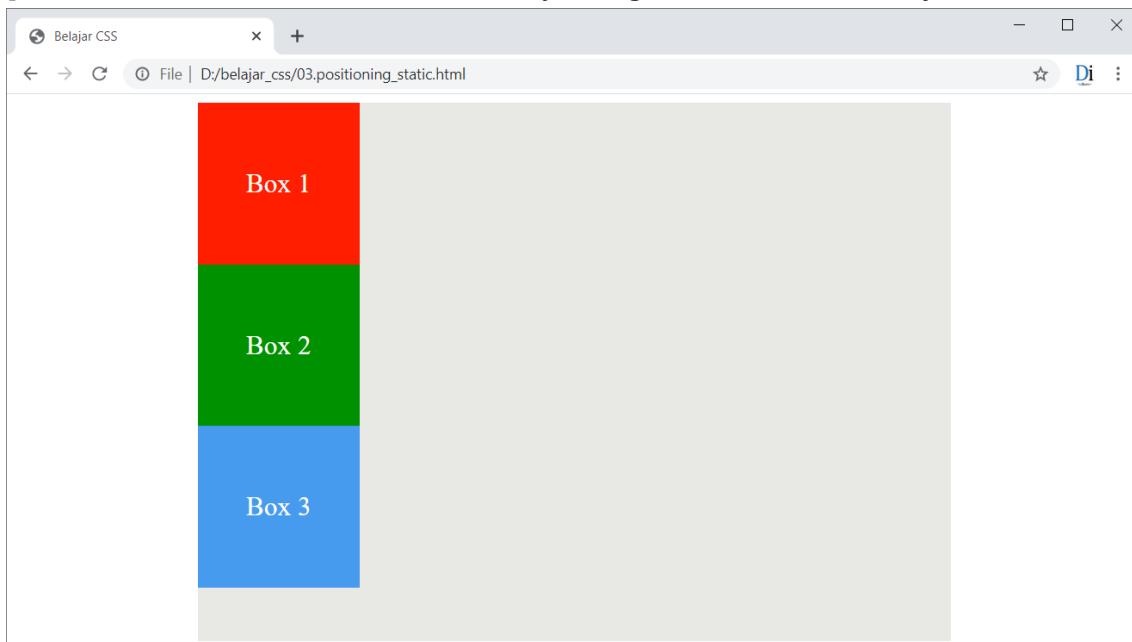
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 700px;
9              height: 500px;
10             background-color: rgb(231, 232, 226);
11             margin: 0 auto;
12         }
13         .satu, .dua, .tiga {
14             width: 150px;
15             height: 150px;
16             font-size: 25px;
17             line-height: 150px;
18             text-align: center;
19             color: white;
20         }
21         .satu {
22             background-color: #FF1A00;
23             position: static;
24         }
25         .dua {
26             background-color: #008C00;
27         }
28         .tiga {
29             background-color: #4096EE;
30         }
31     </style>
32 </head>
33 <body>
```

```
34 <div class="container">
35   <div class="satu"> Box 1 </div>
36   <div class="dua"> Box 2 </div>
37   <div class="tiga"> Box 3 </div>
38 </div>
39 </body>
40 </html>
```

Silahkan pelajari sejenak maksud dari kode CSS di atas. Semua property yang dipakai sudah kita pelajari dalam bab-bab sebelumnya.

Di sini saya membuat 3 buah box dari tag `<div>`. Ketiganya berada dalam parent element `<div class="container">`. Struktur ini akan banyak kita pakai sepanjang pembahasan nanti.

Setelah kode CSS untuk membuat ketiga box tersebut, di baris 23 saya menambah property `position:static` pada box 1. Hasilnya? seperti yang bisa di tebak, tidak tampak perubahan apa-apa. Ini karena `position:static` sama saja dengan `normal document flow`.



Gambar: Efek position static

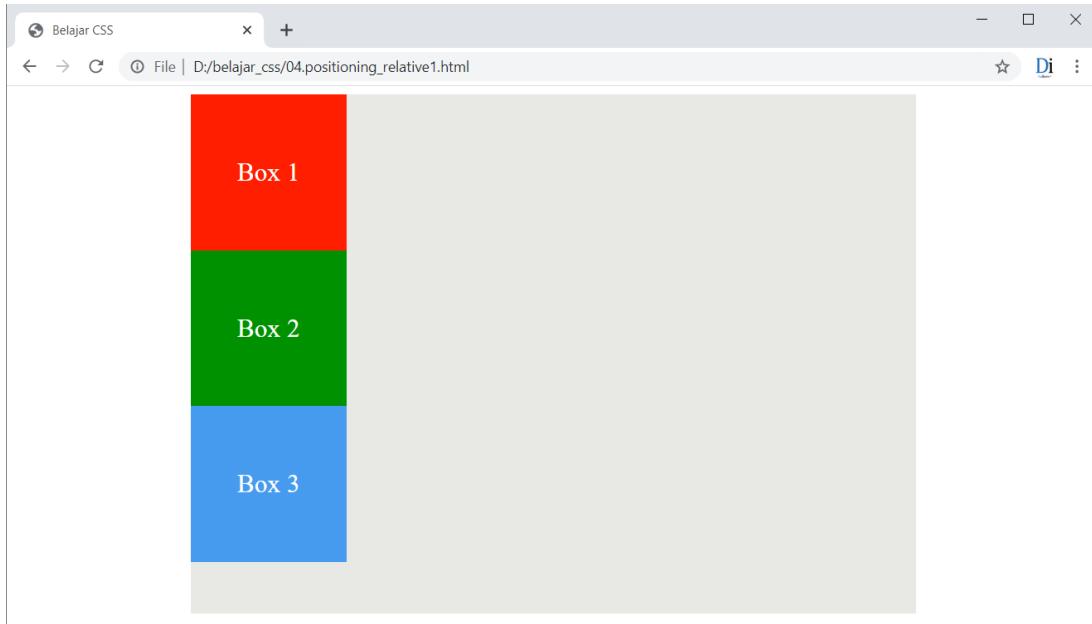
Relative Positioning

Nilai kedua untuk property position adalah **relative**. Kali ini saya akan ubah property box 1 dengan kode CSS berikut:

04.positioning_relative1.html

```
1 .satu {
2   background-color: #FF1A00;
3   position: relative;
4 }
```

CSS Positioning



Gambar: Hasil penggunaan position relative?

Tapi... kenapa juga tidak ada perubahan?

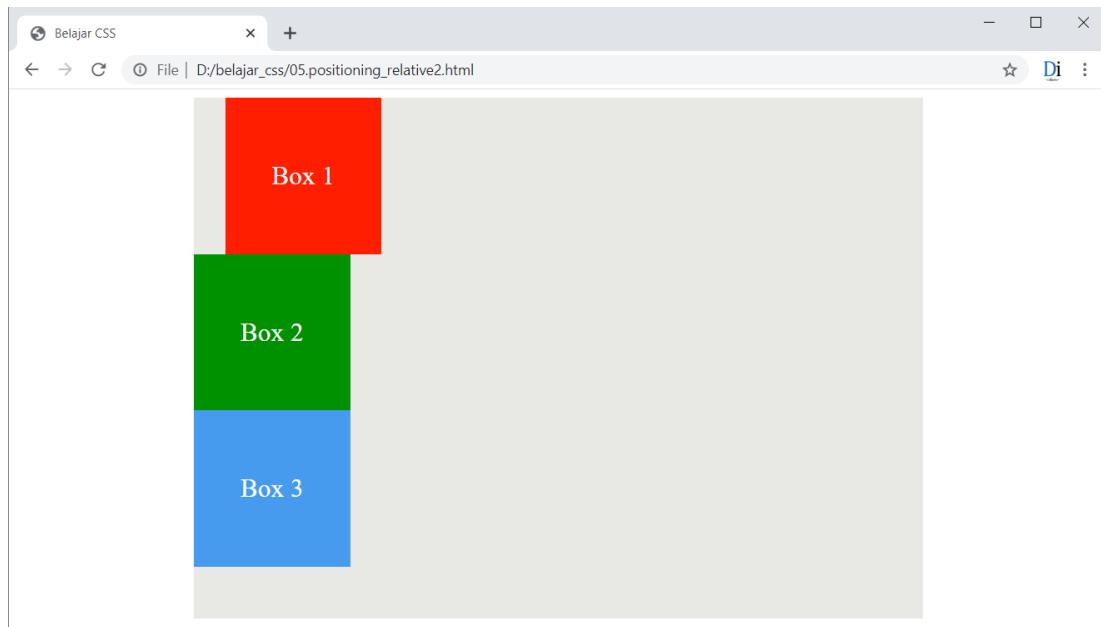
Untuk melihat efek dari property `position: relative`, kita harus minta bantuan 4 property lain, yakni: **top**, **bottom**, **left** dan **right**. Keempat property inilah yang bisa menentukan posisi sebuah element dalam `position: relative`.

Sebagai contoh, untuk menggeser box 1 sebesar 30 pixel ke arah kanan, bisa menggunakan property `left: 30px` seperti contoh berikut:

05.positioning_relative2.html

```
1 .satu {  
2   background-color: #FF1A00;  
3   position: relative;  
4   left: 30px;  
5 }
```

Property `left: 30px` berpasangan dengan `position: relative`. Hasilnya, akan menggeser sisi kiri box sebesar 30 pixel ke arah kanan relatif terhadap posisi element tersebut di dalam *normal document flow*. Jika anda sedikit bingung dengan penjelasan ini, mungkin akan paham ketika kita masuk ke *absolute positioning* sesaat lagi.



Gambar: Box 1 bergeser 30 pixel ke kanan

Bagaimana jika kita ingin box 1 berada 50 pixel ke arah kiri dari posisi normalnya? Terdapat 2 cara, `left:-50px` atau `right:50px`.

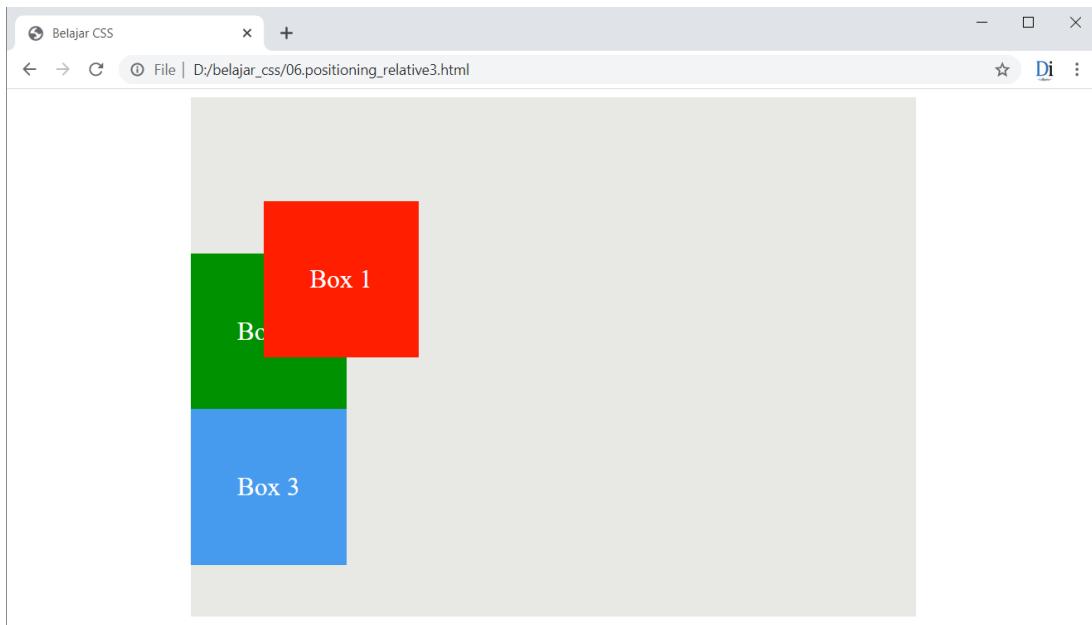
Property `left:-50px` berarti kita ingin menggeser sisi kiri element sebanyak 50 pixel ke arah kiri (perhatikan penambahan tanda minus). Begitu pula dengan property `right:50px` yang berarti akan menggeser sisi kanan element sebesar 50 pixel ke kiri.

Sebagai contoh lain, bagaimana jika saya ingin Box 1 berada pada posisi 70 pixel ke kanan dan 100 pixel ke bawah? Solusinya bisa menggunakan property berikut:

06.positioning_relative3.html

```
1 .satu {  
2   background-color: #FF1A00;  
3   position: relative;  
4   left: 70px;  
5   top: 100px;  
6 }
```

Property `left:70px` akan mendorong box 1 sejauh 70 pixel ke arah kanan relatif terhadap sisi kiri element. Dan property `top:100px` akan mendorong box 1 sejauh 100 pixel ke arah bawah relatif terhadap sisi atas element tersebut.



Gambar: Box 1 bergeser 70 pixel ke kanan, dan 100 pixel ke bawah

Sebagai alternatif, hasil yang sama juga bisa didapat dengan menggunakan property berikut:

```
1 .satu {  
2   background-color: #FF1A00;  
3   position: relative;  
4   right: -70px;  
5   top: 100px;  
6 }
```

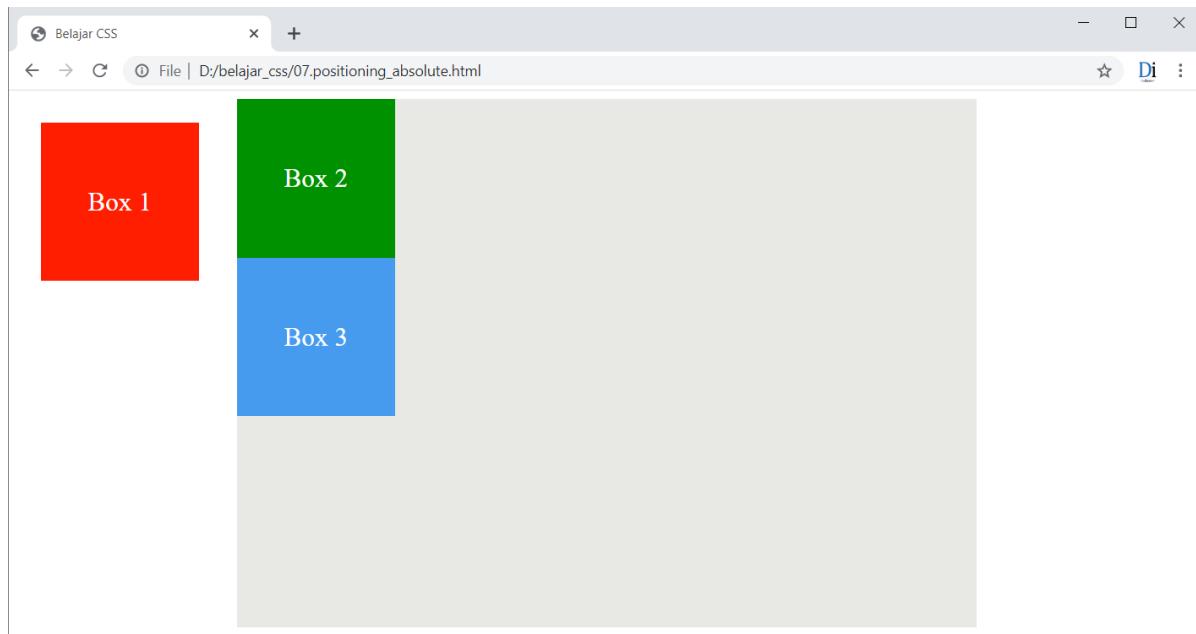
Silahkan anda mencoba kombinasi kode CSS lainnya agar bisa memahami penggunaan nilai positif dan negatif untuk position: relative. Kuncinya adalah, posisi tersebut bergeser relatif terhadap posisi element dalam *normal document flow*.

Absolute Positioning

Pengaturan posisi ketiga adalah **absolute positioning**. Ini dihasilkan dari penggunaan property `position: absolute`. Langsung saja kita lihat contohnya:

07.positioning_absolute.html

```
1 .satu {  
2   background-color: #FF1A00;  
3   position: absolute;  
4   left: 30px;  
5   top: 30px;  
6 }
```



Gambar: Efek penggunaan position: absolute

Ada 2 efek yang bisa kita terlihat. **Pertama**, box 1 sekarang berada 30 pixel dari sisi atas dan 30 pixel dari sisi kiri jendela web browser, ini bukan lagi dihitung berdasarkan posisi asalnya, tapi dari jendela web browser. **Kedua**, perhatikan posisi box 2 dan box 3, keduanya "naik" ke atas untuk mengisi posisi yang sebelumnya di tempati oleh box 1.

Apabila kita menggunakan `position: absolute`, sebuah element akan dipaksa keluar dari *normal document flow*, atau bisa juga dikatakan bahwa element tersebut diangkat dari posisi normalnya. Itulah yang jadi penyebab box 2 dan box 3 naik ke atas, seolah-olah box 1 tidak pernah ada.

Acuan posisi **top**, **bottom**, **right** dan **left** sekarang mengacu ke jendela web browser. Posisi box 1 yang ditulis sebagai `left: 30px` dan `top: 30px`, menempatkan box 1 di 30 pixel dari sisi atas web browser, dan 30 pixel dari sisi kiri web browser.

Bagaimana jika kita ingin box 1 tampil di sudut kanan bawah web browser? Berikut kode yang bisa dipakai:

```

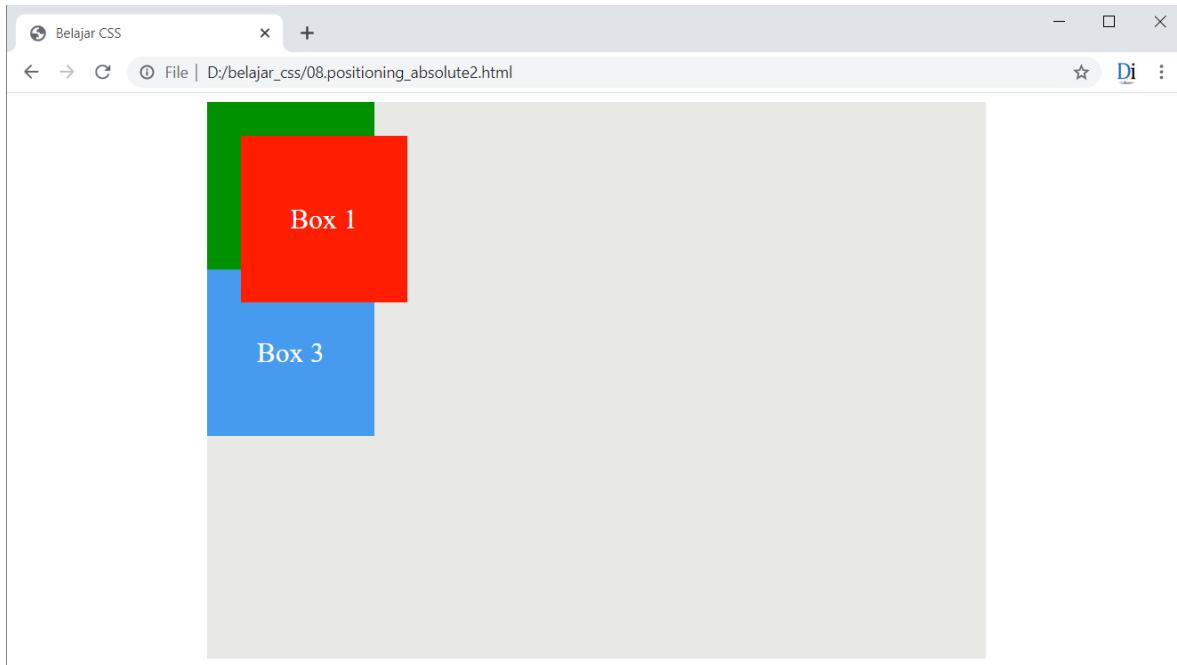
1 .satu {
2   background-color: #FF1A00;
3   position: absolute;
4   right: 0px;
5   bottom: 0px;
6 }
```

Dengan property `right: 0px` dan `bottom: 0px` maka sisi kanan box 1 akan berjarak 0 pixel dari bingkai kanan web browser, serta sisi bawah box 1 akan berjarak 0 pixel dari bingkai bawah web browser. Silahkan coba berbagai kombinasi posisi lain agar bisa memahami konsep ini.

Baik, kita akan masuk ke pemahaman yang mungkin agak sedikit rumit.

Sebenarnya, `position: absolute` bukan berpatokan ke jendela web browser, tetapi kepada element `<html>`. Sebelum itu perhatikan contoh kode program berikut:

08.positioning_absolute2.html



Gambar: Hasil penggunaan `position: absolute`?

Kode CSS yang digunakan untuk box 1 sama persis dengan contoh kita sebelumnya, yakni:

```
1 .satu {  
2   background-color: #FF1A00;  
3   position: absolute;  
4   left: 30px;  
5   top: 30px;  
6 }
```

Tetapi mengapa posisi box 1 berubah? Ini karena saya menambah 1 property khusus kepada selector `.container`, yakni:

```
1 .container {  
2   ...  
3   position: relative;  
4 }
```

Kenapa bisa begini?

Patokan dari `position: absolute` sebenarnya ke parent element terdekat yang memiliki property `position: relative`. Jika tidak ditemukan, ia akan mencari ke parent element selanjutnya, dan seterusnya. Jika tidak ada juga, barulah ke tag `<html>` sebagai patokan posisi, yang tidak lain merujuk ke bingkai jendela web browser.

Kombinasi property `position: absolute` yang terdapat di box 1 dan property `position: relative` dari `.container`, menjadikan tag `<div class="container">` sebagai patokan posisi dari box 1. Property `left:30px` dan `top:30px` dari box 1 akan dihitung dari sisi kiri dan sisi atas tag `<div class="container">`.

Pengertian ini memang agak susah dipahami, oleh karena itu kita masuk ke contoh kedua:

09.positioning_absolute3.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .outer-container {
8              width: 700px;
9              height: 500px;
10             background-color: rgb(90, 232, 200);
11             margin: 0 auto;
12             position: relative;
13         }
14         .inner-container {
15             width: 500px;
16             height: 400px;
17             background-color: rgb(231, 232, 226);
18             margin: 0 auto;
19         }
20         .satu, .dua, .tiga {
21             width: 150px;
22             height: 150px;
23             font-size: 25px;
24             line-height: 150px;
25             text-align: center;
26             color: white;
27         }
28         .satu {
29             background-color: #FF1A00;
30             position: absolute;
31             left: 30px;
32             top: 30px;
33         }
34         .dua{
35             background-color: #008C00;
36         }
37         .tiga{
38             background-color: #4096EE;
39         }
40     </style>
41 </head>
42 <body>
43     <div class="outer-container">
44         <div class="inner-container">
```

```
45      <div class="satu"> Box 1 </div>
46      <div class="dua"> Box 2 </div>
47      <div class="tiga"> Box 3 </div>
48  </div>
49 </div>
50 </body>
51 </html>
```

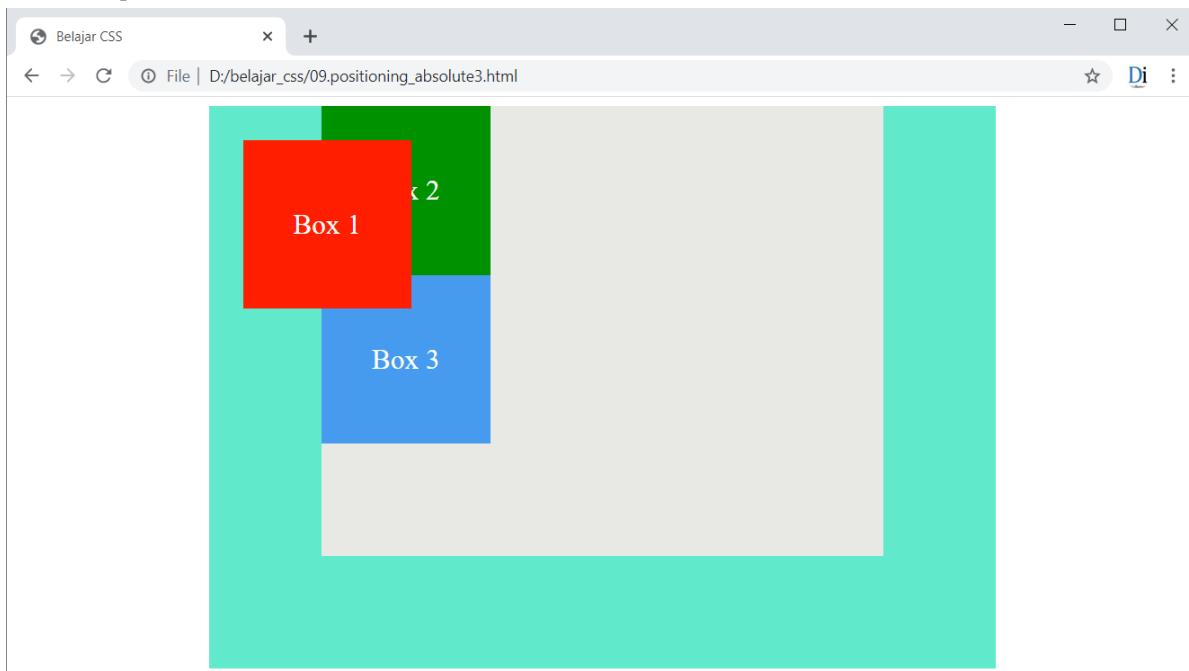
Kali ini saya memodifikasi sedikit struktur HTML. Perhatikan bahwa box 1, box 2, dan box 3 sekarang berada di dalam `<div class="inner-container">`. Tag `<div class="inner-container">` ini juga berada di dalam `<div class="outer-container">`.

Di dalam struktur DOM, `<div class="inner-container">` merupakan parent element pertama dari box 1. Sedangkan tag `<div class="outer-container">` menjadi parent element kedua, atau bisa juga disebut sebagai *grand parent element* dari box 1.

Mari kita ulangi konsep absolute positioning. Karena box 1 memiliki property `position: absolute`, maka ia harus mencari patokan. Langkah pertama, apakah pada class `.inner-container` memiliki property `position: relative`? Tidak.

Lanjut ke atas, apakah class `.outer-container` memiliki property `position: relative`? Yup, ada. Oleh karena itu box 1 akan mengambil posisi 30 pixel dari atas dan 30 pixel dari kanan class `.outer-container`.

Berikut tampilan akhir dari kode di atas:



Gambar: Penggunaan `position: absolute` yang berpatokan kepada parent element dengan `position: relative`

Bagaimana jika class `.outer-container` juga tidak terdapat property `position: relative`? Maka pencarian akan naik ke tag `<body>`, jika tidak ada juga, naik lagi ke tag `<html>`, jika tetap tidak ditemukan, web browser akan memakai tag `<html>` ini sebagai patokan terakhir, sehingga

seolah-olah absolute positioning diukur dari bingkai jendela web browser.

Sama seperti sebelumnya, saya sangat sarankan untuk mencoba mengubah nilai yang ada dan perhatikan efek yang dihasilkan.

Fixed Positioning

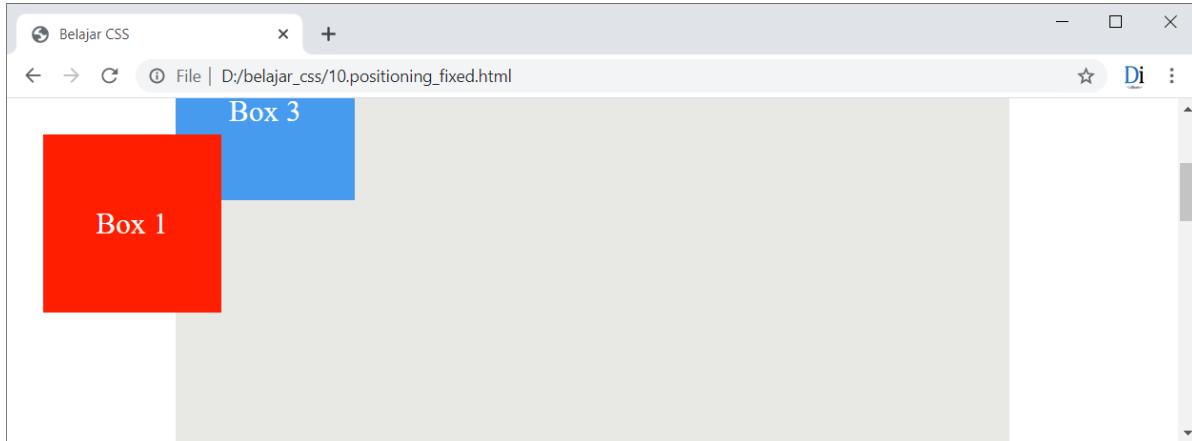
Sesuai dengan namanya, **fixed positioning** bersifat tetap dengan berpatokan ke *viewport*, yakni tampilan jendela web browser yang sedang terlihat. Berikut contoh praktik dari fixed positioning:

10.positioning_fixed.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 700px;
9              height: 1500px;
10             background-color: rgb(231, 232, 226);
11             margin: 0 auto;
12         }
13         .satu, .dua, .tiga {
14             width: 150px;
15             height: 150px;
16             font-size: 25px;
17             line-height: 150px;
18             text-align: center;
19             color: white;
20         }
21         .satu {
22             background-color: #FF1A00;
23             position: fixed;
24             left: 30px;
25             top: 30px;
26         }
27         .dua {
28             background-color: #008C00;
29         }
30         .tiga {
31             background-color: #4096EE;
32         }
33     </style>
34 </head>
35 <body>
36     <div class="container">
37         <div class="satu"> Box 1 </div>
38         <div class="dua"> Box 2 </div>
39         <div class="tiga"> Box 3 </div>
40     </div>
```

```
41 </body>
42 </html>
```



Gambar: Penggunaan position: fixed

Kali ini saya memperpanjang tinggi container menjadi 1500px agar kita bisa lihat efek fixed positioning. Silahkan scroll halaman ke atas dan ke bawah, maka box 1 selalu tetap di posisi 30 pixel dari atas dan 30 pixel dari kiri jendela web browser, atau lebih tepatnya dari viewport.

Anda juga bisa perkecil jendela web browser, dan box 1 akan tetap berada di posisinya, apapun yang terjadi.

Beberapa sumber juga sering menyebut posisi seperti ini sebagai *floating*, karena element HTML seolah-olah melayang (*float*) di atas halaman web. Penyebutan ini bisa menimbulkan kebingungan karena di dalam CSS juga terdapat property *float* yang fungsinya berbeda.

Selain itu sama seperti absolute positioning, box 1 juga diangkat dari *normal document flow*, sehingga box 2 dan box 3 akan naik ke atas untuk mengisi ruang kosong yang ditinggalkan box 1.

Efek element seperti ini juga sering dipakai untuk membuat menu yang selalu menempel di bagian atas halaman web seperti contoh berikut:

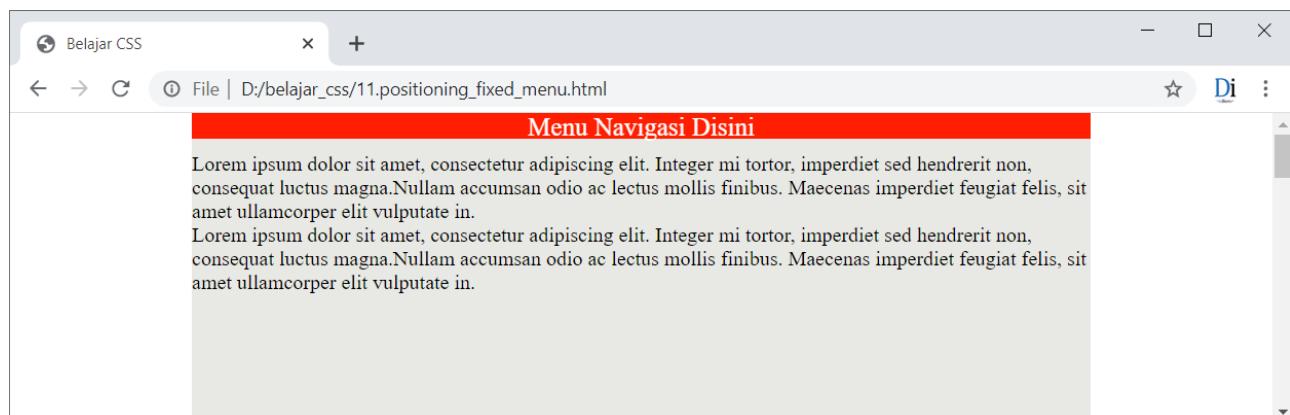
11.positioning_fixed_menu.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          *{
8              padding:0px;
9              margin: 0px;
10         }
11         .container {
12             width: 700px;
13             height: 1500px;
```

```

14     background-color: rgb(231, 232, 226);
15     margin: 0 auto;
16 }
17 .menu {
18     width: 700px;
19     height: 20px;
20     font-size: 20px;
21     text-align: center;
22     color: white;
23     background-color: #FF1A00;
24     position: fixed;
25 }
26 div + p {
27     padding-top: 30px;
28 }
29 </style>
30 </head>
31 <body>
32 <div class="container">
33     <div class="menu"> Menu Navigasi Disini </div>
34     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
35     tortor, imperdiet sed hendrerit non, consequat luctus magna.Nullam
36     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
37     felis, sit amet ullamcorper elit vulputate in.</p>
38     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
39     tortor, imperdiet sed hendrerit non, consequat luctus magna.Nullam
40     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
41     felis, sit amet ullamcorper elit vulputate in.</p>
42 </div>
43 </body>
44 </html>

```



Gambar: Penggunaan position: fixed untuk membuat menu floating

Silakan scroll halaman web ke bawah dan ke atas, menu tersebut akan "ikut" dan selalu menempel di bagian atas jendela web browser

Sticky Positioning

Efek **sticky positioning** mirip seperti fixed positioning, dimana element HTML akan menempel ke bagian atas jendela web browser, tapi hanya jika viewport sudah melewati

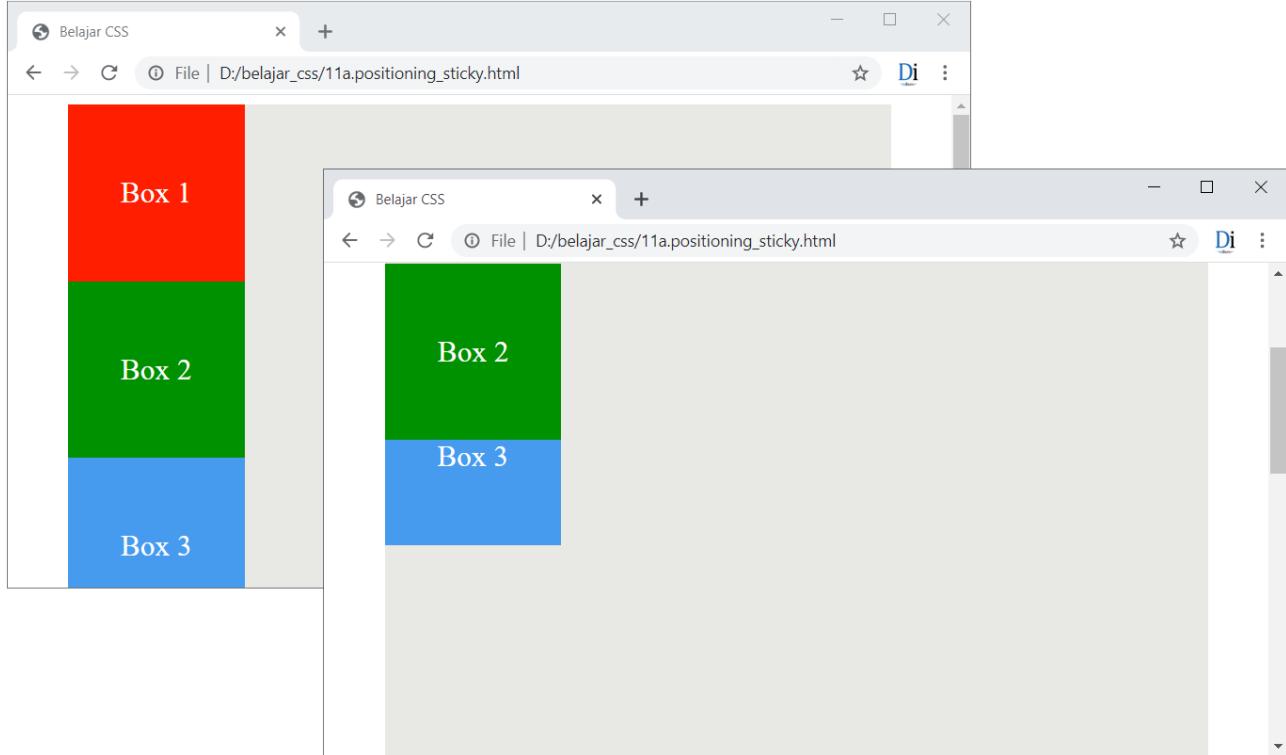
element tersebut.

Saat di jalankan pertama kali, element dengan property `position:sticky` tampil seperti `position:static`, termasuk tetap berada di dalam *normal document flow*. Tetapi begitu halaman di scroll ke bawah dan melewati element, maka ia akan menempel ke sisi atas web browser.

Berikut contoh kode program dari `position:sticky`:

11a.positioning_sticky.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 700px;
9              height: 1500px;
10             background-color: rgb(231, 232, 226);
11             margin: 0 auto;
12         }
13         .satu, .dua, .tiga {
14             width: 150px;
15             height: 150px;
16             font-size: 25px;
17             line-height: 150px;
18             text-align: center;
19             color: white;
20         }
21         .satu {
22             background-color: #FF1A00;
23         }
24         .dua{
25             background-color: #008C00;
26             position: sticky;
27             top: 0px;
28         }
29         .tiga{
30             background-color: #4096EE;
31         }
32     </style>
33 </head>
34 <body>
35     <div class="container">
36         <div class="satu"> Box 1 </div>
37         <div class="dua"> Box 2 </div>
38         <div class="tiga"> Box 3 </div>
39     </div>
40 </body>
41 </html>
```



Gambar: Penggunaan position: sticky

Dalam contoh ini, property position:sticky saya tempatkan ke dalam Box 2. Saat halaman dibuka, semuanya tampak normal. Tapi begitu di scroll melewati Box 2, Box tersebut akan menempel di sisi atas web browser.

Efek ini mungkin sudah sering anda lihat terutama untuk menampilkan iklan di beberapa website.

10.3. Property display

Property **display** dipakai untuk mengatur tampilan dari sebuah element. Nilai property ini cukup banyak namun sebagian besar sangat jarang digunakan. Berikut nilai yang bisa diisi ke dalam property display:

none, inline, block, list-item, inline-block, inline-table, table, table-cell, table-column, table-column-group, table-footer-group, table-header-group, table-row, table-row-group, flex, inline-flex, grid, inline-grid, run-in, ruby, ruby-base, ruby-text, ruby-base-container, ruby-text-container, contents.

Seperti yang terlihat, nilai untuk property **display** cukup banyak, tapi yang sering digunakan hanya beberapa saja.

Nilai yang diawali kata **table-** dipakai untuk membuat element HTML seolah-olah menjadi tabel sel. Sedangkan nilai yang diawali dengan kata **ruby-** dipakai untuk membuat teks yang berisi karakter asia timur seperti jepang, cina, atau korea.

Kali ini kita hanya fokus ke 4 nilai property display, yakni: `none`, `inline`, `block`, dan `inline-block`.

Display none

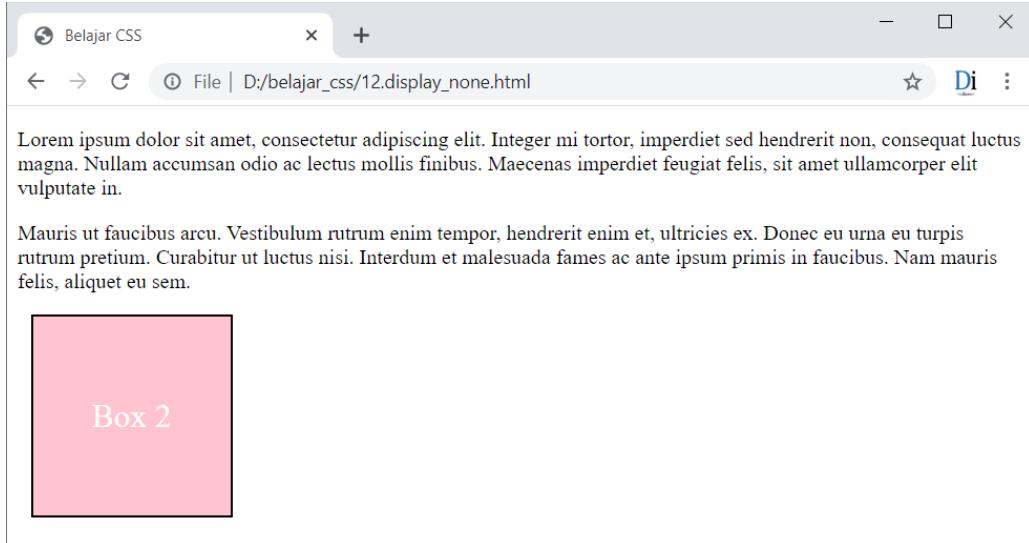
Property `display:none` berfungsi untuk menyembunyikan sebuah element seolah-olah element tersebut tidak pernah ada. Berikut contoh penggunaannya:

12.display_none.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua {
8              width:150px;
9              height: 150px;
10             font-size: 25px;
11             line-height: 150px;
12             text-align: center;
13             color: white;
14             background-color: pink;
15             border: 2px solid black;
16             margin: 10px;
17         }
18         h1 {display: none;}
19         .satu {display: none;}
20     </style>
21 </head>
22 <body>
23     <h1>Dimanakah Saya Sekarang?</h1>
24     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
25         tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
26         accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat
27         felis, sit amet ullamcorper elit vulputate in.</p>
28     <p>Mauris ut faucibus arcu. Vestibulum rutrum enim tempor, hendrerit
29         enim et, ultricies ex. Donec eu urna eu turpis rutrum pretium.
30         Curabitur ut luctus nisi. Interdum et malesuada fames ac ante ipsum
31         primis in faucibus. Nam mauris felis, aliquet eu sem. </p>
32     <div class="satu"> Box 1 </div>
33     <div class="dua"> Box 2 </div>
34 </body>
35 </html>
```

Dalam bagian `<body>`, terdapat satu tag `<h1>`, dua tag `<p>` dan dua tag `<div>`. Akan tetapi di baris 18 dan 19 saya men-set style untuk `h1` dan `.satu` dengan property `display:none`. Berikut hasilnya:



Gambar: Efek dari penggunaan property display: none

Seperti yang terlihat, tag `<h1>` dan tag `<div class="satu">` hilang dan tidak tampil di layar. Inilah efek dari property `display:none` untuk sebuah element HTML.

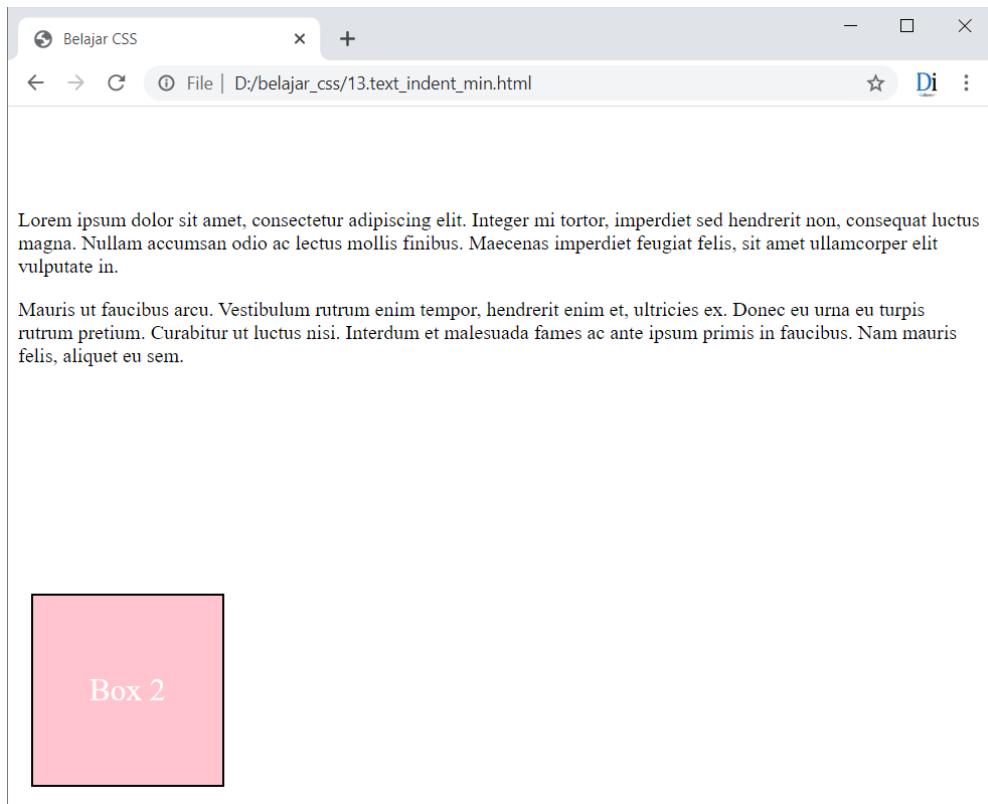
Sama seperti penggunaan `position:absolute` dan `position:fixed`, penggunaan property `display:none` akan mengangkat element tersebut dari *normal document flow*. Efeknya, paragraf di bawah tag `<h1>` akan naik mengisi tempatnya serta Box 2 juga akan naik ke atas mengisi ruang yang seharusnya ditempati oleh box 1.

Sifat lain dari property `display: none` adalah, walaupun element HTML tersebut sudah tidak terlihat, tapi dia tetap ada. Jika kita menggunakan JavaScript, kedua element ini bisa diproses dan tampil kembali. Jadi, element tersebut bukanlah hilang. Ia masih ada, tapi tak terlihat...

Jika anda ingat, saya pernah memakai trik `text-indent: -2000px` untuk menyembunyikan element (ketika membahas property `text-indent` pada bab Typography). Apa bedanya dengan property `display:none`? Mari kita bandingkan:

13.text_indent_min.html

```
1 <style>
2 ...
3   h1 {text-indent: -2000px;}
4   .satu {margin-left: -2000px;}
5 </style>
```



Gambar: Efek penggunaan property text-indent: -2000px

Dalam kode CSS, saya mengganti `display: none` untuk selector `h1` dan `.satu` dengan property `text-indent: -2000px` dan `margin-left: -2000px`.

Karena tag `<div>` tidak hanya berisi text, maka property `text-indent: -2000px` tidak bisa digunakan. Saya menggantinya dengan `margin-left: -2000px` yang berfungsi hampir mirip, yakni mengirim tag `<h1>` dan `<div class="satu">` ke ruangan sebelah.

Property `text-indent: -2000px` dan `margin-left: -2000px` memang menyembunyikan element HTML, namun ruang yang ditinggalkan oleh kedua element ini tetap ada. Dengan kata lain, kedua element masih berada di dalam normal document flow.

Karena alasan inilah paragraf di bawah tag `<h1>` tidak naik ke atas, serta box 2 juga tidak mengisi ruang kosong yang telah ditinggalkan oleh box 1, karena ruang-ruang ini masih kepunyaan element asalnya.

Trik lain yang bisa dipakai untuk menyembunyikan suatu element adalah dengan property **opacity**. Opacity menyatakan seberapa transparan sebuah element dengan nilai antara 0 – 1. Angka 1 berarti tidak ada efek transparan, sedangkan nilai 0 akan membuat element 'hilang' karena transparan 100%.

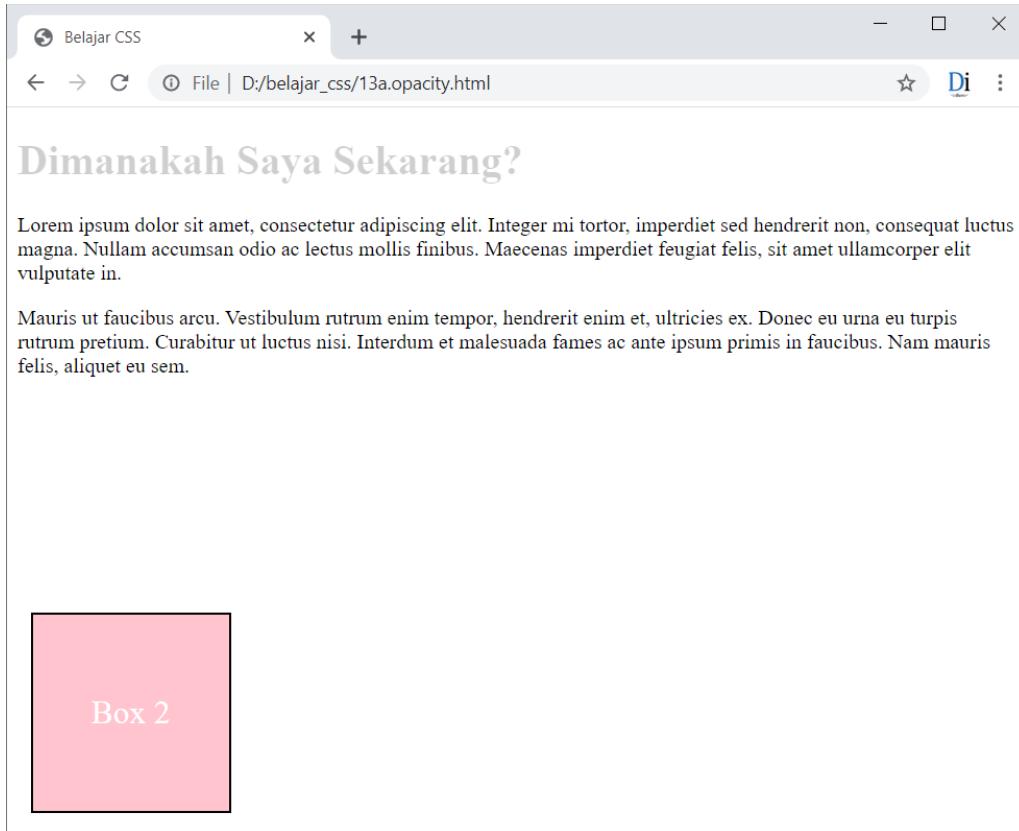
13a.opacity.html

```
1 <style>
2 ...
```

```

3   h1 {opacity: 0.2;}
4   .satu {opacity: 0;}
5 </style>

```



Gambar: Efek penggunaan property opacity

Dalam contoh ini saya men-set tag `<h1>` dengan `opacity:0.2` serta tag `<div class="satu">` dengan `opacity:0`. Hasilnya, tag `<h1>` masih terlihat meskipun sedikit tersamarkan, sedangkan Box 1 sudah sepenuhnya tidak terlihat.

Property `opacity` adalah alternatif yang lebih "elegant" dibandingkan trik `text-indent` dan `margin-left` agar element tetap berada di *normal document flow*. Namun jika kita ingin element tersebut keluar dari *normal document flow*, bisa memakai property `display:none`.

Display inline

Nilai kedua dari property `display` yang akan kita bahas adalah **inline**. Property `display:inline` berguna untuk mengubah element HTML dari **block level element** menjadi **inline level element**.

Sepanjang pembahasan dalam bab sebelumnya, beberapa kali saya menyenggung tentang perbedaan antara **block level element** dengan **inline level element** (dan juga ada di eBook [HTML Uncover](#)).

Semua element yang termasuk ke dalam block level element akan mengambil tempat di baris terpisah. Contoh dari block level element adalah tag `<h1>`, `<p>`, `` dan `<div>`. Berikut tampilan yang dimaksud:

14.display_inline.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6 </head>
7 <body>
8   <h1>Secara default, tag h1 adalah block level element</h1>
9   <p>Tag p juga block level element</p>
10  <div>Tag div masih tetap block level element</div>
11  <ul>
12    <li>List Item juga block level element</li>
13    <li>List Item juga block level element</li>
14    <li>List Item juga block level element</li>
15  </ul>
16 </body>
17 </html>
```



Gambar: Tampilan dari block level element

Dalam kode HTML di atas terdapat berbagai tag yang termasuk ke dalam *block level element*. Seperti yang terlihat, setiap tag menempati baris baru dan saling terpisah satu sama lain.

Bagaimana jika di set dengan property `display:inline`? Berikut hasilnya:

15.display_inline2.html

```
1 <style>
2   h1, p, div, li {display: inline;}
3 </style>
```



Gambar: Tampilan dari display: inline

Sekarang semua element bertumpuk dari kiri ke kanan, sebagaimana layaknya sebuah *inline level element*.

Display block

Jika property `display:inline` dipakai untuk mengkonversi *block level element* menjadi *inline level element*, maka property `display:block` berfungsi sebaliknya, yakni mengubah *inline level element* menjadi *block level element*.

Inline level element adalah jenis tag atau element HTML yang bersambung satu sama lain (tidak mengambil baris baru). Selama masih ada ruang di sisi kanan, inline level element akan terus mengisi tempat tersebut, contohnya seperti tag `<i>`, `` dan `` :

16.display_block.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <i>Secara default, tag i adalah inline level element</i>
9      <strong>Tag strong juga inline level element</strong>
10     <span>Tag span masih tetap block level element</span>
11 </body>
12 </html>

```



Gambar: Tampilan dari inline level element

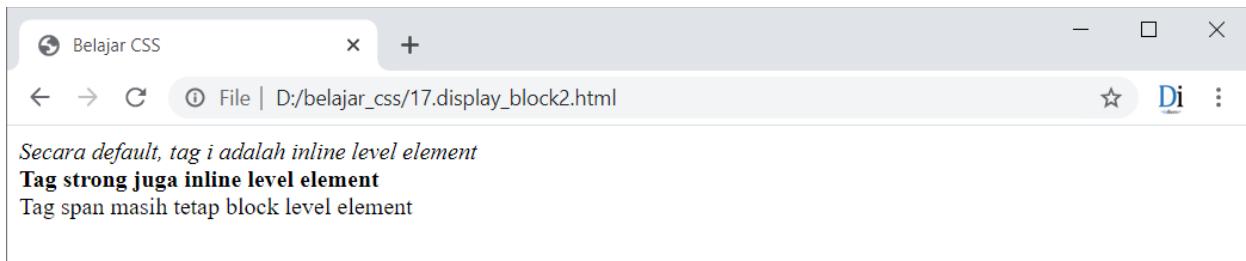
Secara default, inline level element ditampilkan dalam baris yang sama dari kiri ke kanan. Bagaimana jika ditambahkan property `display:block`? Berikut hasilnya:

17.display_block2.html

```

1  <style>
2      i, strong, span {display: block;}
3  </style>

```



Gambar: Tampilan dari display: block

Dapat terlihat sekarang setiap tag menempati baris yang baru, yang menunjukkan ciri khas dari sebuah block level element.

Display inline-block

Property `display:inline-block` cukup menarik. Property ini akan membuat sebuah element berperilaku sebagai mana inline level element, yakni menyambung baris yang sudah ada, tapi masih mempertahankan ciri-ciri block level element, yakni bisa diberikan property **height** dan **padding**.

Sebagai contoh kasus, perhatikan kode HTML berikut:

18.display_inline_block.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width:150px;
9              height: 150px;
10             font-size: 25px;
11             line-height: 150px;
12             text-align: center;
13             color: white;
14             background-color: pink;
15             border: 2px solid black;
16             margin: 10px;
17         }
18     </style>
19 </head>
20 <body>
21     <div> Box 1 </div>
22     <div> Box 2 </div>
23     <div> Box 3 </div>
24 </body>
25 </html>
```

CSS Positioning



Gambar: Tampilan normal tag <div>

Kode di atas mirip dengan contoh ketika membahas tentang positioning, dimana terdapat 3 box yang dibuat dari tag <div>. Dalam HTML, tag <div> termasuk block level element, yang akan tampil terpisah di baris baru.

Jika kita mengkonversinya menjadi inline level element dengan property `display:inline`, hasilnya menjadi sebagai berikut:

19.display_inline_block2.html

```
1 div {  
2   display: inline;  
3 }
```



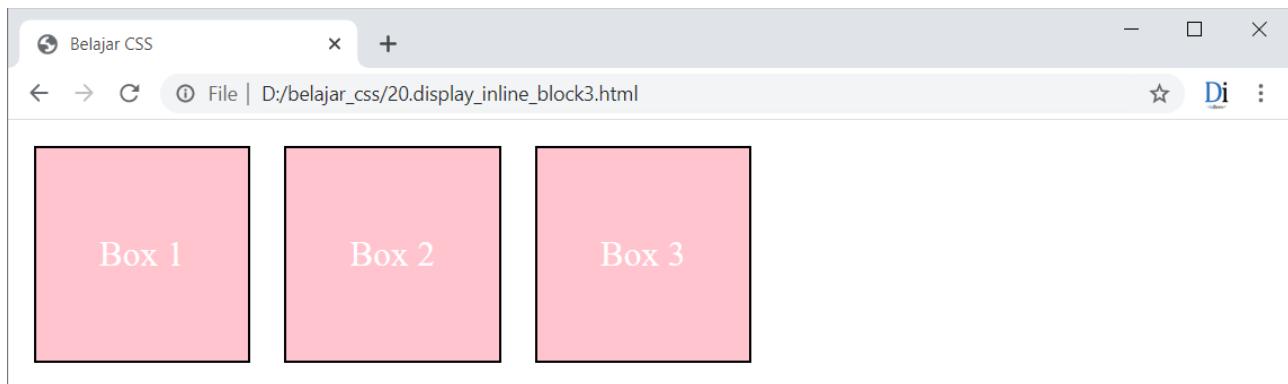
Gambar: Tag <div> dengan penambahan property display: inline

Untuk inline level element, tidak semua property box model bisa kita terapkan. Ketiga box menjadi kecil karena **property height dan padding tidak berefek**, dan ini berlaku untuk semua inline level element.

Jadi bagaimana cara agar tetap bisa mengatur height dan padding pada inline level element? Inilah salah satu fungsi dari `display:inline-block`:

20.display_inline_block3.html

```
1 div {
2   display: inline-block;
3 }
```



Gambar: Dengan penambahan property `display: inline-block`

Hasilnya ketiga box tampil dalam satu baris dari kiri ke kanan sebagaimana layaknya inline level element, namun height dan padding bisa di set seperti block level element.

Nilai lain untuk property display yang cukup penting adalah **flex** dan **grid**. Karena bahasannya cukup banyak, akan kita pecah menjadi bab tersendiri.

10.4. Property z-index

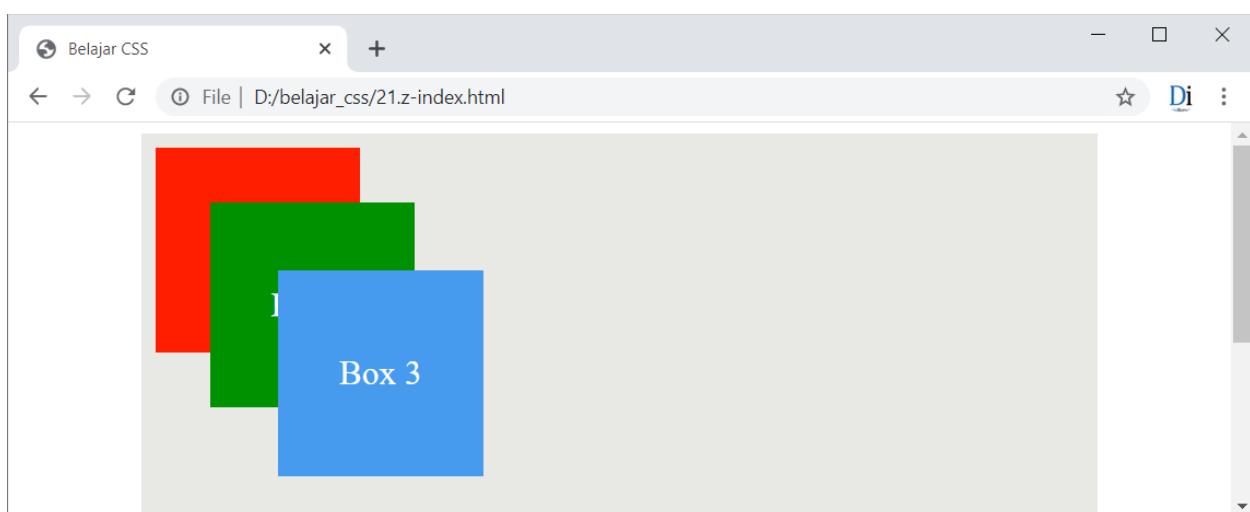
Property **z-index** berfungsi untuk mengatur urutan element saat element tersebut saling bertumpuk. Sebagai contoh, perhatikan kode HTML dan CSS berikut ini:

21.z-index.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .container {
8       width: 700px;
9       height: 500px;
10      background-color: rgb(231, 232, 226);
11      margin: 0 auto;
12      position: relative;
13    }
14    .satu, .dua, .tiga {
```

CSS Positioning

```
15     width:150px;
16     height: 150px;
17     font-size: 25px;
18     line-height: 150px;
19     text-align: center;
20     color: white;
21   }
22   .satu {
23     background-color: #FF1A00;
24     position: absolute;
25     left: 10px;
26     top: 10px;
27   }
28   .dua{
29     background-color: #008C00;
30     position: absolute;
31     left: 50px;
32     top: 50px;
33   }
34   .tiga{
35     background-color: #4096EE;
36     position: absolute;
37     left: 100px;
38     top: 100px;
39   }
40   </style>
41 </head>
42 <body>
43   <div class="container">
44     <div class="satu"> Box 1 </div>
45     <div class="dua"> Box 2 </div>
46     <div class="tiga"> Box 3 </div>
47   </div>
48 </body>
49 </html>
```



Gambar: Hasil penumpukan 3 box dengan absolute positioning

Saya menumpuk ketiga box dengan menggunakan *absolute positioning*. Secara default, box

yang urutannya paling akhir di kode HTML akan tampil di posisi paling depan. Jika kita menambah 1 box lagi, maka box tersebut akan tampil di depan box ketiga, dst.

Property `z-index` bisa dipakai untuk mengatur urutan ini. Sebagai contoh, jika saya ingin Box 1 (merah) tampil paling depan, bisa menambah property `z-index:1`. Berikut hasilnya:

22.z-index2.html

```
1 .satu {  
2   ...  
3   z-index: 1;  
4 }
```



Gambar: Penambahan property `z-index: 1` pada box 1

Hasilnya, Box 1 sekarang berada pada urutan paling atas.

Jika sebuah element tidak menggunakan property `z-index`, maka dianggap `z-index:0`. Element yang memiliki angka `z-index` paling tinggi, akan berada di urutan teratas.

Contoh lain, bagaimana jika sekarang saya ingin Box 2 di urutan paling atas? Caranya bisa dengan menambah nilai `z-index` yang lebih tinggi daripada `z-index` Box 1:

23.z-index3.html

```
1 .dua{  
2   ...  
3   z-index: 2;  
4 }
```



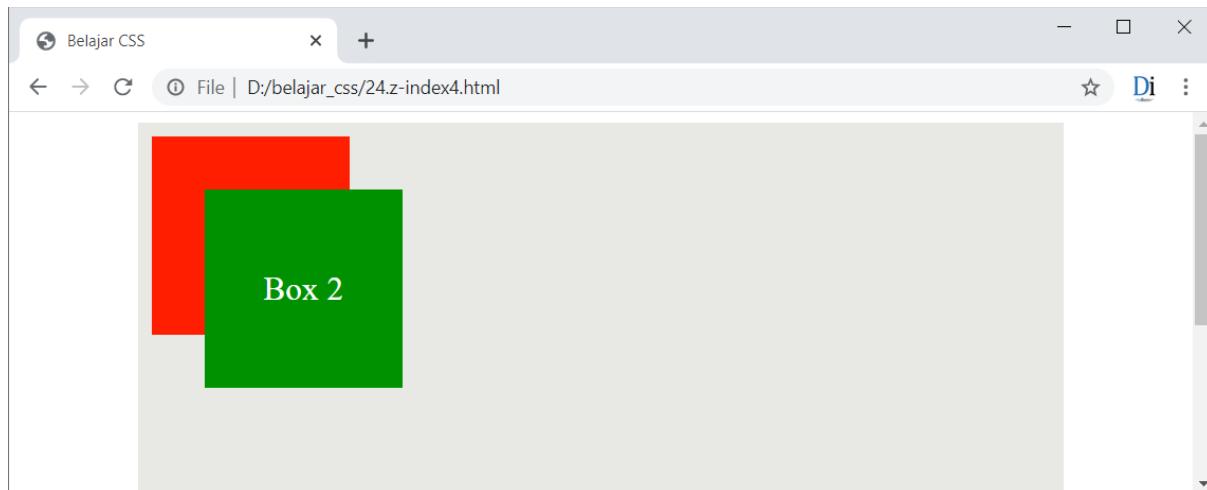
Gambar: Box 2 berada di atas box 1 karena penambahan z-index: 2

Karena Box 2 memiliki nilai `z-index` lebih besar daripada box 1, maka akan tampil paling atas.

Kita bebas ingin menambahkan nilai `z-index` berapa pun, bahkan `z-index:10000` jika ingin agar sebuah element selalu tampil paling atas, selama tidak ada element lain dengan nilai `z-index:10001`. Yang menarik, juga bisa memberikan nilai negatif untuk `z-index`:

24.z-index4.html

```
1 .tiga{  
2     ...  
3     z-index: -100;  
4 }
```



Gambar: Box 3 hilang karena penambahan property z-index: -100

Hasilnya, Box 3 "hilang" tersembunyi di belakang `.container`. Alasannya karena `<div class="container">` secara default memiliki `z-index:0` yang lebih tinggi daripada `z-index:-100`.

10.5. Property float

Property **float** juga bisa dipakai untuk mengatur posisi dari sebuah element, bahkan menjadi cara utama untuk mengatur layout sebuah web. Property **float** memiliki 4 nilai: **none**, **inherit**, **left** dan **right**.

Nilai **none** berguna untuk menghapus efek float sebelumnya, sedangkan nilai **inherit** berfungsi agar nilai float sama dengan parent element saat ini.

Yang akan banyak kita bahas adalah 2 nilai terakhir, yakni **float:left** dan **float:right**.

Keduanya dipakai untuk memindahkan posisi element ke sisi kiri (left) atau ke sisi kanan (right) di dalam parent element.

Langsung saja kita masuk ke contoh praktek:

25.no_float.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              width: 960px;
9              margin: 0 auto;
10         }
11         img {
12             width: 200px;
13             padding: 5px;
14             border: 2px solid black;
15             margin: 0 10px 10px 0;
16         }
17     </style>
18 </head>
19 <body>
20     <h1>Penggunaan Property float</h1>
21     
22     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
23 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
24 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
25 sit amet ullamcorper elit vulputate in. Morbi euismod est ac nisl
26 pulvinar, quis auctor neque pretium. Phasellus molestie ac diam nec
27 sagittis. Quisque tempor purus est, nec facilisis massa sollicitudin nec.
28     </p>
29 </body>
30 </html>
```



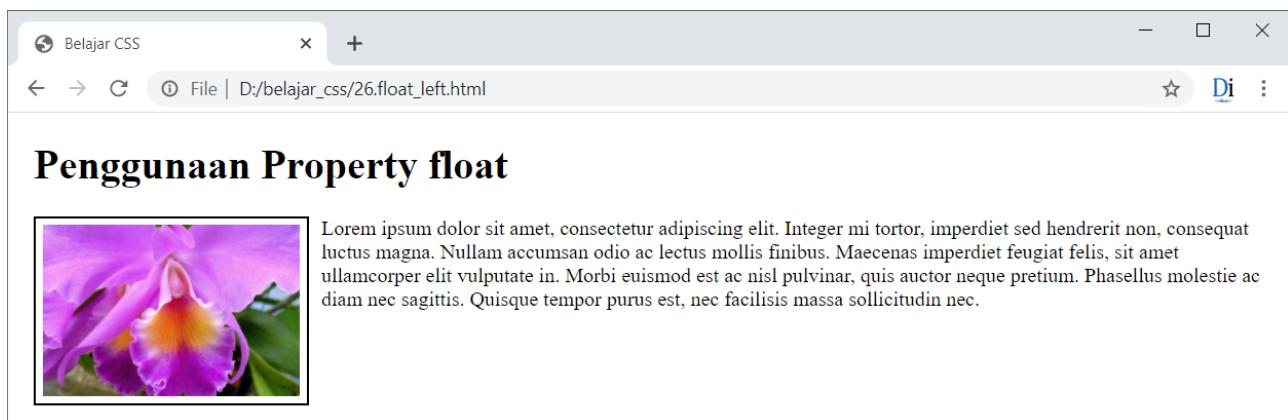
Gambar: Normal document flow tanpa menggunakan float

Di sini saya membuat kode HTML yang terdiri dari tag `<h1>`, tag `` dan tag `<p>`, serta beberapa kode CSS untuk mempercantik tampilan border gambar. Semua element ini tampil dalam *normal document flow*.

Berikutnya, mari kita tambah `float:left` kepada selector `img`:

26.float_left.html

```
1  img {  
2      ...  
3      float: left;  
4 }
```



Gambar: Hasil penggunaan property float: left pada tag ``

Efeknya, paragraf yang berada di bawah gambar akan pindah ke samping tag ``. Ini terjadi karena property `float:left` membuat gambar didorong ke sisi kiri parent element, sehingga di sisi kanan gambar, paragraf akan naik.

Bagaimana jika menggunakan nilai `float:right`?

27.float_right.html

```
1  img {  
2      ...  
3      float: right;  
4  }
```



Gambar: Hasil penggunaan property float: right pada tag

Seperti yang bisa diperkirakan, kali ini gambar akan pindah ke sisi kanan, serta paragraf akan menempati sisi kiri gambar.

Inilah fungsi paling sederhana dari property float, dimana kita bisa mengatur penempatan sebuah element agar berada di sisi kiri atau di kanan parent element.

Efek float pada Normal Document Flow

Property float memiliki beberapa "efek samping" yang harus kita bahas. Sebagai bahan praktek, saya kembali ke bentuk 3 box sebelumnya:

28.float_box.html

```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4      <meta charset="UTF-8">  
5      <title>Belajar CSS</title>  
6      <style>  
7          .container {  
8              width: 700px;  
9              background-color: rgb(231, 232, 226);  
10             margin: 0 auto;  
11         }  
12         .satu, .dua, .tiga {  
13             width: 150px;  
14             height: 150px;  
15             font-size: 25px;  
16             line-height: 150px;  
17             text-align: center;  
18             color: white;  
19     }
```

CSS Positioning

```
20     .satu {
21         background-color: #FF1A00;
22     }
23     .dua{
24         background-color: #008C00;
25     }
26     .tiga{
27         background-color: #4096EE;
28     }
29     </style>
30 </head>
31 <body>
32     <div class="container">
33         <div class="satu"> Box 1 </div>
34         <div class="dua"> Box 2 </div>
35         <div class="tiga"> Box 3 </div>
36     </div>
37 </body>
38 </html>
```



Gambar: 3 div box untuk contoh float

Kode ini mirip seperti praktik di property position. Mari langsung bahas bagaimana efek float pada normal document flow:

29.float_box_left.html

```
1  .satu {
2      background-color: #FF1A00;
3      float: left;
4 }
```

CSS Positioning



Gambar: Kemana kotak kedua (hijau)?

Saya menambah property `float:left` pada box 1. Apa yang terjadi? Kemana box 2 (hijau)?

Box 2 seolah-olah hilang karena property `float:left` pada box 1 akan mengangkat box tersebut dari *normal document flow*. Lalu box 2 dan box 3 akan naik ke atas mengisi ruang yang ditinggalkan oleh box 1.

Dengan kata lain, box 2 sebenarnya masih ada dan berada tepat di belakang box merah (tertutupi). Agar lebih jelas, saya akan tempatkan box 1 ke sisi kanan:

30.float_box_right.html

```
1 .satu {  
2   background-color: #FF1A00;  
3   float: right;  
4 }
```



Gambar: Box pertama di float ke kanan

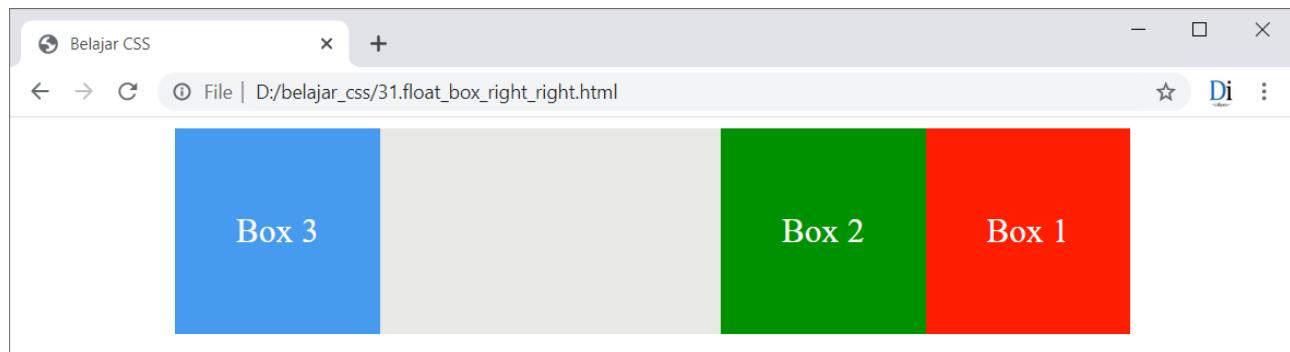
Yes, box 2 kembali terlihat...

Praktek berikutnya, bagaimana jika box 2 juga di float ke kanan dengan menambahkan property `float:right`? Mari kita coba:

31.float_box_right_right.html

```

1 .satu {
2   background-color: #FF1A00;
3   float: right;
4 }
5 .dua{
6   background-color: #008C00;
7   float: right;
8 }
```



Gambar: Box 1 dan 2 di float ke kanan

Terlihat box 2 berada di sebelah kiri box 1. Ini terjadi karena pada kode HTML, box 1 lah yang ditulis terlebih dahulu, setelah itu baru box 2. Ketika kedua box sama-sama dikenakan property `float:right`, box 1 akan mengambil tempat paling kanan.

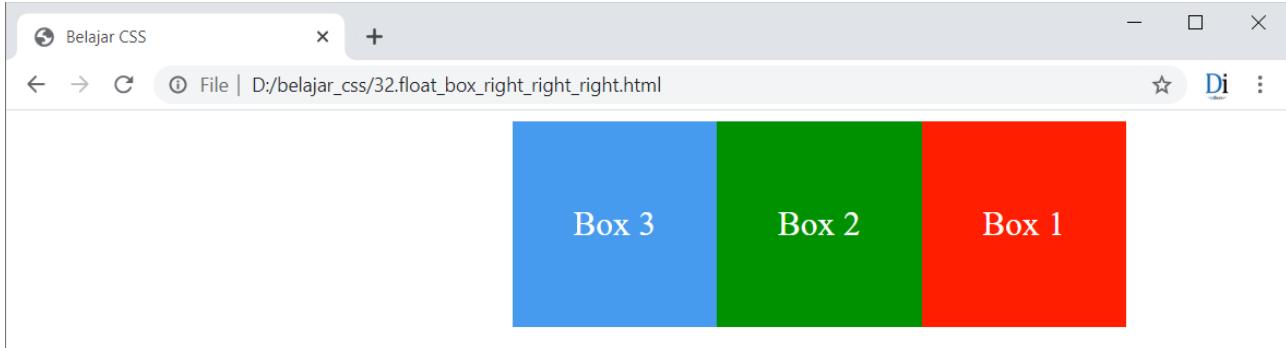
Jika diperhatikan, efek lain dari percobaan ini adalah warna background abu-abu dari parent element ketiga box, yakni tag `<div class="container">`, akan ikut "menyusut". Ini terjadi karena saya tidak memberikan property `height` kepada `.container`. Oleh karena itu, tinggi container akan berubah-ubah sesuai apa yang ditampungnya.

Lanjut, saya akan tambah property `float:right` ke box 3:

32.float_box_right_right_right.html

```

1 .satu {
2   background-color: #FF1A00;
3   float: right;
4 }
5 .dua{
6   background-color: #008C00;
7   float: right;
8 }
9 .tiga{
10  background-color: #4096EE;
11  float: right;
12 }
```



Gambar: Ketiga box di float ke kanan

Kali ini ketiga kotak berdempetan ke sisi kanan, mulai dari box 1 di sisi paling kanan, kemudian diikuti oleh box 2, dan box 3.

Tapi, tunggu dulu. Kemana warna background abu-abu dari `.container`?

Inilah efek dari property `float` yang sering membuat pusing. Kasus di atas terjadi karena ketiga box sudah tidak berada di *normal document flow*. Akibatnya tag `<div class="container">` akan *collapse* atau hilang.

Dalam CSS, ini di kenal dengan istilah **container collapse**, yakni suatu kondisi dimana element HTML akan "hilang" jika di dalamnya tidak berisi element apapun.

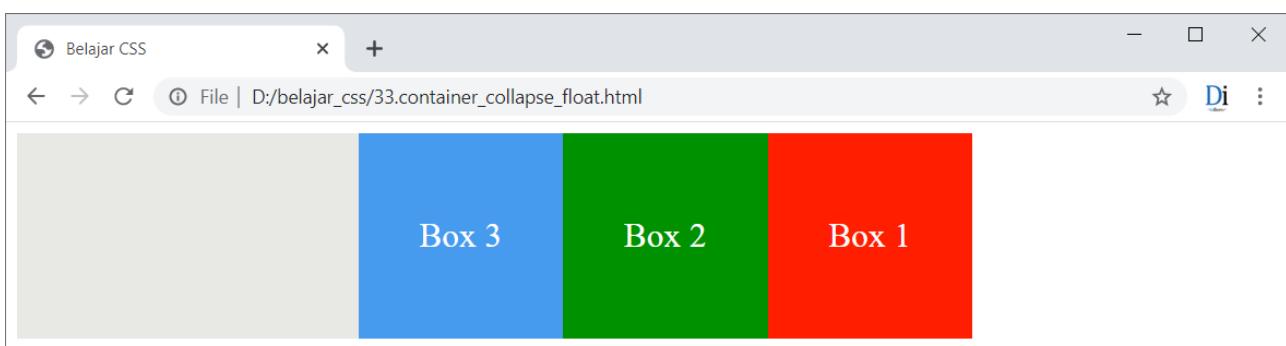
Mengatasi Container Collapse

Terdapat beberapa cara untuk mengatasi efek container collapse.

Cara **pertama** adalah dengan men-float-kan juga parent element. Jika saya menambah property `float` pada container, warna background akan muncul kembali:

33.containerCollapse_float.html

```
1 .container {  
2   width: 700px;  
3   background-color: rgb(231, 232, 226);  
4   margin: 0 auto;  
5   float: left;  
6 }
```



Gambar: Efek yang terjadi dengan men-float-kan `.container`

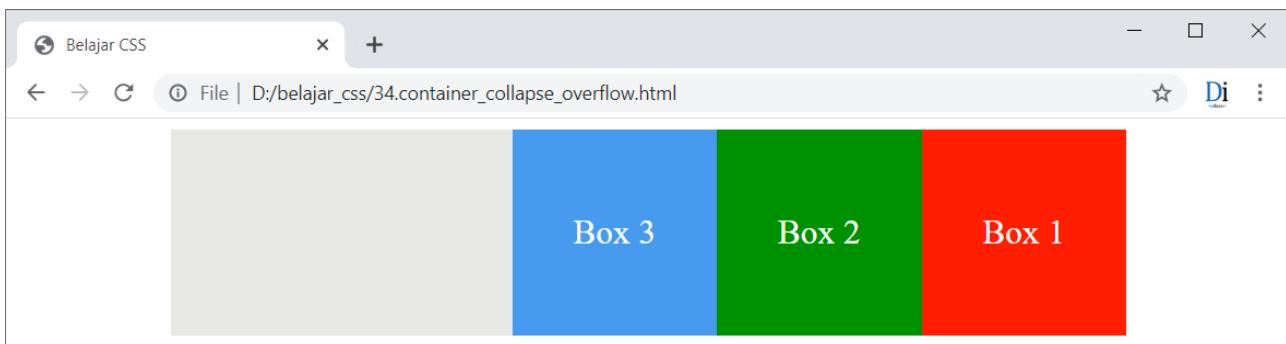
Namun seperti yang terlihat, container ini juga ikut float ke sebelah kiri parent elementnya, yakni tag <body>.

Selain itu, container ikut diangkat dari *normal document flow*. Apabila container tersebut berada di dalam container lain, kita juga terpaksa men-float-kan parent element-nya, dan begitu seterusnya. Ini cukup merepotkan jika halaman tersebut memiliki banyak container.

Cara **kedua** (yang mungkin tidak terduga), adalah dengan menambah property `overflow: hidden` pada parent element:

34.containerCollapseOverflow.html

```
1 .container {
2   width: 700px;
3   background-color: rgb(231, 232, 226);
4   margin: 0 auto;
5   overflow: hidden;
6 }
```



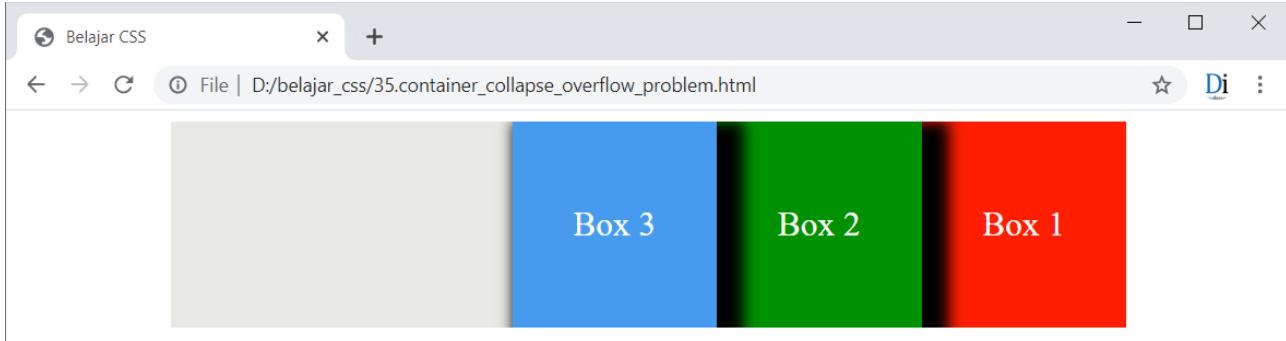
Gambar: Mencegah container collapse dengan `overflow: hidden`

Fungsi dari property `overflow` pernah kita bahas pada bab box model. Property `overflow` digunakan untuk mengatur apa yang terjadi ketika konten tidak muat ditampung oleh sebuah element. Namun property `overflow` juga membawa efek lain, yakni menormalkan hasil float.

Penggunaan property `overflow` seperti ini bisa dipakai pada hampir 90% kasus *container collapse*. Namun pada kasus tertentu, property `overflow` akan memotong isi konten yang ada seperti contoh berikut:

35.containerCollapseOverflowProblem.html

```
1 .container {
2   ...
3   overflow: hidden;
4 }
5 .satu, .dua, .tiga {
6   ...
7   box-shadow: 10px 10px 10px 10px black;
8 }
```



Gambar: Efek bayangan terpotong karena property overflow: hidden

Pada kode ini saya menambahkan property `box-shadow` untuk membuat efek bayangan. Hasilnya, bayangan tersebut ikut terpotong karena sudah berada di luar element (efek dari property `overflow:hidden`).

Untuk kasus seperti ini, kita bisa pakai cara **ketiga**: menggunakan property `clear` pada sebuah tag yang ditempatkan setelah element float terakhir:

36.container_clear.html

```

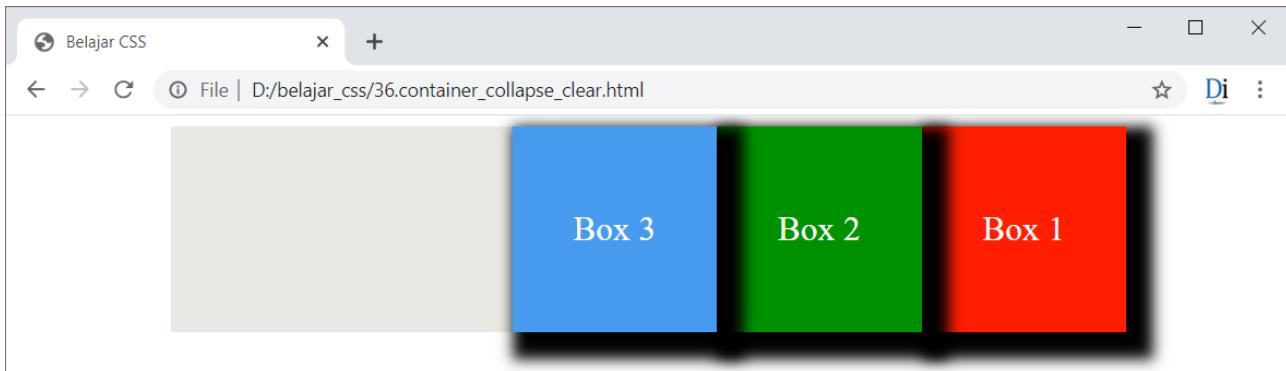
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 700px;
9              background-color: rgb(231, 232, 226);
10             margin: 0 auto;
11         }
12         .satu, .dua, .tiga {
13             width: 150px;
14             height: 150px;
15             font-size: 25px;
16             line-height: 150px;
17             text-align: center;
18             color: white;
19             box-shadow: 10px 10px 10px 10px black;
20         }
21         .satu {
22             background-color: #FF1A00;
23             float: right;
24         }
25         .dua{
26             background-color: #008C00;
27             float: right;
28         }
29         .tiga{
30             background-color: #4096EE;
31             float: right;
32     }

```

```

33  </style>
34 </head>
35 <body>
36  <div class="container">
37    <div class="satu"> Box 1 </div>
38    <div class="dua"> Box 2 </div>
39    <div class="tiga"> Box 3 </div>
40    <br style="clear:both">
41  </div>
42 </body>
43 </html>

```



Gambar: Mengatasi container collapse dengan property clear

Untuk menggunakan trik ini, kita perlu menambah sebuah tag `
` di bagian paling bawah container. Ke dalam tag `
` saya menulis property CSS `clear:both`. Hasilnya, container tidak collapse dan bayangan juga tidak terpotong.

Fungsi dari property `clear` sebenarnya bukan untuk mencegah container collapse, namun memang selalu berkaitan dengan property `float`. Mengenai hal ini akan kita bahas sesaat lagi.

10.6. Float Positioning

Masih membahas property `float`, kali ini kita akan coba latihan mengatur posisi element menggunakan `float` dan konsep box model.

Sampai beberapa waktu lalu, float dan box model masih jadi pilihan favorit untuk membuat layout, setidaknya sampai muncul alternatif CSS3 flexbox dan CSS3 grid.

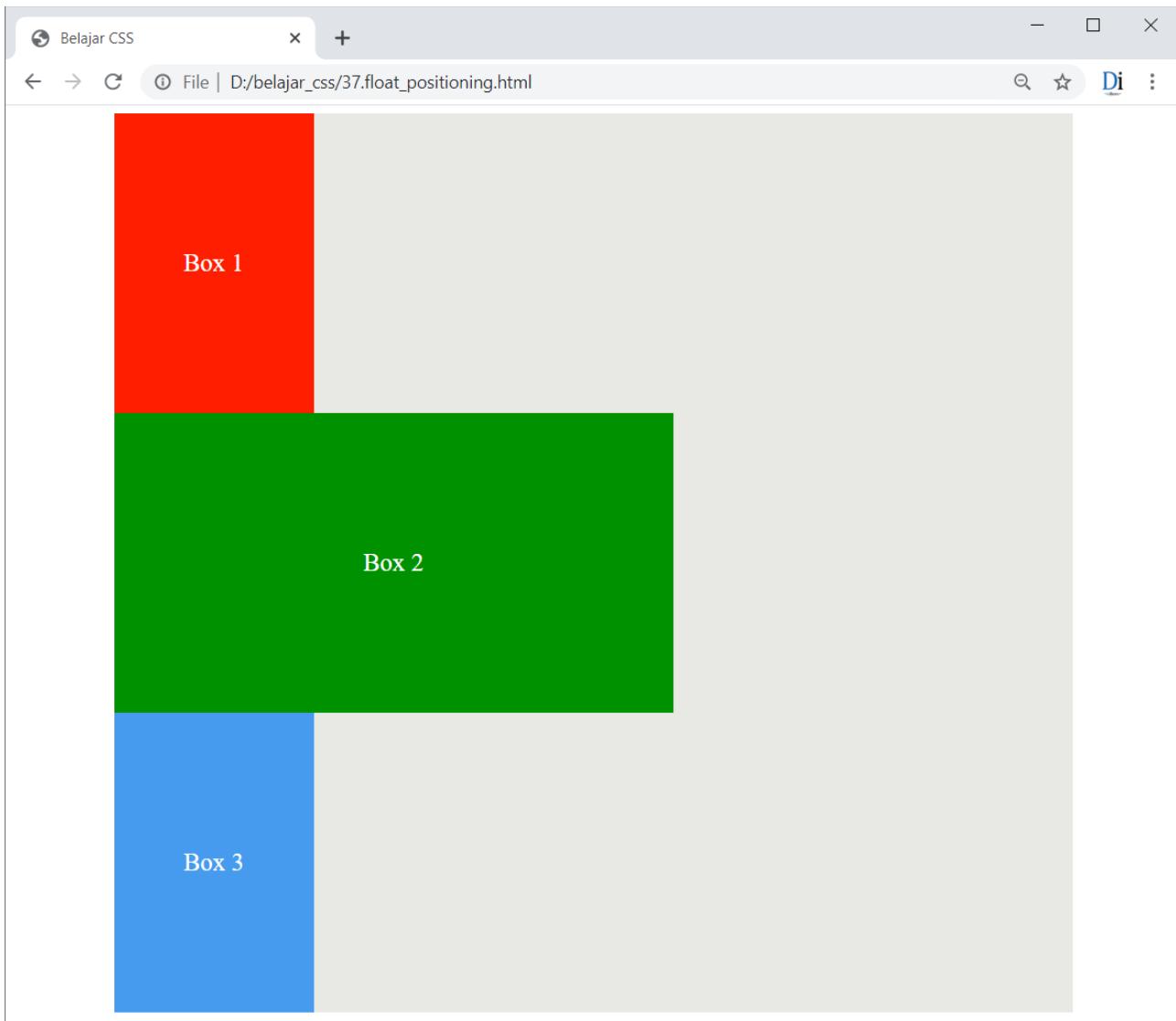
Meskipun saat ini lebih disarankan untuk beralih ke flexbox atau grid, masih ada template dan layout yang dibuat menggunakan float positioning. Oleh karena itu tidak ada salahnya kita tetap mempelajari konsep pembuatan layout menggunakan float.

Sebagai bahan praktik, saya akan memperlebar ukuran container dan ketiga box. Lebar container saya set menjadi 960 pixel. Box 1 dan box 3 memiliki lebar 200 pixel, serta box 2 dengan lebar 560 pixel. Jika ditambah, ketiga box ini memiliki lebar persis 960 pixel:

37.float_positioning.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 960px;
9              background-color: rgb(231, 232, 226);
10             margin: 0 auto;
11         }
12         .satu, .dua, .tiga {
13             height: 300px;
14             font-size: 25px;
15             line-height: 300px;
16             text-align: center;
17             color: white;
18         }
19         .satu {
20             width: 200px;
21             background-color: #FF1A00;
22         }
23         .dua{
24             width: 560px;
25             background-color: #008C00;
26         }
27         .tiga{
28             width: 200px;
29             background-color: #4096EE;
30         }
31     </style>
32 </head>
33 <body>
34     <div class="container">
35         <div class="satu"> Box 1 </div>
36         <div class="dua"> Box 2 </div>
37         <div class="tiga"> Box 3 </div>
38     </div>
39 </body>
40 </html>
```

CSS Positioning



Gambar: Tampilan latihan float positioning

Silahkan pelajari sejenak kode HTML dan CSS yang ada. Latihan pertama, bisakah anda membuat bentuk layout seperti gambar berikut?



Gambar: Latihan float positioning 1

Untuk membuatnya, cukup tambah property `float:left` ke setiap box. Hasilnya, box 1 akan mengambil tempat paling kiri, diikuti box 2 dan box 3:

38.float_positioning2.html

```
1 .satu {  
2     width: 200px;  
3     background-color: #FF1A00;  
4     float: left;  
5 }  
6 .dua{  
7     width: 560px;  
8     background-color: #008C00;  
9     float: left;  
10 }  
11 .tiga{  
12     width: 200px;  
13     background-color: #4096EE;  
14     float: left;  
15 }
```

Latihan kedua, tanpa mengutak-atik kode HTML silahkan coba membuat tampilan sebagai berikut:



Gambar: Latihan float positioning 2

Kali ini saya ingin box 2 berada di sisi paling kanan. Caranya adalah dengan membuat box 1 dan box3 `float:left`, lalu men-set box 2 dengan `float:right`. Berikut perubahan kode yang diperlukan:

39.float_positioning3.html

```
1 .satu {  
2     width: 200px;  
3     background-color: #FF1A00;  
4     float: left;  
5 }  
6 .dua{  
7     width: 560px;
```

CSS Positioning

```
8     background-color: #008C00;
9     float: right;
10 }
11 .tiga{
12     width: 200px;
13     background-color: #4096EE;
14     float: left;
15 }
```

Latihan terakhir, silahkan coba buat tampilan berikut:



Gambar: Latihan float positioning 3

Berikut kode CSS yang diperlukan:

40.float_positioning4.html

```
1 .satu {
2     width: 200px;
3     background-color: #FF1A00;
4     float: right;
5 }
6 .dua{
7     width: 560px;
8     background-color: #008C00;
9     float: right;
10 }
11 .tiga{
12     width: 200px;
13     background-color: #4096EE;
14     float: left;
15 }
```

Dalam praktek sebenarnya, box 1 dan box 3 ini bisa menjadi sebuah komponen sidebar website, sedangkan box 2 sebagai komponen konten utama. Dengan mengatur penggunaan property `float`, kita bisa menata letak layout halaman web.

10.7. Property Clear

Property `clear` hampir selalu digunakan bersama property `float`. Property ini berfungsi untuk membersihkan (*clearing*) efek float dari element sebelumnya dan memulai *normal document flow* baru. Penjelasan ini akan lebih mudah dipahami dengan menggunakan contoh kode program.

Nilai untuk property `clear` adalah salah satu dari: **none**, **right**, **left**, atau **both**. Nilai `none` dipakai untuk menghapus efek clear yang sudah ada sebelumnya (relatif jarang terpakai). Kita akan fokus ke nilai `right`, `left`, dan `both`.

Sebagai contoh pertama, perhatikan kode berikut ini:

41.clear_paragraf_problem.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              width: 700px;
9              margin: 0 auto;
10         }
11         img {
12             width: 200px;
13             padding: 5px;
14             border: 2px solid black;
15             margin: 0 10px 10px 0;
16             float: left;
17         }
18     </style>
19 </head>
20 <body>
21     <h1>Penggunaan Property clear</h1>
22     
23     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
24 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
25 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
26 sit amet ullamcorper elit vulputate in. Morbi euismod est ac nisl
27 pulvinar, quis auctor neque pretium. Phasellus molestie ac diam nec
28 sagittis. Quisque tempor purus est, nec facilisis massa sollicitudin nec.
29 </p>
30     <p>Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
31 sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate
32 eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget nibh
33 dignissim ipsum interdum maximus at scelerisque odio. Nullam elementum
34 ultrices arcu, hendrerit molestie erat malesuada tincidunt.
35     </p>
36 </body>
37 </html>
```



Gambar: Baris awal paragraf kedua nyangkut di samping gambar

Kode ini mirip seperti yang kita pakai di awal pembahasan property `float`. Gambar memiliki property `float:left` sehingga akan berada di sisi kiri halaman. Seluruh teks akan menempati sisi kanan gambar selagi masih muat.

Perhatikan awal paragraf kedua, 2 baris awal kedua berada di samping gambar karena memang masih muat. Namun bagaimana jika kita ingin semua teks di paragraf kedua mulai persis setelah gambar, bukan disampingnya?

Caranya adalah tempatkan property `clear:left` ke dalam paragraf kedua:

42.clear_paragraf_solution.html

```
<p style="clear: left;">Aliquam a lectus porta, bibendum enim id, dictum nunc.  
Duis tincidunt non suscipit. Nunc id mi molestie, porta
```



Gambar: Efek property clear: left pada paragraf kedua

Saya menggunakan kode inline `style="clear: left;"` ke dalam tag `<p>` agar kode kita lebih singkat. Hasilnya, paragraf kedua akan mulai tepat di bawah gambar.

Penggunaan `clear:left` pada tag `<p>` berarti saya ingin seluruh element di sisi kiri paragraf "bersih" dari efek float. Jika ditemukan element float, maka mulai baris baru setelah element tersebut.

Berikut contoh kedua:

43.clear_box_problem.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 960px;
9              background-color: rgb(231, 232, 226);
10             margin: 0 auto;
11         }
12         .sidebar {
13             width: 200px;
14             height: 400px;
15             background-color: #FF1A00;
16             float: right;
17         }
18         .main{
19             width: 760px;
20             height: 200px;
21             background-color: #008C00;
22             float: left;
23         }
24         .footer{
25             width: 760px;
26             height: 100px;
27             background-color: #4096EE;
28             float: left;
29         }
30     </style>
31 </head>
32 <body>
33     <div class="container">
34         <div class="sidebar"></div>
35         <div class="main"></div>
36         <div class="footer"></div>
37     </div>
38 </body>
39 </html>
```

CSS Positioning



Gambar: Masalah dari floating layout

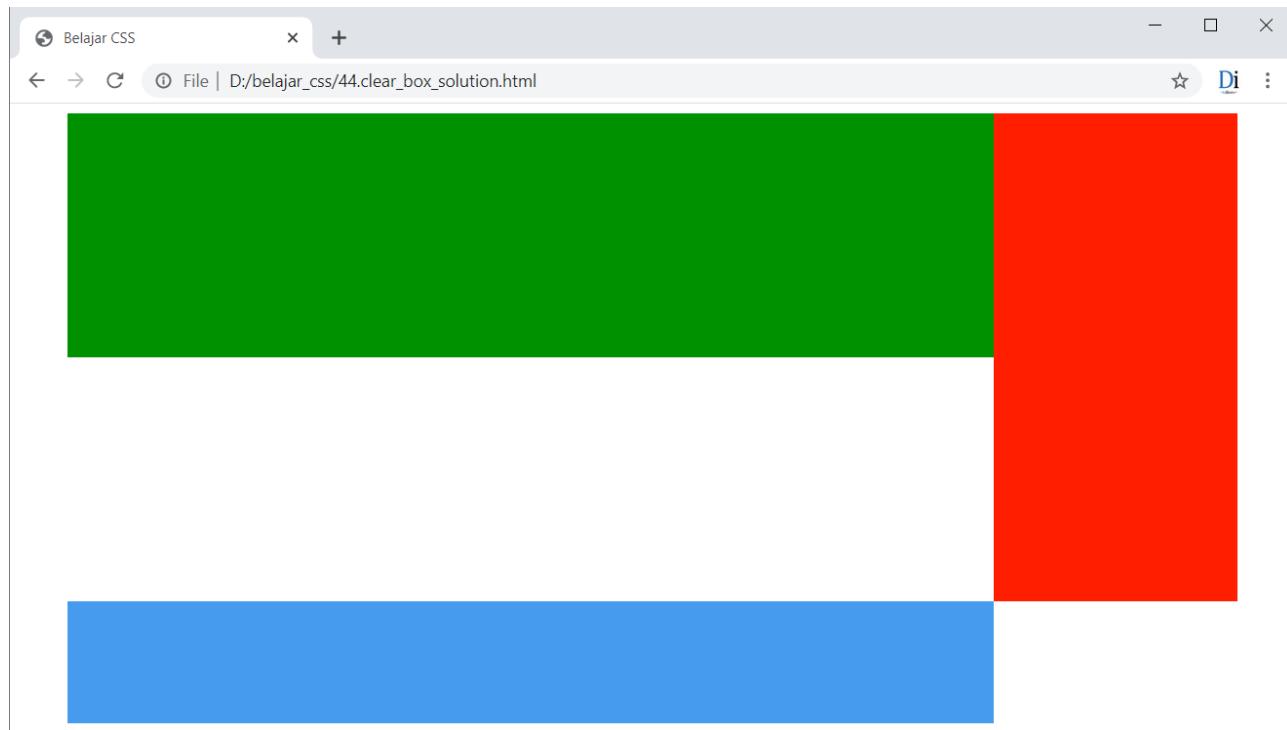
Ini adalah rancangan layout halaman sederhana. Kali ini saya ingin kotak warna biru (class footer) berada di bagian paling bawah. Tapi karena box warna merah cukup tinggi dan di浮akan ke kanan, box footer ini "terjebak" dibagian tengah. Jadi bagaimana solusinya?

Salah satu cara adalah dengan menambah property `clear:right` pada box biru. Hasilnya, box biru akan mulai pada baris baru tepat setelah box merah. Dengan tambahan `clear:right`, blok footer akan "membersihkan" sisi kanan dari element dengan efek float.

44.clear_box_solution.html

```
1 .footer{  
2   width: 760px;  
3   height: 100px;  
4   background-color: #4096EE;  
5   float: left;  
6   clear: right;  
7 }
```

CSS Positioning



Gambar: Box footer (biru) sekarang berada di baris paling bawah

Bagaimana dengan `clear:both`? Sesuai dengan namanya, property ini akan men-clearkan efek float di sisi kiri dan kanan element. Berikut contoh kasusnya:

45.clear_both_problem.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body {
8              width: 700px;
9              margin: 0 auto;
10         }
11         img {
12             width: 200px;
13             padding: 5px;
14             border: 2px solid black;
15             margin: 0 10px;
16         }
17     </style>
18 </head>
19 <body>
20     <h1>Penggunaan Property clear</h1>
21     
22     
23     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
24     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
25     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
```

```

26  sit amet ullamcorper elit vulputate in. Morbi euismod est ac nisl
27  pulvinar, quis auctor neque pretium. Phasellus molestie ac diam nec.
28  sagittis. Quisque tempor purus est, nec facilisis massa sollicitudin nec.
29  </p>
30  <p>Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
31  sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate
32  eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget nibh
33  dignissim ipsum interdum maximus at scelerisque odio. Nullam elementum
34  ultrices arcu, hendrerit molestie erat malesuada tincidunt.
35  </p>
36  </body>
37  </html>

```

Penggunaan Property clear

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis, sit amet ullamcorper elit vulputate in. Morbi euismod est ac nisl pulvinar, quis auctor neque pretium. Phasellus molestie ac diam nec sagittis. Quisque tempor purus est, nec facilisis massa sollicitudin nec.

Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget nibh dignissim ipsum interdum maximus at scelerisque odio. Nullam elementum ultrices arcu, hendrerit molestie erat malesuada tincidunt.

Gambar: Teks di antara 2 gambar

Dalam kode ini saya memiliki 2 gambar yang di float ke kiri dan ke kanan. Akibatnya, teks akan naik mengisi ruang antara kedua gambar tersebut. Bagaimana cara agar teks pindah ke bawah kedua gambar?

Jika kita memakai `clear:left`, teks memang muncul di bawah gambar kiri, tapi di sisi kanan masih ada gambar kedua. Solusi yang paling pas adalah menggunakan `clear:both`:

46.clear_both_solution.html

```
<p style="clear: both;">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam accumsan odio ac lectus mollis finibus. Maecenas imperdiet
```



The screenshot shows a browser window titled "Belajar CSS". The address bar indicates the file is located at "D:/belajar_css/46.clear_both_solution.html". The main content area features a large heading "Penggunaan Property clear". Below the heading are two images of flowers: a purple orchid on the left and a pink and white flower on the right, both enclosed in black-bordered boxes. A block of placeholder text follows:

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis, sit amet ullamcorper elit vulputate in. Morbi euismod est ac nisl pulvinar, quis auctor neque pretium. Phasellus molestie ac diam nec sagittis. Quisque tempor purus est, nec facilisis massa sollicitudin nec.

Aliquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget nibh dignissim ipsum interdum maximus at scelerisque odio. Nullam elementum ultrices arcu, hendrerit molestie erat malesuada tincidunt.

Gambar: Teks tampil pada baris baru di bawah kedua gambar

Dengan tambahan property `clear:both`, teks yang ada di dalam tag `<p>` akan mulai di baris baru, yakni baris dimana tidak ada lagi element lain yang terkena efek float di sisi kiri dan kanan paragraf.

Dalam bab ini kita telah membahas berbagai property CSS yang berhubungan dengan posisi element. Materi ini akan sangat membantu dalam perancangan layout web nantinya.

Untuk beberapa bab ke depan, saya ingin mengajak anda mendalami beberapa teknologi terbaru dari CSS3, yang akan dimulai dari **CSS3 Gradient**.

11. CSS3 Gradient

Bab kali ini akan membahas cara membuat *gradient* dengan CSS3. **Gradient** adalah istilah grafis untuk menyebut perubahan warna secara bertahap dari satu warna ke warna lain. Bagi yang sering menggunakan aplikasi pengolah gambar seperti Adobe Photoshop, tentu tidak asing dengan istilah yang satu ini.

11.1. Membuat Gradient dengan Gambar

Sebelum CSS3, efek gradient bisa didapat dengan cara memasukkan gambar gradient (yang dibuat manual) sebagai background image. Teknik ini sebenarnya sudah jarang dipakai, namun tetap ingin saya praktikkan sekedar pengetahuan umum.

Caranya, siapkan sebuah gambar gradient yang dibuat dari aplikasi pengolah gambar seperti Photoshop, kemudian pakai property `background-image` dan `background-repeat` untuk mengulang gambar.

Di dalam folder `belajar_css/bab_11_css3_gradient`, terdapat gambar `gradient.png` yang memiliki efek gradient dari warna biru ke putih. Gambar seperti inilah yang harus dipersiapkan:



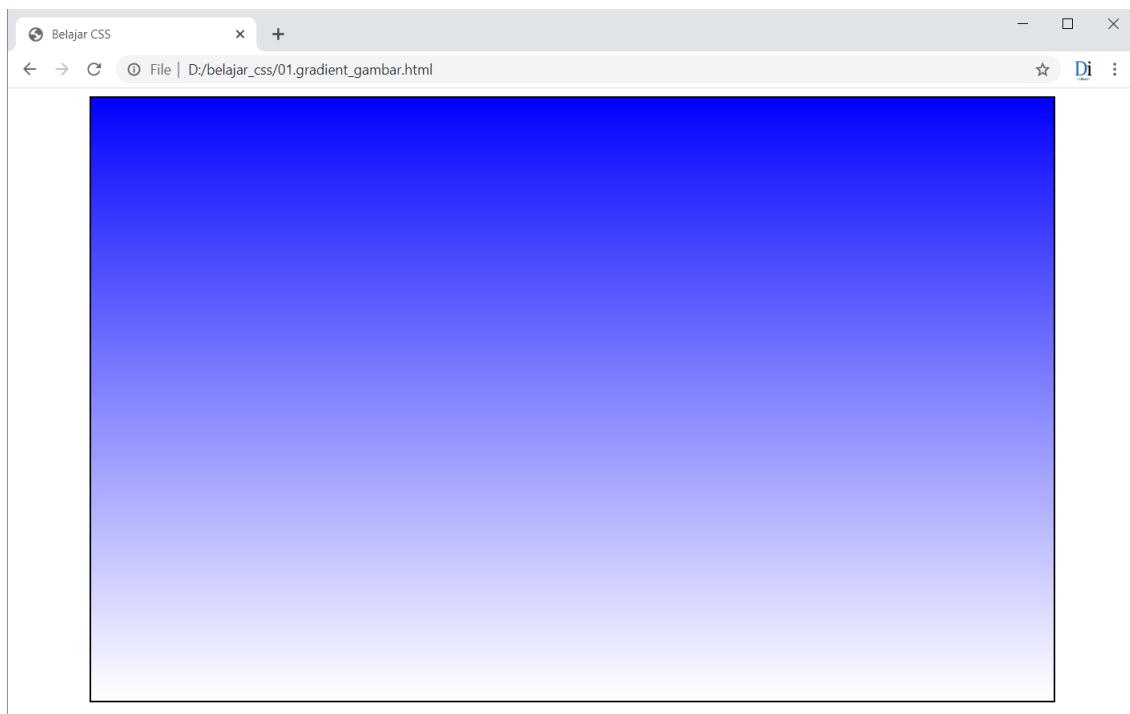
Gambar: `gradient.png` untuk membuat efek gradient

Lebar gambar tidak begitu penting karena kita bisa pakai property `background-repeat` untuk mengulang gambar ini. Gambar gradient di atas berdimensi 13 x 600 pixel. Jika ingin menghemat bandwith, bisa menggunakan lebar gambar 1 x 600 pixel.

Selanjutnya, pakai gambar ini sebagai background dari sebuah element. Berikut contoh penggunaannya:

01.gradient_gambar.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 960px;
9              height: 600px;
10             margin: 0 auto;
11             border: 2px solid black;
12             background-image: url('gradient.png');
13             background-repeat: repeat-x;
14         }
15     </style>
16 </head>
17 <body>
18     <div></div>
19 </body>
20 </html>
```



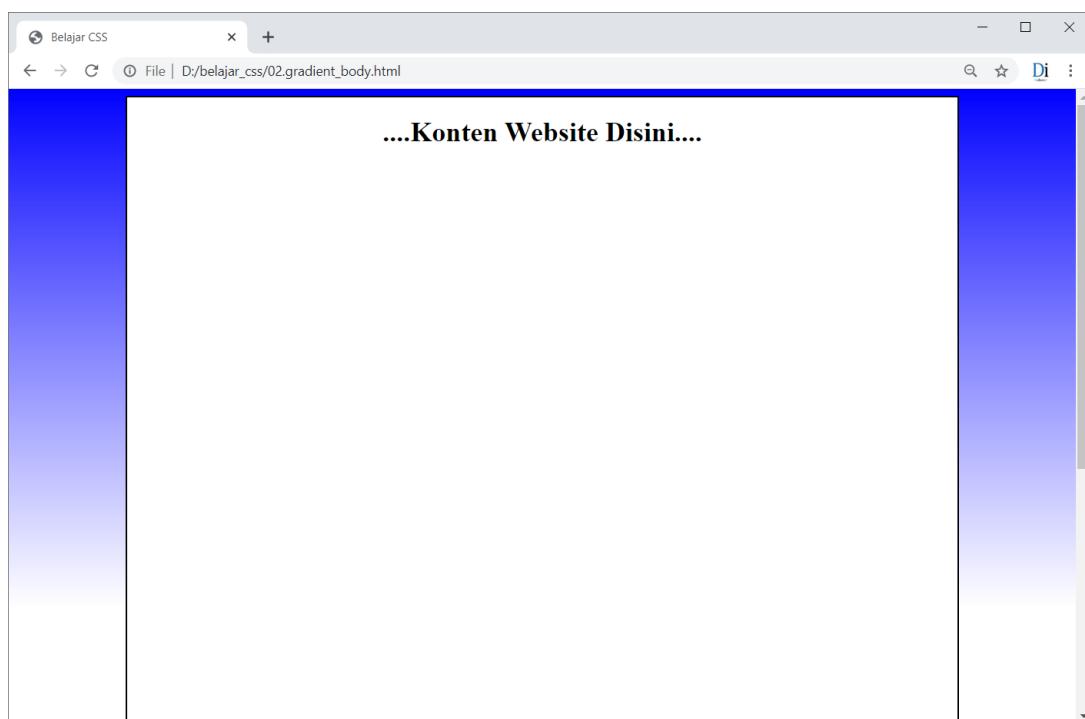
Gambar: Hasil penggunaan gambar gradient

Inti dari teknik di atas adalah tempatkan gambar `gradient.png` pada sumbu x secara berulang (`repeat-x`). Ini dihasilkan dari penggunaan property `background-image: url('gradient.png')` dan `background-repeat: repeat-x`.

Gambar gradient ini juga bisa digunakan untuk background dari tag <body>:

02.gradient_body.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{
8              background-image: url('gradient.png');
9              background-repeat: repeat-x;
10         }
11         div {
12             width: 960px;
13             height: 1200px;
14             margin: 0 auto;
15             border: 2px solid black;
16             background-color: white;
17             text-align: center;
18         }
19     </style>
20 </head>
21 <body>
22     <div>
23         <h1>....Konten Website Disini....</h1>
24     </div>
25 </body>
26 </html>
```



Gambar: Efek gradient untuk background <body>

Dengan mengubah gambar `gradient.png` dengan warna lain, kita bisa menghasilkan efek gradient yang didukung oleh hampir semua web browser, termasuk web browser jadul seperti Internet Explorer. Namun sesuai dengan judul bab, mari kita lihat apa yang ditawarkan oleh CSS3 gradient.

11.2. Property Gradient CSS3

Untuk membuat gradient di CSS3, tempatkan perintah gradient di dalam property `background-image`. Dengan kata lain, gradient adalah sejenis gambar background di dalam CSS. Berikut penulisan dasarnya:

```
selector {
    background-image: efek_gradient_disini;
}
```

CSS3 mendukung 2 jenis efek gradient, yakni **linear gradient** (gradient garis) dan **radial-gradient** (gradient lingkaran). Kita akan mulai dari linear gradient terlebih dahulu.

11.3. Linear Gradient

Efek gradient yang paling sederhana adalah linear gradient, dimana gradasi warna (perubahan warna) terjadi pada sebuah garis lurus. Berikut contoh pembuatan linear gradient dari biru ke putih:

```
background-image: linear-gradient(blue, white);
```

Hasil dari property di atas sama persis dengan gambar gradient yang dipakai pada awal bab.

Jika anda pernah belajar bahasa pemrograman komputer, cara penulisan gradient ini mirip dengan penulisan function (fungsi). Dimana fungsi `linear-gradient()` butuh 2 buah nilai input atau argument, yakni nama warna seperti `blue` dan `white`. Tentu saja kita bisa tukar dengan warna lain.

Berikut contoh praktek dari kode di atas:

03.linear-gradient_basic.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4     <meta charset="UTF-8">
5     <title>Belajar CSS</title>
6     <style>
7         div {
8             width: 960px;
9             height: 600px;
10            margin: 0 auto;
```

```

11      border: 2px solid black;
12      background-image: linear-gradient(blue, white);
13  }
14 </style>
15 </head>
16 <body>
17   <div></div>
18 </body>

```

Hasilnya sama persis dengan contoh kita yang menggunakan gambar, namun kali ini murni dibuat dari CSS. Silahkan anda coba mengubah warna ini dengan nilai lain seperti *green*, *yellow*, atau nilai kode warna CSS seperti RGB dan HSL.

Linear Gradient Color Stop

Warna gradient seperti blue dan white pada contoh di atas dikenal dengan istilah **color-stop**. Color-stop merujuk ke warna apa saja yang ada pada sebuah gradient. Perintah `linear-gradient` setidaknya butuh 2 warna, yakni warna awal dan warna akhir.

Bagaimana jika kita ingin membuat 3, 4 atau bahkan 5 warna? Cukup tambah color-stop lain ke dalam fungsi `linear-gradient()`. Setiap nilai warna dipisah dengan tanda koma seperti contoh berikut:

04.linear-gradient_color_stop.html

```

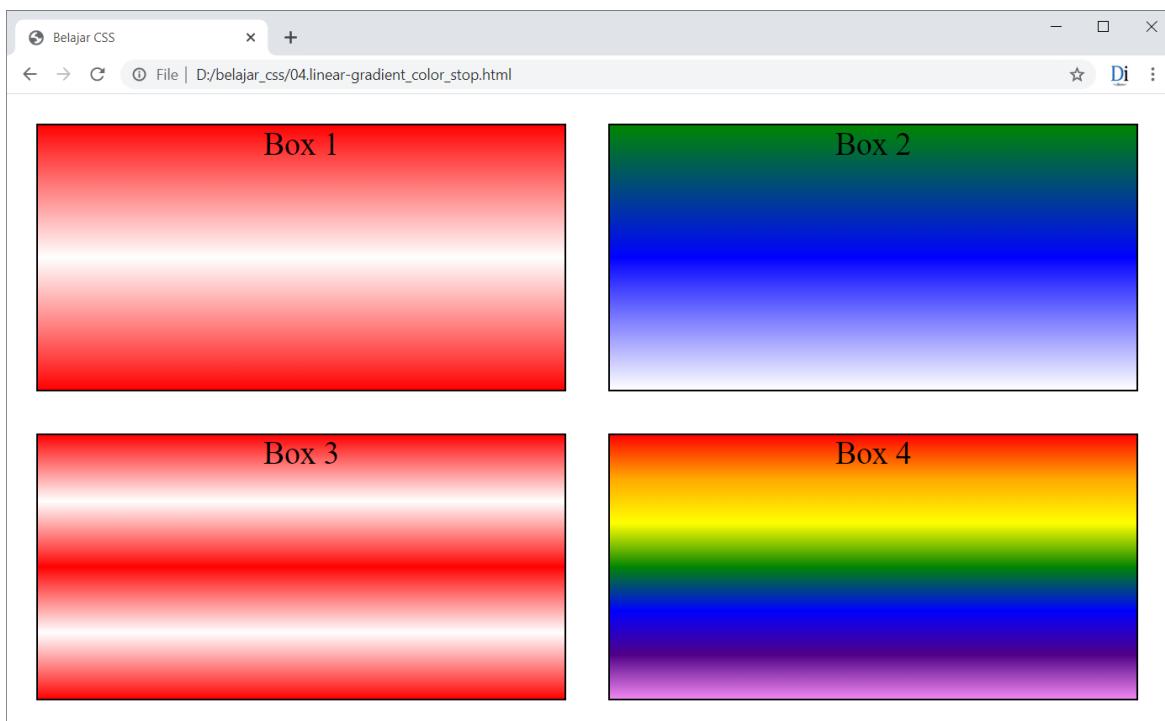
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      div {
8        width: 500px;
9        height: 250px;
10       margin: 0 auto;
11       border: 2px solid black;
12       float: left;
13       margin: 20px;
14       font-size: 30px;
15       text-align: center;
16     }
17     .satu {
18       background-image: linear-gradient(red, white, red);
19     }
20     .dua {
21       background-image: linear-gradient(green, blue, white);
22     }
23     .tiga {
24       background-image: linear-gradient(red, white, red, white, red);
25     }
26     .empat {
27       background-image: linear-gradient(red, orange, yellow, green,

```

```

28                         blue, indigo, violet);
29     }
30   </style>
31 </head>
32 <body>
33   <div class="satu">Box 1</div>
34   <div class="dua">Box 2</div>
35   <div class="tiga">Box 3</div>
36   <div class="empat">Box 4</div>
37 </body>
38 </html>

```



Gambar: Contoh penggunaan linear gradient dengan berbagai color stop

Pada contoh ini saya membuat 4 buah box menggunakan tag `<div>`. Untuk setiap kotak, terdapat nilai linear-gradient dengan beragam variasi warna.

Untuk menghemat tempat, pada contoh-contoh berikutnya saya tidak akan mencantumkan seluruh kode HTML dan CSS, tapi hanya property `gradient` saja.

Linear Gradient Length

Jika diperhatikan, setiap warna dari perintah linear-gradient di sebar secara merata. Misalnya ketika saya menggunakan 2 warna, warna pertama akan mengambil tempat sekitar 50% dari tinggi element, kemudian 50% sisanya menggunakan warna kedua. Jika kita memakai 4 warna, setiap warna akan mengambil $\frac{1}{4}$ bagian dari tinggi element, dst.

CSS3 gradient membolehkan kita mengatur pembagian warna ini. Caranya dengan menambahkan satuan length setelah nilai warna seperti contoh berikut:

```
background-image: linear-gradient(red 75%, white);
```

Nilai 75% berarti warna merah akan tampil hingga 75% dari tinggi element, baru kemudian berganti menjadi warna putih hingga penuh 100%. Property di atas juga bisa ditulis menjadi:

```
background-image: linear-gradient(red 75%, white 100%);
```

Jika kita tidak membuat length, web browser akan menampilkan warna hingga akhir (jika tinggal 1 warna), atau dibagi secara merata (jika terdapat 2 warna atau lebih).

Berikut praktik dari konsep ini:

05.linear-gradient_color_length.html

```
1 .satu {
2   background-image: linear-gradient(red 70%, white);
3 }
4 .dua {
5   background-image: linear-gradient(red 10%, white);
6 }
7 .tiga {
8   background-image: linear-gradient(red 200px, white);
9 }
10 .empat {
11   background-image: linear-gradient(red 90%, white);
12 }
```



Gambar: Contoh penggunaan linear gradient length

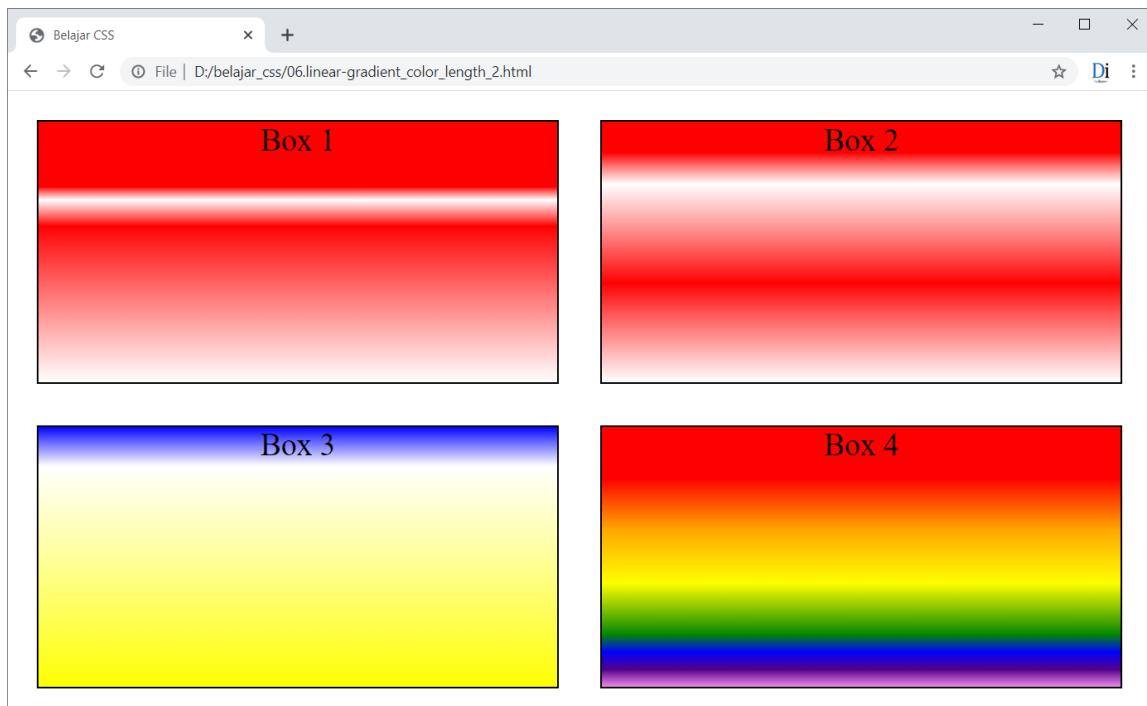
Sebagaimana jumlah color-stop yang tidak dibatasi, kita juga bisa menentukan besaran tiap-tiap warna dan juga menggunakan satuan length lain seperti pixel. Berikut contoh variasinya:

06.linear-gradient_color_length_2.html

```

1 .satu {
2   background-image: linear-gradient(red 25%, white 30%, red 40%, white);
3 }
4 .dua {
5   background-image: linear-gradient(red 30px, white 60px, red, white);
6 }
7 .tiga {
8   background-image: linear-gradient(blue , white 15%, yellow);
9 }
10 .empat {
11   background-image: linear-gradient(red 20%,orange 40%,yellow 60%,
12                                     green 80%, blue , indigo, violet);
13 }

```



Gambar: Contoh penggunaan berbagai linear gradient length

Property `background-image: linear-gradient(red 25%, white 30%, red 40%, white)` bisa dibaca sebagai: buat linear-gradient yang dimulai dengan warna merah hingga posisi 25% dari tinggi element, kemudian berganti dengan warna putih pada posisi 30%, berganti lagi menjadi merah pada posisi 40%, dan sisanya dengan warna putih hingga 100%.

Property `background-image: linear-gradient(red 30px, white 60px, red, white)` berarti buat linear-gradient yang dimulai dengan warna merah hingga 30 pixel, kemudian berganti dengan warna putih hingga 60 pixel, dan sisanya dibagi rata dari merah ke putih.

Linear Gradient Direction

Sejauh ini, linear gradient ditampilkan dari atas ke bawah. Kita juga bisa mengatur arah gradient ini menggunakan satuan keyword atau satuan sudut (*angle unit*). Untuk satuan

keyword, bisa diisi dengan nilai: '**to top**', '**to right**', '**to bottom**', '**to left**', '**to bottom right**', dst. Keyword ini ditulis sebelum warna gradient seperti contoh berikut:

```
background-image: linear-gradient(to top, blue, white);
```

Efek gradient di atas akan mulai dari warna biru di bawah hingga warna putih ke atas (to top)

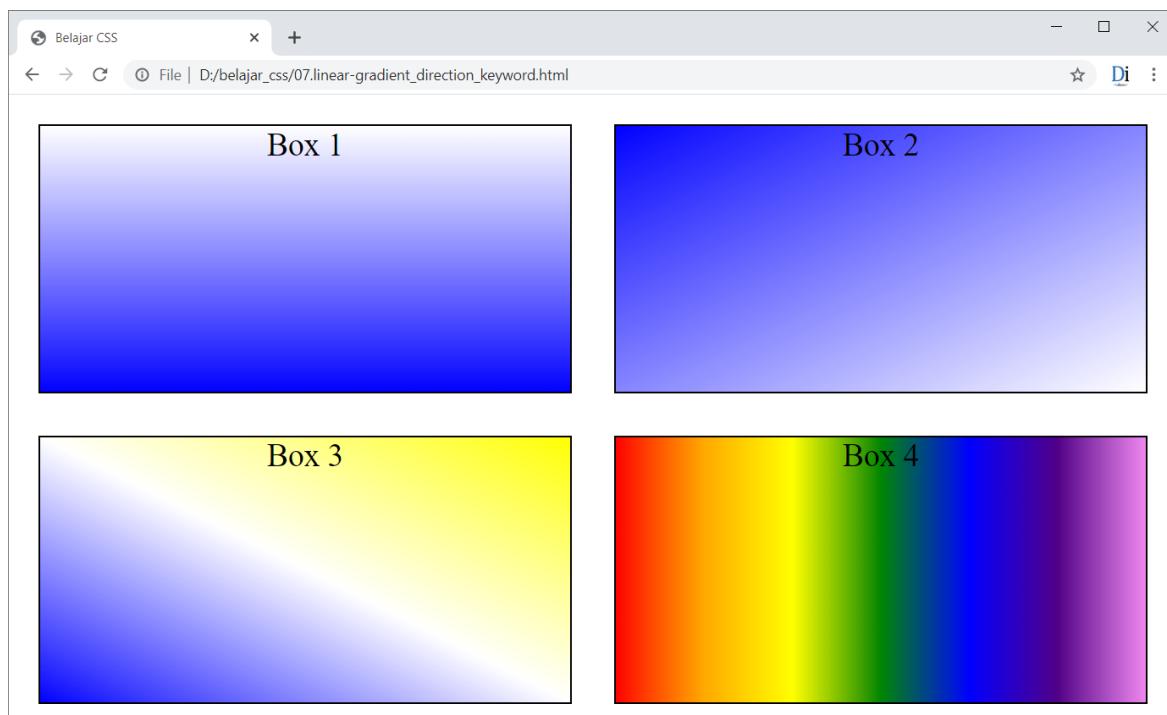
Sebagai contoh lain, property berikut :

```
background-image: linear-gradient(to bottom right, blue, white);
```

Akan menghasilkan efek gradient dari kiri atas ke kanan bawah (bottom right). Berikut contoh penggunaan berbagai nilai keyword linear gradient direction:

07.linear-gradient_direction_keyword.html

```
1 .satu {
2     background-image: linear-gradient(to top, blue, white);
3 }
4 .dua {
5     background-image: linear-gradient(to bottom right, blue, white);
6 }
7 .tiga {
8     background-image: linear-gradient(to top right, blue , white , yellow);
9 }
10 .empat {
11     background-image: linear-gradient(to right, red, orange ,yellow ,
12 green , blue , indigo, violet);
13 }
```



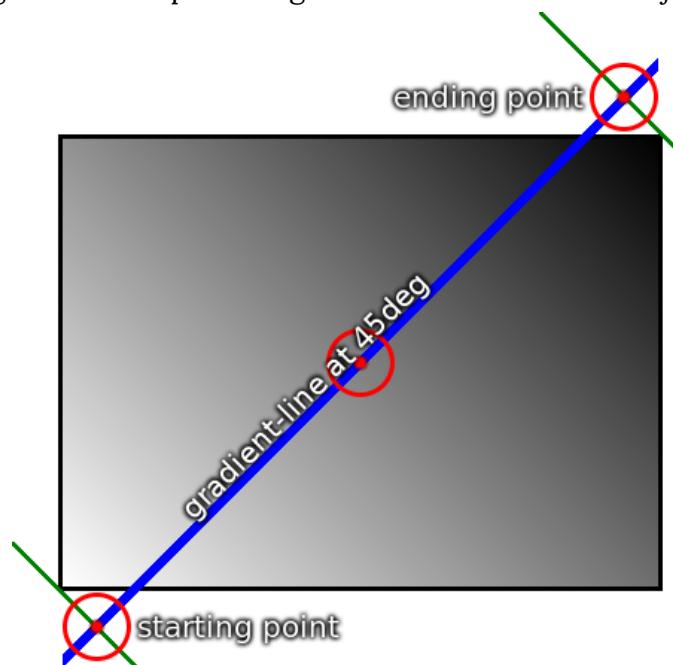
Gambar: Contoh penggunaan linear gradient direction dengan satuan keyword

Selain melalui keyword, arah gradient juga bisa diatur dengan satuan sudut (angle unit). Di dalam CSS3 terdapat 4 satuan sudut, yakni: **degree** (deg), **gradian** (grads), **radian** (rad) dan **turn**. Dari keempat nilai ini, nilai derajat (degrees) lah yang paling mudah digunakan dan didukung luas oleh web browser.

Arah 0deg berarti dari bawah ke atas, atau sama dengan 'to top', 90deg sama dengan 'to right', 180deg sama dengan 'to bottom', dan 270deg sama dengan 'to left'. Satu putaran adalah 360deg dengan kenaikan 1 derajat searah jarum jam.

Selain itu kita juga bisa memberikan nilai negatif, seperti -90deg yang sama artinya dengan 'to left'. Untuk nilai negatif, perhitungannya berlawanan dengan arah jarum jam.

Gambar berikut mengilustrasikan perhitungan arah dalam satuan derajat:



Gambar: Hasil dari 45deg gradient

Berikut contoh penggunaannya:

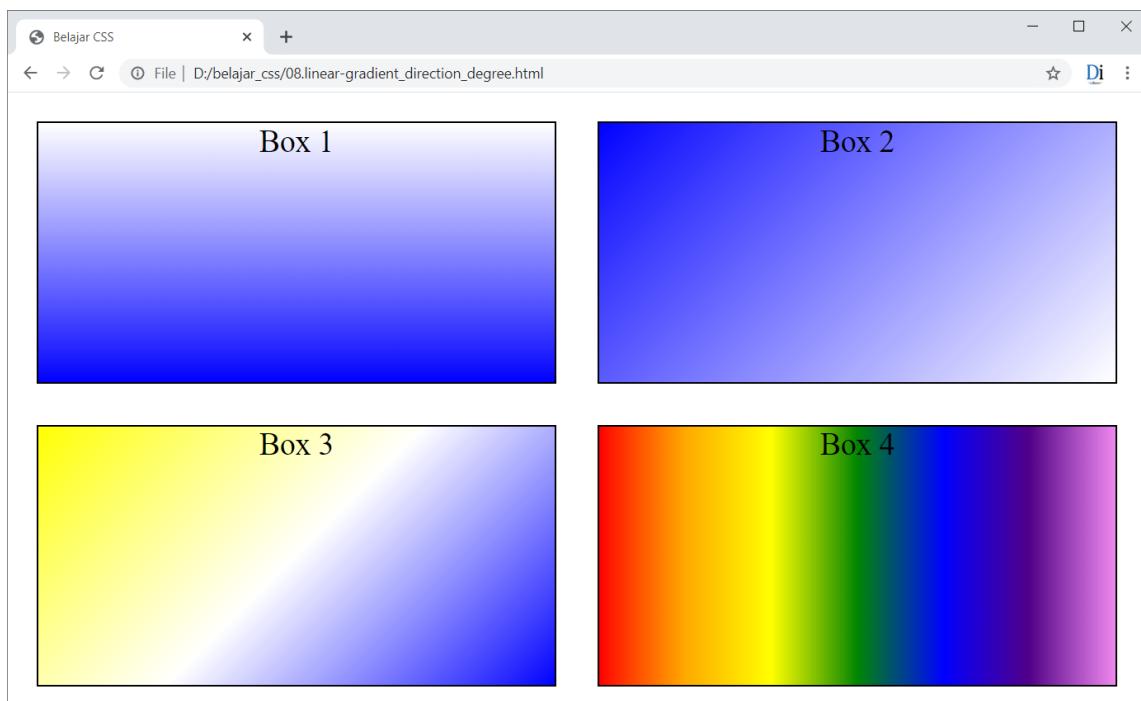
08.linear-gradient_direction_degree.html

```

1 .satu {
2   background-image: linear-gradient(0deg, blue, white);
3 }
4 .dua {
5   background-image: linear-gradient(135deg, blue, white);
6 }
7 .tiga {
8   background-image: linear-gradient(-45deg, blue , white , yellow);
9 }
10 .empat {
11   background-image: linear-gradient(90deg, red, orange ,yellow ,
12   green , blue , indigo, violet);

```

13 }



Gambar: Contoh penggunaan linear gradient direction dengan satuan deg

Repeating Linear Gradients

Untuk menghasilkan efek lanjutan, tersedia perintah `repeating-linear-gradient`. Perintah ini berfungsi untuk mengulang efek gradient beberapa kali. Berikut contoh penulisannya:

```
background-image: repeating-linear-gradient(red, white 25%);
```

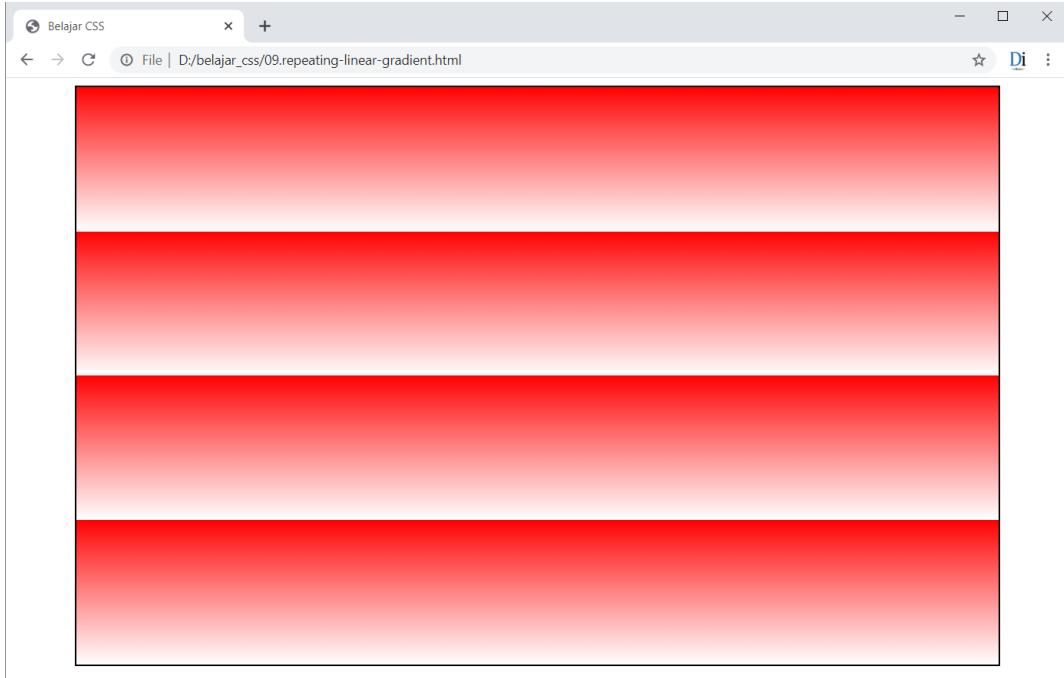
Sewaktu kita mempelajari linear gradients length, warna terakhir harus bernilai 100% (atau dikosongkan, yang artinya sama dengan 100%). Ini bermakna bahwa warna inilah yang akan menutupi 100% panjang element.

Khusus untuk efek gradient yang berulang (`repeating linear gradients`), nilai warna terakhir harus kurang dari 100%, karena nilai inilah yang menentukan berapa kali efek gradient akan diulang. Jika kita menulis nilai 25% seperti di atas, maka warna akan diulang sebanyak 4 kali ($25 \times 4 = 100\%$). Berikut tampilannya:

09.repeating-linear-gradient.html

```
1 div {
2   width: 960px;
3   height: 600px;
4   margin: 0 auto;
5   border: 2px solid black;
6   background-image: repeating-linear-gradient(red, white 25%);
7 }
```

CSS3 Gradient

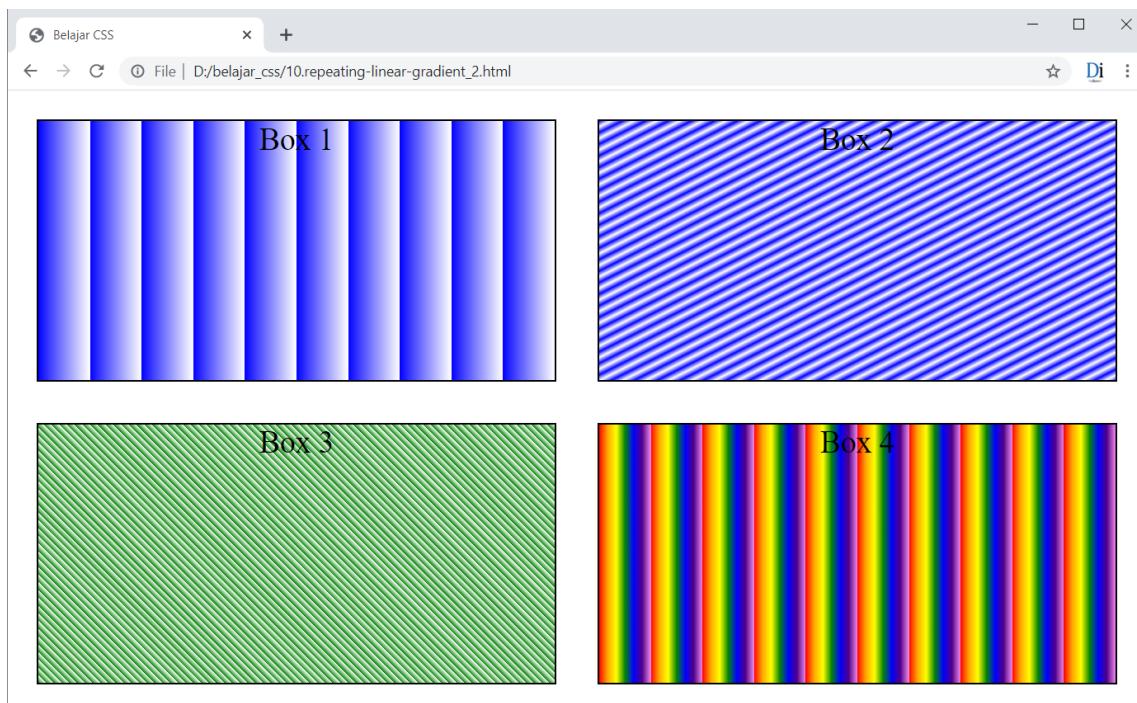


Gambar: Contoh penggunaan repeating linear gradients

Dengan menggunakan berbagai nilai `repeating-linear-gradient`, kita bisa menghasilkan efek gradient yang sangat menarik, seperti 4 contoh berikut:

10.repeating-linear-gradient_2.html

```
1  .satu {
2      background-image: repeating-linear-gradient(to right, blue, white 10%);
3  }
4  .dua {
5      background-image: repeating-linear-gradient(to bottom right, blue,
6          white, blue 10px);
7  }
8  .tiga {
9      background-image: repeating-linear-gradient(45deg, green, white 1%);
10 }
11 .empat {
12     background-image: repeating-linear-gradient(to right, red, orange,
13         yellow , green , blue , indigo, violet 10%);
14 }
```



Gambar: Contoh penggunaan berbagai repeating linear gradients

Kunci dari perintah `repeating-linear-gradient` adalah pada satuan length terakhir. Jika satuan terakhir ditulis 25%, gradient akan di ulang sebanyak 4 kali. Jika satuan terakhir ditulis 5%, maka gradient akan di ulang sebanyak 20 kali.

11.4. Radial Gradient

Radial gradient adalah efek gradient dengan gradasi warna yang berbentuk lingkaran dan elips. Berikut contoh format penulisannya:

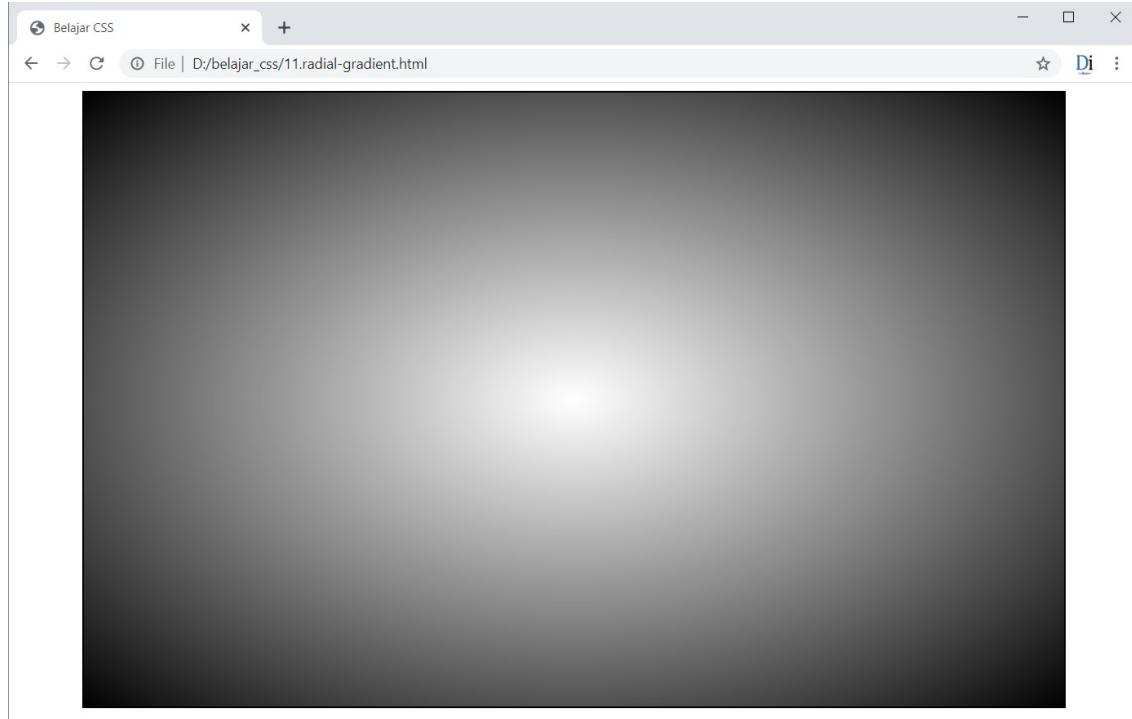
```
background-image: radial-gradient(white, black);
```

Property di atas akan menghasilkan radial gradient berbentuk elips tepat di tengah element. Mari masuk ke praktek:

11.radial-gradient.html

```
1 div {
2   width: 960px;
3   height: 600px;
4   margin: 0 auto;
5   border: 2px solid black;
6   background-image: radial-gradient(white, black);
7 }
```

CSS3 Gradient



Gambar: Contoh penggunaan radial gradient

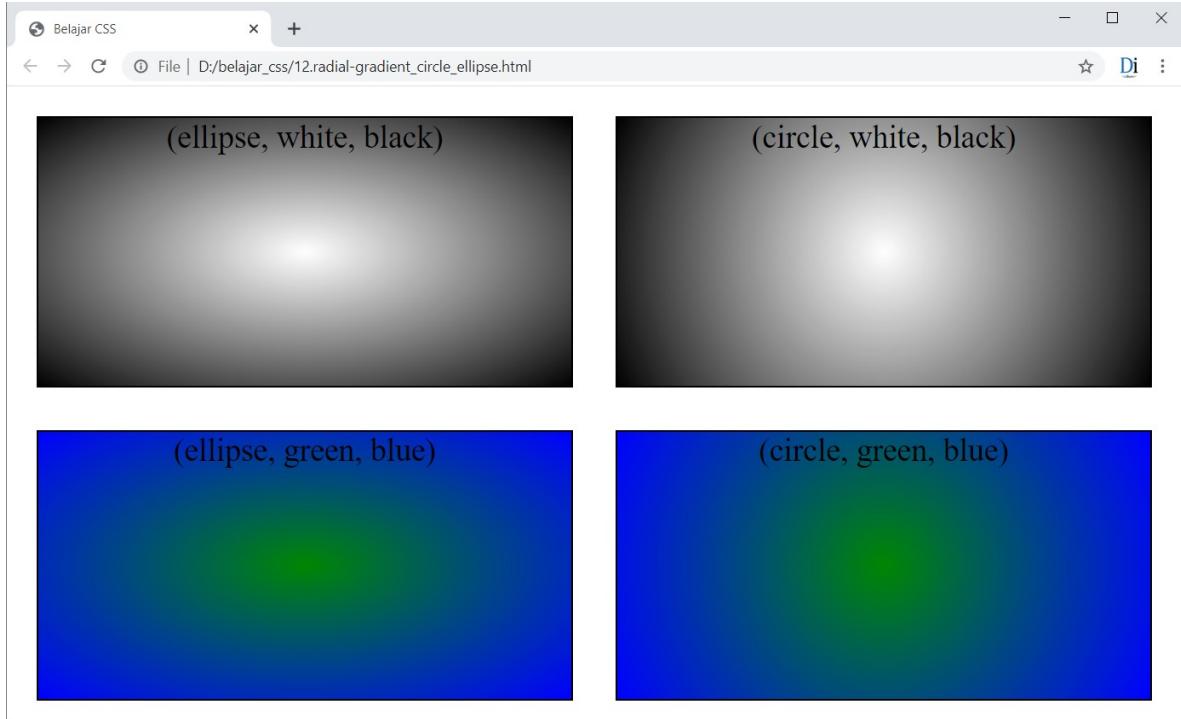
Ellips (`ellipse`) adalah bentuk default dari property radial-gradient. Bentuk lain adalah lingkaran (`circle`) yang ditulis sebagai berikut:

```
background-image: radial-gradient(circle, white, black);
```

Kode di bawah ini memperlihatkan perbedaan antara gradient ellipse dan gradient circle:

12.radial-gradient_circle_ellipse.html

```
1 .satu {
2   background-image: radial-gradient(ellipse, white, black);
3 }
4 .dua {
5   background-image: radial-gradient(circle, white, black);
6 }
7 .tiga {
8   background-image: radial-gradient(ellipse, green, blue);
9 }
10 .empat {
11   background-image: radial-gradient(circle, green, blue);
12 }
```



Gambar: Contoh perbedaan ellipse radial-gradient dengan circle radial-gradient

Radial Gradient Position

Secara default, titik lingkaran gradient berada tepat di tengah element. Kita bisa mengatur posisi titik ini dengan menggunakan keyword dan satuan length.

Format penulisannya menggunakan keyword 'at'. Sebagai contoh, untuk membuat titik pusat radial gradient di sudut kanan atas, bisa ditulis sebagai berikut:

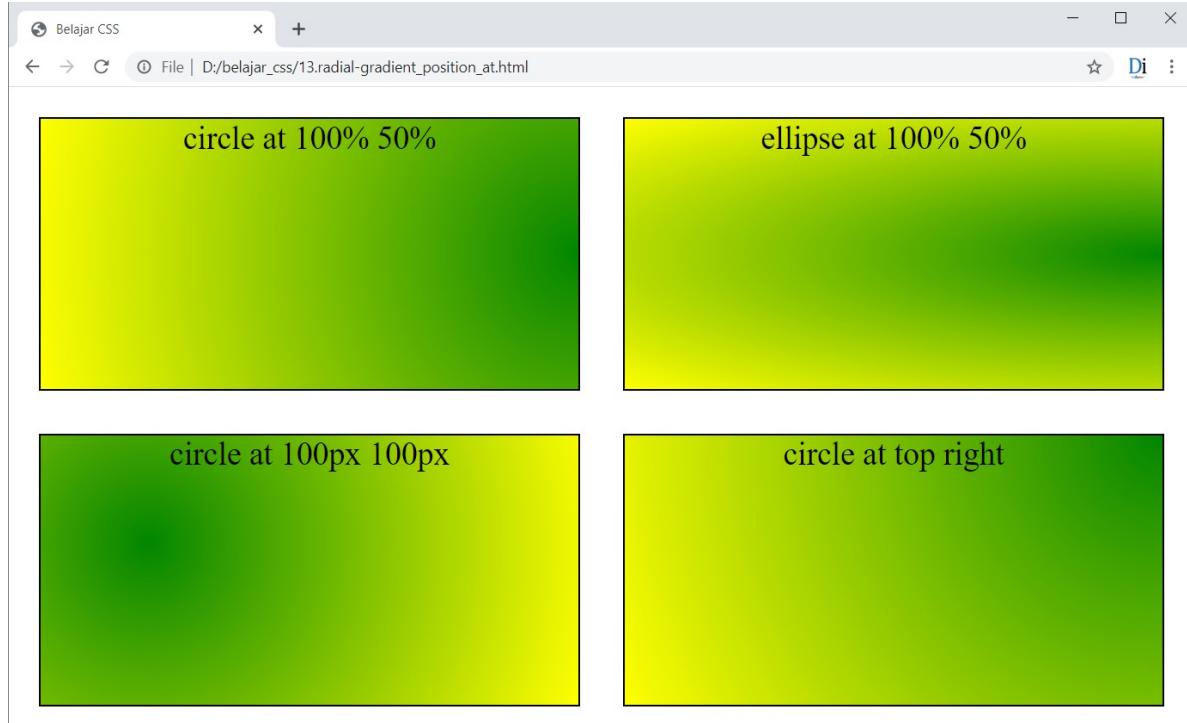
```
background-image: radial-gradient(circle at top right, green, yellow);
```

Nilai yang bisa diinput sama seperti nilai untuk property background-position, yakni 'top', 'top right', 'bottom left', serta bisa juga dalam bentuk nilai length pixel dan persen.

[13.radial-gradient_position_at.html](#)

```

1 .satu {
2   background-image: radial-gradient(circle at 100% 50%, green, yellow);
3 }
4 .dua {
5   background-image: radial-gradient(ellipse at 100% 50%, green, yellow);
6 }
7 .tiga {
8   background-image: radial-gradient(circle at 100px 100px, green, yellow);
9 }
10 .empat {
11   background-image: radial-gradient(circle at top right, green, yellow);
12 }
```



Gambar: Contoh penggunaan radial gradient position untuk menentukan titik pusat gradient

Radial Gradient Extends

Radial gradients extends adalah istilah yang merujuk ke seberapa jauh batas lingkaran gradient. Sebagai contoh, untuk membuat radial gradient yang berjarak 200 pixel dari titik pusat, bisa menggunakan property berikut:

```
background-image: radial-gradient(circle 200px, black, yellow);
```

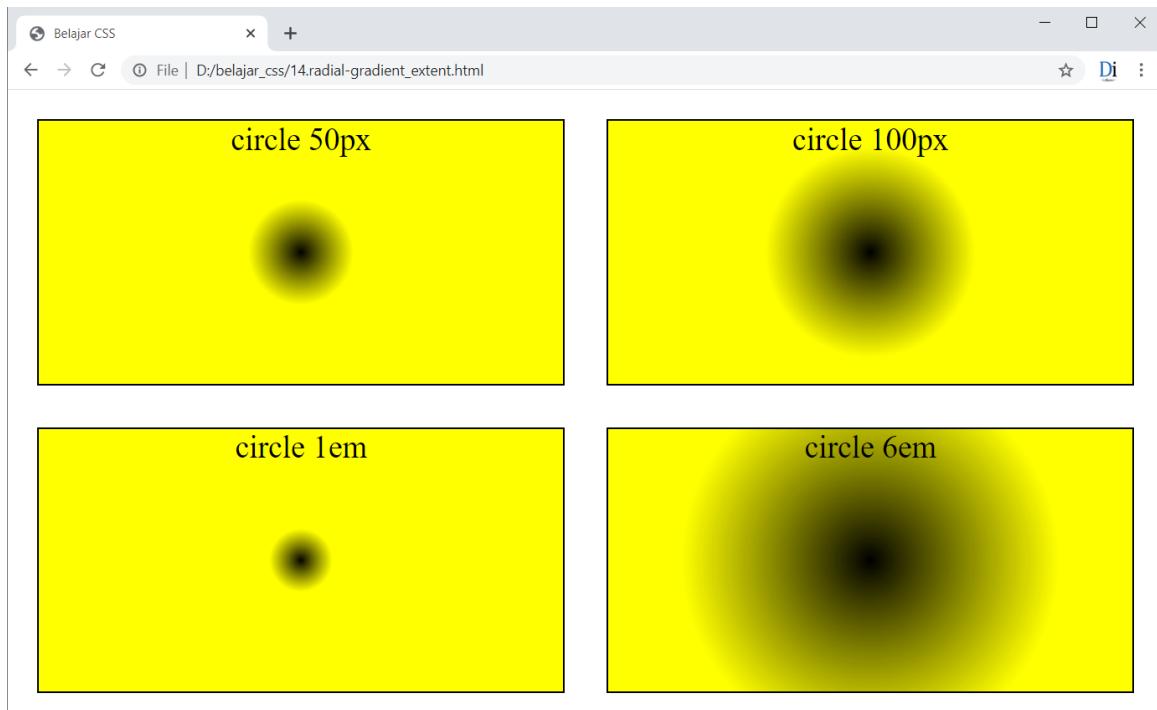
Property ini akan menghasilkan gradient warna hitam yang melebar hingga 200px dari titik pusat lingkaran, kemudian baru beralih menjadi kuning.

Berikut contoh penggunaan berbagai nilai radial gradients extends:

14.radial-gradient_extent.html

```

1 .satu {
2   background-image: radial-gradient(circle 50px, black, yellow);
3 }
4 .dua {
5   background-image: radial-gradient(circle 100px, black, yellow);
6 }
7 .tiga {
8   background-image: radial-gradient(circle 1em, black, yellow);
9 }
10 .empat {
11   background-image: radial-gradient(circle 6em, black, yellow);
12 }
```



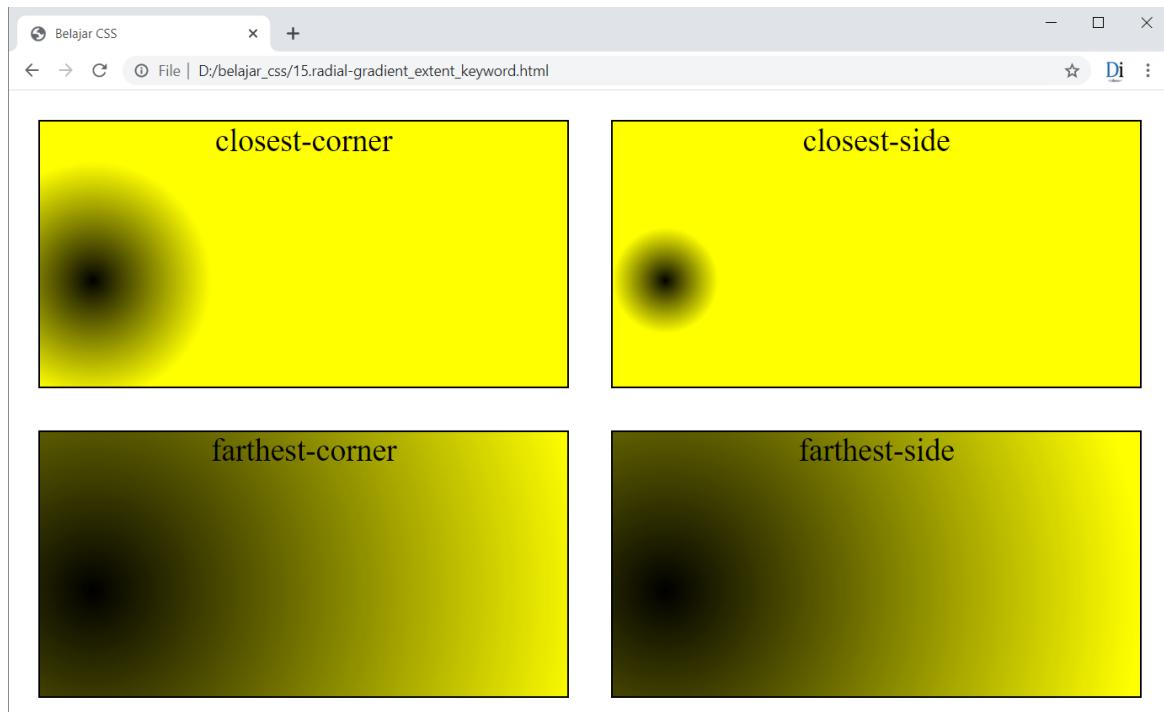
Gambar: Contoh penggunaan radial gradient extends

Selain ukuran panjang, kita juga bisa menentukan lebar atau batas lingkaran gradient berdasarkan sisi tertentu dari parent element. Nilai yang saat ini didukung adalah: `closest-corner`, `closest-side`, `farthest-corner` (default), dan `farthest-side`. Langsung saja kita bahas dengan contoh:

15.radial-gradient_extent_keyword.html

```

1  .satu {
2    background-image: radial-gradient(closest-corner circle at 10% 60%,
3                                     black, yellow);
4  }
5  .dua {
6    background-image: radial-gradient(closest-side circle at 10% 60%,
7                                     black, yellow);
8  }
9  .tiga {
10   background-image: radial-gradient(farthest-corner circle at 10% 60%,
11                                     black, yellow);
12 }
13 .empat {
14   background-image: radial-gradient(farthest-side circle at 10% 60%,
15                                     black, yellow);
16 }
```



Gambar: Contoh penggunaan radial gradient extends dengan keyword

Seperti yang terlihat, nilai `closest-corner` akan membuat gradient melebar hingga sisi sudut (corner) terdekat. Pada contoh di atas, sudut terdekat adalah di kiri bawah.

Nilai `closest-side` akan membuat gradient melebar hingga sisi border terdekat, dimana dalam contoh kita adalah sisi kiri box.

Nilai `farthest-corner` akan melebarkan efek gradient hingga sudut terjauh, yang dalam contoh di atas adalah sudut kanan atas.

Serta nilai `farthest-side` akan menampilkan efek gradient hingga sisi terjauh. Dalam contoh di atas adalah border sisi kanan.

Radial Gradient Color Stop

Sama seperti linear gradient, kita juga bisa menentukan kapan sebuah warna berhenti dan kapan bergabung dengan warna lain. Konsep **radial gradient color stop** sangat mirip seperti linear gradient. Berikut contoh penggunaannya:

16. radial-gradient_extent_color_stop.html

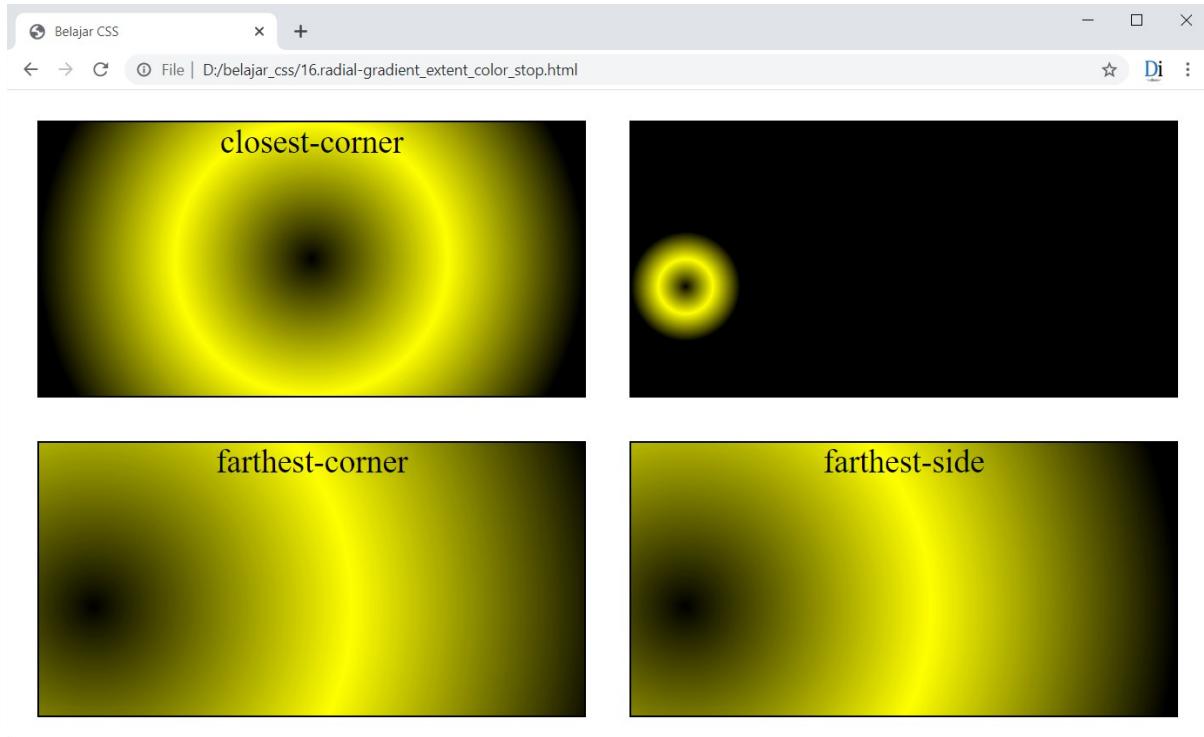
```

1 .satu {
2   background-image: radial-gradient(farthest-side circle,
3   black, yellow, black);
4 }
5 .dua {
6   background-image: radial-gradient(closest-side circle at 10% 60%,
7   black, yellow, black);
8 }
```

```

9 .tiga {
10  background-image: radial-gradient(farthest-corner circle at 10% 60%,
11    black, yellow, black);
12 }
13 .empat {
14  background-image: radial-gradient(farthest-side circle at 10% 60%,
15    black, yellow, black);
16 }

```



Gambar: Contoh penggunaan radial gradient color stop

Pada contoh ini saya menggabung efek radial gradient extends dengan color stop. Hasilnya cukup mengagumkan.

Repeating Radial Gradients

Efek terakhir yang kita pelajari dari radial gradients dan juga paling rumit, adalah menggunakan fungsi **repeating-radial-gradient**. Ini bisa dipakai untuk mengulang warna gradient seperti contoh berikut:

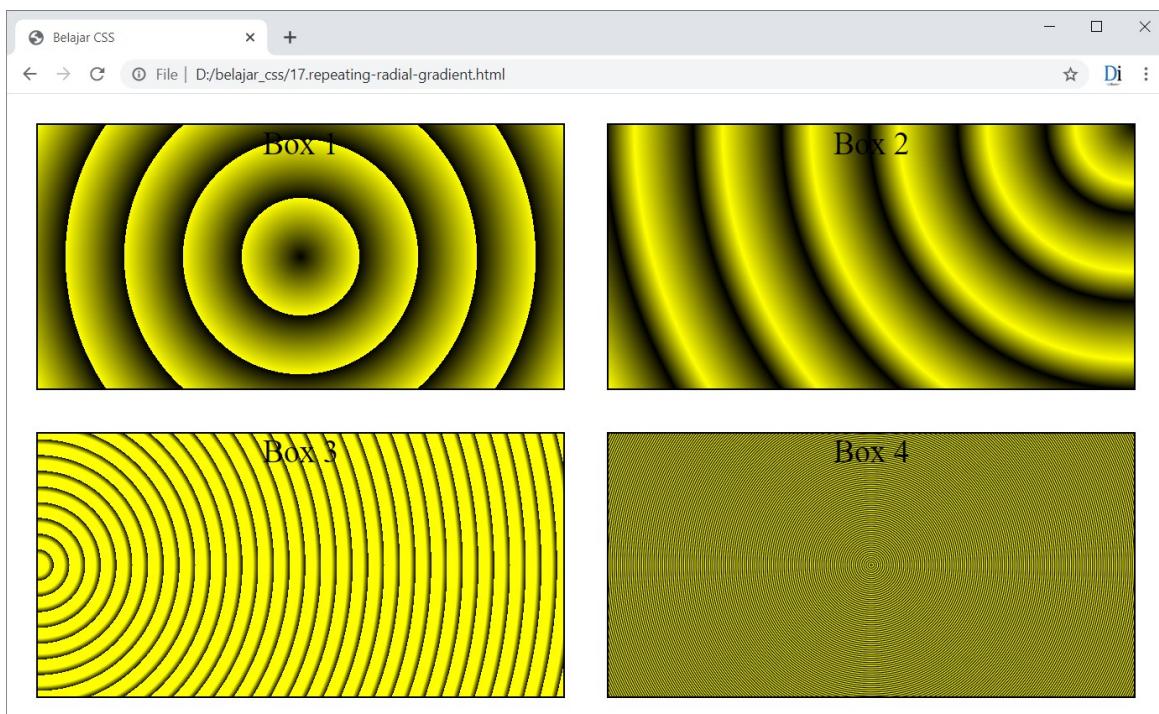
18.repeating-radial-gradient.html

```

1 .satu {
2   background-image: repeating-radial-gradient(circle, black, yellow 20%);
3 }
4 .dua {
5   background-image: repeating-radial-gradient(circle farthest-corner
6     at right top, black, yellow 10%, black 15%);
7 }
8 .tiga {

```

```
9  background-image: repeating-radial-gradient(circle farthest-corner  
10 at left, yellow, yellow 10px, black 15px); }  
11 .empat {  
12   background-image: repeating-radial-gradient(circle farthest-corner,  
13   yellow, black 1px, yellow 2px);  
14 }
```



Gambar: Contoh penggunaan repeating radial gradients

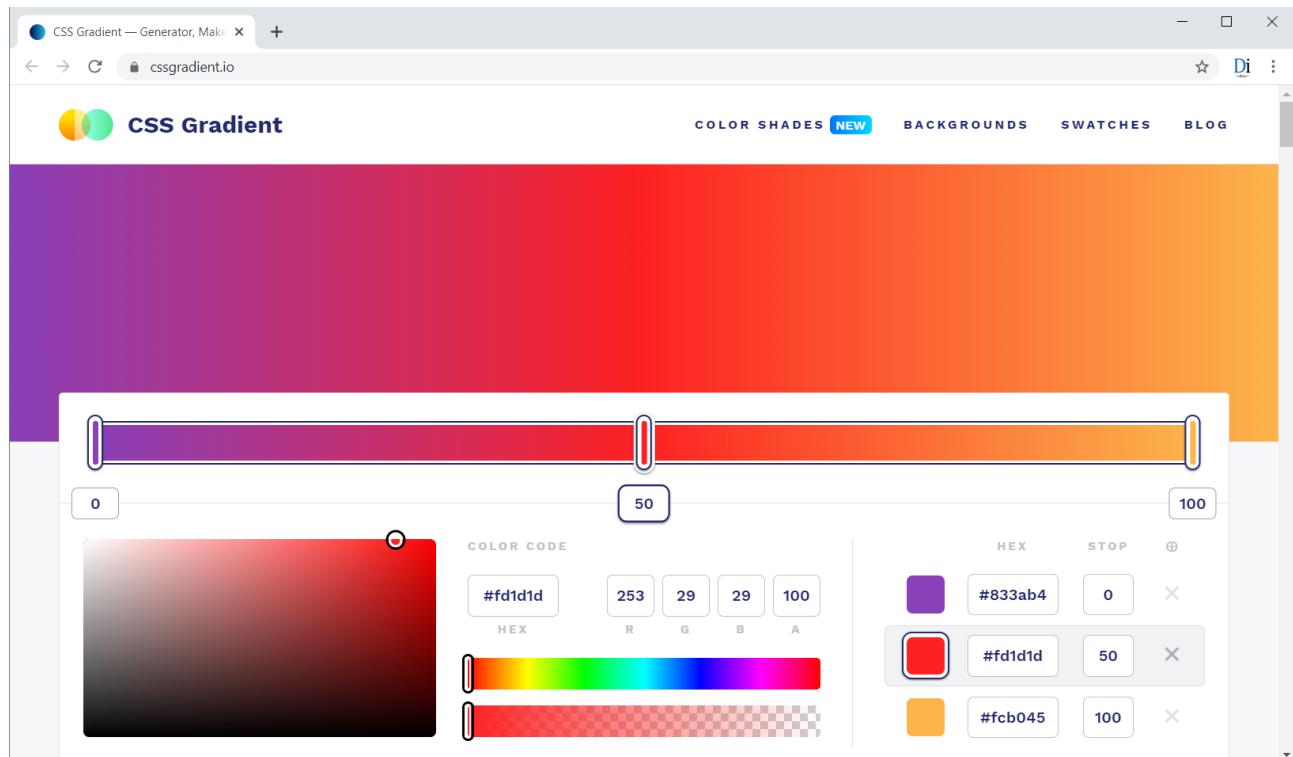
Prinsip dari repeating-radial-gradient sama dengan repeating-linear-gradient, dimana gradient akan terus di ulang hingga menutupi seluruh element.

11.5. CSS Gradient Generator

Seperti yang telah kita pelajari dalam bab ini, perintah untuk efek gradient cukup rumit, apalagi jika ingin mencampur berbagai warna. Untungnya, terdapat berbagai tools online yang bisa dipakai untuk men-generate CSS gradient.

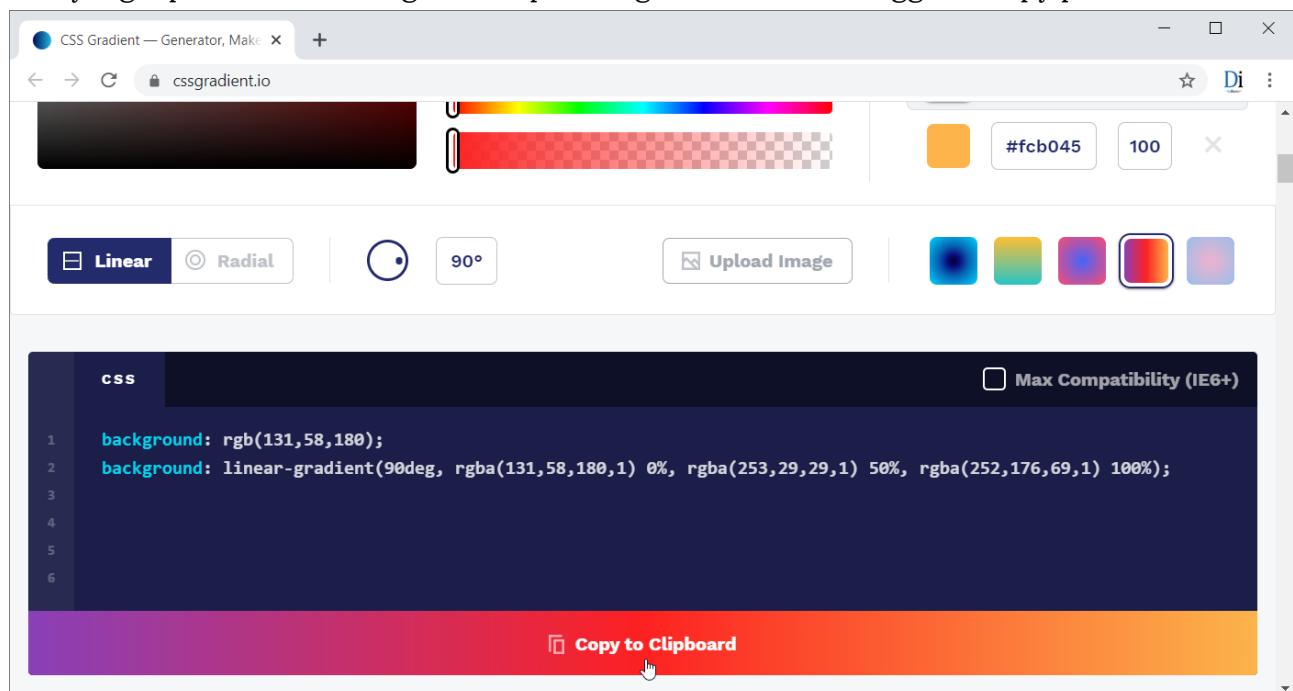
Salah satunya bisa ke <https://cssgradient.io>:

CSS3 Gradient



Gambar: Tampilan website <https://cssgradient.io>

Sesuai dengan nama website, kita bisa membuat gradien melalui tombol dan yang ada. Kode CSS yang diperlukan akan di generate pada bagian bawah dan tinggal di copy paste:



Gambar: Kode CSS yang di generate oleh <https://cssgradient.io>

Selain itu juga ada beberapa website lain yang juga menyediakan layanan untuk men-generate gradient, misalnya:

- ✓ mycolor.space/gradient
- ✓ www.css-gradient.com
- ✓ www.colorzilla.com/gradient-editor

Selain menggenerate kode CSS3, tools ini juga menyediakan kode tambahan vendor prefix untuk web browser lawas seperti `-moz-linear-gradient` atau `-webkit-linear-gradient`. Namun karena penggunaan web browser ini sudah mulai berkurang, lebih praktis jika ditulis tanpa vendor prefix.

Efek yang dihasilkan CSS3 gradient memang sangat menarik. Namun karena mayoritas konten utama web adalah teks, variasi warna-warni gradient tetap harus dibatasi. Jangan sampai warna gradient ini terlalu "mencolok" yang berakibat negatif kepada sisi *user experience* dari website kita.

Lanjut, kita akan bahas tentang **CSS3 Multiple Column**, yakni property CSS yang bisa dipakai untuk membuat tampilan teks seperti di koran.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

12. CSS3 Multiple Column

Salah satu konsep *user experience (UX)* yang kadang sering luput dari perhatian adalah terkait panjang baris text. Baris yang terlalu panjang membuat teks susah dibaca. Karena alasan itu pula majalah atau koran memecah tulisan menjadi kolom kecil yang memanjang ke bawah. Tujuannya supaya mata pembaca lebih mudah mengikuti alur teks yang ada.

Sebelum CSS3, tampilan layout seperti koran lumayan sulit untuk dibuat. Salah satu cara bisa dengan tabel HTML. Tapi di HTML, isi teks dari satu sel tabel tidak bisa pindah ke kolom sebelah, sehingga sangat tidak praktis untuk dibuat.

Atas dasar inilah CSS3 menghadirkan konsep baru bernama *multiple column*. **CSS3 multiple column** bisa dipakai untuk membuat efek kolom yang mirip di dalam koran atau majalah. Kita tinggal menentukan jumlah kolom dan web browser secara otomatis membagi teks yang ada ke masing-masing kolom.

12.1. Property column-count

Property **column-count** bisa jadi merupakan satu-satunya property yang diperlukan untuk membagi kolom teks. Property ini berfungsi untuk menentukan jumlah kolom teks. Nilai yang bisa diisi berupa angka seperti 2, 3 atau 4. Web browser kemudian akan membagi teks secara otomatis.

Sebagai contoh, jika saya ingin membuat teks dengan 3 kolom, bisa menggunakan kode berikut:

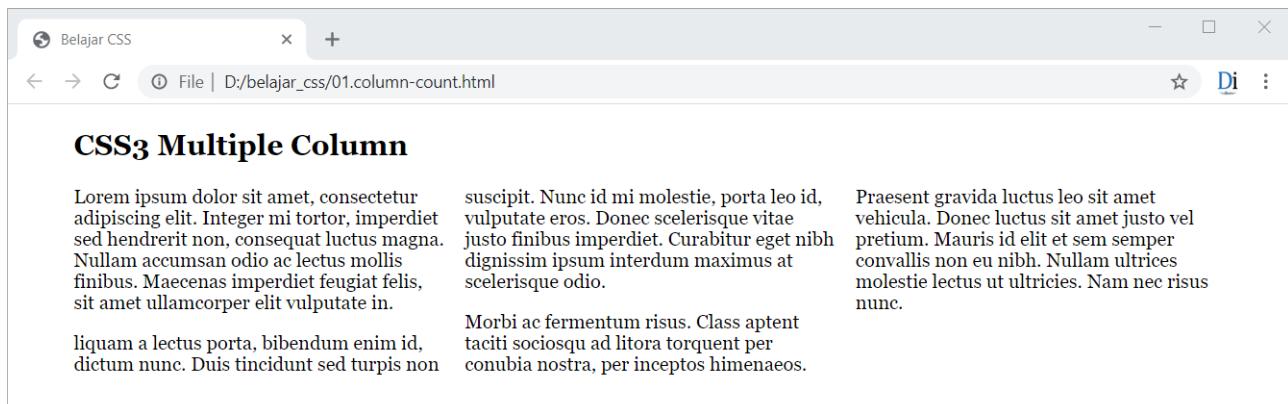
01.column-count.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{
8              width: 960px;
9              margin: 0 auto;
10             font-family: Georgia, "Times new Roman", serif;
11         }
12         .content {
13             column-count: 3;
14     }
```

```

15      p {
16          margin-top: 0;
17      }
18  </style>
19 </head>
20 <body>
21     <h2>CSS3 Multiple Column</h2>
22     <div class="content">
23         <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
24             tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
25             accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
26             sit amet ullamcorper elit vulputate in.
27         </p>
28         <p>liquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
29             sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate
30             eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget
31             nibh dignissim ipsum interdum maximus at scelerisque odio.</p>
32         <p>
33             Morbi ac fermentum risus. Class aptent taciti sociosqu ad litora
34                 torquent per conubia nostra, per inceptos himenaeos. Praesent gravida
35                 luctus leo sit amet vehicula. Donec luctus sit amet justo vel pretium.
36                 Mauris id elit et sem semper convallis non eu nibh. Nullam ultrices
37                 molestie lectus ut ultricies. Nam nec risus nunc.</p>
38     </div>
39 </body>
40 </html>

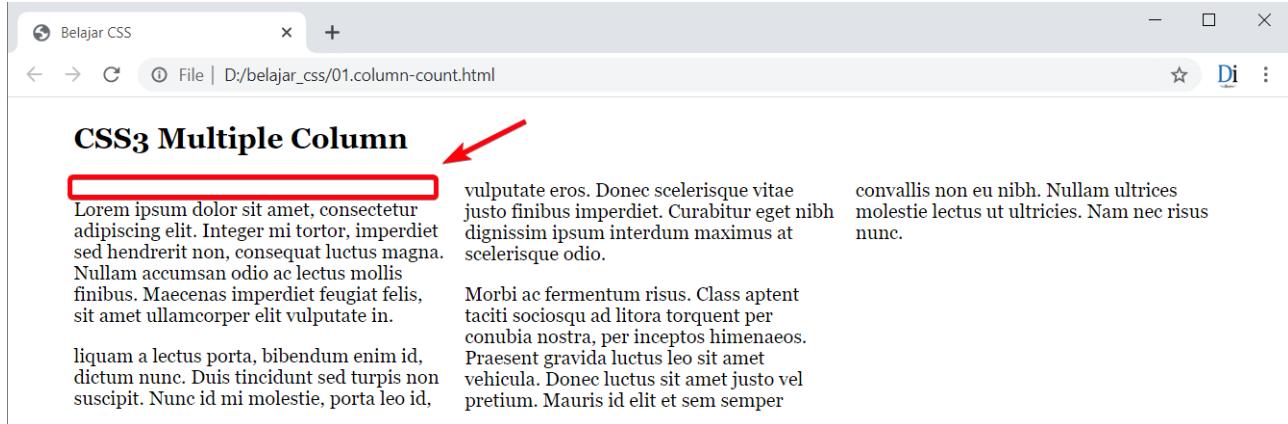
```



Tampilan dari property column-count: 3

Hasilnya, teks otomatis dibagi menjadi 3 kolom. Perhatikan juga bahwa selain dibagi ke dalam 3 kolom, web browser juga menyesuaikan tinggi kolom agar seragam.

Selector `p { margin-top: 0 }` saya tambah agar margin dari paragraf pertama tidak mendorong teks ke bawah. Jika ini tidak ditambah, hasilnya menjadi sebagai berikut:



Gambar: Paragraf pertama akan terdorong sedikit ke bawah

12.2. Property column-width

Selain mengatur pembagian berdasarkan jumlah kolom, kita juga bisa mengatur jumlah kolom berdasarkan lebar kolom yang diinginkan dari property `column-width`.

Sebagai contoh, apabila lebar class `.content` 960 pixel. Maka apabila property `column-width` diisi nilai 200px, maka bisa menampung $960/200 = 4,8$ kolom. Web browser akan membulatkan hasil ini ke bawah menjadi 4 kolom:

02.column-width.html

```
1 .content {
2   column-width: 200px;
3 }
```



Tampilan dari property column-width: 200px

Dengan mengatur lebar `column-width`, kita bisa membuat tampilan yang lebih fleksibel.

12.3. Property column-fill

Jika diperhatikan, secara default web browser membagi rata jumlah teks ke setiap kolom.

Hasilnya, masing-masing kolom memiliki tinggi baris yang sama. Property `column-fill` bisa dipakai untuk mengatur efek ini. Nilai yang bisa diisi adalah keyword `balance` atau `auto`.

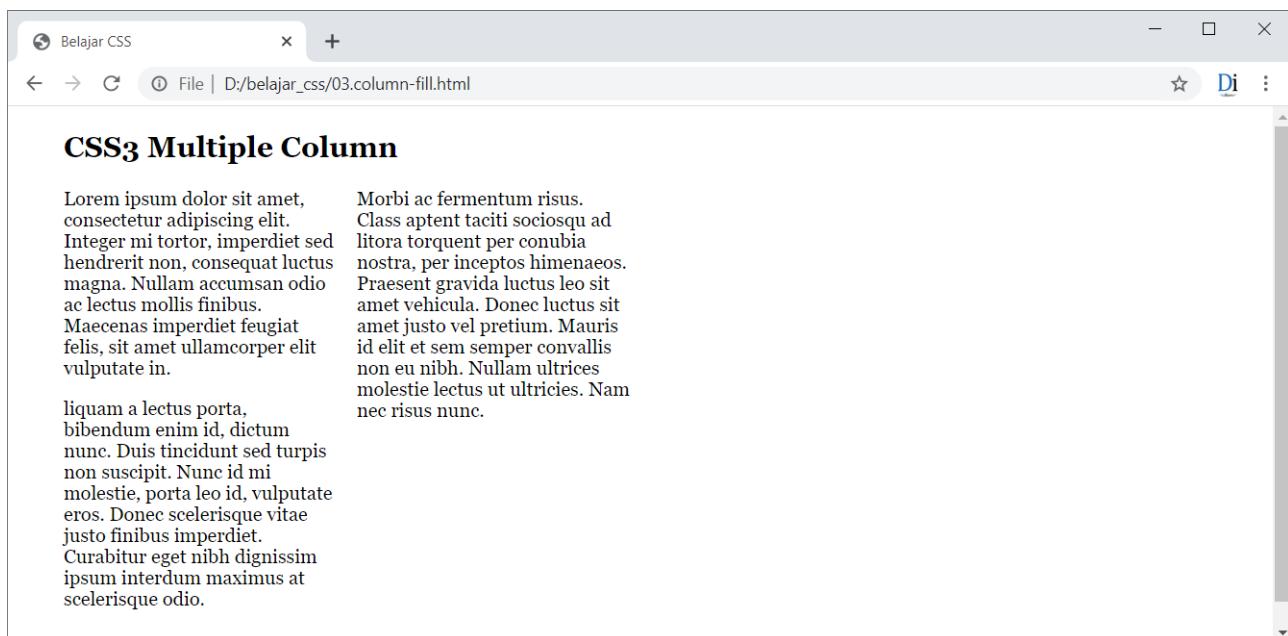
Property `column-fill:balance` akan membagi teks sama rata ke dalam setiap kolom, dan ini merupakan nilai default.

Sedangkan property `column-fill:auto` akan memaksa web browser untuk mengisi kolom pertama terlebih dahulu, baru kemudian pindah ke kolom berikutnya apabila sudah mencapai tinggi maksimal. Supaya efek ini bisa diterapkan, kita harus men-set tinggi parent element.

Berikut contoh penggunaan dari property `column-fill`:

03.column-fill.html

```
1 .content {  
2   height: 400px;  
3   column-width: 200px;  
4   column-fill: auto;  
5 }
```



Tampilan dari property `column-fill: auto`

Kali ini saya men-set tinggi class `.content` menjadi 400 pixel serta menambahkan `column-fill:auto`. Hasilnya, teks akan mengisi penuh kolom pertama terlebih dahulu hingga 400 pixel. Kemudian jika terdapat sisa teks, baru mengambil tempat di kolom yang sebelahnya. Demikian seterusnya hingga seluruh kolom terisi.

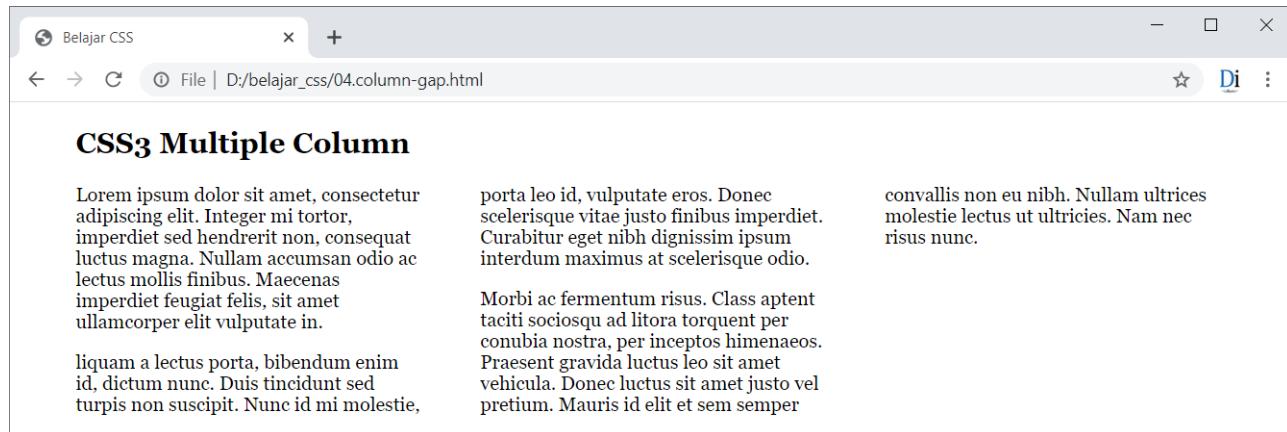
Apabila nilai property `column-fill` diganti menjadi `balanced`, teks akan kembali di sebar merata ke setiap kolom.

12.4. Property column-gap

Property `column-gap` berguna untuk mengatur besar jarak (gap) antar kolom. Nilai yang bisa diisi adalah satuan length seperti pixel. Sebagai contoh, dalam kode berikut saya membuat jarak spasi antar kolom sebesar 50 pixel:

04.column-gap.html

```
1 .content {
2   column-count: 3;
3   column-gap: 50px;
4 }
```



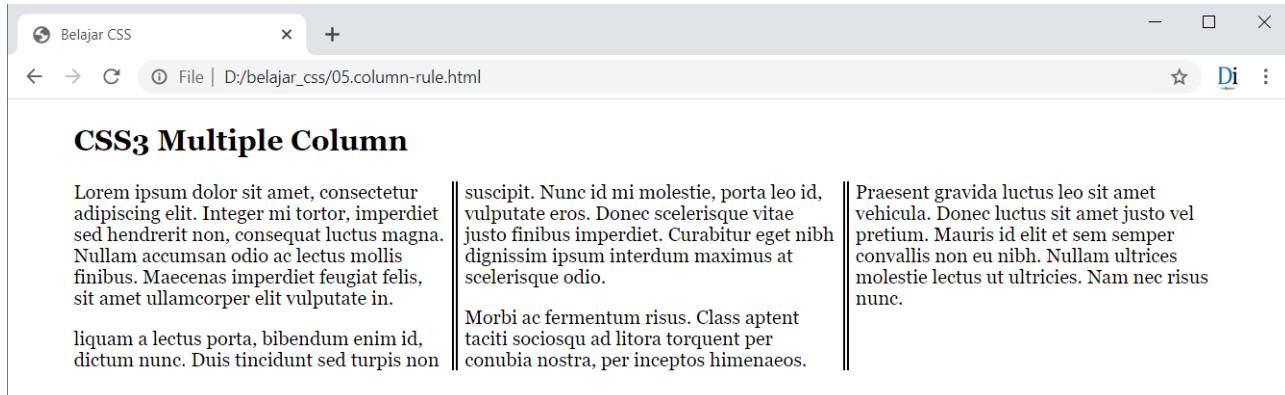
Tampilan dari property `column-gap: 50px`

12.5. Property column-rule

Daripada menggunakan spasi kosong antara kolom, kita juga bisa menambah garis. Inilah fungsi dari property `column-rule`. Nilai yang harus ditulis sama seperti property `border`, dimana kita harus menentukan lebar garis, jenis garis, dan warna garis.

05.column-rule.html

```
1 .content {
2   column-count: 3;
3   column-rule: 5px double black;
4 }
```



Tampilan dari property column-rule: 5px double black

Property `column-rule: 5px double black` akan menghasilkan garis setebal 5 pixel, berjenis double dan berwarna hitam pada spasi antar kolom.

12.6. Property column-span

Property `column-span` berguna untuk "keluar" dari efek kolom. Ini mirip seperti atribut `colspan` di tabel HTML. Property ini bisa diisi dengan salah satu dari 2 nilai: `none` atau `all`. Nilai default dari `column-span` adalah `none`, yang artinya tidak ada efek span sama sekali. Berikut contoh dari penggunaan nilai `column-span: all`:

06.column-span.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{
8              width: 960px;
9              margin: 30px auto;
10             font-family: Georgia, "Times new Roman", serif;
11         }
12         .content {
13             column-count: 3;
14         }
15         .span {
16             column-span: all;
17             background-color: aqua;
18         }
19         p {
20             margin-top: 0;
21         }
22     </style>
23 </head>
24 <body>
25     <div class="content">
```

```

26 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
27 tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
28 accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
29 sit amet ullamcorper elit vulputate in.
30 </p>
31 <p>liquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
32 sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate
33 eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget
34 nibh dignissim ipsum interdum maximus at scelerisque odio.</p>
35 <h2 class="span">CSS3 Multiple Column</h2>
36 <p>
37 Morbi ac fermentum risus. Class aptent taciti sociosqu ad litora
38 torquent per conubia nostra, per inceptos himenaeos. Praesent gravida
39 luctus leo sit amet vehicula. Donec luctus sit amet justo vel pretium.
40 Mauris id elit et sem semper convallis non eu nibh. Nullam ultrices
41 molestie lectus ut ultricies. Nam nec risus nunc.</p>
42 </div>
43 </body>
44 </html>
```



Tampilan dari property column-span: all

Terlihat judul header "CSS3 Multiple Column" melebar (span) hingga ke seluruh kolom.

Berbagai property CSS3 multiple column yang kita bahas di sini bisa disatukan untuk mendapatkan efek yang lebih "pas" seperti contoh berikut:

07.column-final.html

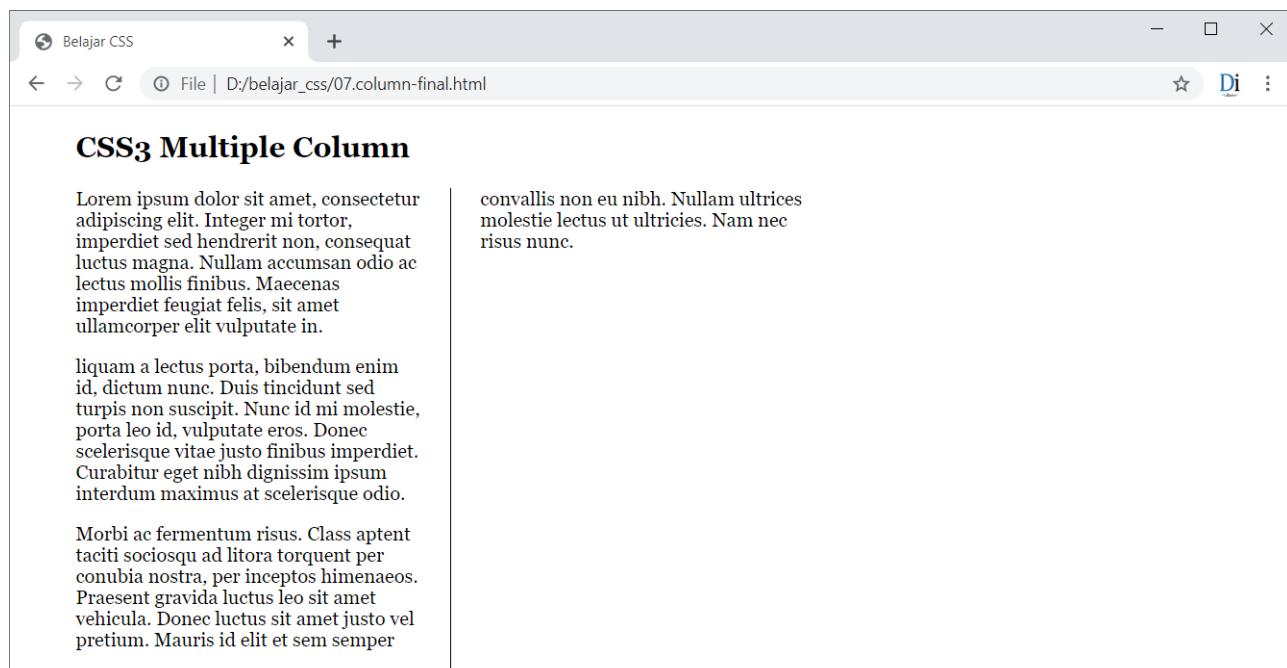
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          body{
8              width: 960px;
9              margin: 0 auto;
10             font-family: Georgia, "Times new Roman", serif;
11         }
12         .content {
```

```

13     height: 400px;
14     column-count: 3;
15     column-gap: 50px;
16     column-rule: 1px solid black;
17     column-fill: auto;
18   }
19   p {
20     margin-top: 0;
21   }
22 </style>
23 </head>
24 <body>
25   <h2 class="span">CSS3 Multiple Column</h2>
26   <div class="content">
27     <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer mi
28     tortor, imperdiet sed hendrerit non, consequat luctus magna. Nullam
29     accumsan odio ac lectus mollis finibus. Maecenas imperdiet feugiat felis,
30     sit amet ullamcorper elit vulputate in.
31   </p>
32   <p>liquam a lectus porta, bibendum enim id, dictum nunc. Duis tincidunt
33     sed turpis non suscipit. Nunc id mi molestie, porta leo id, vulputate
34     eros. Donec scelerisque vitae justo finibus imperdiet. Curabitur eget
35     nibh dignissim ipsum interdum maximus at scelerisque odio.</p>
36   <p>
37     Morbi ac fermentum risus. Class aptent taciti sociosqu ad litora
38     torquent per conubia nostra, per inceptos himenaeos. Praesent gravida
39     luctus leo sit amet vehicula. Donec luctus sit amet justo vel pretium.
40     Mauris id elit et sem semper convallis non eu nibh. Nullam ultrices
41     molestie lectus ut ultricies. Nam nec risus nunc.</p>
42 </div>
43 </body>
44 </html>

```



Hasil penggunaan berbagai property CSS3 multiple column

Dalam kode di atas saya menggunakan 4 property multiple column sekaligus, yakni: `column-count`, `column-gap`, `column-rule` dan `column-fill`.

Efek yang dihasilkan dari CSS3 multiple column memang sangat menarik, tetapi dari segi user experience hanya cocok untuk teks yang pas tampil satu layar. Jika panjang teks butuh scroll ke bawah, maka akan cukup repot bagi user untuk scroll naik-turun untuk pindah ke kolom berikutnya.

Dalam bab setelah ini kita akan masuk ke materi tentang **CSS3 Border Image**.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

13. CSS3 Border Image

Sejak awal kemunculan CSS, menambah bingkai element menjadi salah satu efek yang paling sering dipakai. Property `border` yang kita pelajari pada bab box model sudah menyediakan kebutuhan dasar. Namun agar lebih lengkap lagi, CSS3 menghadirkan **Border Image**.

13.1. CSS3 Border Image

Fitur **CSS3 border image** memungkinkan kita membuat bingkai dari gambar. Sebelum property ini hadir, efek yang sama harus dibuat dengan trik yang cukup rumit. Misalnya menempatkan sebuah gambar di background, kemudian menggeser element dengan property `positioning`.

Sama seperti property `border`, CSS3 border image bisa ditulis dengan penulisan singkat (*shorthand notation*) maupun dengan penulisan panjang (*longhand notation*). Kita akan bahas versi longhand notation terlebih dahulu, baru kemudian masuk bentuk shorthand notation.

Secara konsep, CSS3 border image tetap butuh property `border` sebagai tempat untuk gambar, akan tetapi warna dari border tersebut harus set sebagai `transparent`. Berikut contoh kode yang dimaksud:

```
.satu {  
    width: 200px;  
    height: 200px;  
    margin: 10px auto;  
    background-color: #57ac5e;  
    border: 30px solid transparent;  
}
```

Di sini saya membuat sebuah element dengan tinggi 200 pixel dan lebar 200 pixel, kemudian warna background di set sebagai `#57ac5e`, yakni warna hijau tua. Margin di set sebagai `10px auto` agar element akan memiliki margin 10px di sisi atas dan bawah serta berada di tengah-tengah parent element-nya.

Property `border: 30px solid transparent` berarti element ini akan memiliki border sebesar 30px di semua sisi, solid tapi "tanpa warna" atau transparan. Jika kode di atas dijalankan, warna border akan sama seperti warna background element (`#57ac5e`). Di ruang sebesar 30px inilah bingkai gambar dari CSS3 border image akan tampil.

13.2. Property border-image-source

Property **border-image-source** berfungsi untuk menentukan lokasi gambar border. Nilai yang dibutuhkan berupa alamat path gambar dalam format `url('alamat\ke\gambar')`. Sebagai contoh, jika saya ingin memakai `border1.png` yang tersimpan di dalam folder `gambar`, maka penulisan property-nya adalah:

```
border-image-source: url('gambar/border1.png');
```

Gambar `border1.png` ini sudah saya siapkan dengan tampilan sebagai berikut:



Gambar: `border1.png`

Seluruh gambar dalam bab ini juga tersedia di folder `belajar_css\bab_13_css3_border_image\gambar\`.

Berikut contoh praktik dari penggunaan property `border-image-source`:

01.border-image-source.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              width: 200px;
9              height: 200px;
10             background-color: #57ac5e;
11             margin: 10px auto;
12             border: 30px solid transparent;
13             border-image-source: url('gambar/border1.png');
14         }
15     </style>
16 </head>
17 <body>
18     <div class="satu"></div>
19 </body>
20 </html>
```



Gambar: Border image?

Gambar border sudah tampil, tapi hanya di bagian sudut saja. Kenapa bisa seperti ini? Karena CSS3 border image ternyata masih butuh perhitungan lanjutan (akan kita bahas sesaat lagi).

Tapi setidaknya dari tampilan di atas sudah terlihat efek dari property `border-image-source`, dimana gambar border akan muncul pada 4 sudut element. Lebar dari gambar adalah 30 pixel, sesuai dengan property `border: 30px solid transparent`.

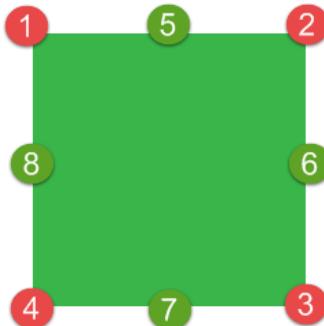
13.3. Property `border-image-slice`

Property `border-image-slice` berfungsi untuk memotong (*slicing*) gambar yang akan dijadikan border. Pembahasannya mungkin sedikit rumit, tapi semoga bisa dipahami.

Property `border-image-slice` akan membagi gambar menjadi 9 bagian. Bagian-bagian inilah yang nantinya dipakai sebagai border untuk setiap sudut serta untuk setiap sisi element.

Dalam sebuah element HTML (box model), terdapat 4 sudut dan 4 sisi. 4 sudut ini terdiri dari: sudut *top left*, *top right*, *bottom right* dan *bottom left*. Serta 4 sisi: *top*, *right*, *bottom* dan *left*.

Total terdapat 8 bagian yang diilustrasikan dalam gambar berikut:



Gambar: Posisi 4 sudut + 4 sisi element

Property `border-image-slice` akan memotong gambar agar bisa menempati ke-8 posisi ini. Sebagai contoh, jika saya menggunakan property:

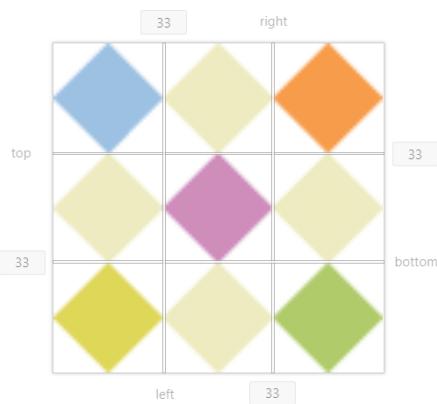
```
border-image-slice: 33 33 33 33;
```

Maka gambar border akan dipotong dari setiap sisi dengan jarak 33 pixel. Yang perlu diperhatikan, nilai di atas ditulis tanpa satuan "px". Alasannya untuk mengakomodasi berbagai jenis format gambar.

Jika menggunakan gambar berbasis pixel seperti *.jpg, *.giv atau *.png, maka satuan yang dipakai adalah pixel, namun jika menggunakan format gambar *.svg, satuan yang dipakai adalah titik koordinat. Nilai lain yang bisa kita gunakan adalah dalam satuan persen, seperti contoh berikut:

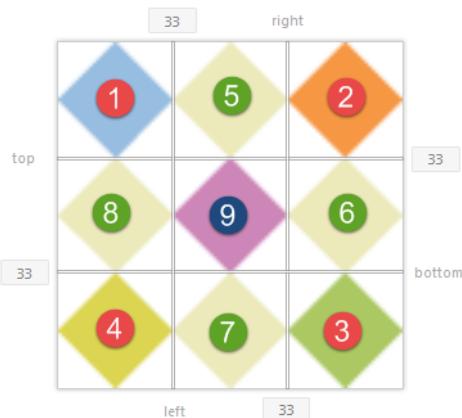
```
border-image-slice: 33% 33% 33% 33%;
```

Kembali ke contoh property `border-image-slice`, berikut "pemotongan" yang dilakukan ke gambar `border1.png`:



Ilustrasi pemotongan gambar "border1.png" oleh property `border-image-slice: 33 33 33 33`

Gambar `border1.png` berukuran 100×100 pixel, sehingga garis yang ada seolah-olah membagi pas menjadi 9 bagian. Setiap bagian dari pemotongan ini akan dipakai untuk ke-8 sisi gambar (kecuali bagian tengah):



Gambar: Pembagian sisi gambar `border1.png`

Berikut tampilan dari penggunaan property border-image-slice: 33 33 33 33:

02.border-image-slice.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              width: 200px;
9              height: 200px;
10             background-color: #57ac5e;
11             margin: 10px auto;
12             border: 33px solid transparent;
13
14             border-image-source: url('gambar/border1.png');
15             border-image-slice: 33 33 33 33;
16         }
17     </style>
18 </head>
19 <body>
20     <div class="satu"></div>
21 </body>
22 </html>
```



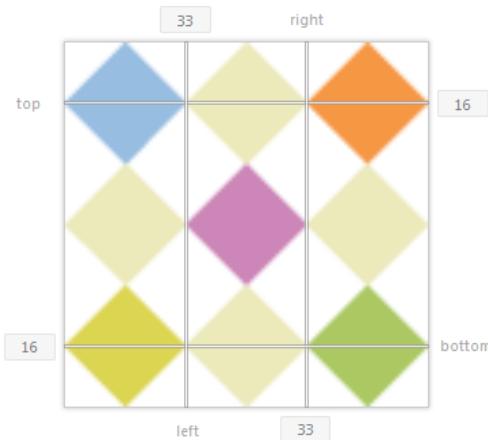
Hasil penggunaan property border-image-slice

Sekarang gambar border sudah tampil di semua sudut dan sisi box. Khusus untuk bagian sisi top, right, bottom dan left tampak sedikit berbeda. Ini terjadi karena secara default web browser "meregangkan" (stretch) gambar di ke-4 sisi border. Nantinya kita bisa mengatur efek ini lewat property border-image-repeat yang akan dibahas sesaat lagi.

Kembali ke cara penulisan property border-image-slice, saya menulis angka "33" sebanyak 4 kali. Sebenarnya, empat angka ini juga menggunakan konsep TrouBLE, yang artinya kita bisa

CSS3 Border Image

atur posisi pemotongan. Sebagai contoh, jika saya ubah jadi `border-image-slice: 16 33 16 33`, maka pemotongan gambarnya menjadi sebagai berikut:



Ilustrasi pemotongan gambar dengan property border-image-slice: 16 33 16 33

Berikut hasilnya:

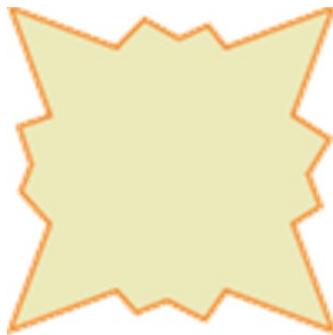
03.border-image-slice2.html

```
1 .satu {  
2   width: 200px;  
3   height: 200px;  
4   background-color: #57ac5e;  
5   margin: 10px auto;  
6   border: 33px solid transparent;  
7  
8   border-image-source: url('gambar/border1.png');  
9   border-image-slice: 16 33 16 33;  
10 }
```



Hasil dari property border-image-slice: 16 33 16 33

Sebagai contoh lain, saya akan pakai gambar border2.png:

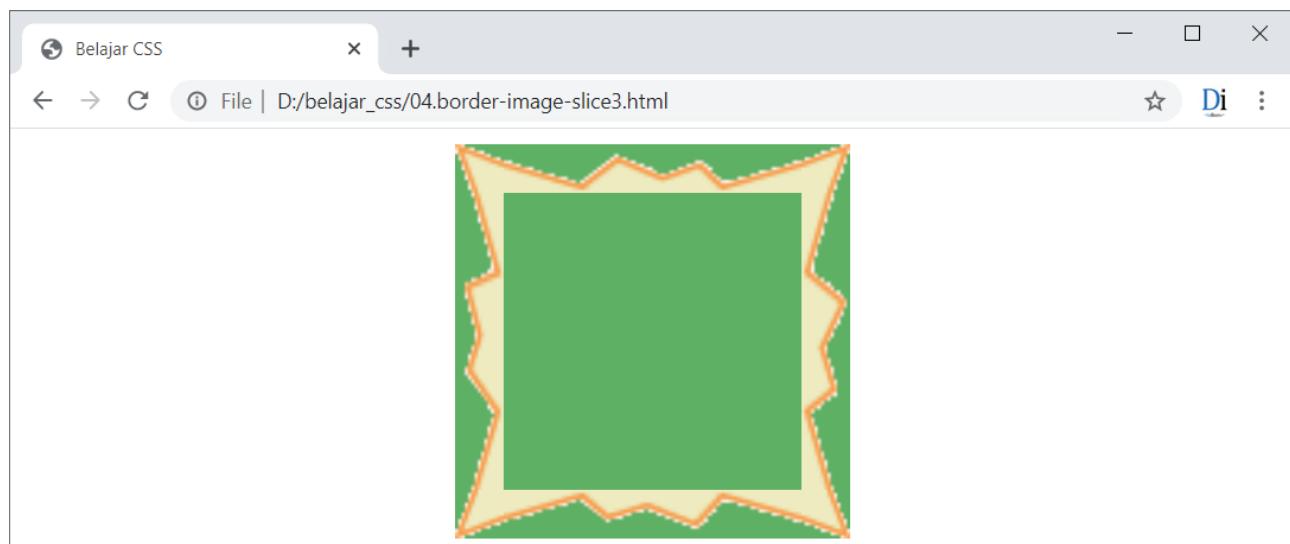


Gambar: border2.png

Berikut contoh prakteknya:

04.border-image-slice3.html

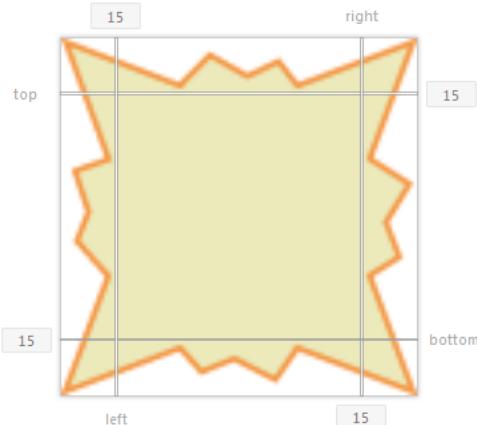
```
1 .satu {  
2   width: 200px;  
3   height: 200px;  
4   background-color: #57ac5e;  
5   margin: 10px auto;  
6   border: 33px solid transparent;  
7  
8   border-image-source: url('gambar/border2.png');  
9   border-image-slice: 15%;  
10 }
```



Gambar: Hasil penggunaan property border-image-slice: 15% pada gambar 'border2.png'

Kali ini saya menggunakan `border-image-slice:15%`. Sesuai dengan konsep TRouBLE, jika ditulis satu nilai saja, maka itu berlaku untuk setiap sisi gambar. Berikut ilustrasi pemotongan gambar border tersebut:

CSS3 Border Image

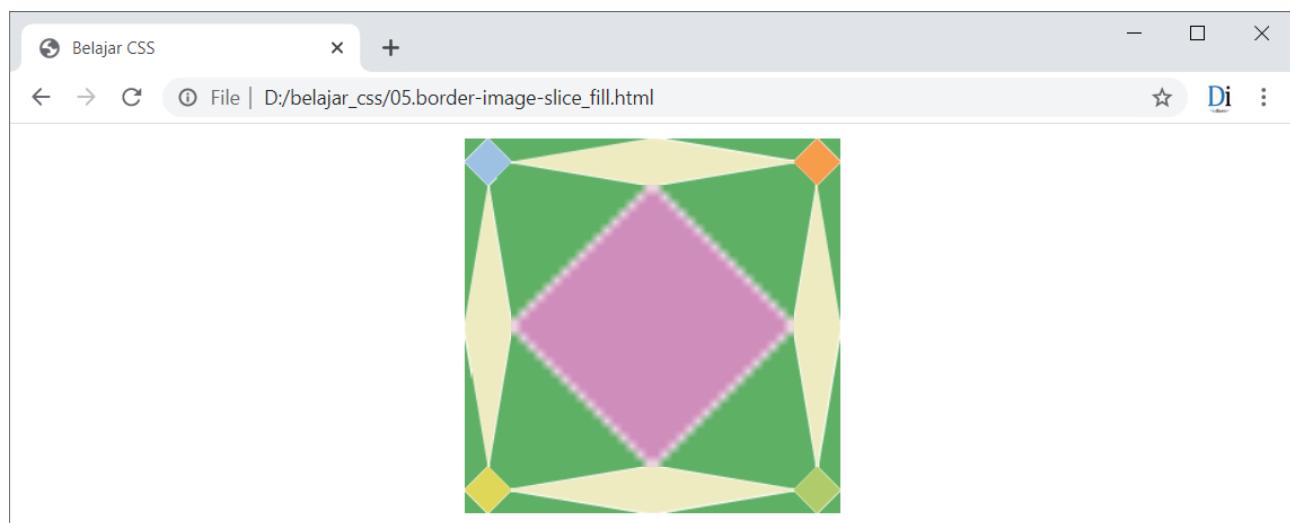


Gambar: Ilustrasi pemotongan gambar "border2.png"

Property `border-image-slice` memiliki 1 nilai opsional, yakni **fill**. Nilai ini bisa ditulis setelah ukuran slice seperti contoh berikut:

05.border-image-slice_fill.html

```
1 .satu {  
2   width: 200px;  
3   height: 200px;  
4   background-color: #57ac5e;  
5   margin: 10px auto;  
6   border: 33px solid transparent;  
7  
8   border-image-source: url('gambar/border1.png');  
9   border-image-slice: 33 fill;  
10 }
```



Hasil dari penggunaan property `border-image-slice: 33% fill`

Di baris 9, property `border-image-slice: 33 fill` akan membuat potongan gambar ke-9 (yang tepat berada di tengah-tengah hasil pemotongan) juga ikut ditampilkan.

13.4. Property border-image-width

Property `border-image-width` berfungsi untuk menentukan lebar dari gambar border. Nilai yang bisa diinput berupa satuan length seperti pixel atau persen. Berikut contoh penggunaannya:

06.border-image-width.html

```

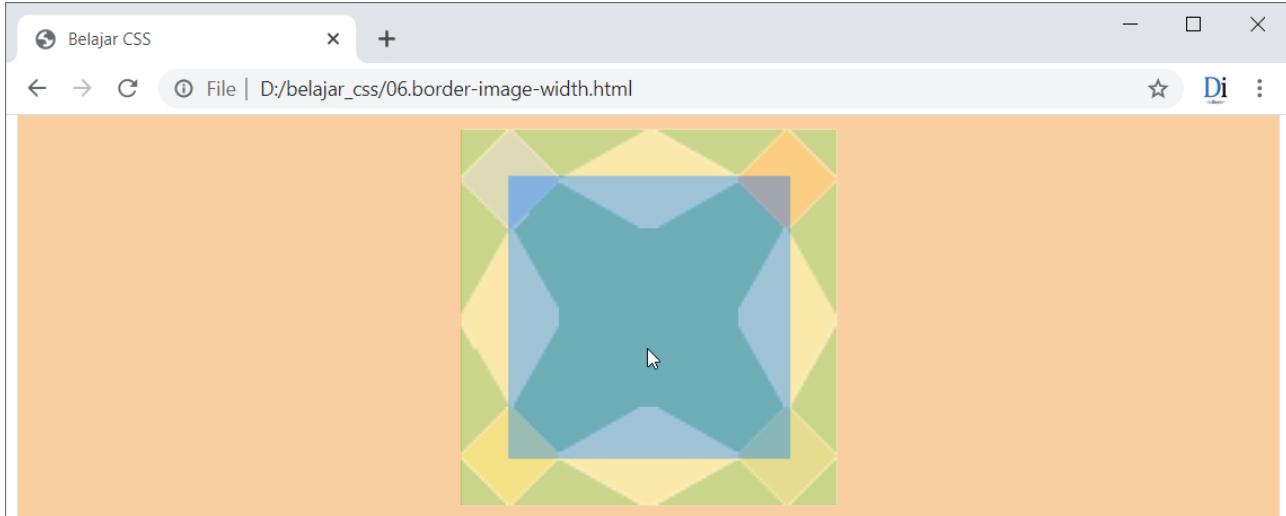
1  .satu {
2    width: 200px;
3    height: 200px;
4    background-color: #57ac5e;
5    margin: 10px auto;
6    border: 33px solid transparent;
7
8    border-image-source: url('gambar/border1.png');
9    border-image-slice: 33;
10   border-image-width: 70px;
11 }
```



Gambar: Tampilan border image dengan tambahan `border-image-width: 70px`

Tambahan property `border-image-width: 70px` di baris 10 artinya saya ingin gambar border tampil dengan lebar 70 pixel.

Lebar gambar border ini berbeda dengan lebar border yang berasal dari property `border`. Dalam contoh di atas, lebar border sebenarnya hanya 33 pixel, yakni berasal dari property `border: 33px solid transparent`. Sehingga jika digabung dengan `border-image-width: 70px`, maka ada bagian gambar border yang "melimpah" ke sisi dalam element. Berikut ilustrasinya:



Gambar: Gambar border menempati sisi dalam element

Kotak dengan warna lebih gelap adalah sisi dalam element box `.satu`. Terlihat gambar border masuk ke arah dalam kurang lebih sebesar 37 pixel, yang didapat dari 70px (`border-image-width`) dikurangi 33px (lebar border).

Idealnya nilai `border-image-width` harus sama dengan nilai property `border`, namun tidak salah juga dibedakan apabila ingin membuat efek tertentu.

13.5. Property `border-image-repeat`

Dalam beberapa praktik sebelum ini, border di 4 sisi element tampil dengan efek `stretch`, dimana gambar border dipaksa melebar untuk menutupi sisi top, right, bottom dan left. Kita bisa mengubahnya dari property `border-image-repeat`.

Property `border-image-repeat` bisa diisi dengan salah satu nilai keyword berikut: `stretch` (nilai default), `repeat`, `round` dan `space`. Berikut perbedaan ketiga nilai ini:

07.border-image-repeat.html

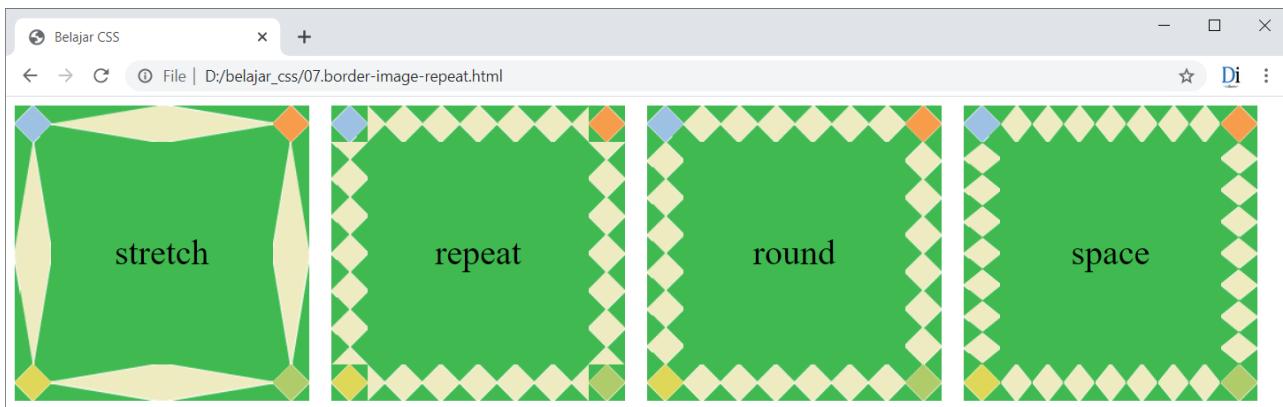
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua, .tiga, .empat {
8              width: 200px;
9              height: 200px;
10             background-color: #39B54A;
11             float: left;
12             margin-right: 20px;
13             border: 33px solid transparent;
14             font-size: 2em;
15             text-align: center;

```

CSS3 Border Image

```
16     line-height: 200px;
17
18     border-image-source: url('gambar/border1.png');
19     border-image-slice: 33%;
20 }
21 .satu {
22     border-image-repeat: stretch;
23 }
24 .dua {
25     border-image-repeat: repeat;
26 }
27 .tiga {
28     border-image-repeat: round;
29 }
30 .empat {
31     border-image-repeat: space;
32 }
33 </style>
34 </head>
35 <body>
36 <div class="satu">stretch</div>
37 <div class="dua">repeat</div>
38 <div class="tiga">round</div>
39 <div class="empat">space</div>
40 </body>
41 </html>
```



Gambar: Perbedaan 4 nilai border-image-repeat

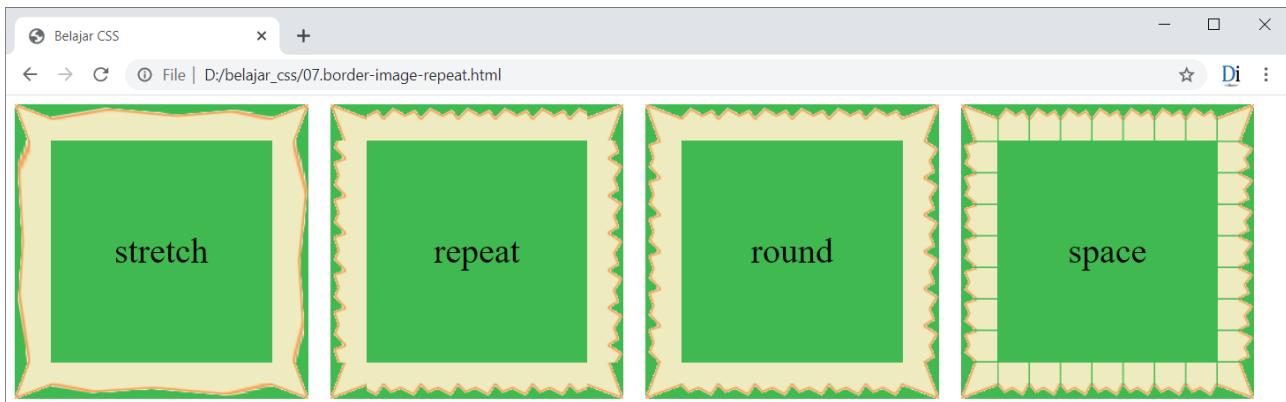
Nilai **stretch** merupakan nilai default, dimana untuk setiap sisi border, gambar akan ditarik untuk memenuhi sisi tersebut. Inilah hasil yang tampil jika property **border-image-repeat** tidak ditulis.

Nilai **repeat** akan mengulang gambar border terus menerus.

Nilai **round** mirip seperti **repeat**, tetapi perulangan hanya dilakukan selama gambar tidak terpotong (web browser akan memperbesar/memperkecil sedikit ukuran gambar).

Nilai **space** juga mirip seperti **repeat**, namun jika terdapat ruang yang tidak cukup, akan dibagi rata ke semua sisi.

Berikut hasil dari kode yang sama tapi menggunakan gambar border2.png:



Gambar: Penggunaan border-image-repeat untuk gambar border2.png

Hasil yang didapat dari property `border-image-repeat` akan berbeda-beda tergantung bentuk dan ukuran pemotongan gambar.

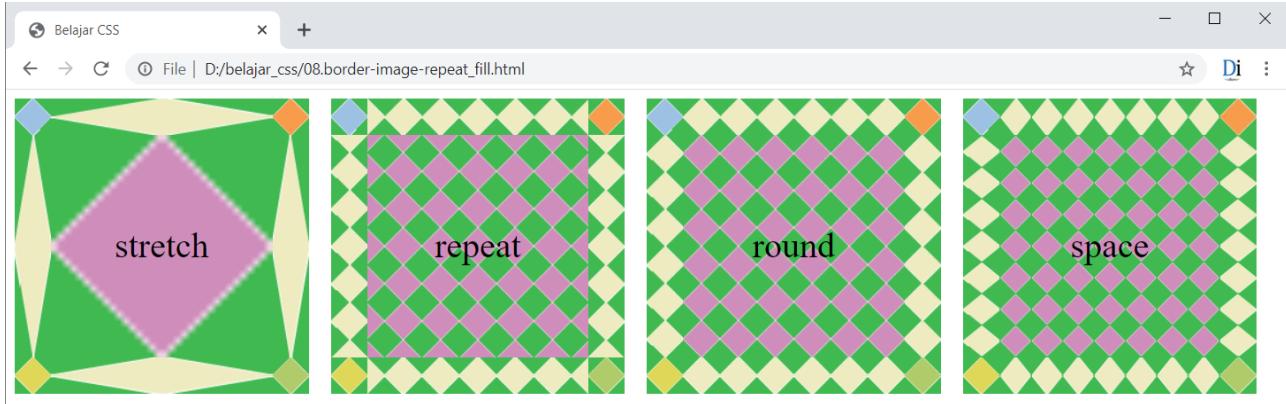
Efek dari property `border-image-repeat` juga berdampak ke penggunaan `border-image-slice:fill`, dan efek yang dihasilkan jadi cukup menarik:

08.border-image-repeat_fill.html

```

1  .satu, .dua, .tiga, .empat {
2    width:200px;
3    height: 200px;
4    background-color: #39B54A;
5    float: left;
6    margin-right: 20px;
7    border: 33px solid transparent;
8    font-size: 2em;
9    text-align: center;
10   line-height: 200px;
11
12  border-image-source: url('gambar/border1.png');
13  border-image-slice: 33% fill;
14 }
15 .satu {
16   border-image-repeat: stretch;
17 }
18 .dua {
19   border-image-repeat: repeat;
20 }
21 .tiga {
22   border-image-repeat: round;
23 }
24 .empat {
25   border-image-repeat: space;
26 }
```

CSS3 Border Image



Gambar: Penggunaan property border-image-repeat dan border-image-slice: 33% fill

Penggunaan nilai `round` akan menghasilkan efek yang lebih rapi daripada `repeat`, karena web browser akan mengkalkulasi lebar gambar agar tidak terpotong.

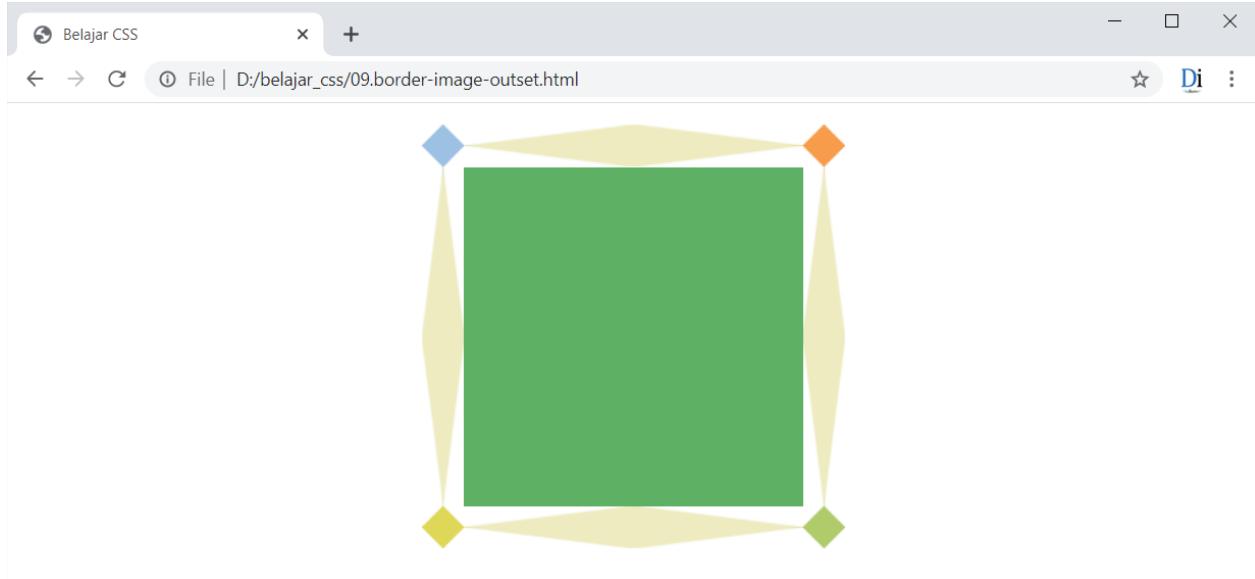
13.6. Property border-image-outset

Property **border-image-outset** bisa dipakai untuk "memindahkan" gambar border keluar dari element, yakni berada di posisi garis outset (mirip seperti yang kita pelajari dalam box model).

Property `border-image-outset` butuh nilai dalam satuan length yang berfungsi untuk menggeser border keluar. Berikut contoh penggunaannya:

09.border-image-outset.html

```
1 .satu {  
2   width: 200px;  
3   height: 200px;  
4   background-color: #57ac5e;  
5   margin: 50px auto;  
6   border: 33px solid transparent;  
7  
8   border-image-source: url('gambar/border1.png');  
9   border-image-slice: 33;  
10  border-image-outset: 33px;  
11 }
```



Gambar: Penggunaan property border-image-outset: 30px

Terlihat sekarang gambar border berada diluar gambar. Namun ini tidak otomatis menambah ukuran element. Saya harus menyesuaikan ukuran margin agar border tidak terpotong oleh sisi jendela web browser.

13.7. Property border-image (shorthand)

Sampai di sini kita telah mempelajari 5 property border image. Kelimanya bisa disatukan dalam satu property `border-image shorthand` dengan format penulisan sebagai berikut:

```
border-image: source slice / width / outset repeat;
```

Sebagai contoh, penulisan property:

```
border-image: url('gambar/border1.png') 33 fill / 30px / 30px round;
```

Sama artinya dengan:

```
border-image-source: url('gambar/border1.png');
border-image-slice: 33 fill;
border-image-width: 30px;
border-image-outset: 30px;
border-image-repeat: round;
```

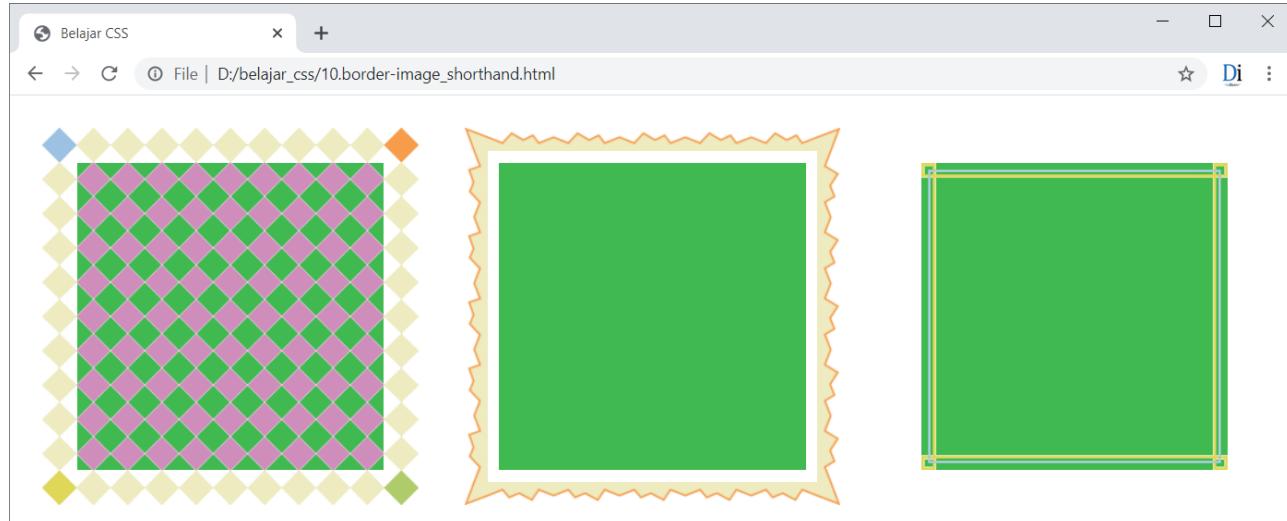
Berikut contoh penggunaannya:

10.border-image_shorthand.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
```

CSS3 Border Image

```
5 <title>Belajar CSS</title>
6 <style>
7   .satu, .dua, .tiga {
8     width:200px;
9     height: 200px;
10    background-color: #39B54A;
11    float: left;
12    margin: 50px;
13    border: 33px solid transparent;
14  }
15  .satu {
16    border-image: url('gambar/border1.png') 33 fill / 30px / 30px round;
17  }
18  .dua {
19    border-image: url('gambar/border2.png') 20 / 20px / 30px round;
20  }
21  .tiga {
22    border-image: url('gambar/border3.png') 30% / 30px / 0px round;
23  }
24 </style>
25 </head>
26 <body>
27   <div class="satu"></div>
28   <div class="dua"></div>
29   <div class="tiga"></div>
30 </body>
31 </html>
```



Contoh penggunaan property border-image shorthand

Meskipun lebih singkat, property `border-image` shorthand sedikit lebih sulit untuk ditulis karena kita harus hafal urutan dari nilai-nilai yang ada.

13.8. Border Image dari Gradient

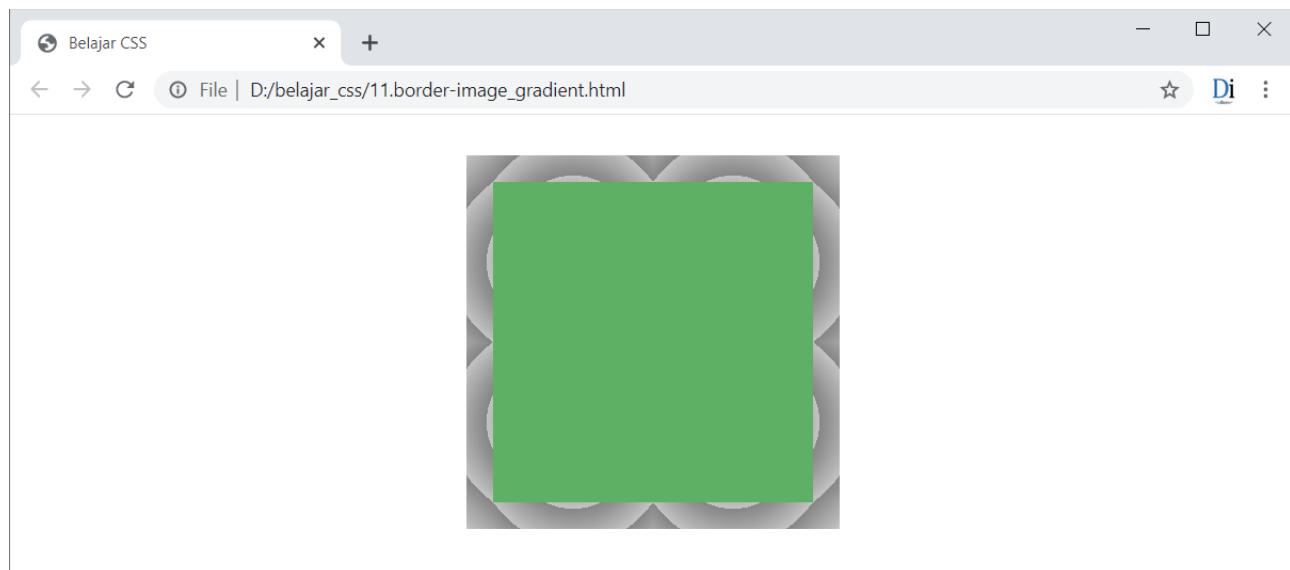
CSS3 border image juga mendukung gradient sebagai gambar border. Caranya, ganti alamat

CSS3 Border Image

url sebagai nilai property `border-image-source` dengan fungsi gradient. Berikut contoh penggunaannya:

11.border-image_gradient.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              width: 200px;
9              height: 200px;
10             background-color: #57ac5e;
11             border: 20px solid transparent;
12             margin: 50px auto;
13
14             border-image-width: 20px;
15             border-image-source: repeating-radial-gradient(circle, gray, silver 20%);
16             border-image-slice: 10%;
17             border-image-repeat: round;
18             border-image-outset: 20px;
19         }
20     </style>
21 </head>
22 <body>
23     <div class="satu"></div>
24 </body>
25 </html>
```



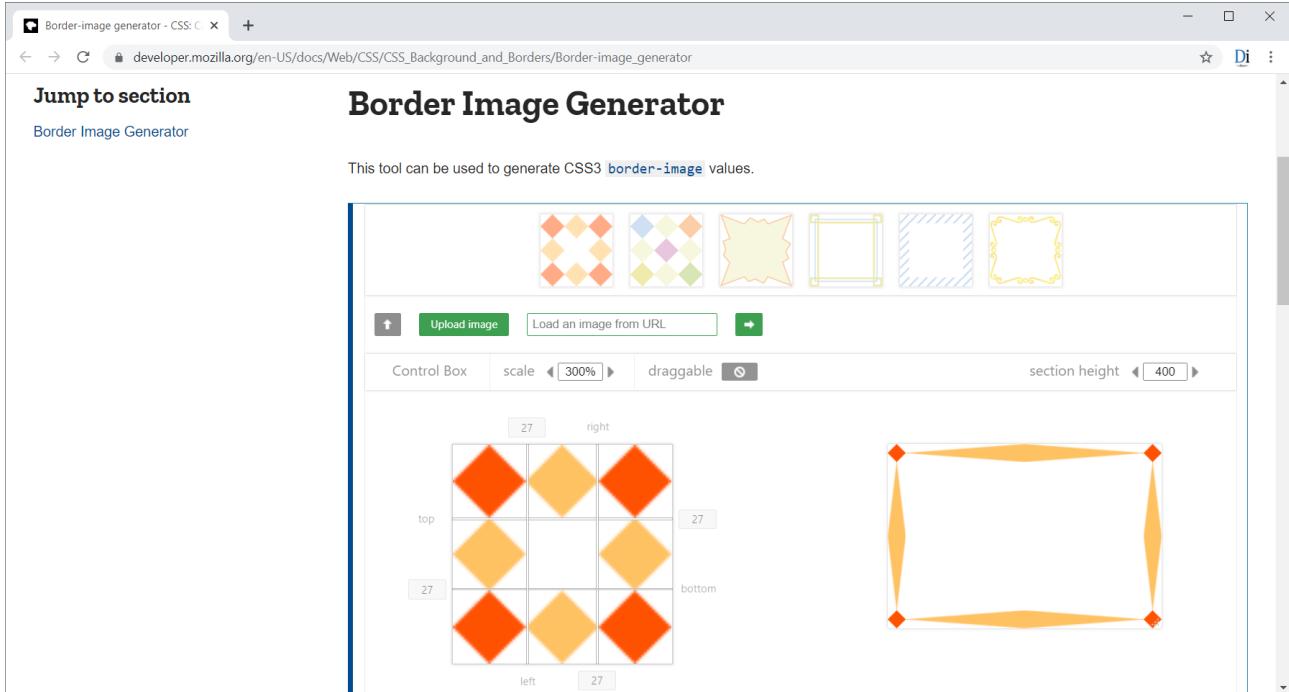
Gambar: Cara membuat gradient sebagai border image

Dengan gradient, kita bisa membuat efek gambar border secara langsung dari CSS. Silahkan berkreasi dengan merancang tampilan gradient-gradient lain.

13.9. CSS Border Image Generator

Sepanjang bab ini kita telah membahas cara membuat gambar border menggunakan CSS. Namun tidak bisa dipungkiri kalau caranya cukup rumit. Gambar border harus dibuat sedemikian rupa agar pas untuk setiap sisi border.

Untungnya, terdapat berbagai tools online yang bisa kita pakai. Salah satunya adalah Border Image Generator yang disediakan web [developer.mozilla.org](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Border-image_generator)²¹:



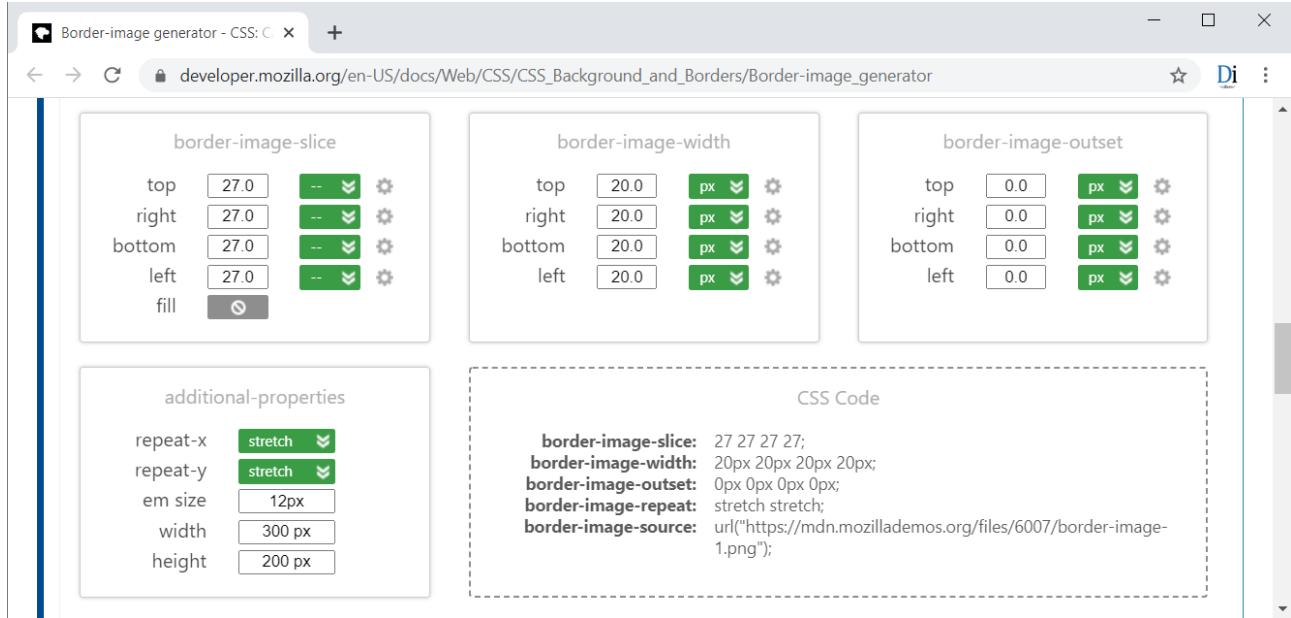
Gambar: Tampilan developer.mozilla.org untuk CSS Border Image Generator

Semua gambar border yang saya pakai sebenarnya juga berasal dari web ini. Selain itu kita bisa mengupload gambar lain dan mengatur nilai property border image secara interaktif.

Hasil pemotongan atau preview bisa dilihat dari sisi sebelah kanan, serta kode CSS akan di generate di bagian bawah halaman.

²¹ https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Border-image_generator

CSS3 Border Image



Gambar: Proses generate property border image

Ini sangat membantu dalam perancangan border image menggunakan CSS.

Masih membahas tentang border, dalam bab berikutnya akan masuk ke materi tentang cara membuat efek bayangan sepanjang border, yakni **CSS3 Box Shadow**.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

=====

14. CSS3 Box Shadow

Ketika saya pertama kali belajar CSS sekitar 12 tahun lalu, efek bayangan (*shadow*) harus dibuat menggunakan trik yang cukup rumit. Caranya adalah membuat 1 atau 2 gambar bayangan yang sudah di-crop, lalu tempatkan ke dalam 2 lapis tag `<div>` (sebagai container).

Belum lagi jika ingin memperbesar ukuran bayangan atau mengganti warna, maka harus menyediakan berbagai gambar bayangan.

Untungnya, semua tinggal masa lalu... CSS3 menghadirkan property `box-shadow` untuk membuat efek bayangan. Cara penggunaannya sangat mirip dengan property `text-shadow` yang telah kita pelajari dalam bab Typography.

14.1. Property box-shadow

Untuk membuat efek bayangan hanya butuh 1 property saja, yakni `box-shadow`. Nilai yang bisa diinput cukup beragam. Format lengkap penulisannya adalah sebagai berikut:

```
box-shadow: inset offset-x offset-y blur-radius spread color;
```

Kita akan bahas satu per satu.

14.2. Box Shadow Offset

Nilai paling minimal untuk membuat efek bayangan adalah `offset-x` dan `offset-y`. Kedua nilai ini berfungsi untuk mengatur besar dan lokasi bayangan. Nilai yang bisa diinput berupa angka positif maupun negatif dengan satuan width (px, em, %, dll).

Untuk `offset-x`, jika bernilai positif maka bayangan akan berada di sisi kanan element. Jika diisi angka negatif, bayangan akan berada di sisi kiri element.

Untuk `offset-y`, jika bernilai positif maka bayangan akan berada di sisi bawah element. Jika diisi angka negatif, bayangan akan berada di sisi atas element.

Berikut contoh penggunaannya:

```
01.box-shadow_offset_xy.html
```

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
```

```

4  <meta charset="UTF-8">
5  <title>Belajar CSS</title>
6  <style>
7      .satu, .dua, .tiga, .empat {
8          width: 200px;
9          height: 200px;
10         background-color: #39B54A;
11         float: left;
12         margin: 30px 0 0 50px;
13     }
14     .satu { box-shadow: 5px 5px;      }
15     .dua { box-shadow: 20px 20px;    }
16     .tiga { box-shadow: -8px -10px;  }
17     .empat { box-shadow: -15px 20px; }
18 </style>
19 </head>
20 <body>
21     <div class="satu"></div>
22     <div class="dua"></div>
23     <div class="tiga"></div>
24     <div class="empat"></div>
25 </body>
26 </html>

```



Gambar: Beberapa contoh penggunaan box shadow offset

Nilai `offset-x` dan `offset-y` ini adalah syarat minimum untuk membuat efek box shadow. Nilai lain yang akan kita bahas setelah ini merupakan opsional dan boleh tidak ditulis.

14.3. Box Shadow Blur-radius

Setelah penulisan `offset-x` dan `offset-y`, urutan ketiga dari property box shadow dipakai untuk **blur-radius**. Blur-radius adalah sebutan dari efek 'mengaburkan' bayangan. Nilainya bisa diisi dengan seluruh satuan width positif, seperti contoh berikut:

02.box-shadow_blur-radius.html

```
1  .satu { box-shadow: 10px 10px 2px; }
```

CSS3 Box Shadow

```
2 .dua { box-shadow: 10px 10px 5px; }
3 .tiga { box-shadow: 10px 10px 15px; }
4 .empat { box-shadow: 10px 10px 30px; }
```



Gambar: Beberapa contoh penggunaan box shadow blur-radius

14.4. Box Shadow Spread

Urutan keempat property `box-shadow` diisi dengan **spread**. Spread berfungsi untuk mengatur seberapa jauh efek bayangan yang terjadi. Jika diberikan nilai negatif, bayangan akan lebih kecil daripada dimensi element. Berikut contohnya:

03.box-shadow_spread.html

```
1 .satu { box-shadow: 10px 10px 10px 10px; }
2 .dua { box-shadow: 10px 10px 10px 20px; }
3 .tiga { box-shadow: 10px 10px 10px -10px; }
4 .empat { box-shadow: 10px 10px 10px -20px }
```



Beberapa contoh penggunaan box shadow spread

Pada kotak ketiga saya menggunakan nilai `spread: -10px`. Hasilnya, bayangan lebih kecil daripada ukuran element, tapi efeknya masih bisa terlihat sedikit.

Sedangkan pada kotak keempat, efek bayangan sudah tidak terlihat sama sekali, karena ukuran spread jauh lebih kecil daripada ukuran element.

14.5. Box Shadow Color

Nilai kelima dari property `box-shadow` adalah warna (color). Kita bisa menggunakan seluruh satuan warna CSS, mulai dari keyword, #RRGGBB hingga format warna HSL. Berikut contoh penggunaannya:

04.box-shadow_color.html

```
1 .satu { box-shadow: 10px 10px 10px 10px red; }
2 .dua { box-shadow: 10px 10px 10px 10px #C0C0C0; }
3 .tiga { box-shadow: 10px 10px 10px 10px rgba(255,88,75,0.8); }
4 .empat { box-shadow: 10px 10px 10px 10px hsl(100,100%,50%); }
```



Gambar: Beberapa contoh penggunaan box shadow color

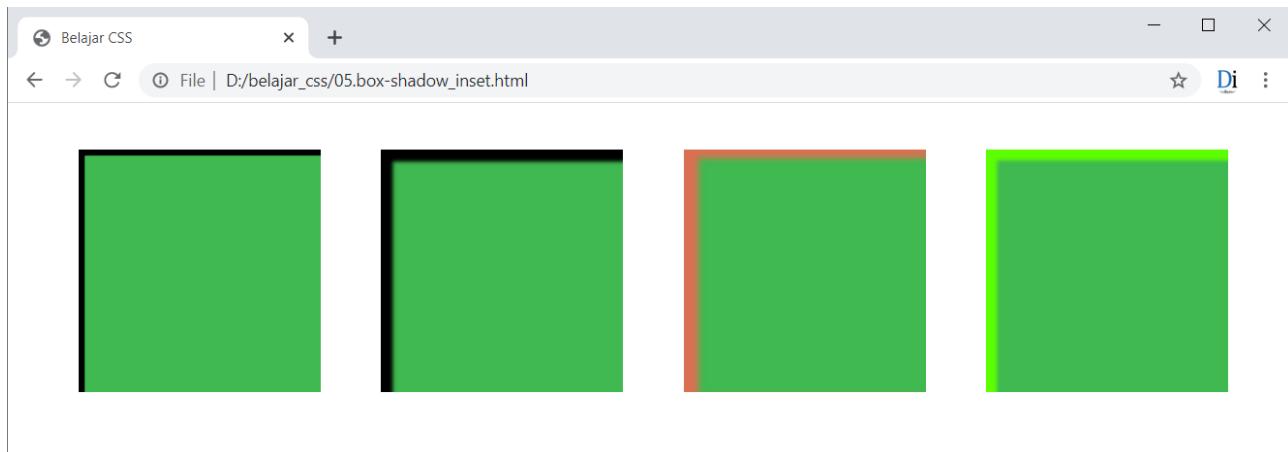
Apabila nilai warna tidak ditulis, web browser akan menggunakan warna hitam sebagai nilai default.

14.6. Box Shadow Inset

Jika kita menambahkan keyword `inset` di awal penulisan nilai `box-shadow`, maka bayangan akan tampil di dalam element HTML:

05.box-shadow_inset.html

```
1 .satu { box-shadow: inset 5px 5px ; }
2 .dua { box-shadow: inset 10px 10px 3px ; }
3 .tiga { box-shadow: inset 10px 5px 5px 2px rgba(255,88,75,0.8); }
4 .empat { box-shadow: inset 7px 7px 3px 2px hsl(100,100%,50%); }
```



Gambar: Berbagai contoh penggunaan box shadow inset

Seperti yang terlihat, efek bayangan berada di dalam element. Meminjam istilah photoshop, ini biasa disebut sebagai *inner shadow*. Yang perlu diperhatikan adalah, penulisan keyword `inset` harus ditempatkan pada posisi pertama penulisan nilai box-shadow (sebelum offset).

14.7. Multiple Box Shadow

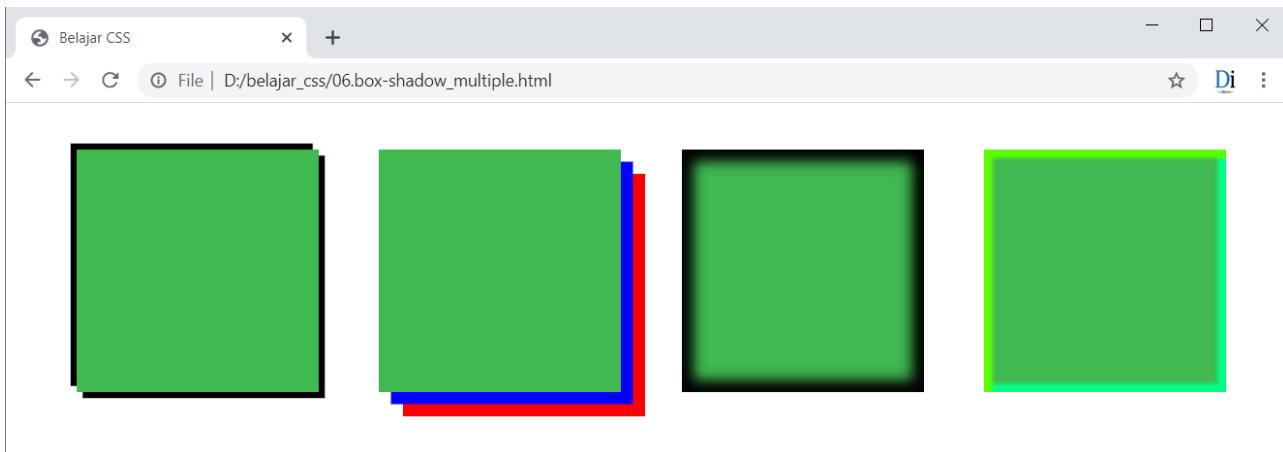
Jika 1 efek bayangan belum mencukupi, kita bisa membuat 2, 3 atau lebih bayangan ke dalam satu element HTML. Seluruh bayangan ini cukup ditulis dengan 1 property `box-shadow` dengan pemisah tanda koma. Berikut contoh yang dimaksud:

06.box-shadow_multiple.html

```

1  .satu {
2    box-shadow: 5px 5px, -5px -5px ;
3  }
4  .dua {
5    box-shadow: 10px 10px blue, 20px 20px red ;
6  }
7  .tiga {
8    box-shadow: inset 10px 10px 10px, inset -10px -10px 10px ;
9  }
10 .empat {
11   box-shadow: inset 7px 7px 3px hsl(100,100%,50%),
12                 inset -7px -7px 3px hsl(150,100%,50%);
13 }
```

CSS3 Box Shadow



Berbagai contoh penggunaan multiple box shadow

Dengan menggabungkan beberapa efek bayangan sekaligus, kita bisa membuat berbagai efek yang menarik.

14.8. CSS Box Shadow Generator

Tidak mau ketinggalan dengan berbagai "generator" lain, web developer.mozilla.org²² juga menyediakan **Box-shadow generator**:

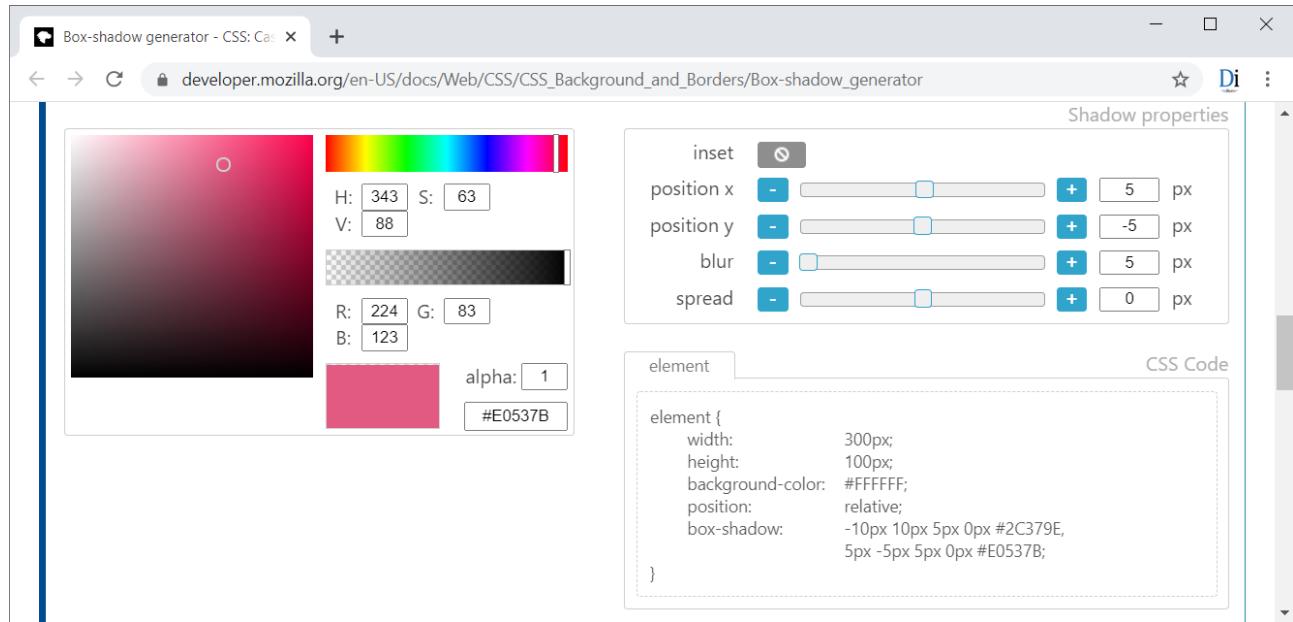
A screenshot of a web browser window titled "Box-shadow generator - CSS: Cat". The address bar shows the URL "developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Box-shadow_generator". The main content area has a large heading "box-shadow generator". Below it, a sub-headline says "This tool lets you construct CSS `box-shadow` effects, to add box shadow effects to your CSS objects.". On the left, there is a sidebar with a "+" button, up and down arrows, and tabs for "element", ":before", and ":after". Under the "element" tab, there is a list with "shadow 1" and "shadow 0". To the right, there is a preview area showing a white rectangle with a multi-layered shadow effect, consisting of a blue outer shadow and a pink inner shadow.

Gambar: Tampilan developer.mozilla.org untuk CSS Box-shadow generator

Di halaman tersebut kita bisa membuat berbagai efek bayangan secara interaktif, termasuk multiple shadow. Caranya, klik tombol plus "+" di kiri atas sesuai dengan jumlah bayangan yang ingin dibuat.

²² https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders/Box-shadow_generator

Di bagian bawah, setiap nilai bayangan bisa diatur dan kodennya tersedia untuk di copy paste:



Gambar: Proses generate property box-shadow

Fitur ini sangat membantu dalam perancangan box shadow menggunakan CSS.

Efek box shadow yang kita pelajari dalam bab ini bisa dipakai sebagai pemanis tampilan terutama untuk membuat efek 3D. Namun tetap batasi penggunaannya agar tidak terlalu over.

15. CSS3 2D Transform

Sejak pertama kali dikembangkan, setiap element HTML diproses sebagai sebuah kotak (yang kemudian menjadi konsep CSS *box model*). Satu-satunya cara untuk memanipulasi tampilan ini adalah dengan menggantinya menjadi gambar.

Sekitar tahun 2008, tim dari webkit CSS engine mengusulkan penambahan property baru agar CSS bisa memanipulasi bentuk kotak ini. Misalnya dengan menambahkan efek memperbesar atau merotasi sebuah element.

Akhirnya tim dibalik CSS3 menerima usul tersebut dan menghadirkan **CSS Transforms Module**. Dengan modul ini, kita bisa membuat berbagai efek visual 2 dimensi ke dalam element HTML, seperti *rotate*, *translate*, *scale* dan *skew*.

15.1. Property transform

Seluruh efek 2D Transformations yang di dalam CSS3, dilakukan hanya dari 1 property, yakni: **transform**. Nilai dari property inilah yang akan menentukan bagaimana efek yang dihasilkan. Cara penulisannya mirip dengan gradient, yakni berbentuk fungsi:

```
transform: function(value);
```

Jika terdiri dari 2 fungsi, penulisannya dipisah dengan tanda spasi:

```
transform: function(value) function(value);
```

Kita akan bahas satu per satu nilai yang bisa diinput ke dalam property `transform` di atas.

15.2. Transform Rotate

Rotate digunakan untuk memutar element dengan sudut tertentu. Berikut format penulisannya:

```
transform: rotate(value);
```

Value atau nilai dari fungsi ini berupa angka dalam satuan sudut (*angle unit*). Sebagaimana yang telah kita praktekkan pada bab gradient, CSS3 mendukung berbagai satuan sudut, yakni degree (`deg`), gradian (`grads`), radian (`rad`) dan turn. Dari keempat nilai ini, nilai degree-lah yang paling mudah digunakan dan didukung luas oleh web browser.

Sebagai contoh, jika saya ingin memutar sebuah element sebesar 15 derajat searah jarum jam, bisa menggunakan property berikut:

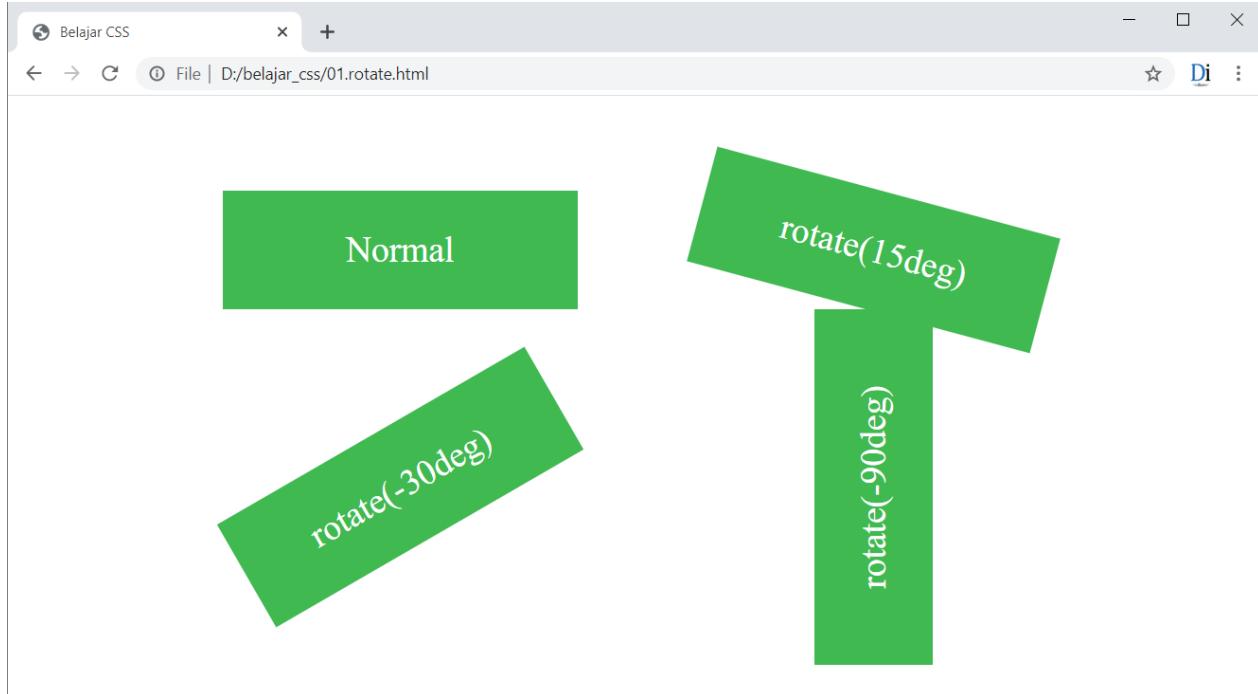
```
transform: rotate(15deg);
```

Jika kita menggunakan nilai negatif, maka nilai derajat akan dihitung berlawanan dengan arah jarum jam. Berikut contoh penggunaannya:

01.rotate.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 800px;
9              margin: 30px auto;
10         }
11         .satu, .dua, .tiga, .empat {
12             width: 300px;
13             height: 100px;
14             background-color: #39B54A;
15             float: left;
16             margin: 50px;
17             color: white;
18             font-size: 30px;
19             line-height: 100px;
20             text-align: center;
21         }
22         .dua { transform: rotate(15deg);    }
23         .tiga { transform: rotate(-30deg);  }
24         .empat { transform: rotate(-90deg); }
25     </style>
26 </head>
27 <body>
28     <div class="container">
29         <div class="satu">Normal</div>
30         <div class="dua">rotate(15deg)</div>
31         <div class="tiga">rotate(-30deg)</div>
32         <div class="empat">rotate(-90deg)</div>
33     </div>
34 </body>
35 </html>
```



Gambar: Tampilan berbagai efek transform: rotate

Dalam contoh di atas saya membuat 4 box dengan berbagai efek rotasi. Yang perlu diperhatikan adalah, ketika kita merotasi sebuah element, element tersebut tetap berada di tempatnya, yakni masih berada dalam *normal document flow*.

Jika kita terlalu jauh merotasi element dan tidak memperhitungkan ruang di sekitar, element lain bisa tertutupi seperti pada kotak ke empat.

Transform Rotate dengan transform-origin

Efek rotate yang kita praktikkan sebelum ini akan merotasi element dari titik pusatnya, yakni tepat di tengah-tengah element tersebut. CSS3 juga menyediakan property `transform-origin` untuk mengatur dari titik mana rotasi dilakukan.

Konsep `transform-origin` bisa diilustrasikan dengan memutar kertas yang diberi paku gabus (*push pin*), seperti gambar berikut:



Gambar: Paku gabus sebagai ilustrasi dari transform-origin

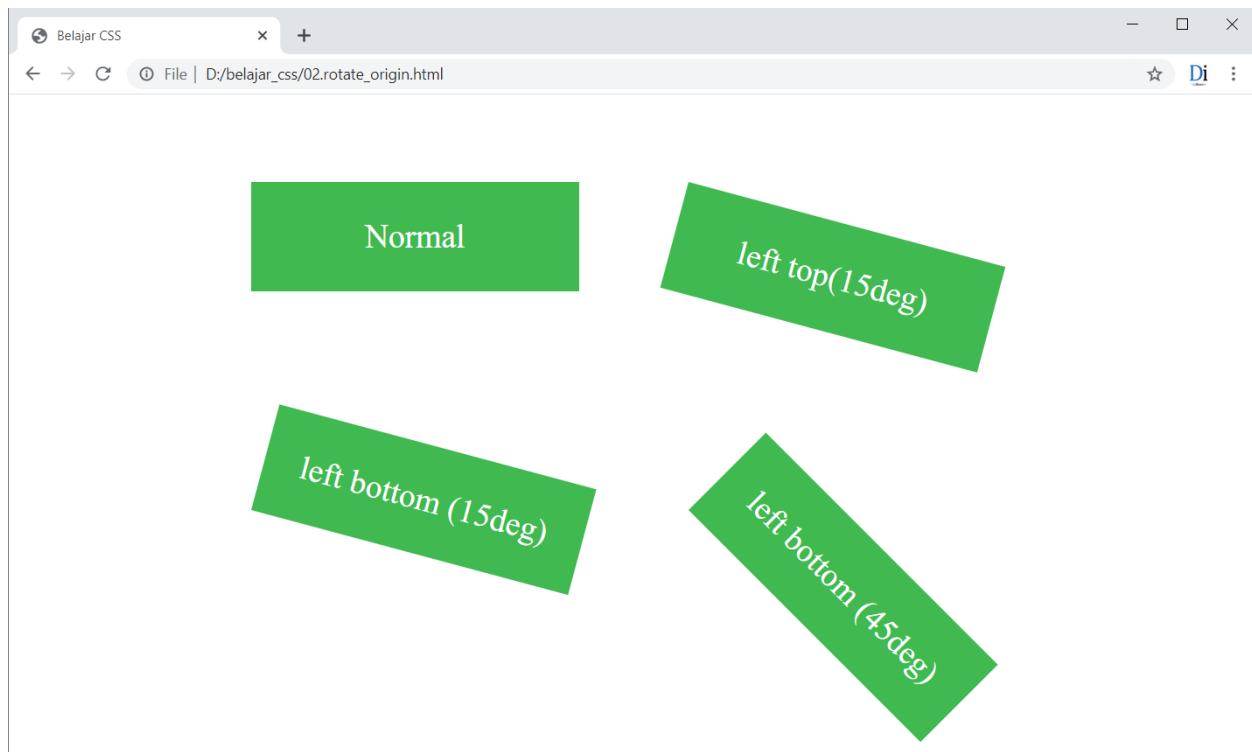
Dengan memindahkan posisi paku gabus tersebut, efek rotasi yang dihasilkan juga akan berbeda.

Nilai untuk property `transform-origin` adalah posisi element sebagaimana nilai yang digunakan untuk property `background-position`. Berikut contoh penggunaanya:

02.rotate_origin.html

```

1  .satu {
2 }
3  .dua {
4      transform-origin: left top;
5      transform: rotate(15deg);
6 }
7  .tiga {
8      transform-origin: left bottom;
9      transform: rotate(15deg);
10 }
11 .empat {
12     transform-origin: left bottom;
13     transform: rotate(45deg);
14 }
```



Gambar: Tampilan efek transform: rotate dengan berbagai posisi transform-origin

Di sini saya menggabung property `transform` dengan `rotate` dan `transform-origin`. Sekilas tidak terlihat perbedaan, tapi pusat rotasi element akan berubah tergantung nilai property `transform-origin`. Silahkan coba nilai lain dan lihat efek yang dihasilkan.

15.3. Transform Translate

Transform **translate** dipakai untuk memindahkan posisi suatu element. Jika dilihat dari hasilnya, efek ini mirip dengan relative positioning.

Untuk memindahkan element secara horizontal (sepanjang sumbu X), bisa menggunakan fungsi **translateX**, sedangkan untuk posisi vertikal (sepanjang sumbu Y), bisa menggunakan fungsi **translateY**.

Sebagai contoh, untuk memindahkan posisi sebuah element 20 pixel ke kanan dan 20 pixel ke bawah, bisa ditulis dengan kode **transform: translateX(20px) translateY(20px)**.

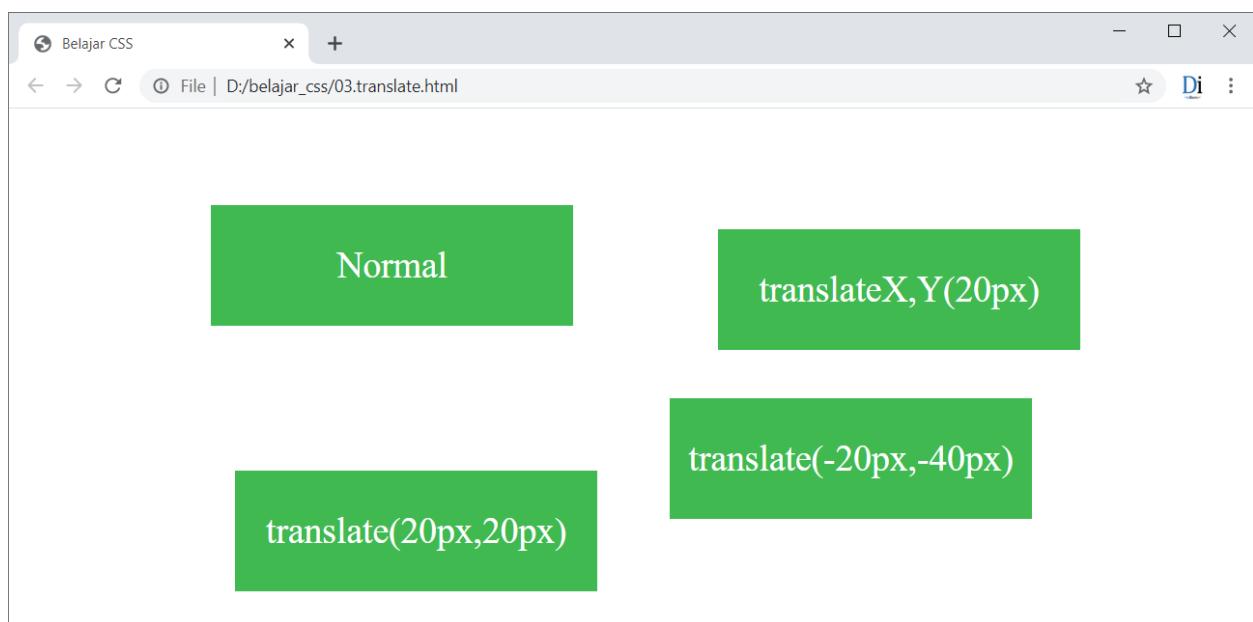
Sebagai alternatif, tersedia juga fungsi shorthand **transform: translate(20px, 20px)**. Dimana nilai pertama adalah untuk sumbu X, dan nilai kedua untuk sumbu Y.

Berikut contoh penggunaan efek transform translate di dalam CSS:

03.translate.html

```

1  .satu {
2  }
3  .dua {
4      transform: translateX(20px) translateY(20px);
5  }
6  .tiga {
7      transform: translate(20px,20px);
8  }
9  .empat {
10     transform: translate(-20px,-40px);
11 }
```



Gambar: Tampilan berbagai efek transform: translate

Selain nilai positif, kita bisa memberikan nilai negatif untuk translate, contohnya seperti pada box ke-4. Nilai negatif ini akan membuat element di geser ke kanan untuk sumbu X, dan ke atas untuk sumbu Y.

15.4. Transform Scale

Efek scale berguna untuk memperbesar atau memperkecil sebuah element. Nilai dari scale berupa angka tanpa satuan. Angka ini bisa berupa pecahan, baik positif maupun negatif.

Efek scale juga terdiri dari 2 dimensi, yakni `scaleX` untuk membuat skala pada sumbu X (lebar element), dan `scaleY` untuk skala sumbu Y (tinggi element).

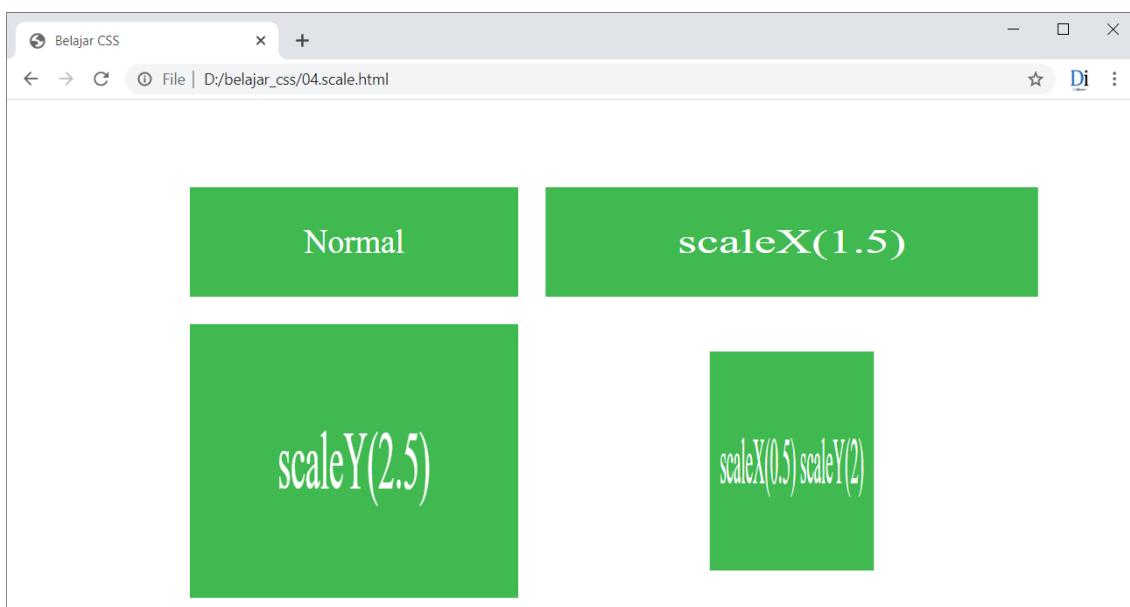
Sebagai contoh, untuk membuat sebuah element menjadi 2 kali lebih besar, kita bisa men-set fungsi `scaleX` dan `scaleY` sebagai `transform: scaleX(2) scaleY(2)`. Tapi jika hanya ingin memperbesar tinggi element saja, cukup dengan: `transform: scaleY(2)`.

Berikut contoh prakteknya:

04.scale.html

```

1  .satu {
2 }
3  .dua {
4      transform: scaleX(1.5);
5 }
6  .tiga {
7      transform: scaleY(2.5);
8 }
9  .empat {
10     transform: scaleX(0.5) scaleY(2);
11 }
```



Gambar: Tampilan berbagai efek transform: scaleX dan scaleY

Hasilnya, bukan saja tinggi dan lebar element yang berubah, tetapi seluruh "isi" element juga ikut di-scale, termasuk ukuran font.

Selain men-set kedua sisi secara terpisah, terdapat fungsi shorthand `scale()`. Sebagai contoh, untuk memperbesar sebuah element menjadi 1,5 kali, bisa menggunakan penulisan sebagai berikut:

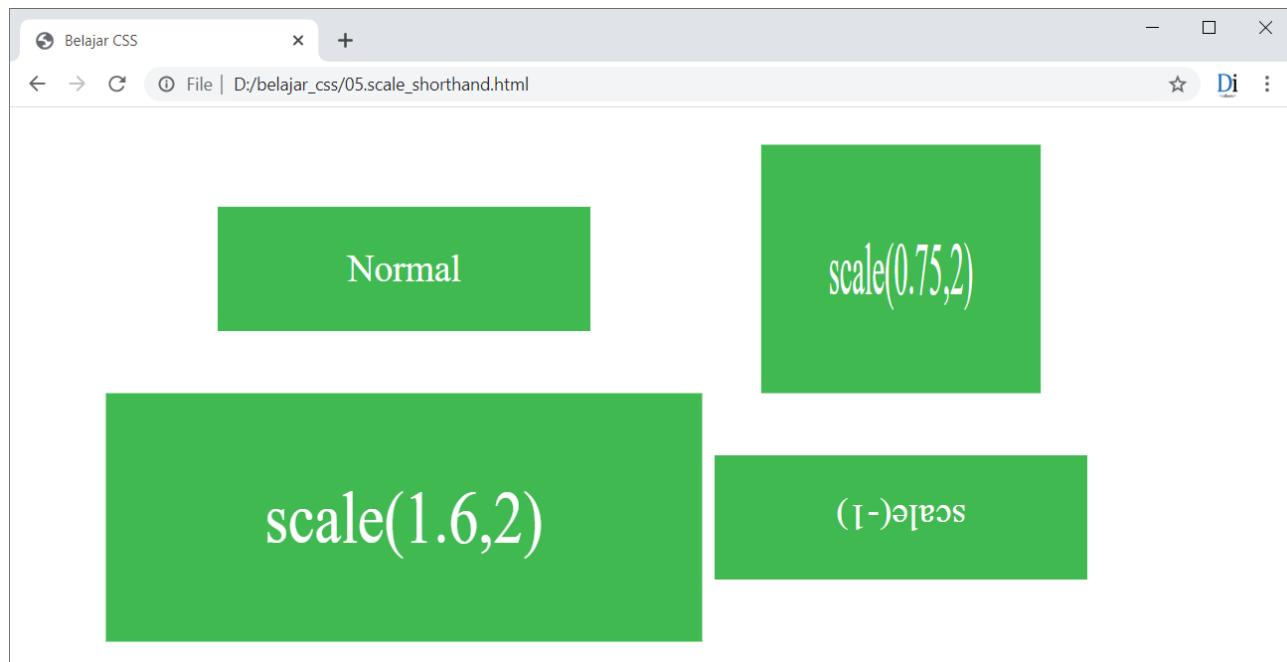
```
transform: scale(1.5, 1.5);
```

Angka pertama dipakai untuk mengatur lebar element, dan angka kedua untuk kala tinggi element. Jika hanya terdapat 1 nilai saja, nilai tersebut akan dipakai untuk kedua dimensi.

Berikut praktek penggunaannya:

`05.scale_shorthand.html`

```
1 .satu {
2 }
3 .dua {
4   transform: scale(0.75,2);
5 }
6 .tiga {
7   transform: scale(1.6,2);
8 }
9 .empat {
10  transform: scale(-1);
11 }
```



Gambar: Tampilan berbagai efek transform scale shorthand

Perhatikan box keempat, pada kotak tersebut saya memberi nilai negatif untuk scale. Hasilnya, element tampil terbalik. Efek seperti ini bisa dipakai untuk menghasilkan efek cermin

(mirroring), yakni dengan memberikan nilai negatif untuk sumbu Y, sumbu X, atau keduanya.

15.5. Transform Skew

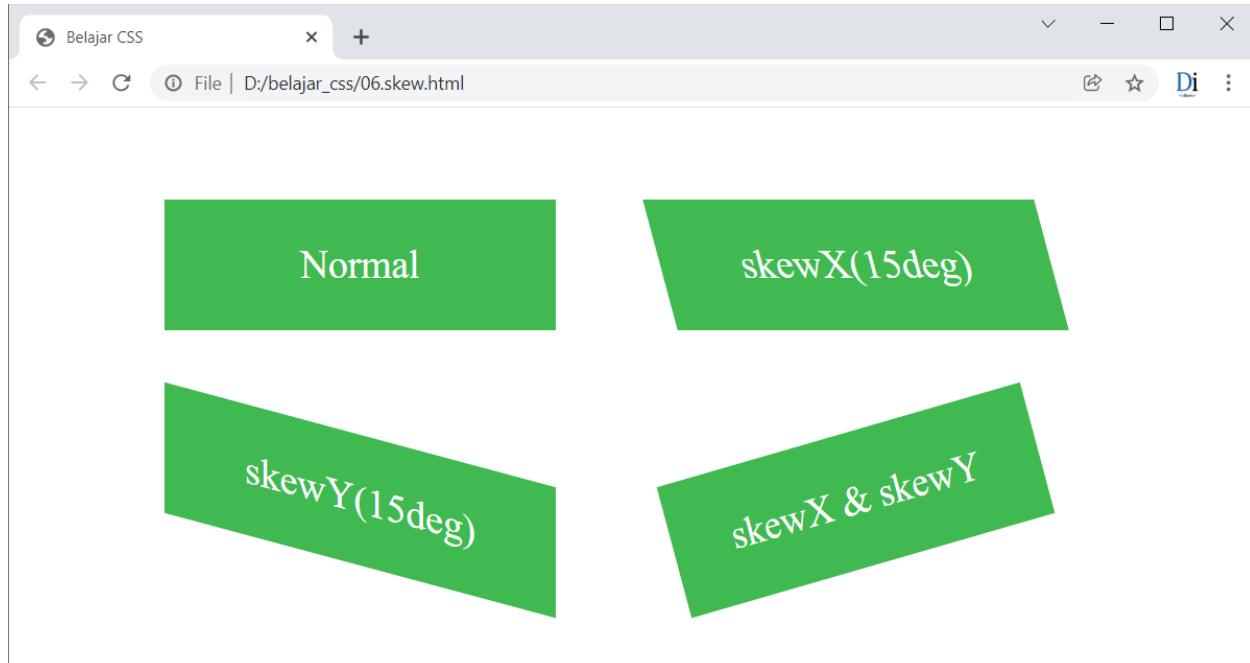
Efek **skew** adalah mengubah sudut element pada sumbu vertikal, horizontal, atau keduanya. Efek yang dihasilkan seolah-olah element tersebut "ditarik" ke arah tertentu.

Untuk membuat **skew**, kita bisa menggunakan fungsi **skewX** dan **skewY**. Kedua fungsi ini butuh nilai dalam satuan derajat. Agar lebih mudah dipahami, langsung saja masuk ke contoh:

06.skew.html

```

1  .satu {
2 }
3  .dua {
4      transform: skewX(15deg);
5 }
6  .tiga {
7      transform: skewY(15deg);
8 }
9  .empat {
10     transform: skewX(15deg) skewY(-15deg);
11 }
```



Gambar: Tampilan berbagai efek transform skewX dan skewY

Pada box kedua, saya menggunakan property **transform: skewX(15deg)**, hasilnya element seolah2 'ditarik' sepanjang sumbu X. Begitu juga dengan property **transform: skewY(15deg)** yang akan menarik element pada sumbu Y sebanyak 15 derajat.

Khusus untuk fungsi **skew**, tidak terdapat penulisan shorthand.

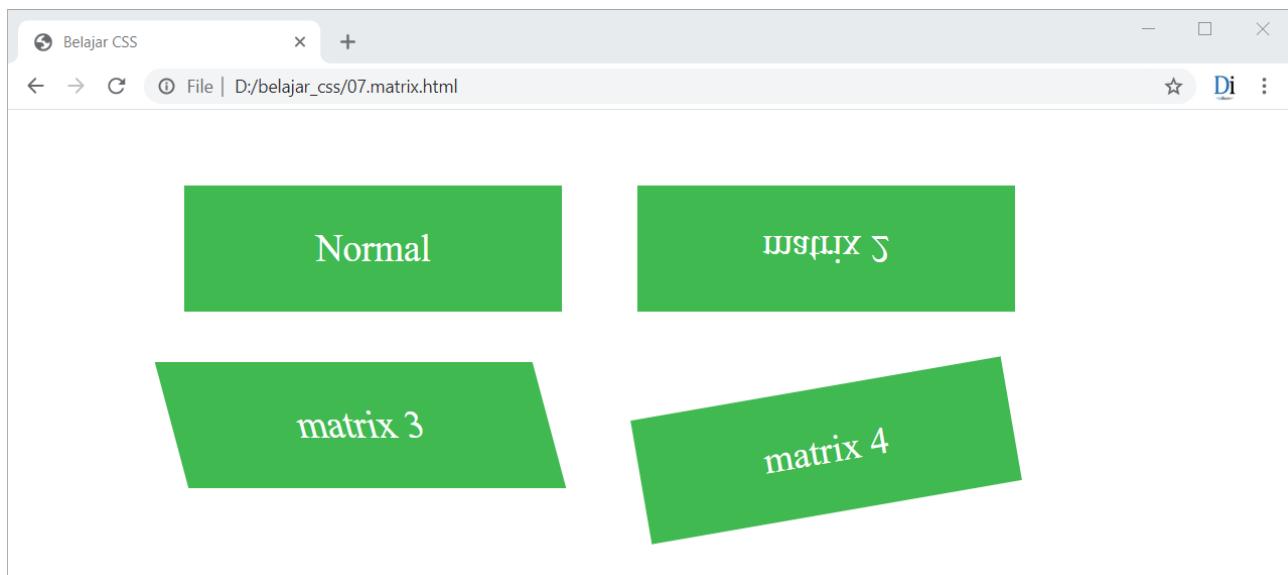
15.6. Transform dengan Matriks

Selain dengan fungsi-fungsi yang kita bahas di atas, efek transform juga bisa ditulis dengan matriks khusus. Tapi karena perhitungannya lumayan "njelimet" (melibatkan perkalian matriks), saya tidak akan bahas maksud dari nomor-nomor yang ada. Kita hanya lihat contoh hasil akhir saja:

07.matrix.html

```

1  .satu {
2  }
3  .dua {
4    transform: matrix(1,0,0,-1,0,0);
5  }
6  .tiga {
7    transform: matrix(1,0,0.268,1,-10,-20);;
8  }
9  .empat {
10   transform: matrix(0.98,-0.17,0.17,0.98,0,0);
11 }
```



Gambar: Tampilan berbagai efek transform dengan menggunakan matrix

Jika anda tertarik mempelajari cara penulisan fungsi matrix ini, bisa lanjut ke artikel di website dev.opera.com berikut: [Understanding the CSS Transforms Matrix²³](#).

Efek transform yang kita pelajari di sini bisa dipakai untuk membuat berbagai efek menarik dari CSS, terutama saat digabung dengan CSS3 Transitions dan CSS3 Animations. Inilah yang akan dibahas pada bab berikutnya.

²³ <https://dev.opera.com/articles/understanding-the-css-transforms-matrix/>

16. CSS3 Transitions dan CSS3 Animations

Dari semua modul CSS3, sepertinya inilah yang paling menarik. Apalagi yang bisa dibandingkan dengan membuat animasi langsung dari CSS tanpa perlu kode JavaScript.

Fitur animasi dalam CSS3 memberikan ruang berkreasi yang lebih luas, sekaligus mengaburkan konsep pemisahan antara CSS dan JavaScript. Fitur animasi selama ini sangat identik dengan JavaScript, karena hanya dengan JavaScript-lah kita bisa membuat efek interaksi dan perubahan ke dalam kode HTML.

Terlepas dari pro dan kontra ini (apakah animasi seharusnya "milik" JavaScript atau CSS), fitur animasi sudah tersedia di CSS.

Di dalam CSS3, efek animasi ini berada dalam 2 modul terpisah: **CSS3 Transitions Module**, dan **CSS3 Animations Module**. CSS 3 Transitions (transisi) boleh disebut sebagai bentuk sederhana dari CSS3 Animations (animasi). Oleh karena itu kita akan bahas efek transisi terlebih dahulu.

16.1. CSS3 Transitions

Di dalam CSS3, **transitions** atau transisi adalah perubahan style dari sebuah element HTML secara bertahap. Apakah itu perubahan warna background, lebar element, tinggi element, dll.

Sebenarnya di CSS 2.1 juga sudah terdapat efek transisi, yakni dengan menggunakan selector `:hover`. Efek ini sering dipakai untuk membuat menu navigasi yang berubah warna ketika cursor mouse berada di atasnya. Selain itu, selector `:hover` juga bisa digunakan untuk element HTML lain.

Akan tetapi efek transisi yang dihasilkan langsung "lompat" dari satu kondisi ke kondisi lain tanpa ada efek perantara. Sebagai contoh, perhatikan kode berikut:

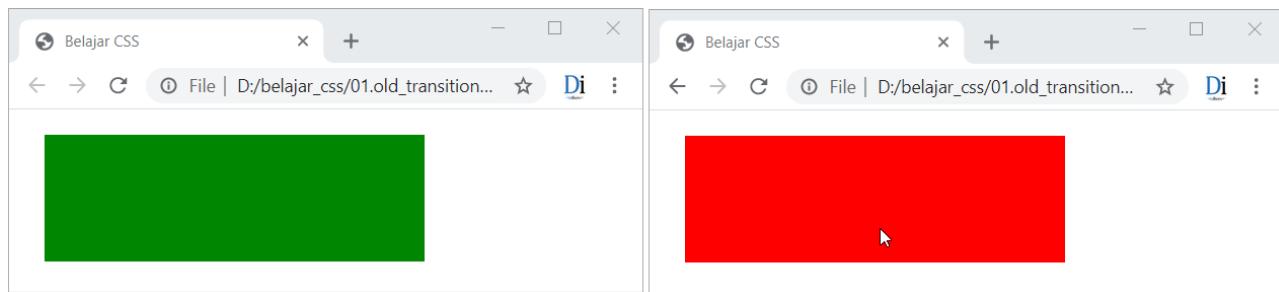
01.old_transitions.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu {
8              width: 300px;
```

```

9      height: 100px;
10     margin: 20px;
11     background-color: green;
12   }
13   .satu:hover {
14     background-color: red;
15   }
16 </style>
17 </head>
18 <body>
19   <div class="satu"></div>
20 </body>
21 </html>

```



Gambar: Efek transisi perubahan warna pada CSS 2.1

Di sini saya menggunakan selector `:hover` kepada element `.satu`. Ketika cursor mouse berada di atas kotak, warna background langsung berganti menjadi merah, namun begitu cursor mouse keluar dari element, akan kembali berwarna hijau. Inilah efek transisi di CSS 2.1.

Pada CSS3 Transitions, kita bisa membuat transisi yang lebih halus, dimana perubahan warna dari hijau ke merah akan berlangsung secara perlahan (gradual):

02.css3_transitions.html

```

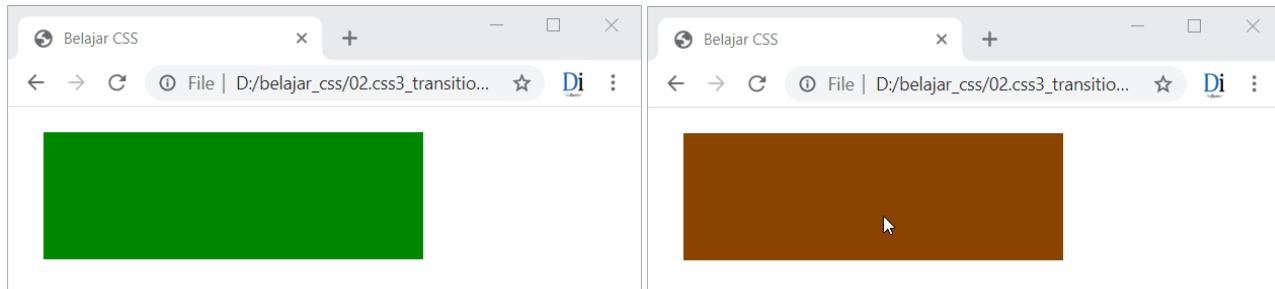
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      .satu {
8        width: 300px;
9        height: 100px;
10       margin: 20px;
11       background-color: green;
12
13       transition-duration: 1s;
14     }
15     .satu:hover {
16       background-color: red;
17     }
18   </style>
19 </head>

```

```

20 <body>
21   <div class="satu"></div>
22 </body>
23 </html>

```



Gambar: Efek transisi perubahan dari warna hijau ke merah secara berlahan

Tambahannya ada di baris 13, yakni dari property transition-duration: 1s. Penjelasan lebih lengkap tentang property ini akan kita bahas sesaat lagi. Tapi silahkan test jalankan kode tersebut, transisi dari warna hijau ke warna merah menjadi bertahap dan lebih menarik.

Selain itu, ketika cursor mouse meninggalkan box, warna background merah juga kembali secara perlahan ke warna hijau. Inilah efek transisi di dalam CSS3.

Karena sifat transisi dan animasi yang dinamis, saya sarankan untuk langsung praktik menjalankan kode-kode yang ada.

Pengertian Event

Untuk memulai sebuah transisi (dan juga animasi), kita memerlukan **event**. Secara sederhana, event adalah suatu 'kejadian' yang dialami element HTML. Sebagai contoh, ketika sebuah kotak di-klik, bisa dikatakan bahwa kotak tersebut mengalami event click. Ketika kotak di double-klik, maka element tersebut mengalami event double-click.

Istilah event sebenarnya lebih pas di bahas dalam JavaScript, karena dengan JavaScript-lah kita bisa membuat kode program yang bisa berjalan ketika suatu event terjadi.

Tanpa JavaScript, event yang tersedia sangat terbatas. Salah satu yang bisa kita pakai di CSS adalah event **mouseover**, yakni event ketika cursor mouse berada di atas sebuah element. Ini bisa diakses menggunakan selector :hover.

Selector :hover inilah yang akan selalu saya gunakan untuk membuat efek transisi dan animasi sepanjang bab ini.

16.2. Property transition-property

Untuk membuat efek transisi di dalam CSS3, kita butuh 3 syarat minimal:

- ✓ Style awal sebelum transisi

- ✓ Style akhir setelah transisi
- ✓ Durasi transisi

Property `transition-property` berfungsi untuk membuat apa saja property yang mendapat efek transisi. Nilai yang bisa diberikan adalah `all`, `none` atau beberapa nama property yang dipisah dengan koma.

Sebagai contoh, untuk membuat efek transisi warna background pada selector `.satu`, bisa ditulis sebagai berikut:

```

1 .satu {
2   background-color: green;
3   transition-property: background-color;
4 }
5 .satu:hover {
6   background-color: red;
7 }
```

Hasilnya, efek transisi hanya berlaku untuk property `background-color` saja.

Sekarang bagaimana dengan kode berikut ini?

```

1 .satu {
2   background-color: green;
3   color: white;
4   transition-property: background-color;
5 }
6 .satu:hover {
7   background-color: red;
8   color: black;
9 }
```

Tetap, efek transisi hanya berlaku pada `background-color` saja. Untuk property `color` akan langsung berubah dari warna putih menjadi hitam (tidak mendapat efek transisi).

Nilai default dari property `transition-property` adalah `all`, yang berarti semua property sebelum dan sesudah event akan ber-transisi, termasuk jika property ini tidak ditulis.

Mayoritas property CSS bisa dipakai untuk transisi dan animasi (tapi tidak semua). Di dalam modul W3C, property yang mendukung transisi dan animasi disebut sebagai *animatable properties*. Daftar lengkapnya bisa dilihat ke [MDN Web Docs: Animatable CSS properties²⁴](#).

16.3. Property `transition-duration`

Property `transition-duration` berfungsi untuk menentukan durasi efek transisi. Nilai yang bisa diinput adalah angka dalam satuan s (*second*), atau ms (*millisecond*). Sebagai tambahan, 1

²⁴ https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties

detik (1 second) sama dengan 1000 milidetik (1000 millisecond).

Misalnya untuk membuat efek transisi warna background dalam durasi 1 detik, bisa ditulis sebagai berikut:

```

1 .satu {
2   background-color: green;
3   transition-property: background-color;
4   transition-duration: 1s;
5 }
6 .satu:hover {
7   background-color: red;
8 }
```

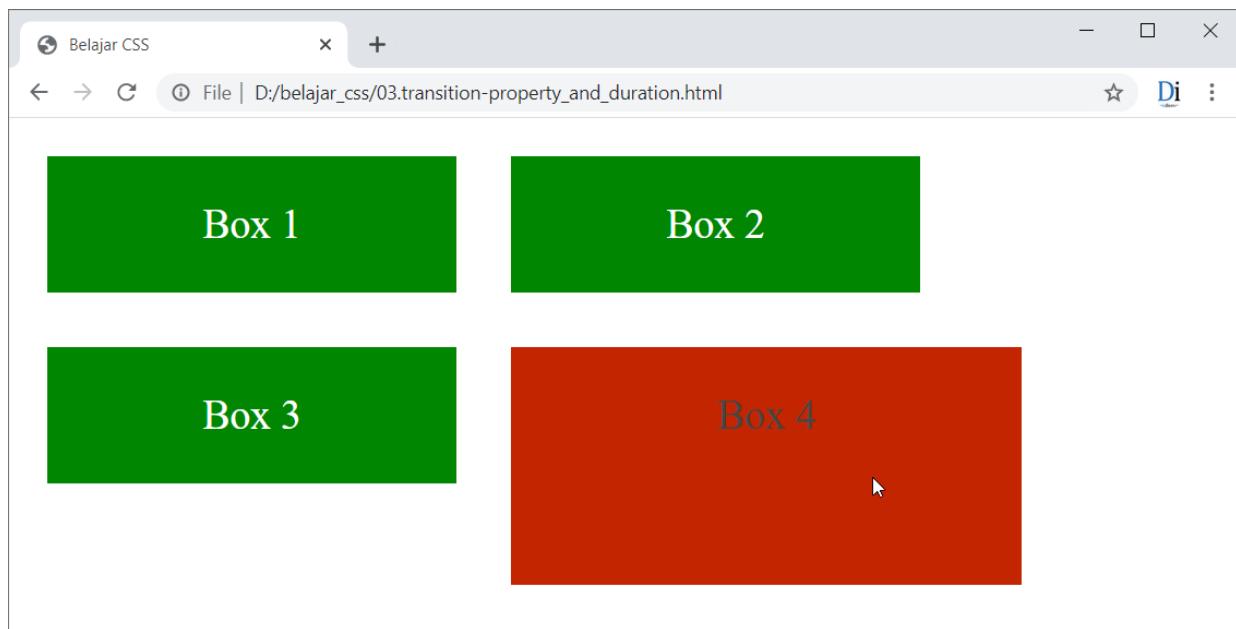
Dengan menggabung property **transition-property** dan **transition-duration**, kita sudah bisa membuat berbagai efek transisi:

03.transition-property_and_duration.html

```

1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .satu, .dua, .tiga, .empat {
8       width: 300px;
9       height: 100px;
10      background-color: green;
11      float: left;
12      margin: 20px;
13      color: white;
14      font-size: 30px;
15      line-height: 100px;
16      text-align: center;
17    }
18
19   /* Tanpa efek transisi*/
20   .satu:hover {
21     background-color: red;
22   }
23
24   /* Dengan efek transisi untuk background-color; */
25   .dua {
26     transition-property: background-color;
27     transition-duration: 1s;
28   }
29   .dua:hover {
30     background-color: red;
31   }
32
33   /* Dengan efek transisi untuk background-color dan color; */
34   .tiga {
```

```
35     transition-property: color, background-color;
36     transition-duration: 2000ms;
37 }
38 .tiga:hover {
39     background-color: red;
40     color: black;
41 }
42
43 /* Dengan efek transisi untuk semua property */
44 .empat {
45     transition-property: all;
46     transition-duration: 1s;
47 }
48 .empat:hover {
49     background-color: red;
50     color: black;
51     width: 400px;
52     height: 200px;
53 }
54 </style>
55 </head>
56 <body>
57     <div class="satu">Box 1</div>
58     <div class="dua">Box 2</div>
59     <div class="tiga">Box 3</div>
60     <div class="empat">Box 4</div>
61 </body>
62 </html>
```



Gambar: Efek transisi menggunakan property transition-property dan transition-duration

Kode di atas cukup panjang karena saya membuat 4 kotak dengan berbagai efek transisi. Juga karena keterbatasan media, tidak bisa menampilkan semua efek yang ada.

Pada box 1, tidak ada efek transisi. Saya hanya menambahkan event :hover untuk membuat

warna background berubah dari warna hijau ke merah tepat saat cursor mouse berada di atasnya.

Pada box 2, terdapat tambahan property `transition-property: background-color` dan `transition-duration: 1s`. Hasilnya, efek transisi warna background dilakukan dalam waktu 1 detik.

Pada box 3, terdapat 2 efek transisi, yakni `background-color` untuk warna background dan `color` untuk warna teks. Keduanya akan diproses dalam waktu 2000ms atau 2 detik.

Pada box 4 saya menambah beberapa property lain. Dengan membuat `transition-property: all`, semua *animatable properties* akan di transisikan selama 1 detik. Kita cukup tulis efek akhir di selektor `:hover`. Transisi akan terjadi untuk 4 jenis property: `background-color`, `color`, `width`, dan `height`.

Silahkan anda coba tambah property-property lain, dan lihat hasilnya.

Satu hal yang bisa diperhatikan, keempat transisi akan kembali ke posisi awal dengan cara membalik efek yang ada. Misalnya saat kita memindahkan cursor mouse dari box 4, maka ukuran box akan kembali seperti semula secara perlahan.

16.4. Property transition-timing-function

Property `transition-timing-function` berfungsi untuk menentukan bagaimana efek transisi berjalan. Terdapat beberapa nilai keyword untuk property ini: `linear`, `ease`, `ease-in`, `ease-out`, dan `ease-in-out`. Bagi yang pernah membuat animasi dengan aplikasi Adobe Flash, tentu sudah familiar dengan nilai-nilai ini.

Berikut penjelasan masing-masing nilai dari property `transition-timing-function`:

- **Linear:** efek transisi berlangsung merata mulai dari awal hingga akhir.
- **Ease:** efek transisi mulai dengan pelan, kemudian dipercepat, dan kembali pelan di akhir proses. Ini adalah nilai default jika property `transition-timing-function` tidak dituliskan.
- **Ease-in:** efek transisi mulai dengan pelan, kemudian dipercepat hingga akhir proses.
- **Ease-out:** efek transisi mulai dengan cepat, kemudian diperlambat hingga akhir proses.
- **Ease-in-out:** efek transisi mulai dengan pelan, kemudian dipercepat, dan kembali pelan di akhir proses. Ini mirip dengan efek ease, tetapi tidak se-dramatis ease.

Mari kita lihat praktik penggunaannya:

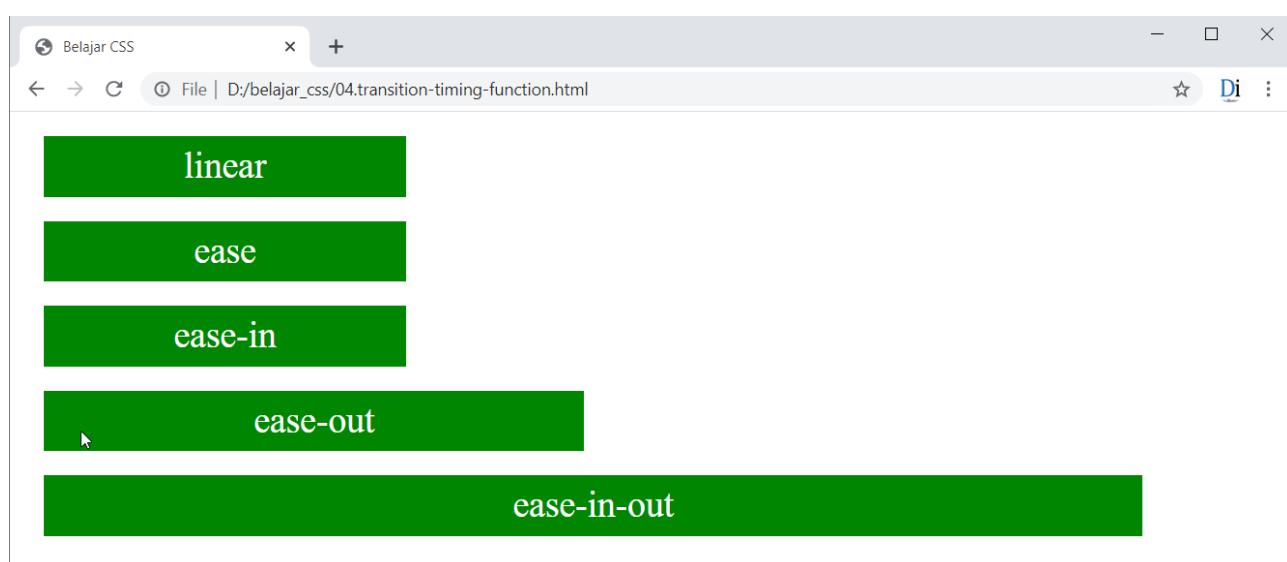
04.transition-timing-function.html

```
1 <!DOCTYPE html>
2 <html lang="id">
```

```

3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .satu, .dua, .tiga, .empat, .lima {
8       width: 300px;
9       height: 50px;
10      background-color: green;
11      margin: 20px;
12      color: white;
13      font-size: 30px;
14      line-height: 50px;
15      text-align: center;
16
17      transition-duration: 1s;
18    }
19    .satu { transition-timing-function: linear; }
20    .dua { transition-timing-function: ease; }
21    .tiga { transition-timing-function: ease-in; }
22    .empat { transition-timing-function: ease-out; }
23    .lima { transition-timing-function: ease-in-out; }
24
25    .satu:hover, .dua:hover, .tiga:hover,
26    .empat:hover, .lima:hover {
27      width: 1000px;
28    }
29  </style>
30 </head>
31 <body>
32   <div class="satu">linear</div>
33   <div class="dua">ease</div>
34   <div class="tiga">ease-in</div>
35   <div class="empat">ease-out</div>
36   <div class="lima">ease-in-out</div>
37 </body>
38 </html>

```



Gambar: Berbagai efek transition-timing-function

Kali ini saya membuat 5 box dengan efek `transition-timing-function` yang berbeda-beda.

Selain kelima nilai keyword ini, property `transition-timing-function` masih mendukung 2 nilai lain, yakni `cubic bézier curve`, dan `steps function`. Namun tidak saya bahas karena penggunaannya cukup rumit. Jika tertarik bisa dibaca-baca ke [MDN Web Docs <easing-function>](#)²⁵.

16.5. Property transition-delay

Property `transition-delay` berfungsi untuk membuat jeda waktu sebelum efek transisi berlangsung. Nilai yang bisa diinput adalah angka dalam satuan second (s) atau millisecond (ms). Berikut contoh penggunaannya:

05.transition-delay.html

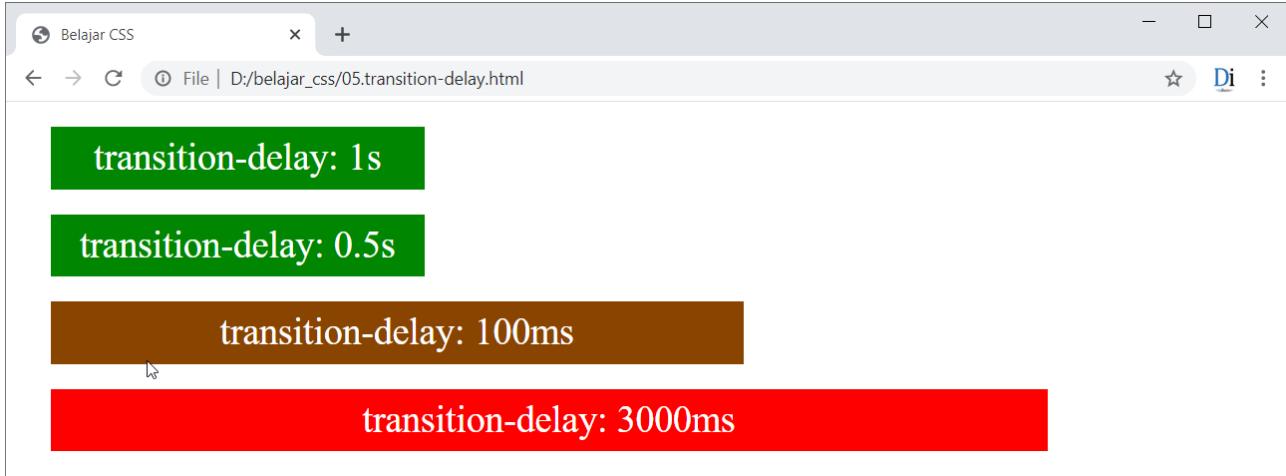
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua, .tiga, .empat {
8              width: 300px;
9              height: 50px;
10             background-color: green;
11             margin: 20px;
12             color: white;
13             font-size: 30px;
14             line-height: 50px;
15             text-align: center;
16
17             transition-duration: 1s;
18         }
19
20         .satu {transition-delay: 1s;}
21         .dua {transition-delay: 0.5s;}
22         .tiga {transition-delay: 100ms; }
23         .empat {transition-delay: 3000ms; }
24
25         .satu:hover, .dua:hover,
26         .tiga:hover, .empat:hover {
27             width: 800px;
28             background-color: red;
29         }
30     </style>
31 </head>
32 <body>
33     <div class="satu">transition-delay: 1s</div>

```

²⁵ <https://developer.mozilla.org/en-US/docs/Web/CSS/easing-function>

```
34 <div class="dua">transition-delay: 0.5s</div>
35 <div class="tiga">transition-delay: 100ms</div>
36 <div class="empat">transition-delay: 3000ms</div>
37 </body>
38 </html>
```



Gambar: Contoh penggunaan property transition-delay

Dalam contoh ini saya memakai berbagai nilai `transition-delay`. Kita harus menunggu beberapa saat sebelum efek transisi muncul (sesuai dengan nilai waktu yang diinput).

Jeda waktu ini juga berlaku setelah efek transisi. Jika kita men-set delay 3 detik sebelum efek transisi terjadi, maka kita juga harus menunggu 3 detik untuk element tersebut kembali ke bentuknya semula.

16.6. Property transition (shorthand)

Dari awal bab hingga sekarang, kita telah membahas 4 property transisi, yakni:

- ◆ `transition-property`
- ◆ `transition-duration`
- ◆ `transition-timing-function`
- ◆ `transition-delay`

Untuk menyederhanakan penulisan, terdapat versi shorthand notation dengan format penulisan sebagai berikut:

```
transition: property duration timing-function delay;
```

Sebagai contoh, `transition: background-color 2s ease-out 250ms` sama artinya dengan:

```
1 transition-property: background-color;
2 transition-duration: 2s;
3 transition-timing-function: ease-out;
4 transition-delay: 250ms;
```

Berikut contoh praktik penggunaan property transition shorthand:

06.transition_shorthand.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua, .tiga, .empat {
8              width: 300px;
9              height: 50px;
10             background-color: green;
11             margin: 20px;
12         }
13
14         .satu { transition: all 1s ease-out 250ms; }
15         .dua { transition: width 1.5s linear 100ms; }
16         .tiga { transition: background-color 0.5s ease-in-out 1000ms; }
17         .empat { transition: none 2s ease-out 250ms; }
18
19         .satu:hover, .dua:hover,
20         .tiga:hover, .empat:hover {
21             width: 1000px;
22             background-color: red;
23         }
24     </style>
25 </head>
26 <body>
27     <div class="satu"></div>
28     <div class="dua"></div>
29     <div class="tiga"></div>
30     <div class="empat"></div>
31 </body>
32 </html>
```



Gambar: Hasil penggunaan property transition shorthand

Dalam keempat element di atas, saya membuat 2 property transisi, yakni `width` dan `background-color`.

Untuk box pertama, kedua property ini mengalami efek transisi.

Untuk box kedua, property yang akan ditransisikan hanya `width` saja. Oleh karena itu ketika event `:hover` terjadi, warna background langsung berubah menjadi merah, sedangkan lebar element (`width`) akan bertransisi selama 1,5 detik.

Untuk box ketiga, adalah kebalikannya. Kali ini saya mentransisikan `background-color`, sehingga saat event `:hover` terjadi, lebar box akan langsung menjadi 1000px, tetapi warna background bertransisi selama 0,5 detik setelah delay 1 detik.

Untuk box keempat, tidak ada efek transisi sama sekali. Kenapa? karena saya memberikan nilai `none` untuk `transition-property`. Dengan kata lain, tidak ada property yang diizinkan bertransisi.

16.7. Multiple Transition

Contoh yang kita bahas sebelum ini hanya terdiri dari 1 proses transisi, dan semuanya terjadi dalam 1 waktu. CSS3 mendukung pemisahan setiap property agar berjalan pada waktu yang berbeda-beda.

Sebagai contoh, kita bisa membuat transisi `background` berjalan selama 1 detik, transisi `width` berjalan selama 5 detik, serta transisi `color` selama 10 detik. Setiap efek transisi bisa ditulis pada 1 property `transition` yang dipisah dengan tanda koma:

```
transition: background-color 1s, width 5s, color 10s;
```

Kode di atas akan menjalankan efek transisi pada waktu yang berbeda-beda. Berikut contoh prakteknya:

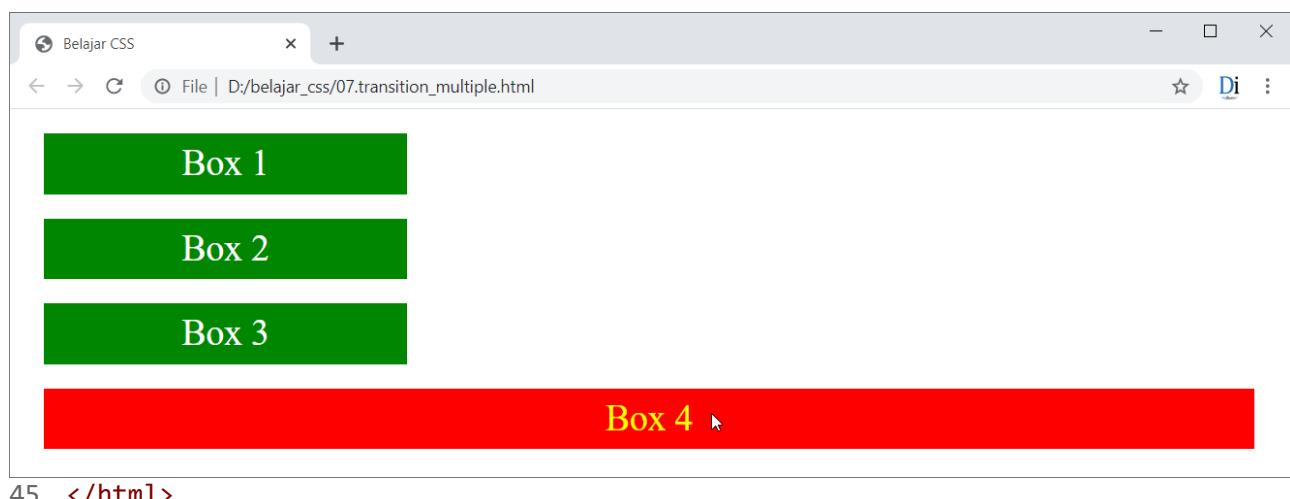
07.transition_multiple.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua, .tiga, .empat {
8              width: 300px;
9              height: 50px;
10             background-color: green;
11             margin: 20px;
12             color: white;
13             font-size: 30px;
14             line-height: 50px;
15             text-align: center;
```

```

16      }
17
18      .satu:hover, .dua:hover,
19      .tiga:hover, .empat:hover    {
20          width: 1000px;
21          background-color: red;
22          color: yellow;
23      }
24
25      .satu {
26          transition: all 2s ease-out 250ms;
27      }
28      .dua {
29          transition: width 1s, background-color 10s;
30      }
31      .tiga {
32          transition: width 1s, background-color 5s, color 10s ;
33      }
34      .empat {
35          transition: width 100ms 1s, background-color 1s 3s, color 100ms 5s;
36      }
37  </style>
38 </head>
39 <body>
40     <div class="satu">Box 1</div>
41     <div class="dua">Box 2</div>
42     <div class="tiga">Box 3</div>
43     <div class="empat">Box 4</div>
44 </body>

```



45 </html>

Gambar: Hasil penggunaan multiple transition

Untuk box pertama, `transition: all 2s ease-out 250ms` berarti saya ingin membuat efek transisi untuk semua property (all) selama 2 detik (2s) dengan efek ease-out yang baru dimulai setelah 250ms. Tidak ada yang baru di sini.

Pada box kedua, `transition: width 1s, background-color 10s` berarti saya ingin transisi

width berjalan selama 1 detik, dan transisi background-color berjalan selama 10 detik. Karena nilai efek tidak ditulis, maka akan memakai nilai default (ease). Hasilnya ketika box kedua sudah melebar, perubahan warna akan terus berjalan hingga 10 detik.

Pada box ketiga, `transition: width 1s, background-color 5s, color 10s` mirip dengan box kedua, tapi saya menambah property ketiga, yakni color yang transisinya berlangsung selama 10 detik.

Untuk box keempat, `transition: width 100ms 1s, background-color 1s 3s, color 100ms 5s` berarti ketika mouse berada di atas box 4, efek pertama yang akan di proses adalah width, tapi baru berjalan setelah 1 detik. Box kemudian melebar selama 100ms. Setelah detik ke-4, warna background akan berganti. Terakhir di detik ke 5 barulah warna teks ikut berubah.

Efek transisi yang kita praktekkan di sini hanya versi sederhana. Silahkan berkreasi dengan berbagai property lain. Namun jika butuh control yang lebih kepada efek transisi yang terjadi, saatnya masuk ke **animation**.

16.8. CSS3 Animations

Secara umum, property CSS3 animation hampir mirip seperti transition. Bedanya, jika di transisi kita hanya bisa mengatur efek awal dan akhir saja, maka di animasi bisa mengatur setiap tahap yang terjadi.

Animations Keyframe

Untuk membuat animasi di CSS, kita harus merangkai apa saja efek yang terjadi dan kapan waktu terjadinya. Waktu perubahan ini dikenal dengan istilah **keyframe**.

Sebagai contoh, saya ingin membuat efek animasi perubahan lebar sebuah element dengan alur sebagai berikut: saat awal animasi lebarnya 300px, kemudian mengecil menjadi 100px, lalu di akhir animasi melebar menjadi 1000px. Langkah-langkah ini bisa dipecah menjadi 3 keyframe:

```
0%   -> width: 300px;
50%  -> width: 100px;
100% -> width: 1000px;
```

Angka persen di sebelah kiri menunjukkan kapan efek tersebut terjadi. Mulai dari 0% untuk awal animasi, hingga 100% di akhir animasi. Berikut hasilnya saat di konversi menjadi kode CSS:

```
@keyframes lebar {
  0%  { width: 300px; }
  50% { width: 100px; }
  100% { width: 1000px; }
}
```

Perintah `@keyframes` memberitahu web browser bahwa kode ini merupakan kumpulan `keyframe` untuk membuat animasi. Setelah itu, kita harus menyertakan 'label' atau 'nama' untuk kumpulan keyframe ini. Dalam contoh di atas, **lebar** adalah nama keyframe yang saya buat. Nama keyframe ini bebas diganti dengan nama lain.

Sebagai contoh lain, saya ingin membuat efek perubahan warna background dari merah, ke biru, ke kuning, lalu menjadi hitam. Karena ada 4 perubahan, kita tinggal memecahnya menjadi 4 keyframe:

```
@keyframes warna {
    0% { background-color: red; }
    30% { background-color: blue; }
    80% { background-color: yellow; }
    100% { background-color: black; }
}
```

Angka persen di sisi kiri bisa di edit sesuka kita, selama nilainya antara 0% hingga 100%.

Lalu, bagaimana cara menggunakan CSS keyframe ini? Kita butuh setidaknya 2 property lain: `animation-name` dan `animation-duration`.

16.9. Property `animation-name`

Property `animation-name` berfungsi untuk menghubungkan keyframe dengan CSS selector. Sebagai contoh, keyframe "lebar" yang telah didefinisikan sebelumnya bisa kita terapkan ke class `.satu:hover` sebagai berikut:

```
.satu:hover {
    animation-name: lebar;
}
```

16.10. Property `animation-duration`

Property `animation-duration` berfungsi untuk menentukan durasi sebuah animasi. Satuannya dalam bentuk second (s) atau millisecond (ms):

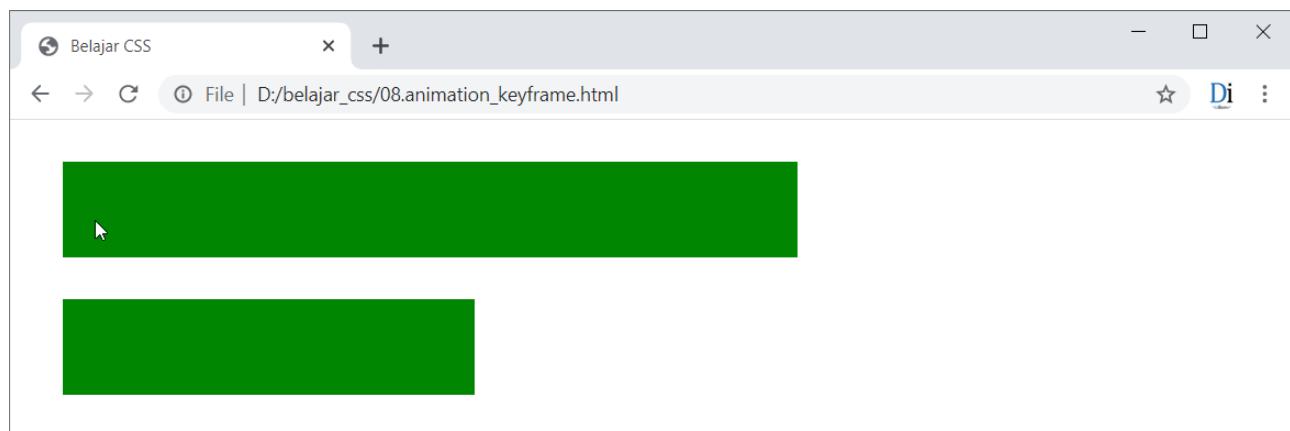
```
.satu:hover {
    animation-name: lebar;
    animation-duration: 5s;
}
```

Dengan menggabungkan keyframe, `animation-name` dan `animation-duration`, kita sudah bisa membuat efek animasi lengkap menggunakan CSS. Berikut contoh prakteknya:

08.animation_keyframe.html

```
1  <!DOCTYPE html>
```

```
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .satu, .dua{
8       width: 300px;
9       height: 70px;
10      background-color: green;
11      margin: 30px;
12    }
13
14   @keyframes lebar {
15     0% { width: 300px; }
16     50% { width: 100px; }
17     100% { width: 1000px; }
18   }
19   .satu:hover {
20     animation-name: lebar;
21     animation-duration: 5s;
22   }
23
24   @keyframes warna {
25     0% { background-color: red; }
26     30% { background-color: blue; }
27     80% { background-color: yellow; }
28     100% { background-color: black; }
29   }
30   .dua:hover {
31     animation-name: warna;
32     animation-duration: 5s;
33   }
34 </style>
35 </head>
36 <body>
37   <div class="satu"></div>
38   <div class="dua"></div>
39 </body>
40 </html>
```



Gambar: Akhirnya, sebuah animasi dengan CSS

Saya menggabung 2 efek animasi dari keyframe yang telah kita buat sebelumnya. Silahkan anda coba jalankan kode di atas dan letakkan cursor mouse pada masing-masing kotak.

Satu hal yang langsung bisa terlihat, kotak tersebut akan kembali ke bentuk asalnya tanpa efek mundur sebagaimana transisi. Setelah animasi selesai, animasi akan langsung terputus dan kembali ke bentuk awal.

Jika kita ingin agar pembalikan ini terjadi secara bertahap, bisa dengan cara membuat keyframe terakhir lebih kecil dari 100%, misalnya menjadi 95% atau kurang. Makin kecil angkanya, makin perlahan juga efek pembalikan yang terjadi:

09.animation_keyframe2.html

```

1  @keyframes lebar {
2    0% { width: 300px; }
3    40% { width: 100px; }
4    80% { width: 1000px; }
5  }
6
7  @keyframes warna {
8    0% { background-color: red; }
9    30% { background-color: blue; }
10   60% { background-color: yellow; }
11   90% { background-color: black; }
12 }
```

16.11. Property animation-timing-function

Property `animation-timing-function` mirip seperti `transition-timing-function`, yakni untuk mengatur bagaimana proses animasi berlangsung. Apakah itu `linear`, `ease`, `ease-in`, `ease-out`, atau `ease-in-out`. Berikut contoh penggunaannya:

```
.satu {
  animation-timing-function: ease;
}
```

Karena efeknya sama persis seperti di transisi, tidak akan saya bahas lagi.

16.12. Property animation-delay

Property ini juga mirip dengan `transition-delay`, yakni berfungsi untuk memberikan jeda sebelum efek animasi. Nilai yang bisa diinput berupa satuan second (s) atau millisecond (ms). Berikut contoh penggunaannya:

```
.satu {
  animation-delay: 2s;
}
```

16.13. Property animation-iteration-count

Tidak seperti transisi yang hanya berjalan sekali, efek animasi bisa di ulang beberapa kali, dan ini di set melalui property `animation-iteration-count`.

Nilai yang bisa diisi berupa angka yang menentukan berapa kali animasi terjadi, atau juga bisa diisi dengan keyword `infinite` agar animasi terus berlangsung selamanya (hingga halaman ditutup).

Sebagai contoh, untuk membuat animasi berlangsung sebanyak 5 kali, bisa menggunakan kode CSS berikut:

```
.satu {
  animation-iteration-count: 5;
}
```

16.14. Property animation-direction

Animasi sebelumnya hanya bergerak dari awal ke akhir, kemudian selesai. Tapi kita bisa mengontrol efek ini melalui property `animation-direction`. Property ini bisa diisi dengan nilai keyword `normal` atau `alternate`.

Apabila diberikan nilai `normal`, animasi akan berjalan dari awal hingga akhir. Jika kita mengulang animasi (dengan menambah property `animation-iteration-count`), animasi tersebut kembali di ulang dari awal. Ini merupakan nilai default jika property `animation-direction` tidak ditulis.

Jika diberikan nilai `alternate`, animasi juga akan berjalan dari awal hingga akhir, namun untuk perulangan kedua, animasi akan mundur, yakni dari efek terakhir kembali ke bentuk semula. Untuk perulangan ketiga akan kembali mulai dari awal, dst.

Sebagai contoh, perhatikan property berikut:

```
.satu {
  animation-iteration-count: 3;
  animation-direction: normal;
}
```

Efek yang dihasilkan akan di putar sebanyak 3 kali, dimana setiap putarannya dimulai dari awal hingga akhir. Namun jika ditulis menjadi:

```
.satu {
  animation-iteration-count: 3;
  animation-direction: alternate ;
}
```

Animasi juga akan diputar sebanyak 3 kali. Pada putaran pertama akan mulai dari awal hingga

akhir. Namun untuk putaran kedua animasi diputar mundur dari akhir hingga kembali ke awal, dan pada putaran ketiga kembali lagi dari awal ke akhir.

Berikut contoh praktik dari 4 property yang kita bahas sebelum ini: `animation-timing-function`, `animation-delay`, `animation-iteration-count`, dan `animation-direction`:

10.animation-timing_delay_count_iteration.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua, .tiga, .empat, .lima{
8              width: 400px;
9              height: 50px;
10             background-color: green;
11             margin: 20px;
12             color: white;
13             font-size: 20px;
14             line-height: 50px;
15             padding-left: 10px;
16         }
17
18         @keyframes lebar-warna {
19             25% { width: 500px; background-color: red; }
20             50% { width: 200px; background-color: black; }
21             100% { width: 900px; background-color: pink; }
22         }
23
24         .satu:hover,.dua:hover, .tiga:hover,.empat:hover,.lima:hover {
25             animation-name: lebar-warna;
26             animation-duration: 5s;
27         }
28
29         .satu {
30             animation-timing-function: linear;
31         }
32         .dua {
33             animation-timing-function: ease;
34             animation-delay: 2s;
35         }
36         .tiga {
37             animation-timing-function: ease-in;
38             animation-delay: 2s;
39             animation-iteration-count: 3;
40         }
41         .empat {
42             animation-timing-function: ease-out;
43             animation-delay: 2s;
44             animation-iteration-count: 3;
45             animation-direction: normal;
46         }

```

```

47     .lima {
48         animation-timing-function: ease-in-out;
49         animation-delay: 2s;
50         animation-iteration-count: 3;
51         animation-direction: alternate;
52     }
53 
```

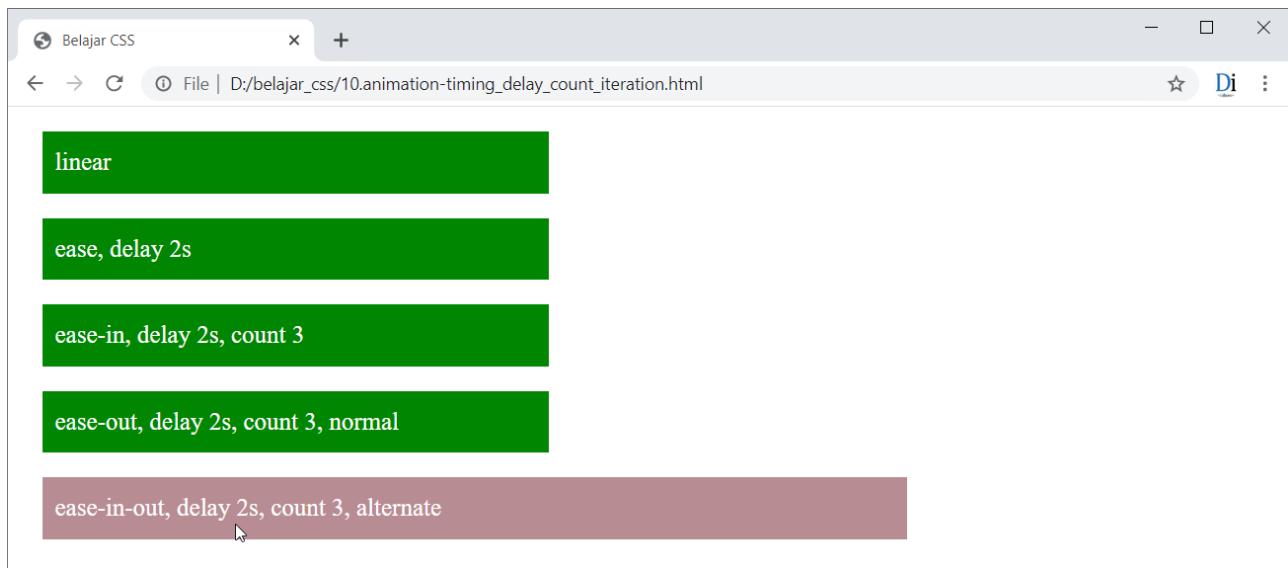
```
</style>
```

```
</head>
```

```
<body>
```

```
56     <div class="satu">linear</div>
```

```
57     <div class="dua">ease, delay 2s</div>
58     <div class="tiga">ease-in, delay 2s, count 3</div>
59     <div class="empat">ease-out, delay 2s, count 3, normal</div>
60     <div class="lima">ease-in-out, delay 2s, count 3, alternate</div>
61 </body>
62 </html>
```



Gambar: Contoh penggunaan 4 property animasi

Di baris 18 saya membuat keyframe "lebar-warna" yang terdiri dari 3 keyframe, yakni di titik 25%, 50%, dan 75%. Setiap keyframe akan mengubah lebar (`width`) dan warna background (`background-color`).

Event `:hover` kemudian di set untuk kelima box di baris 24. Animasi yang dipakai adalah keyframe "lebar-warna" yang berjalan selama 5 detik.

Pada box pertama hanya terdapat 1 property, yakni `animation-timing-function: linear`. Dengan demikian kecepatan animasi dibuat seragam.

Pada box kedua, terdapat property `animation-timing-function: ease` dan `animation-delay: 2s`. Maka animasi akan dipercepat di tengah-tengah proses (ease), dan baru berjalan setelah delay 2 detik.

Pada box ketiga, animasi akan di ulang sebanyak 3 kali dengan property `animation-iteration-count: 3` dan efek `ease-in`. Agar bisa melihat ketiga animasi ini, cursor mouse harus selalu

berada di atas box supaya event :hover tetap aktif.

Animasi box keempat mirip seperti box ketiga, namun dengan efek ease-out. Tambahan property `animation-direction: normal` tidak berpengaruh apa-apa karena secara default nilai inilah yang dipakai. Efeknya animasi akan di ulang mulai dari awal untuk setiap perulangan.

Pada box kelima, animasi dijalankan 3 kali dengan efek ease-in-out, delay 2 detik, dan berjalan mundur. Ini terjadi karena penambahan property `animation-direction: alternate`. Hasilnya, untuk perulangan kedua animasi akan mundur secara perlahan ke bentuk awal.

16.15. Property `animation-fill-mode`

Property `animation-fill-mode` dipakai untuk menentukan bagaimana 'style' element sebelum dan sesudah animasi. Sebagai contoh, perhatikan potongan kode CSS berikut:

```

1  .satu{
2      width: 400px;
3      height: 50px;
4      background-color: green;
5
6      animation-timing-function: linear;
7  }
8
9  @keyframes lebar-warna {
10     0% { width: 500px; background-color: red; }
11     50% { width: 200px; background-color: black; }
12     100% { width: 900px; background-color: pink; }
13 }
14
15 .satu:hover{
16     animation-name: lebar-warna;
17     animation-duration: 5s;
18     animation-delay: 2s;
19 }
```

Jika mengabaikan keyframe, dapat disimpulkan bahwa element `.satu` akan berwarna hijau dan memiliki lebar 400px.

Namun di dalam keyframe paling awal (0%) saya men-set warna element menjadi red dan lebar 500px. Jadi nilai apa yang dipakai di awal animasi? Perhatikan juga terdapat jeda selama 2 detik sebelum animasi dimulai (`animation-delay: 2s`).

Secara default ketika event :hover di panggil, element tersebut akan berwarna hijau selama 1 detik (terdapat delay), kemudian di detik ke-2 langsung berubah menjadi merah dan animasi dimulai. Setelah 3 detik, animasi selesai dan element langsung kembali ke bentuk awal (warna hijau dan lebar 400px). Efek ini sama dengan maksud property `animation-fill-mode: none`.

Nilai `none` dari property `animation-fill-mode` berarti element akan berpindah ke style animasi hanya selama animasi berlangsung. Di luar itu, akan memakai style awal sebelum animasi.

Nilai lain untuk `animation-fill-mode` adalah `backwards`, `forwards`, atau `both`.

Jika kita memakai `animation-fill-mode:backwards`, maka sebelum animasi berlangsung, yang digunakan adalah style yang ada di dalam keyframe.

Jika menggunakan `animation-fill-mode:forwards`, maka saat animasi selesai, style yang dipakai adalah style di akhir keyframe, tidak kembali ke style awal element.

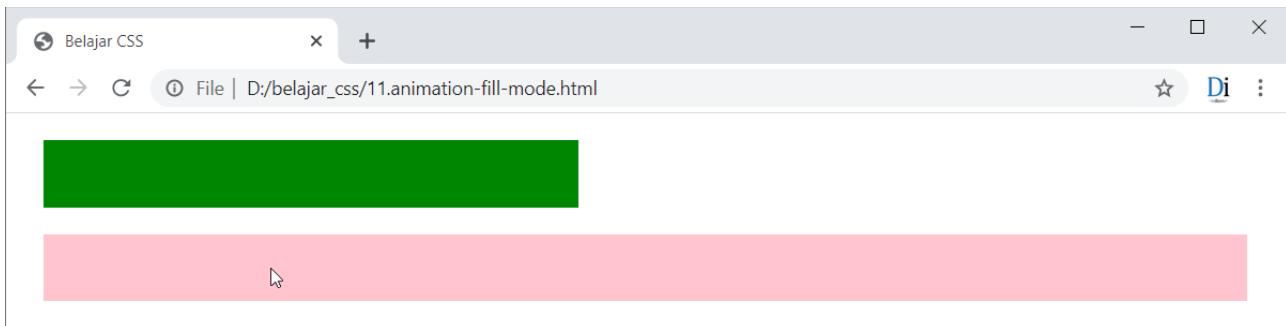
Sedangkan nilai `animation-fill-mode:both` adalah gabungan dari `backwards` dan `forwards`.

Agar lebih memahami pengertian ini, langsung saja kita masuk ke contoh kode program:

11.animation-fill-mode.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .satu, .dua{
8              width: 400px;
9              height: 50px;
10             background-color: green;
11             margin: 20px;
12         }
13
14         @keyframes lebar-warna {
15             0% { width: 500px; background-color: red; }
16             50% { width: 200px; background-color: black; }
17             100% { width: 900px; background-color: pink; }
18         }
19
20         .satu:hover{
21             animation-name: lebar-warna;
22             animation-duration: 3s;
23             animation-delay: 2s;
24         }
25
26         .dua:hover{
27             animation-name: lebar-warna;
28             animation-duration: 3s;
29             animation-delay: 2s;
30             animation-fill-mode: both;
31         }
32     </style>
33 </head>
34 <body>
35     <div class="satu"></div>
36     <div class="dua"></div>
37 </body>
38 </html>
```



Gambar: Contoh perbedaan dari animation-fill-mode: none dengan animation-fill-mode: both

Box pertama saya buat tanpa property `animation-fill-mode`, yang artinya sama dengan `animation-fill-mode:none`. Ketika event `:hover` terjadi, box tetap berwarna hijau selama 2 detik, lalu animasi berjalan dan setelah selesai akan kembali ke bentuk semula.

Sedangkan box kedua menggunakan `animation-fill-mode:both`. Maka begitu cursor mouse berada di atas element, warna box langsung menjadi merah dan melebar ke 500px. Inilah style awal dari *keyframe*.

Setelah 1 detik, animasi akan berjalan dan ketika selesai tetap berwarna pink dengan lebar 1000px. Ini dengan catatan cursor mouse harus selalu di atas element agar event `:hover` tetap aktif. Begitu cursor mouse di jauhkan dari box kedua, barulah style kembali ke bentuk semula.

16.16. Property `animation-play-state`

Property `animation-play-state` benar-benar mengaburkan konsep CSS dan JavaScript karena berfungsi untuk mengontrol jalannya animasi.

Property ini bisa diisi dengan keyword `running` atau `paused`. Jika kita memakai `animation-play-state:paused`, maka animasi akan berhenti, sedangkan `animation-play-state:running` akan membuat animasi berjalan kembali, dengan catatan terdapat event yang memanggilnya.

Langsung saja kita masuk ke contoh praktek:

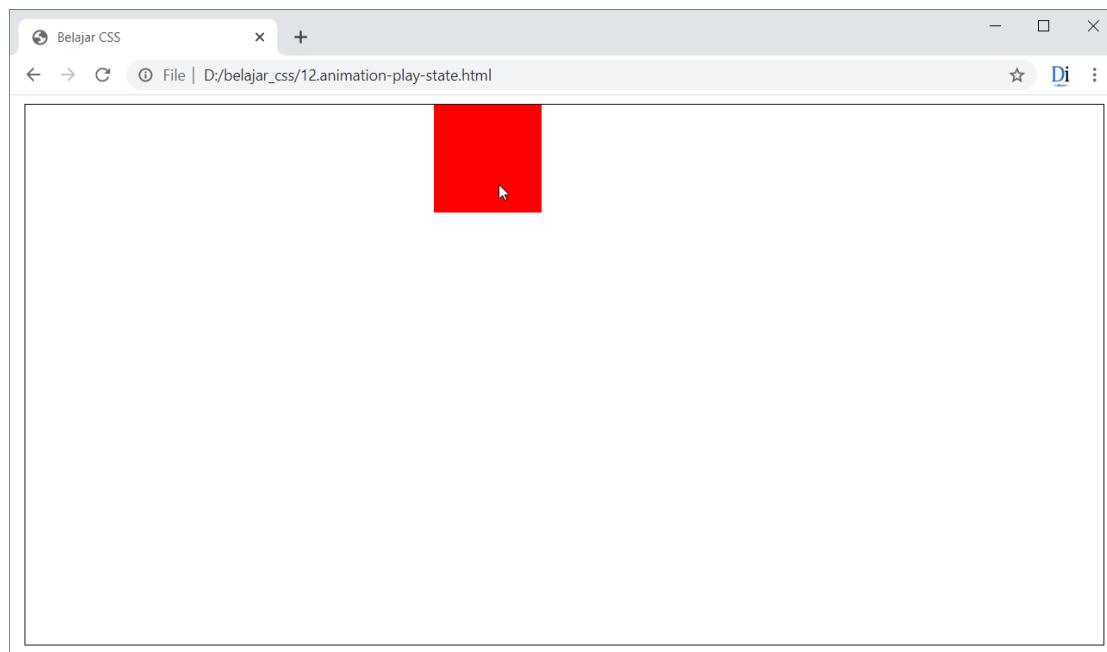
`12.animation-play-state.html`

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 1000px;
9              height: 500px;
10             margin: 0 auto;
11             border: 1px solid black;
12         }
13         .satu{

```

```
14     width: 100px;
15     height: 100px;
16     background-color: green;
17
18     position: relative;
19 }
20
21 @keyframes pindah2 {
22     0% { top: 0px; left:0px; }
23     25% { top: 00px; left:900px; }
24     50% { top: 400px; left:900px; }
25     75% { top: 400px; left:0px; }
26     100% { top: 0px; left:0px; }
27 }
28
29 .satu {
30     animation-name: pindah2;
31     animation-duration: 10s;
32     animation-iteration-count: infinite;
33 }
34 .satu:hover {
35     animation-play-state: paused;
36     background-color: red;
37 }
38
39 </style>
40 </head>
41 <body>
42     <div class="container">
43         <div class="satu"></div>
44     </div>
45 </body>
46 </html>
```



Gambar: Animasi akan berhenti ketika cursor mouse berada di atas element

Kali ini saya meng-animasikan property positioning `left`. Hasilnya, box akan bergerak berputar-putar di dalam container. Silahkan luangkan waktu sebentar untuk mempelajari apa yang terjadi di dalam setiap keyframe.

Selain itu saya tidak menggunakan event `:hover` untuk memulai animasi. Box akan langsung bergerak begitu halaman di load pertama kali, dan animasi terus berjalan karena terdapat property `animation-iteration-count: infinite`.

Ketika box mengalami event `:hover`, dua property berikut akan berjalan:

```
.satu:hover {
    animation-play-state: paused;
    background-color: red;
}
```

Ini berarti ketika cursor mouse berada di atas box, proses animasi akan berhenti dan warna background menjadi merah. Tetapi begitu cursor mouse tidak lagi di atas box (event `:hover` berhenti), animasi akan kembali berjalan karena secara default kembali ke `animation-play-state: running`.

16.17. Property animation shorthand

Total terdapat 8 jenis property animation yang telah kita pelajari. Sama seperti di iterasi, terdapat juga penulisan shorthand dari semua property animasi. Berikut urutan penulisannya:

```
animation: name duration timing-function delay iteration-count direction fill-mode
play-state;
```

Sebagai contoh, penulisan property ini:

```
.satu {
    animation: berubah-warna 6s ease-in 2s 10 alternate forwards running;
}
```

Sama artinya dengan:

```
.satu {
    animation-name: berubah-warna;
    animation-duration: 6s;
    animation-timing-function: ease-in;
    animation-delay: 2s;
    animation-iteration-count: 10;
    animation-direction: alternate;
    animation-fill-mode: forwards;
    animation-play-state: running;
}
```

Cukup panjang? memang, tapi sebenarnya kita tidak perlu menulis semua property di atas untuk membuat animasi. Sekurang-kurangnya, hanya perlu 2 property, yakni `animation-name`

dan `animation-duration`.

16.18. Multiple Animation

Jika satu efek animasi masih kurang, CSS mengizinkan kita untuk menambah efek animasi lanjutan ke dalam 1 penulisan property animation. Cara penggunaannya mirip multiple transition, dimana setiap animasi di pisah dengan tanda koma.

Sebagai contoh, jika saya memiliki 3 keyframe untuk 3 efek animasi, bisa digabung menjadi:

```
.satu {  
    animation: berubah-warna 6s, berubah-posisi 5s, berubah-bentuk 3s;  
}
```

Efek transisi dan animasi yang kita pelajari membuka banyak ruang kreatifitas di dalam CSS. Animasi bisa dibuat dengan kode yang tidak terlalu rumit, apalagi jika dibandingkan dengan kode JavaScript yang diperlukan untuk membuat animasi yang sama.

Walaupun begitu, sama seperti era tag `<blink>` dan `<marquee>` pertama kali diperkenalkan, sebaiknya kita tidak memakai animasi secara berlebihan. Alih-alih membuat halaman web menjadi menarik, pengunjung akan merasa pusing dengan banyaknya efek yang ada.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarluaskan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

17. CSS3 Media Query

Media query merupakan fitur CSS3 paling revolusioner dan mengubah dunia web design secara signifikan. Dengan media query, kita bisa merancang tampilan website yang bisa menyesuaikan diri tergantung ukuran layar.

Membuat design web saat ini lebih kompleks daripada 5 tahun lalu. Ini karena media untuk mengaksesnya juga sangat beragam.

Beberapa waktu lalu untuk mengakses website harus dari komputer atau laptop yang memiliki layar besar. Namun saat ini, pengguna smartphone yang mengakses website sudah melewati jumlah pengguna komputer / laptop.

Sebuah website yang tampil sempurna di monitor 14 inci akan susah diakses dari layar smartphone berukuran 4 inci. Hal ini diperparah dengan beragamnya ukuran layar smartphone dan tablet yang tersedia di pasaran.

Untuk mengatasi masalah ini, beberapa web membuat 2 versi tampilan, satu untuk pengguna dengan layar besar, dan satu lagi untuk penggunaan mobile dengan layar kecil. Sebuah kode JavaScript dipakai untuk mengetahui dari perangkat apa situs tersebut diakses, dan pengguna akan dialihkan ke salah satu web tersebut.

Bagi kebanyakan developer, cara ini kurang efisien dan butuh biaya besar karena kita harus mengupdate kedua versi web secara bersamaan. **CSS3 Media Query** hadir untuk mengatasi hal ini.

17.1. Mengenal Jenis-jenis Layout

Apabila anda mengikuti perkembangan web design, tentu tidak asing dengan istilah *web responsive*. **Web responsive** atau lengkapnya **Responsive Web Design**, adalah sebuah teknik merancang tampilan (*layout*) website yang bisa berubah tergantung media pengaksesnya.

Teknik *responsive web design* pertama kali dicetuskan oleh web desainer **Ethan Marcotte** di tahun 2010. Ethan Marcotte menulis sebuah artikel yang berjudul "Responsive Web Design" di situs web [alistapart.com](https://alistapart.com/article/responsive-web-design/)²⁶. Dalam artikel tersebut, Marcotte memakai fitur CSS3 media query untuk membuat desain web yang berubah-ubah tergantung perangkat pengaksesnya. Sejak itu, responsive web design menjadi standar baru di dunia web.

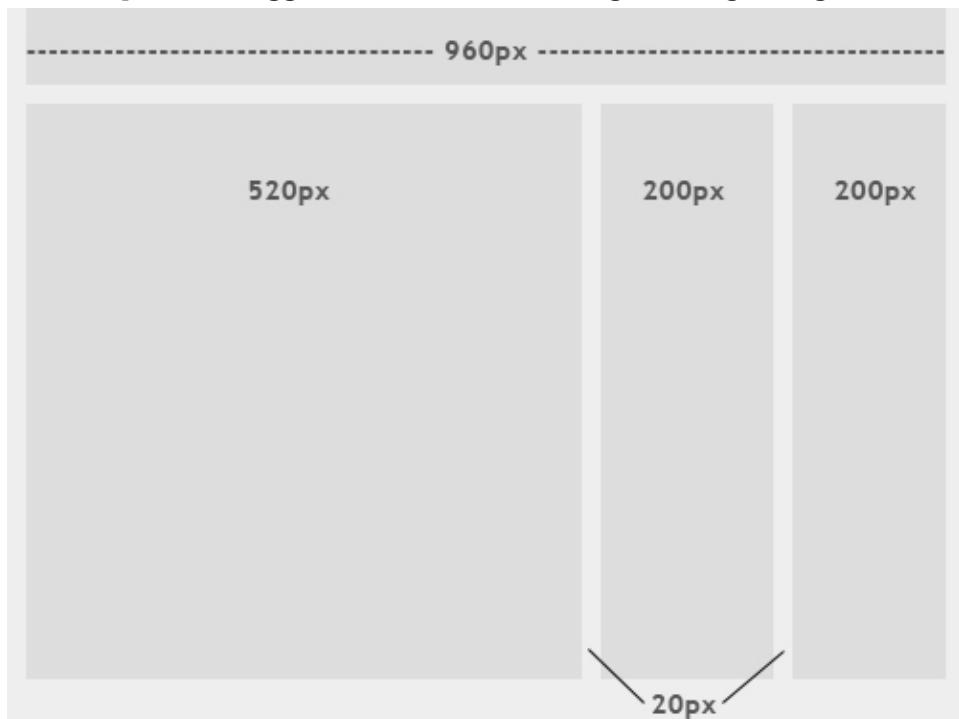
²⁶ <https://alistapart.com/article/responsive-web-design/>

Sebelum era *web responsive*, layout website biasanya hanya terdiri dari 2 jenis: **fixed layout** dan **fluid layout**.

Fixed Layout

Fixed layout adalah sebutan untuk tampilan web yang selalu tetap. Bagi web desainer, fixed layout relatif mudah dirancang karena kita memiliki kontrol pixel demi pixel.

Fixed layout umumnya dibuat untuk monitor dekstop yang cukup lebar dengan resolusi 1024 pixel ke atas. Namun jika diakses dari perangkat dengan layar berukuran kecil seperti smartphone, kita terpaksa menggunakan fitur zoom dengan mengeser-geser scrollbar.

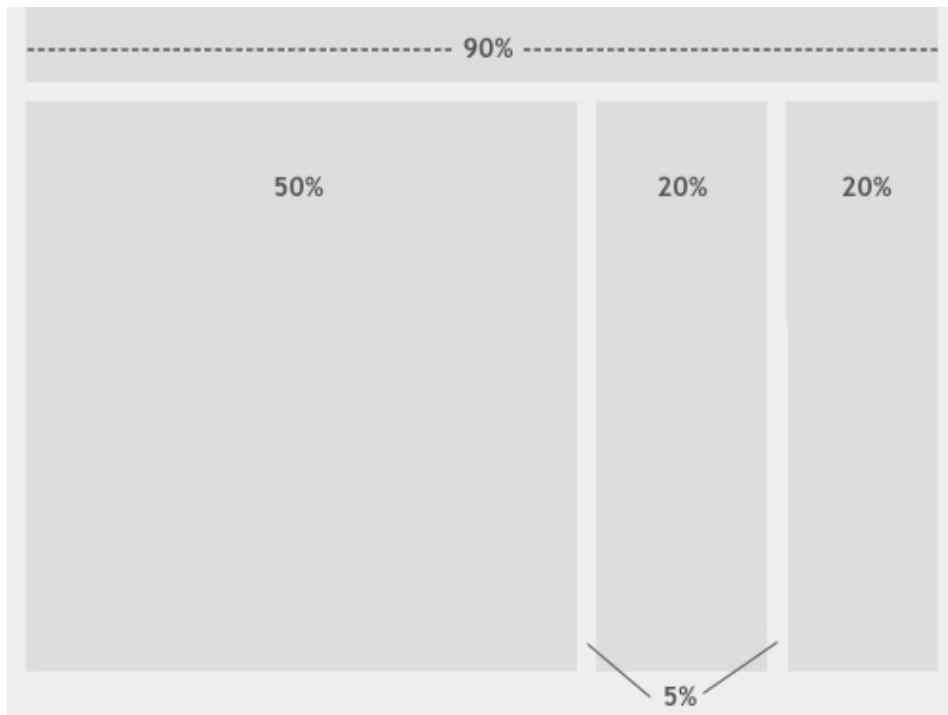


Gambar: contoh rancangan fixed layout, sumber: smashingmagazine.com

Fluid Layout

Fluid layout adalah sebutan untuk tampilan web yang dibuat dengan satuan relatif seperti %. Dengan demikian, lebar dari website akan berubah-ubah tergantung ukuran layar.

Bagi web desainer, fluid layout lebih susah dirancang dari pada fixed layout karena kita tidak memiliki kontrol maksimal untuk penempatan setiap element. Walaupun tampilan website bisa menyesuaikan diri, namun komponen seperti sidebar tetap ada di posisinya yang ketika diakses dari layar kecil menjadi kurang pas karena terlalu sempit.

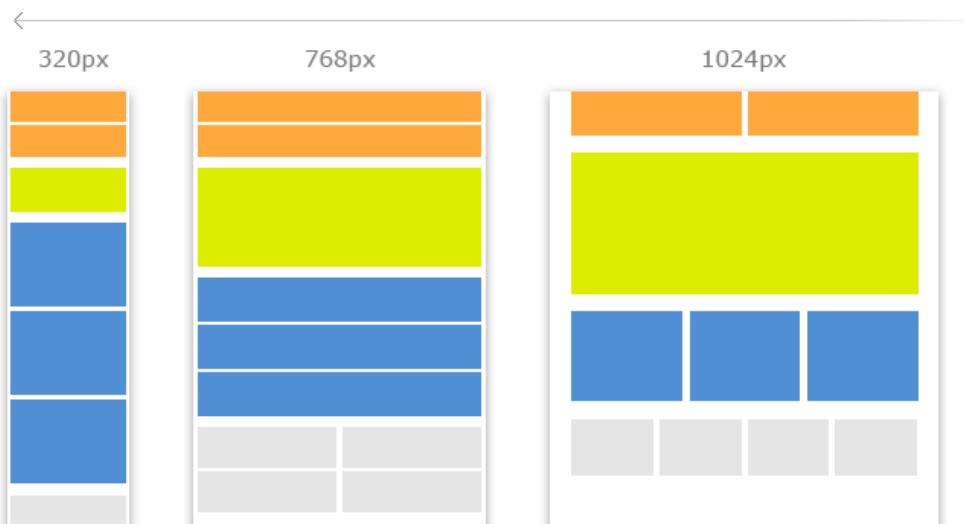


Gambar: contoh rancangan fluid layout, sumber: smashingmagazine.com

Responsive Layout

Responsive layout menggabungkan keunggulan kedua jenis layout sebelumnya. Selain memakai ukuran relatif (sama seperti *fluid layout*), kita bisa menentukan bagaimana tampilan web untuk setiap ukuran layar (seperti *fixed layout*). Pada responsive layout, sidebar di kiri atau kanan bisa ditampilkan, diperkecil, atau disembunyikan jika dibutuhkan.

Kelemahan dari responsive layout adalah lebih susah untuk dirancang. Kita harus buat perencanaan yang matang untuk menentukan element mana yang tampil dan bagaimana urutannya.



Gambar: contoh rancangan responsive layout, sumber: smashingmagazine.com

Cara merancang responsive layout akan di praktekkan pada bab tersendiri. Kali ini kita akan fokus membahas apa itu CSS3 media query, dan bagaimana cara penggunaannya.

17.2. CSS Media Type

Pengertian dan cara penggunaan **CSS media type** sebenarnya telah kita pelajari di awal buku, tepatnya di akhir bab 4. Karena penulisan media query mirip dengan media type, saya ingin kembali membahasnya sekilas.

CSS Media type bisa dipakai untuk menampilkan kode CSS yang berbeda-beda tergantung dari media apa website tersebut diakses, apakah dari monitor, dari printer, dari tv, dll.

Sebagai contoh, untuk membuat kode CSS yang tampil berbeda pada media screen dan print, kita bisa menggunakan penulisan berikut ini:

```
<link href="style.css" rel="stylesheet" media="screen">
<link href="print.css" rel="stylesheet" media="print">
```

Ketika kode ini diakses dari layar monitor, file `style.css` yang akan dipakai. Jika diakses dari print (sewaktu kita akan print halaman tersebut), file `print.css` lah yang akan digunakan.

Selain dengan tag `<link>`, media type juga bisa ditulis menggunakan perintah `@media` seperti berikut:

```
/* Style untuk media screen */
@media screen {
    /* Kode CSS disini */
}

/* Style untuk media print */
@media print {
    /* Kode CSS disini */
}
```

Jika butuh penyegaran tentang maksud kode ini, silahkan buka kembali akhir bab 4.

17.3. Cara Penulisan Media Query

Pada dasarnya, *media query* adalah versi upgrade dari *media type*. Untuk menggunakan media query, penulisannya sangat mirip dengan media type plus tambahan aturan yang lebih fleksibel.

Berikut format dasar penulisan media query:

```
<link href="nama_file.css" rel="stylesheet" media="not|only mediatype and (media
```

`feature)">`

Atau jika menggunakan internal style CSS bisa ditulis menjadi:

```
@media not|only mediatype and (media feature) {  
    /* kode CSS disini */  
}
```

Di sini terdapat tambahan aturan sebelum penulisan `mediatype`, yakni bagian `not|only` dan bagian `media feature`. Kedua aturan inilah yang menjadi dasar dari media query.

Perintah `not|only` dipakai untuk menegaskan pada media mana saja kode CSS akan berjalan. `only` berarti kode CSS hanya akan dijalankan khusus untuk media tersebut, sedangkan `not` berarti kode CSS akan dijalankan pada selain media tersebut.

Sebagai contoh, perhatikan kode berikut:

```
@media only print{  
    /* kode CSS disini */  
}
```

Kode ini hanya akan dijalankan pada media `print`, sedangkan kode:

```
@media not print{  
    /* kode CSS disini */  
}
```

Artinya akan dijalankan selain dari media `print`.

Apabila perintah `not|only mediatype` tidak ditulis, akan dianggap tampil pada semua jenis tipe media.

Bagian media feature merupakan aspek paling menarik dari media query. Media feature berisi informasi tentang perangkat yang akan menjalankan kode CSS, apakah itu dimensinya, resolusinya, dll.

Kita bisa mengatur supaya kode CSS hanya berjalan pada dimensi tertentu saja, resolusi tertentu saja, atau aturan-aturan lain. Inilah yang dipakai untuk membuat responsive web design. Informasi yang bisa digunakan mencakup:

`width, height, device-width, device-height, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, resolution, orientation, scan, dan grid.`

Seluruh media feature ini juga bisa menggunakan awalan (*prefix*) `min-` dan `max-`, seperti `min-width` dan `max-width`. Kita akan bahas dengan lebih detail sesaat lagi.

Untuk mempersingkat pembahasan, saya hanya memakai cara penulisan `@media` saja.
Untuk versi tag `<link>` tidak akan saya tulis lagi.

17.4. Media Feature: Width dan Height

Aturan yang paling sering digunakan untuk media query adalah `width` dan `height`.

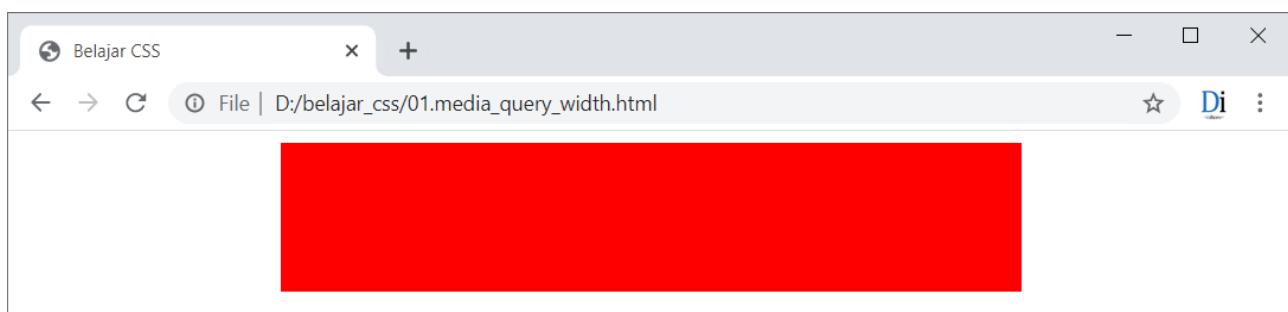
`Width` dan `height` di sini merujuk ke lebar dan tinggi jendela web browser, atau lebih tepatnya disebut dengan `viewport` web browser. **Viewport** merupakan sebutan untuk lebar dan tinggi web browser (termasuk bagian scrollbar).

Jika sebuah web diakses dari perangkat mobile, lebar dan tinggi `viewport` biasanya sama dengan lebar dan tinggi layar smartphone/tablet. Karena kita menjalankan kode program di komputer, efek yang sama bisa didapat dengan cara mengecilkan ukuran jendela web browser.

Dengan media feature: `width`, kita bisa membuat kode CSS yang hanya dijalankan pada lebar tertentu saja. Sebagai contoh, perhatikan kode berikut:

```
01.media_query_width.html

47 <!DOCTYPE html>
48 <html lang="id">
49 <head>
50   <meta charset="UTF-8">
51   <title>Belajar CSS</title>
52   <style>
53     div {
54       width: 500px;
55       height: 100px;
56       margin: 0 auto;
57       background-color: red;
58     }
59     @media (width: 600px) {
60       div {
61         background-color: blue;
62       }
63     }
64   </style>
65 </head>
66 <body>
67   <div></div>
68 </html>
```

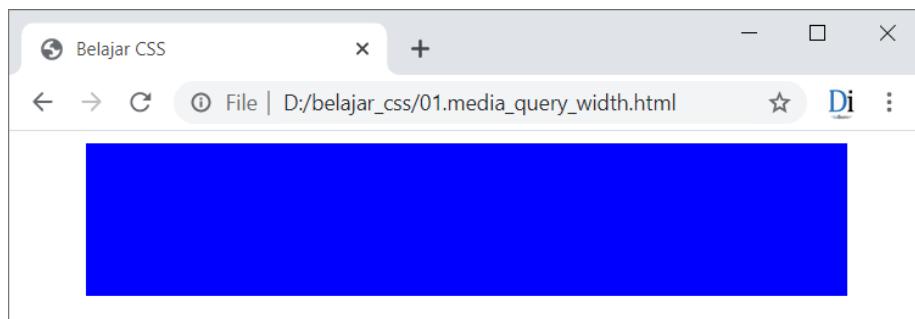


Gambar: penggunaan media query

Ketika dijalankan, box <div> akan berwarna merah. Ini berasal dari property `background-color: red` pada selector `div`. Namun perhatikan kode CSS setelahnya:

```
@media (width: 600px) {  
    div {  
        background-color: blue;  
    }  
}
```

Kode ini berarti, jika lebar jendela viewport sebesar 600px, timpa nilai `background-color` pada selector `div` menjadi `blue` (masih ingat tentang konsep *cascade*?). Agar bisa menampilkan efek ini, kita harus mengatur lebar web browser harus pas sebesar 600px.



Gambar: lebar web browser harus pas 600px

Perintah media feature width ini tampaknya terlalu spesifik, karena sangat sedikit perangkat yang memiliki lebar pas 600 pixel, dan kita juga susah mengatur lebar jendela web browser sebesar 600 pixel. Untuk ini, CSS3 menyediakan prefix `min-` dan `max-`.

Media feature `min-width` bisa digunakan untuk membatasi lebar minimum dari viewport, begitu pula dengan `max-width` yang dipakai untuk membatasi lebar maksimal dari viewport.

Sebagai contoh, saya akan modifikasi kode CSS sebelumnya menjadi:

02.media_query_max-width.html

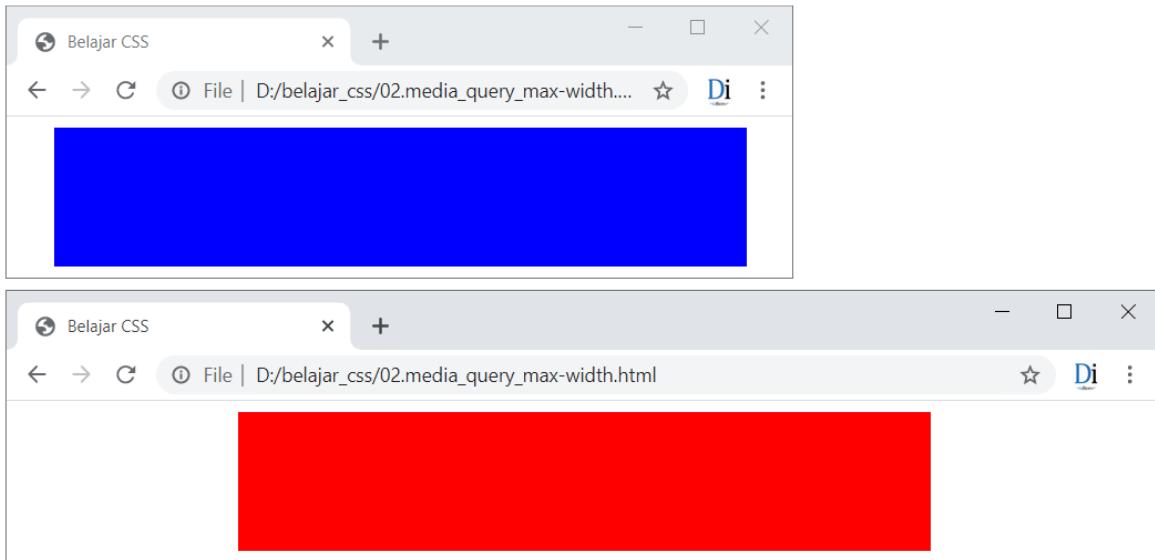
```
1 ...  
2 <style>  
3     div {  
4         width: 500px;  
5         height: 100px;  
6         margin: 0 auto;  
7         background-color: red;  
8     }  
9     @media (max-width: 600px) {  
10         div {  
11             background-color: blue;  
12         }  
13     }  
14 </style>
```

Perintah di baris 9 – 13 bisa dibaca: timpa nilai `background-color` menjadi `blue` jika lebar

CSS3 Media Query

viewport maksimal 600px. Dengan kata lain, jika lebar web browser antara 0 – 600 pixel, box akan berwarna biru, jika lebih dari 600 pixel baru akan berwarna merah.

Silahkan geser ukuran web browser menjadi kecil, kemudian lebarkan lagi. Inilah efek dari media query!



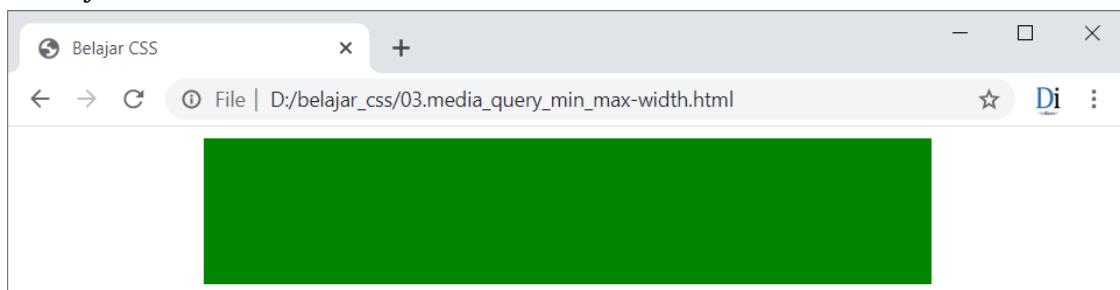
Gambar: media query dengan max-width: 600px

Kita juga tidak terbatas dengan 1 media feature, tapi bisa merangkainya (*di-chaining*) menjadi aturan yang lebih spesifik. Perhatikan kode CSS berikut:

03.media_query_min_max-width.html

```
1 ...  
2     @media (min-width: 600px) and (max-width: 1000px) {  
3         div {  
4             background-color: green;  
5         }  
6     }
```

Perintah ini berarti, ketika lebar web browser antara 600 pixel hingga 1000 pixel, box akan berwarna hijau. Selain itu akan berwarna merah.



Gambar: media query dengan (min-width: 600px) and (max-width: 1000px)

Berikutnya, bagaimana dengan membuat kode CSS dimana ketika lebar web browser kurang dari 500 pixel, box akan berwarna hijau, antara 501 pixel hingga 900 pixel berwarna biru, dan

CSS3 Media Query

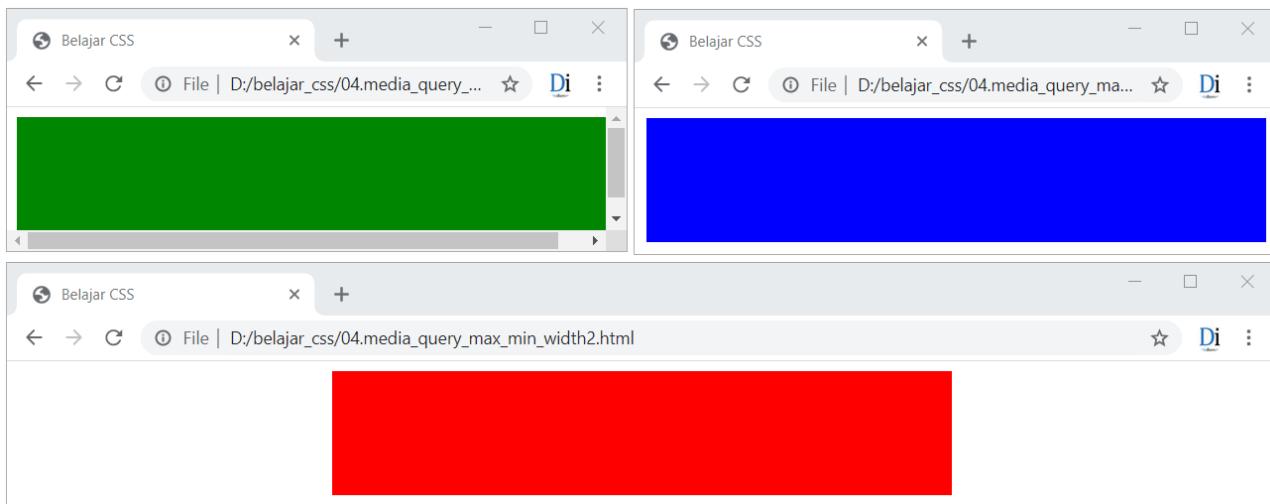
di atas itu berwarna merah?

Untuk mendapatkan efek ini kita butuh 2 kali penulisan media query:

04.media_query_max_min_width2.html

```
1 ...  
2 <style>  
3     div {  
4         width: 500px;  
5         height: 100px;  
6         margin: 0 auto;  
7         background-color: red;  
8     }  
9     @media (max-width: 500px) {  
10        div {  
11            background-color: green;  
12        }  
13    }  
14    @media (min-width: 501px) and (max-width: 900px) {  
15        div {  
16            background-color: blue;  
17        }  
18    }  
19 </style>
```

Silahkan pelajari sejenak kode CSS di atas, dan praktekkan dengan mengubah ukuran web browser untuk melihat perubahan yang terjadi.



Gambar: warna box akan berubah-ubah tergantung lebar dari web browser

Untuk praktek berikutnya, saya melakukan sedikit perubahan:

05.media_query_max_min_width3.html

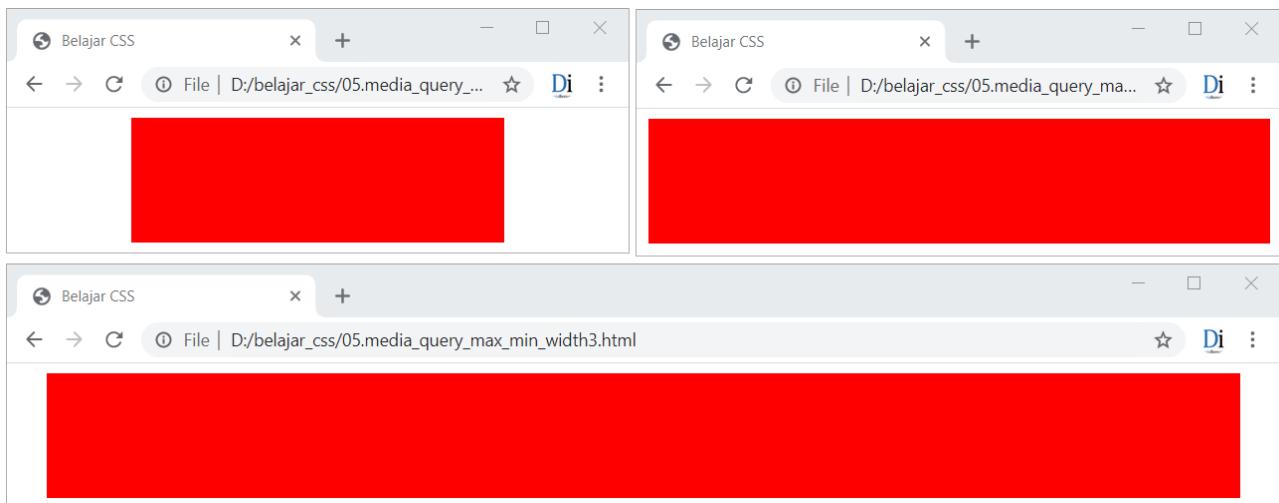
```
1 ...  
2 <style>  
3     div {
```

CSS3 Media Query

```
4     width: 960px;
5     height: 100px;
6     margin: 0 auto;
7     background-color: red;
8   }
9   @media (max-width: 400px) {
10     div {
11       width: 300px;
12     }
13   }
14   @media (min-width: 401px) and (max-width: 900px) {
15     div {
16       width: 500px;
17     }
18   }
19 </style>
```

Kali ini, saya mengganti property `background-color` dengan `width`. Ini artinya pada setiap kode media query, lebar box lah yang akan diubah, bukan lagi warnanya.

Ketika dijalankan, lebar dari box adalah 960 pixel. Namun saat lebar web browser antara 401 pixel hingga 900 pixel, lebar box menjadi 500 pixel. Dan jika lebar web browser kurang dari 400 pixel, lebar box menjadi 300 pixel. Kita tinggal selangkah lagi untuk bisa mendesain sebuah web responsive!



Gambar: Lebar box akan berubah-ubah tergantung lebar dari web browser

Patokan lebar web browser seperti 500 pixel dan 900 pixel yang kita gunakan di sini dikenal juga dengan istilah *breakpoint*. **Breakpoint** adalah titik dimana layout atau tampilan halaman akan berubah mengikuti apa yang ditulis di dalam media query.

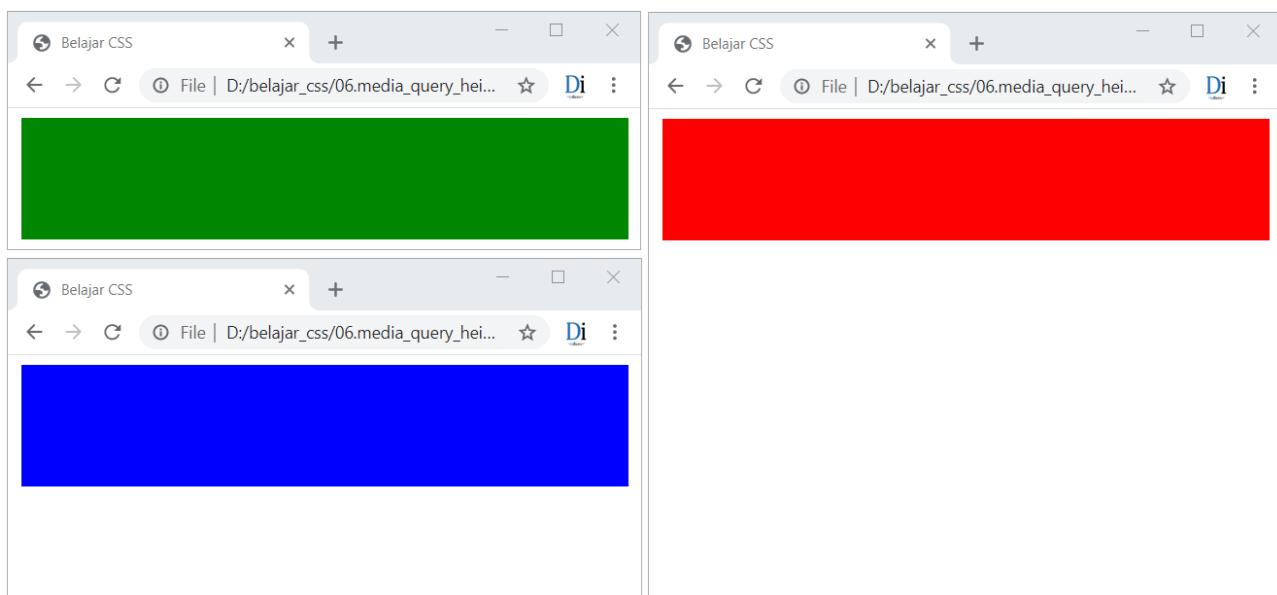
Kita bebas ingin membuat batasan sendiri nilai dari breakpoint ini. Bisa menggunakan 480pixel yang ditujukan untuk layar smartphone, kemudian 640 pixel untuk layar tablet, dan sisanya untuk layar komputer/laptop.

Kembali ke pembahasan tentang media feature, batasan breakpoint juga bisa menggunakan

nilai `height` viewport. Berikut contoh prakteknya:

06.media_query_height.html

```
1 ...  
2 <style>  
3     div {  
4         width: 500px;  
5         height: 100px;  
6         margin: 0 auto;  
7         background-color: red;  
8     }  
9     @media (max-height: 200px) {  
10        div {  
11            background-color: green;  
12        }  
13    }  
14    @media (min-height: 201px) and (max-height: 400px) {  
15        div {  
16            background-color: blue;  
17        }  
18    }  
19 </style>
```



Gambar: Menggunakan tinggi web browser sebagai breakpoint

Kali ini warna box akan berubah ubah tergantung tinggi web browser.

Penggunaan media feature `height` cukup jarang karena tinggi dari sebuah halaman web umumnya tidak dibatasi. Kita bisa dengan mudah menggunakan scroll bar untuk mengakses seluruh isi website dari atas ke bawah. Namun praktek ini memperlihatkan fitur media query untuk membuat efek-efek menarik.

17.5. Media Feature: Resolution

Pada prakteknya, mayoritas media query hanya menggunakan breakpoint `min-width` dan `max-width` seperti yang telah kita coba. Kali ini saya ingin menjelaskan sedikit tentang media feature: `pixel ratio` karena terdapat konsep yang cukup penting untuk dipahami.

Secara umum di dalam CSS, **pixel** merupakan satuan terkecil yang bersesuaian dengan resolusi layar. Sebagai contoh, jika saya memakai monitor dengan resolusi lebar 1366 pixel, maka sebuah box dengan lebar 1366 pixel, akan 'pas' memenuhi seluruh lebar layar.

Begini juga dengan sebuah font yang di set dengan tinggi 18 pixel nyaman dibaca pada monitor laptop 14 inci ber-resolusi 1366 x 768 pixel. Dengan hitungan matematika sederhana, font tersebut akan tampil setinggi 4,2 milimeter. Ini didapat dari $(18 \text{ pixel} / 768 \text{ pixel}) * 18\text{cm} = 0,42 \text{ cm}$. Angka 18cm adalah tinggi dari layar laptop yang saya gunakan.

Bagaimana dengan layar smartphone?

Sebagai contoh, jika sebuah layar smartphone memiliki resolusi 1080 x 1920 pixel dan tinggi layar hanya 11 cm, berapakah besar font 18 pixel? mari kita hitung: $(18 / 1920) * 11 \text{ cm} = 0,103 \text{ cm} = 1 \text{ milimeter!}$ nyaris tidak terlihat.

Tapi mengapa website non responsive tetap tampil bagus di layar smartphone beresolusi tinggi? Jawabannya karena perangkat ini menggunakan teknik *device pixel ratio*.

Device pixel ratio (DPR) adalah perbandingan yang dipakai perangkat beresolusi tinggi agar pixel di dalam CSS bisa tampil dengan ukuran normal.

Sebagai contoh, jika sebuah smartphone menggunakan DPR 2, maka 1 pixel CSS sama dengan 2 pixel di perangkat ini. DPR 1.5 berarti 1 pixel di CSS sama dengan 1,5 pixel di perangkat ini, dst. Sebuah box 5x5 pixel akan tampil sebesar 10x10 pixel pada perangkat dengan DPR 2.

Jika anda mengikuti perkembangan teknologi gadget, iPhone merupakan pelopor yang menggunakan smartphone beresolusi tinggi. Tujuannya supaya tampilan grafis tampak lebih tajam. Oleh Apple, ini di sebut sebagai **retina display**. Di dalam CSS, retina display menggunakan DPR 2.

Permasalahan muncul di kalangan web design karena jika menggunakan gambar, pixel ratio bisa membuat gambar terlihat kabur di perangkat beresolusi tinggi. Oleh karena itu, di dalam CSS tersedia media feature: `resolution`.

Dengan media feature: `resolution`, kita bisa mengatur style apa yang ingin dipakai untuk perangkat beresolusi tinggi.

Terdapat beragam satuan yang bisa digunakan, salah satunya adalah dots per pixel yang disingkat menjadi `dppx`. Untuk membuat breakpoint pada perangkat dengan DPR 2, kita bisa menggunakan nilai `2dppx` seperti contoh berikut:

```

div {
  background-image: url('logo.jpg');
}
@media (min-resolution: 2dppx) {
  div {
    background-image: url('logo_HD.jpg');
  }
}

```

Kode ini bisa dibaca: untuk perangkat biasa, ambil logo.jpg sebagai gambar background. Namun jika perangkat yang mengakses menggunakan minimal DPR 2, gunakan logo.HD.jpg yang beresolusi tinggi.

Praktek media feature: resolution cukup susah karena kita harus menjalankannya di perangkat smartphone. Namun mudah-mudahan anda bisa memahami fungsi dari perintah ini.

Penjelasan tentang *device pixel ratio* dan *media feature: resolution* mungkin terasa kompleks. Tapi karena istilah retina display cukup populer di kalangan web design, saya rasa informasi ini penting untuk disampaikan.

17.6. Media Feature: Device Width dan Device Height

Pada pembahasan tentang media feature: width, yang jadi patokan untuk breakpoint adalah lebar dari jendela web browser (*viewport*).

Untuk media feature: device width, yang menjadi ukuran adalah lebar sebenarnya dari layar. Misalnya, untuk perangkat dengan resolusi asli 1080 x 1920 pixel, kita bisa buat breakpoint sebagai berikut:

```

div {
  background-image: url('logo.jpg');
}
@media (min-device-width: 1080px) {
  div {
    background-image: url('logo_HD.jpg');
  }
}

```

Penggunaan media query seperti ini relatif jarang karena terdapat faktor *device pixel ratio* yang bisa berbeda-beda pada tiap perangkat. Gambar logo_HD.jpg mungkin saja akan terlalu besar pada perangkat tertentu.

Dari penjelasan tentang *media feature resolution*, dapat dipahami bahwa lebar web browser belum tentu sama dengan resolusi sebenarnya dari perangkat yang dipakai (karena ada efek *device pixel ratio*).

Perbedaan antara *viewport* web browser dengan resolusi perangkat ini bisa menimbulkan

masalah tersendiri. Sebagai contoh, untuk mendesain tampilan pada Samsung Galaxy S21, berapakah ukuran resolusi yang digunakan?

Selain itu, walaupun kita telah memakai media query sesuai dengan standarnya, kadang website tetap tampil berbeda. Ini terjadi karena terdapat fitur zoom di web browser smartphone yang seolah-oleh membuat web kita tampil responsive.

Untuk mengatasi hal ini, perlu perintah khusus untuk web browser smartphone agar masuk ke "mobile mode". Caranya dengan menambah sebuah meta tag `viewport` di bagian `<head>`, sebagai berikut:

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

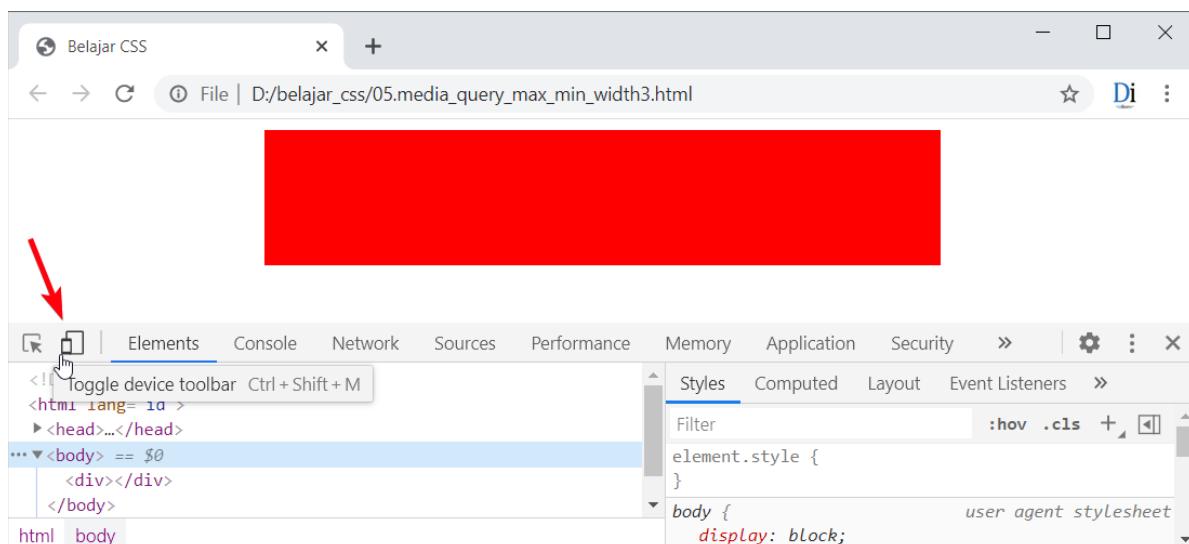
Meta tag ini berfungsi agar ukuran viewport web browser diatur secara otomatis mengikuti ukuran ideal dari perangkat tersebut (`device-width`).

Penjelasan tentang meta tag `viewport` di atas memang cukup rumit. Jika anda tertarik mempelajarinya lebih lanjut, bisa mengunjungi artikel di situs quirksmode.org berikut: [A pixel is not a pixel is not a pixel](#)²⁷.

Tapi satu hal yang pasti, hampir setiap web dengan design responsive menyertakan meta tag ini dibagian `<head>`.

Menguji Fitur Responsive di Developer Tools

Developer tools bawaan web browser memiliki fitur "uji responsive". Kita bisa men-simulasikan tampilan website di berbagai perangkat mobile. Caranya, buka developer tools dengan menekan kombinasi tombol Crtl + Shift + I, lalu klik icon smartphone di toolbar:

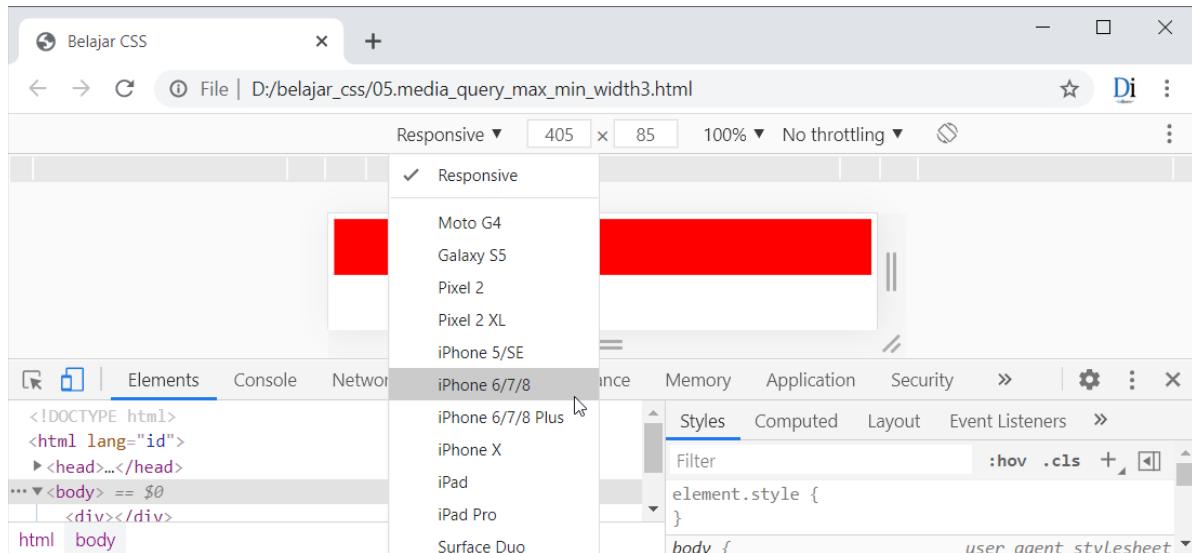


Gambar: Mengaktifkan responsive mode di Developer Tools

Sesaat kemudian akan muncul jendela kecil di dalam web browser sebagai simulasi ukuran

²⁷ https://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html

layar smartphone. Kita bisa menggeser manual dimensi layar smartphone tersebut atau memilih salah satu jenis smartphone yang tersedia:



Gambar: Jenis-jenis smartphone untuk simulasi web responsive

Yang harus diperhatikan saat menggunakan fitur ini adalah, meta tag `viewport` harus ada di bagian `<head>` kode HTML. Jika tidak, halaman tampil dalam versi zoom.

17.7. Media Feature: Orientation

Di perangkat smartphone dan tablet, terdapat 2 model tampilan: *portrait* dan *landscape*.

Portrait adalah ketika perangkat tersebut memiliki tinggi yang lebih besar daripada lebarnya. Sedangkan **landscape** adalah ketika lebar perangkat lebih lebar daripada tingginya. Jika anda memiliki smartphone, tentunya tidak asing dengan kedua istilah ini.

CSS3 juga mengakomodasi penggunaan media query berdasarkan apakah halaman tampil pada posisi *portrait* atau *landscape*. Berikut contoh penggunaannya:

07.media_query_orientation.html

```

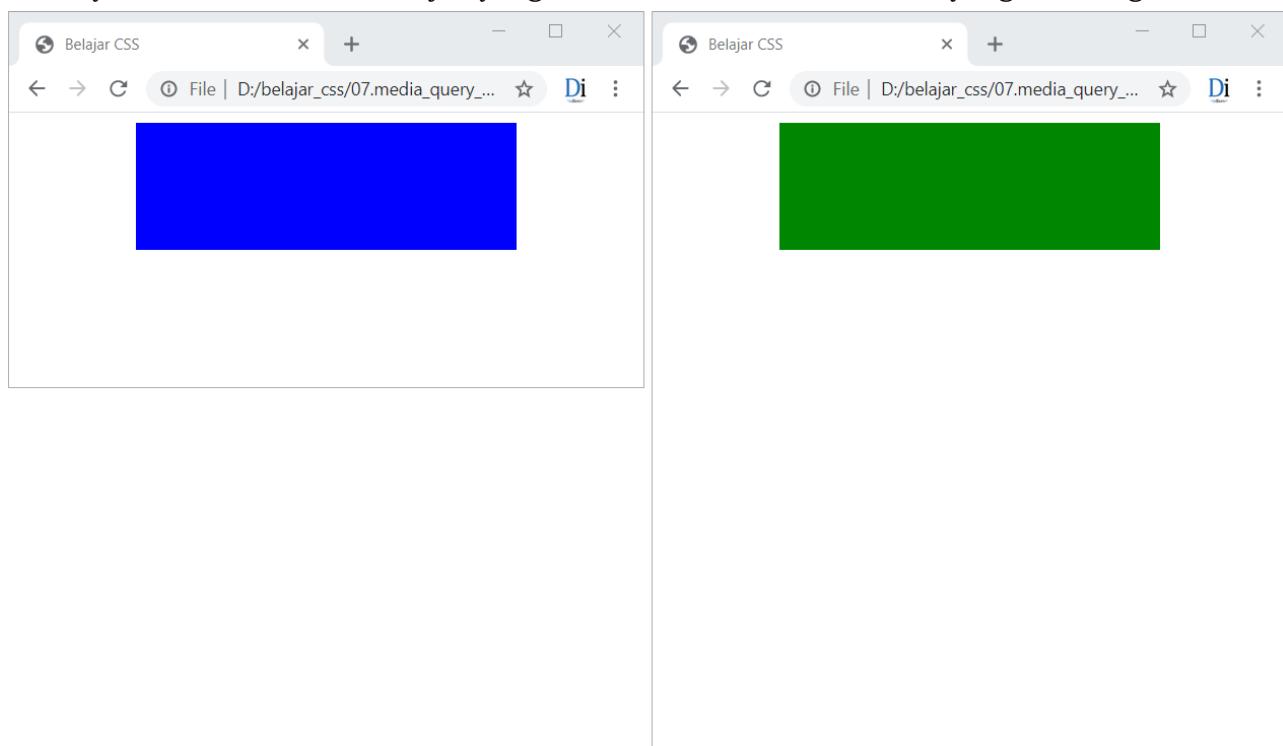
1 ...
2 <style>
3   div {
4     width: 300px;
5     height: 100px;
6     margin: 0 auto;
7   }
8   @media (orientation: portrait) {
9     div {
10       background-color: green;
11     }
12   }
13   @media (orientation: landscape) {
14     div {

```

```
15         background-color: blue;
16     }
17 }
18 </style>
```

Uniknya, kita tidak harus mengakses media feature: orientation dari smartphone, tapi cukup dengan mengubah ukuran jendela web browser.

Jika lebar web browser lebih panjang dari pada tingginya, yang aktif adalah mode `landscape` sehingga box akan berwarna biru. Namun jika tinggi web browser lebih besar daripada lebarnya, box akan berwarna hijau yang menandakan mode `portrait` yang sekarang aktif.

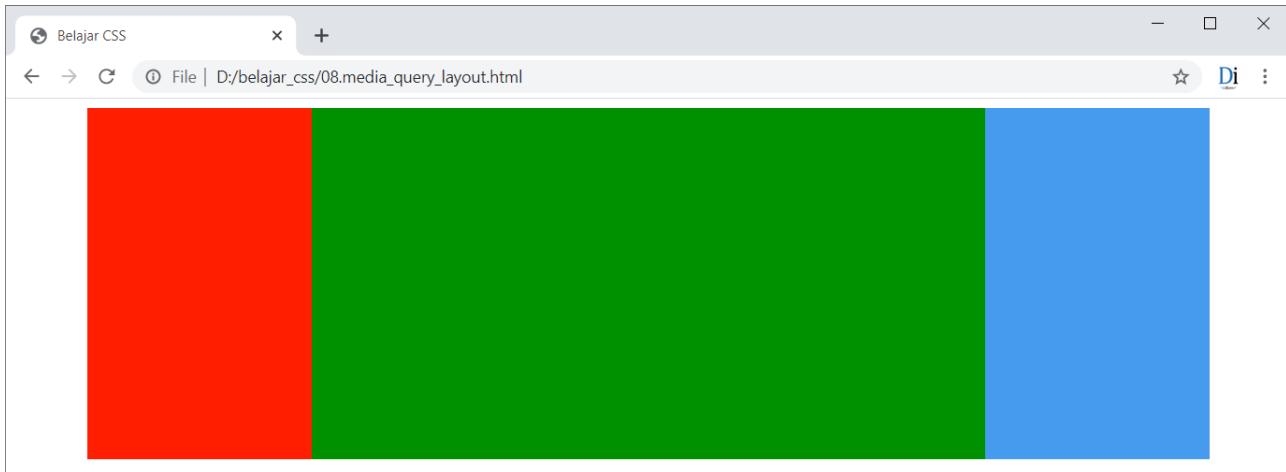


Gambar: Tampilan orientation: landscape (kiri) dan orientation: portrait (kanan)

17.8. Membuat Layout Responsive Sederhana

Berbekal pemahaman tentang media query dan berbagai property CSS, kita sudah bisa merancang layout responsive sederhana. Untuk contoh ini saya akan menggunakan box merah, hijau dan biru seperti contoh ketika membahas property float:

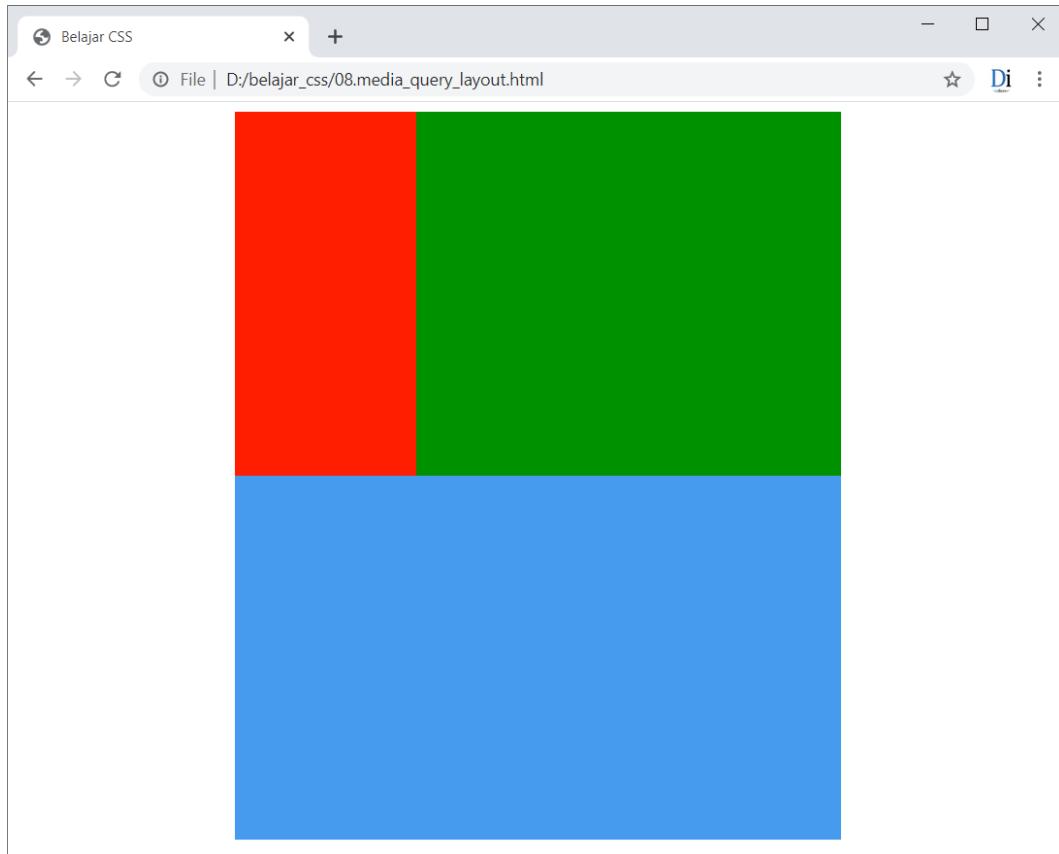
CSS3 Media Query



Gambar: Tampilan layout sederhana: 2 sidebar + 1 konten utama

Untuk tampilan desktop, ketiga box tampil seperti gambar di atas, yakni sidebar merah dan biru mengapit box hijau.

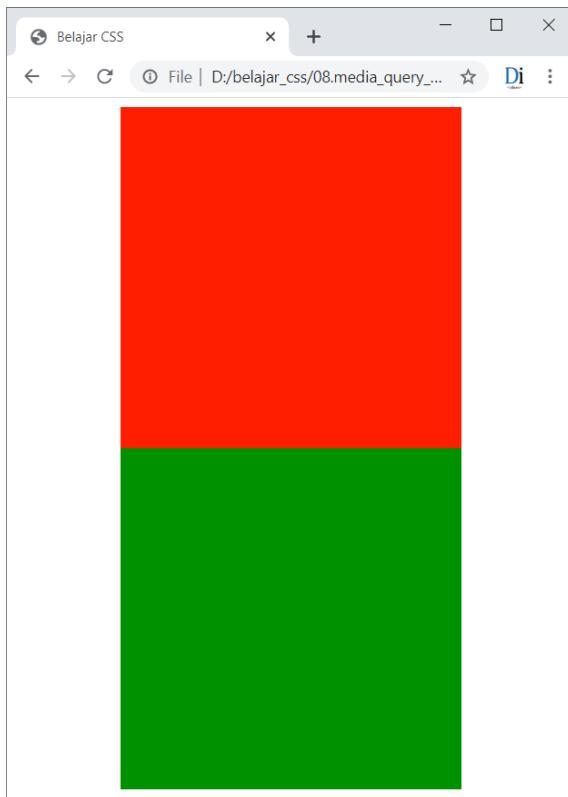
Untuk tampilan tablet, saya ingin agar sidebar biru pindah ke posisi bawah. Sehingga sidebar merah tampil di bagian atas bersebelahan dengan box konten hijau:



Gambar: Tampilan layout untuk perangkat tablet

Untuk tampilan smartphone, saya ingin agar seluruh box saling bertumpuk dari atas ke bawah dengan box merah di posisi pertama, box hijau di bawahnya, dan box biru tidak ditampilkan

(hidden):



Gambar: Tampilan layout untuk perangkat smartphone

Mengenai breakpoint untuk masing-masing tampilan, saya akan memakai posisi 900 pixel dan 500 pixel.

Silahkan anda coba rancang tampilan di atas. Untuk membuatnya kita perlu menggunakan media query, serta manipulasi posisi box dengan property float. Baik, berikut kode lengkap yang saya gunakan:

08.media_query_layout.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          /*==Desktop Style==*/
8          /*=====
9          .container {
10              width: 960px;
11              background-color: rgb(231, 232, 226);
12              margin: 0 auto;
13          }
14          .satu {
15              width: 20%;
16              height: 300px;
```

CSS3 Media Query

```
17     background-color: #FF1A00;
18     float: left;
19 }
20 .dua{
21     width: 60%;
22     height: 300px;
23     background-color: #008C00;
24     float: left;
25 }
26 .tiga{
27     width: 20%;
28     height: 300px;
29     background-color: #4096EE;
30     float: left;
31 }
32 /*==Tablet Style==*/
33 /*=====
34 @media only screen and (min-width: 501px) and (max-width: 900px){
35     .container {
36         width: 500px;
37     }
38     .satu {
39         width: 30%;
40         float: left;
41     }
42     .dua{
43         width: 70%;
44         float: left;
45     }
46     .tiga{
47         width: 100%;
48         float: none;
49         clear: both;
50     }
51 }
52 /*==Smartphone Style==*/
53 /*=====
54 @media only screen and (max-width: 500px){
55     .container {
56         width: 300px;
57     }
58     .satu {
59         width: 100%;
60         float: none;
61     }
62     .dua{
63         width: 100%;
64         float: none;
65     }
66     .tiga{
67         display: none;
68     }
69 }
70 </style>
71 </head>
```

```
72 <body>
73   <div class="container">
74     <div class="satu"></div>
75     <div class="dua"></div>
76     <div class="tiga"></div>
77   </div>
78 </body>
79 </html>
```

Cukup panjang? Betul, dan ini barulah contoh layout sederhana, kita belum contoh website sebenarnya. Seperti yang saya singgung ketika membahas tentang jenis-jenis layout, membuat web responsive memang paling kompleks, karena kita harus merencanakan bukan hanya 1 tampilan, tapi bisa sampai 3 jenis tampilan layout sekaligus.

Konsep perancangan layout web responsive di atas menggunakan metode "desktop first", dimana style global saya buat untuk tampilan desktop terlebih dahulu, baru kemudian menyesuaikan tampilan untuk versi mobile.

Selain itu, terdapat pula konsep "mobile first", yakni style global dirancang untuk versi mobile terlebih dahulu, baru disesuaikan untuk versi tablet dan desktop. Konsep *mobile first* ini lebih populer digunakan pada CSS framework seperti **Bootstrap**.

Untuk pembuatan web responsive yang agak kompleks, saya lebih sarankan memakai framework CSS seperti Bootstrap atau TailwindCSS, karena akan memudahkan kita dalam menentukan breakpoint. Namun pemahaman dasar terkait cara kerja media query tetap menjadi nilai tambah tersendiri.

CSS3 Media Query dan **Responsive Web Design** merupakan topik yang cukup kompleks, bahkan bisa menjadi buku sendiri. Dalam bab kita telah melihat sekilas cara penggunaannya.

Berikutnya saya akan masuk ke pembahasan tentang **CSS3 Flexbox**.

18. CSS3 FlexBox

Sejak awal dikembangkan CSS, **design layout** menjadi fokus utama dari setiap pembuatan web, yakni bagaimana mengatur tata letak element HTML agar tampil menarik dan fungsional.

Yang agak aneh, CSS tidak memiliki property atau kode yang secara khusus mengatur cara membuat layout. Bertahun-tahun sejak era CSS 2, web desainer menggunakan property **float** untuk membuat design layout.

Float pada dasarnya adalah property CSS untuk mengatur posisi gambar di dalam teks, yang kemudian dipakai secara "kreatif" untuk membuat design layout. Ini mirip seperti penggunaan tag `<table>` HTML untuk membuat layout sebelum era CSS.

Jika sebuah element ingin ditempatkan pada sisi kiri, kita perlu menambah perintah `float:left`. Jika ingin diletakkan di sisi kanan, bisa memakai `float:right`. Jika ingin element tersebut pas berada di tengah-tengah halaman, well... terpaksa harus menggunakan trik tertentu karena tidak ada property CSS yang secara khusus bisa dipakai untuk itu (memang ada property `text-align`, tapi hanya bisa dipakai untuk *inline level element*).

Sebagai alternatif yang lebih modern, tim dibalik **CSS3 (W3C)** memperkenalkan 2 buah layout modul baru, yakni [CSS Flexible Box Layout Module](#) atau sering disingkat dengan **Flexbox** dan [CSS Grid Layout Module](#).

Flexbox berisi berbagai property CSS untuk mengatur tampilan yang tersusun secara 1 dimensi (baris saja, atau kolom saja), misalnya mengatur posisi element di dalam baris, mengatur posisi link pada menu bar, atau mengatur urutan kolom. Sedangkan **CSS Grid** bisa dipakai untuk mengatur tampilan dalam 2 dimensi sekaligus (kolom dan baris).

Dalam bab ini kita akan bahas cara penggunaan CSS3 Flexbox terlebih dahulu.

18.1. Flex Container

Agar bisa menggunakan flexbox, satu atau beberapa element perlu berada di dalam sebuah "flex container". Ini dibuat dengan cara menambah property `display:flex` atau `display:inline-flex` ke dalam parent elementnya. Berikut contoh yang dimaksud:

01.flex-container.html

```
1  <!DOCTYPE html>
2  <html lang="id">
```

```

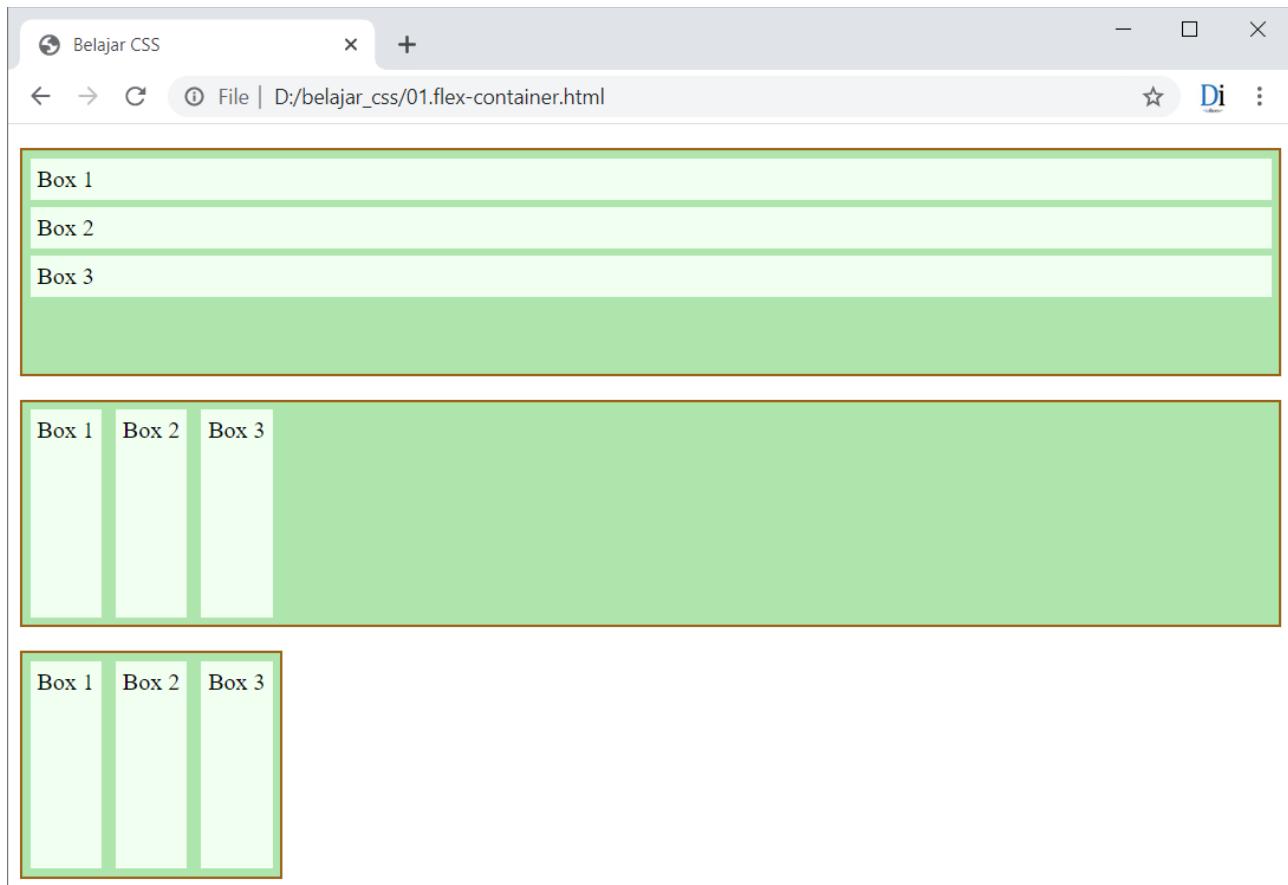
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             margin-top: 1em ;
12         }
13         .box {
14             background-color: honeydew;
15             margin: 0.3em;
16             padding: 0.3em;
17         }
18     </style>
19 </head>
20 <body>
21
22     <div class="container">
23         <div class="box"> Box 1 </div>
24         <div class="box"> Box 2 </div>
25         <div class="box"> Box 3 </div>
26     </div>
27
28     <div class="container" style="display: flex;">
29         <div class="box"> Box 1 </div>
30         <div class="box"> Box 2 </div>
31         <div class="box"> Box 3 </div>
32     </div>
33
34     <div class="container" style="display: inline-flex;">
35         <div class="box"> Box 1 </div>
36         <div class="box"> Box 2 </div>
37         <div class="box"> Box 3 </div>
38     </div>
39
40 </body>
41 </html>

```

Di dalam tag `<style>`, selector `.container` (baris 7 – 12) berisi beberapa property, yakni `height: 150px` untuk mengatur tinggi element sebesar 150px, serta men-set warna background, border dan margin.

Selector kedua, `.box` berisi property untuk men-set warna background menjadi `honeydew` (putih agak kehijauan), serta sedikit padding dan margin sebesar `0.3em`.

Masuk ke kode HTML, saya membuat 3 container dari tag `<div class="container">`. Setiap container ini berisi 3 box `<div class="box">`. Berikut tampilan kode diatas:



Gambar: Penggunaan flexbox container

Container pertama tidak mendapat property tambahan. Panjang Box 1, Box 2 dan Box 3 akan melebar ke kanan untuk memenuhi parent element, sedangkan tinggi ketiga box sesuai dengan tinggi konten yang ada di dalamnya. Ini merupakan perilaku default dari sebuah block level element jika width dan height-nya tidak di atur.

Pada container kedua, saya menambah property `display: flex` menggunakan inline style CSS di baris 28. Akibat penambahan ini, tampilan ketiga box berubah drastis menjadi sejajar dalam satu baris serta tingginya memenuhi parent element. Ini merupakan tampilan default dari flexbox. Secara bertahap nantinya kita akan bahas berbagai property flexbox untuk mengatur tampilan ini.

Container ketiga mendapat tambahan property `display: inline-flex`. Pada dasarnya property ini juga berguna untuk membuat *flex container* (sama seperti `display:flex`), hanya saja container akan tampil dalam bentuk *inline level element*.

Jenis-jenis Flexbox Property

Setelah selesai membuat *flex container*, langkah berikutnya adalah menambah beberapa property flexbox lain untuk mengatur tampilan element. Property flexbox ini bisa dipecah menjadi 2 kelompok: property untuk **flex container**, dan property untuk **flex item**.

Pengertian *flex container* sudah kita bahas sebelumnya, yakni element yang menampung semua flex item. Property yang ditulis ke flex container akan berpengaruh secara global.

Sedangkan *flex item* adalah sebutan dari setiap *child element* yang berada di dalam *flex container*. Dalam contoh sebelumnya, Box 1, Box 2 dan Box 3 adalah flex item.

Berikut daftar property untuk **flex container**:

- `flex-direction`
- `flex-wrap`
- `flex-flow (shorthand)`
- `justify-content`
- `align-items`
- `align-content`

Dan berikut property untuk **flex item**:

- `order`
- `flex-grow`
- `flex-shrink`
- `flex-basis`
- `flex (shorthand)`

Semua property ini akan kita bahas secara bertahap.

18.2. Property **flex-direction**

Property **flex-direction** dipakai untuk mengatur urutan dan posisi element yang terdapat di dalam *flex container*. Nilai yang bisa diisi adalah: `row` (nilai default), `row-reverse`, `column` dan `column-reverse`.

Berikut contoh penggunaan nilai `row` dan `row-reverse`:

02.flex-direction-row.html

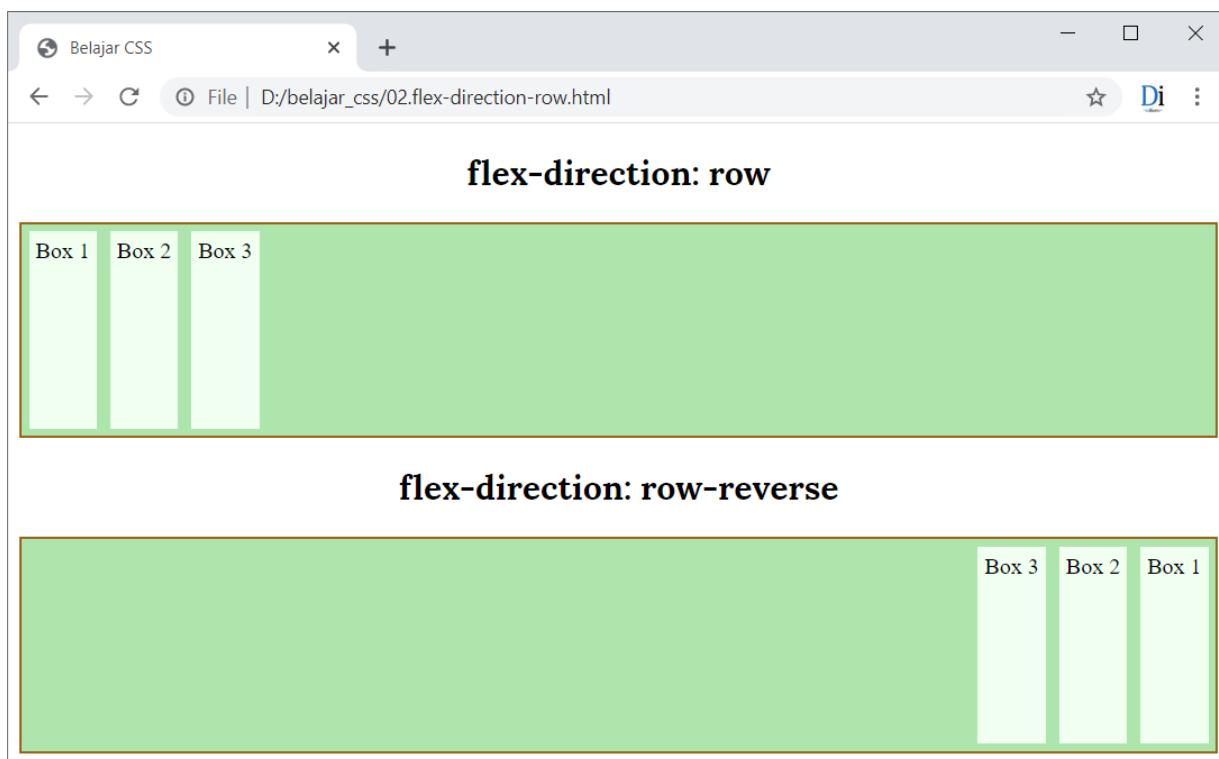
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             margin-top: 1em;
12             display: flex;
13         }
14         .box {

```

CSS3 FlexBox

```
15     background-color: honeydew;
16     margin: 0.3em;
17     padding: 0.3em;
18 }
19 h2{
20     text-align: center;
21     font-family: lora,serif;
22 }
23 </style>
24 </head>
25 <body>
26
27 <h2>flex-direction: row</h2>
28 <div class="container" style="flex-direction: row;">
29     <div class="box"> Box 1 </div>
30     <div class="box"> Box 2 </div>
31     <div class="box"> Box 3 </div>
32 </div>
33
34 <h2>flex-direction: row-reverse</h2>
35 <div class="container" style="flex-direction: row-reverse;">
36     <div class="box"> Box 1 </div>
37     <div class="box"> Box 2 </div>
38     <div class="box"> Box 3 </div>
39 </div>
40
41 </body>
42 </html>
```



Gambar: Hasil penggunaan property flex-direction

Property `display: flex` saya pindahkan ke selector `.container` di baris 12. Dengan demikian, semua element HTML yang menggunakan selector ini otomatis akan menjadi *flex container*.

Container pertama memakai `flex-direction: row`, hasilnya semua flex item akan berkumpul di sisi kiri dengan urutan sesuai penulisan, yakni dari Box 1, Box 2 dan Box 3. Ini merupakan nilai default jika property `flex-direction` tidak ditulis.

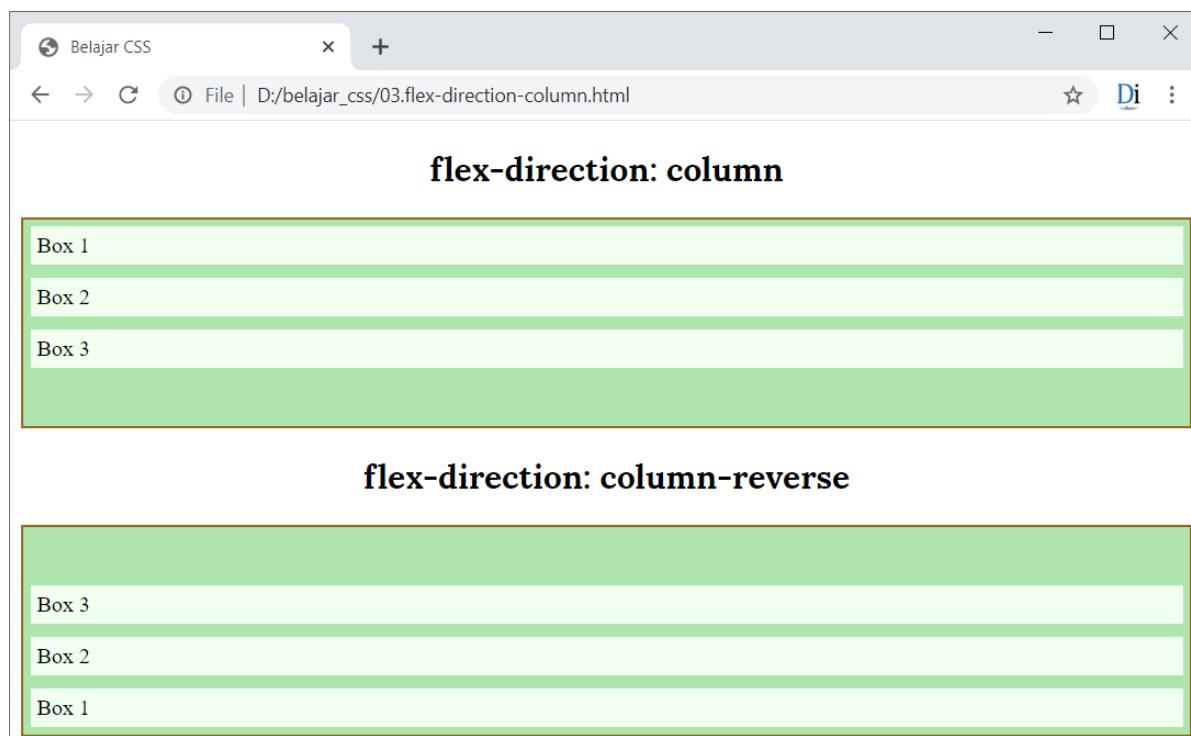
Container kedua menggunakan `flex-direction: row-reverse`, maka kali ini semua flex item akan berada di sisi kanan. Selain itu urutannya juga terbalik, dimana Box 1 berada di sisi paling kanan, lalu diikuti Box 2 dan Box 3.

Sekarang kita akan lihat nilai hasil dari `column` dan `column-reverse`:

03.flex-direction-column.html

```

1 ...
2   <h2>flex-direction: column</h2>
3   <div class="container" style="flex-direction: column;">
4     <div class="box"> Box 1 </div>
5     <div class="box"> Box 2 </div>
6     <div class="box"> Box 3 </div>
7   </div>
8
9   <h2>flex-direction: column-reverse</h2>
10  <div class="container" style="flex-direction: column-reverse;">
11    <div class="box"> Box 1 </div>
12    <div class="box"> Box 2 </div>
13    <div class="box"> Box 3 </div>
14  </div>
```



Gambar: Hasil penggunaan property `flex-direction`

Ketika flex container di tambahkan property `flex-direction: column`, maka flex item akan berjejer dari atas ke bawah membentuk kolom. Sedangkan untuk `flex-direction: column-reverse`, kolom akan dimulai dari bagian bawah terlebih dahulu.

Property `flex-direction` yang kita bahas ini dipakai untuk menentukan **main axis** (sumbu utama) dari flexbox.

Nilai `row` dan `row-reverse` akan membuat sumbu utama pada arah horizontal (sumbu x), sedangkan nilai `column` dan `column-reserve` akan membuat sumbu utama flexbox di arah vertikal (sumbu y).

Arah main axis ini nantinya akan berpengaruh ke beberapa property lain.

18.3. Property `flex-wrap`

Property `flex-wrap` berguna untuk menentukan apakah flex item diizinkan membuat baris baru jika flex container tidak cukup lebar. Nilai yang bisa diisi adalah `nowrap` (default), atau `wrap`.

Property `flex-wrap` hanya berefek jika flex container di set dengan `flex-direction:row` atau `row-reverse` saja, dan akan diabaikan untuk `flex-direction:column` serta `column-reverse`.

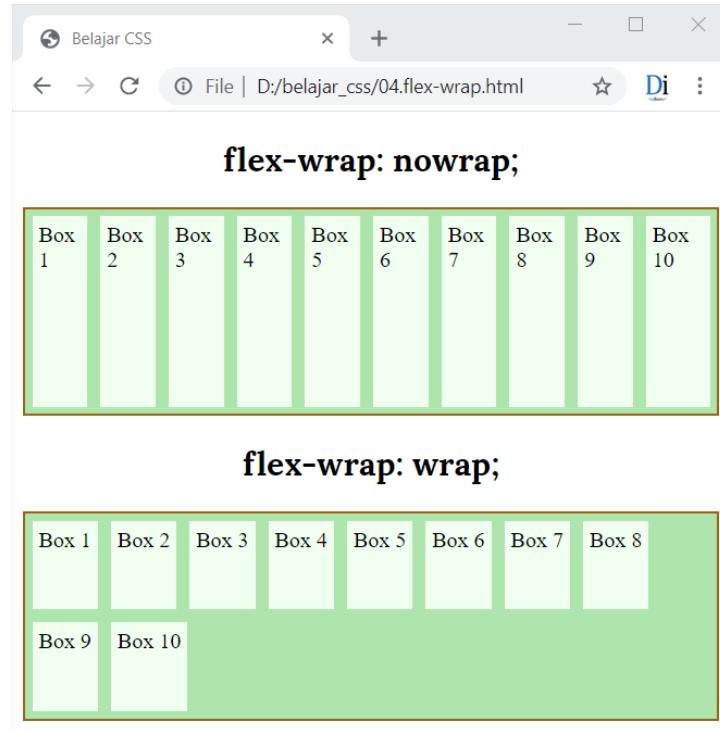
Berikut percobaannya:

04.flex-wrap.html

```

1 ...
2   <h2>flex-wrap: nowrap;</h2>
3   <div class="container" style="flex-direction: row; flex-wrap: nowrap;">
4     <div class="box"> Box 1 </div>
5     <div class="box"> Box 2 </div>
6     ...
7     <div class="box"> Box 9 </div>
8     <div class="box"> Box 10 </div>
9   </div>
10
11 <h2>flex-wrap: wrap;</h2>
12 <div class="container" style="flex-direction: row; flex-wrap: wrap;">
13   <div class="box"> Box 1 </div>
14   <div class="box"> Box 2 </div>
15   ...
16   <div class="box"> Box 9 </div>
17   <div class="box"> Box 10 </div>
18 </div>
19 ...

```



Gambar: Hasil penggunaan property flex-wrap

Jumlah flex item yang ada di dalam flex container saya tambah sampai 10 Box. Ketika lebar jendela web browser diperkecil, pada titik tertentu flex container tidak muat lagi menampung semua item ini.

Jika flex container di set dengan `flex-wrap: nowrap`, maka flex item akan terus diperkecil, sedangkan untuk `flex-wrap: wrap`, flex item akan pindah ke bawah membentuk baris baru.

Juga apabila lebar flex item di set langsung dengan property `width`, pengaturan `flex-wrap: nowrap` akan menganggap `width` tersebut sebagai `max-width` dan masih bisa "dipaksa" untuk mengecil. Berikut contoh praktek dari efek ini:

05.flex-wrap-width.html

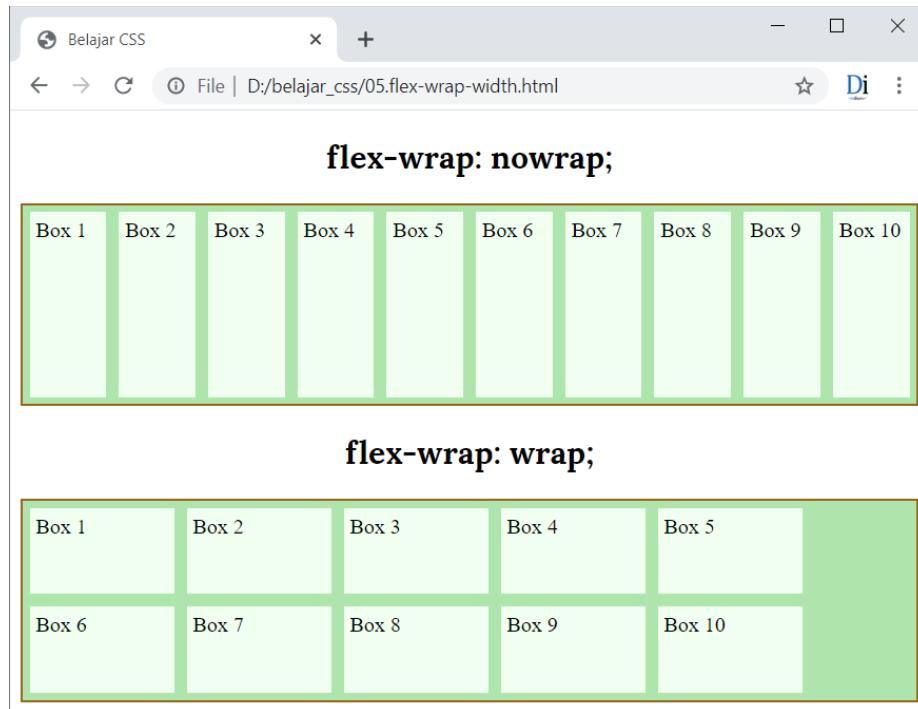
```

1 ...
2 <h2>flex-wrap: nowrap;</h2>
3 <div class="container" style="flex-direction: row; flex-wrap: nowrap;">
4   <div class="box" style="width:100px"> Box 1 </div>
5   <div class="box" style="width:100px"> Box 2 </div>
6   ...
7   <div class="box" style="width:100px"> Box 9 </div>
8   <div class="box" style="width:100px"> Box 10 </div>
9 </div>
10
11 <h2>flex-wrap: wrap;</h2>
12 <div class="container" style="flex-direction: row; flex-wrap: wrap;">
13   <div class="box" style="width:100px"> Box 1 </div>
14   <div class="box" style="width:100px"> Box 2 </div>
15   ...

```

CSS3 FlexBox

```
16      <div class="box" style="width:100px"> Box 9 </div>
17      <div class="box" style="width:100px"> Box 10 </div>
18  </div>
```



Gambar: Efek property flex-wrap pada width flex item

Sekarang semua flex item saya tambah dengan property `width:100px`.

Untuk flex container yang menggunakan `flex-wrap: nowrap`, lebar 100px ini masih bisa dipaksa mengecil. Sedangkan untuk container dengan `flex-wrap: wrap`, flex item akan mempertahankan `width:100px` dan akan membuat baris baru jika sudah tidak muat lagi.

Akan tetapi apabila flex container cukup lebar untuk menampung semua flex item, lebar setiap item tetap dibatasi sampai 100 pixel saja:



Gambar: Efek property flex-wrap pada width flex item

Dari percobaan ini bisa kita simpulkan bahwa jika flex container di set dengan `flex-direction: nowrap`, property `width` pada flex item akan dianggap sebagai `max-width`, yakni boleh berkurang dari nilai tersebut, tapi tidak bisa melebihi nilai `width`.

18.4. Property flex-flow

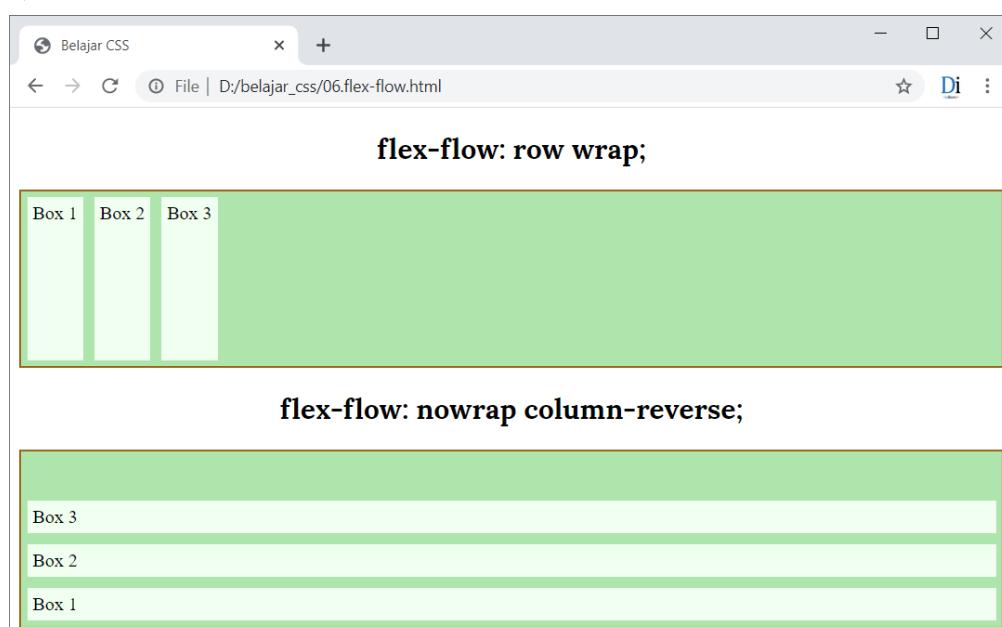
Property **flex-flow** merupakan penulisan singkat (*shorthand*) dari gabungan `flex-direction` dan `flex-flow`. Sebagai contoh, penulisan dua property berikut:

```
flex-direction: row
flex-flow: wrap
```

Bisa disingkat menjadi: `flex-flow: row wrap`, atau `flex-flow: wrap row`. Urutan nilai yang ditulis boleh bolak-balik.

06.flex-flow.html

```
1   ...
2   <h2>flex-flow: row wrap;</h2>
3   <div class="container" style="flex-flow: row wrap;">
4     <div class="box"> Box 1 </div>
5     <div class="box"> Box 2 </div>
6     <div class="box"> Box 3 </div>
7   </div>
8
9   <h2>flex-flow: nowrap column-reverse;</h2>
10  <div class="container" style="flex-flow: nowrap column-reverse;">
11    <div class="box"> Box 1 </div>
12    <div class="box"> Box 2 </div>
13    <div class="box"> Box 3 </div>
14  </div>
```



Gambar: Hasil penggunaan property flex-flow

18.5. Property justify-content

Ketika mempelajari `flex-direction`, saya sempat membahas bahwa nilai dari property itu akan menentukan **main axis** atau sumbu utama dari flex container, yakni apakah searah sumbu x (row) atau searah sumbu y (column).

Property **justify-content** bisa dipakai untuk mengatur posisi flex item pada *main axis* ini, dan hasilnya bisa berbeda tergantung sumbu yang dipakai.

Property **justify-content** bisa diisi dengan 6 nilai:

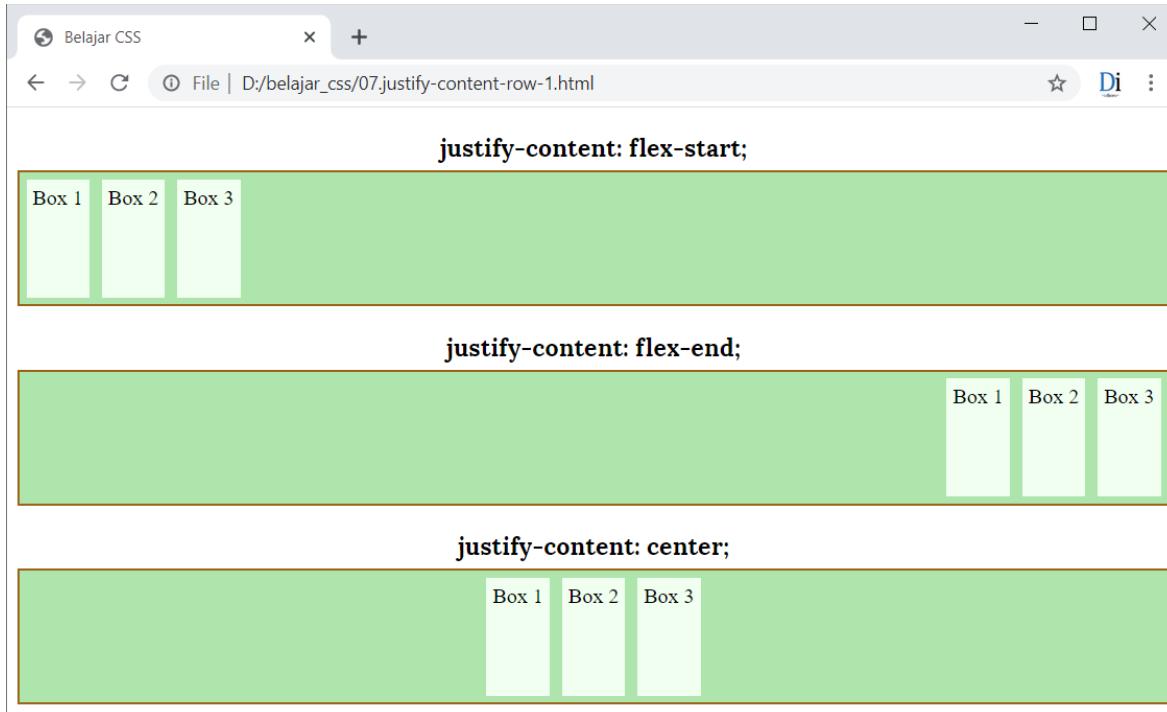
- `flex-start` (default)
- `flex-end`
- `center`
- `space-between`
- `space-around`
- `space-evenly`

Berikut contoh penggunaannya:

07.justify-content-row-1.html

```

1 <style>
2   .container {
3     ...
4     display: flex;
5     flex-direction: row;
6   }
7 </style>
8 ...
9
10 <h3>justify-content: flex-start;</h3>
11 <div class="container" style="justify-content: flex-start;">
12   <div class="box"> Box 1 </div>
13   <div class="box"> Box 2 </div>
14   <div class="box"> Box 3 </div>
15 </div>
16
17 <h3>justify-content: flex-end;</h3>
18 <div class="container" style="justify-content: flex-end;">
19   <div class="box"> Box 1 </div>
20   <div class="box"> Box 2 </div>
21   <div class="box"> Box 3 </div>
22 </div>
23
24 <h3>justify-content: center;</h3>
25 <div class="container" style="justify-content: center;">
26   <div class="box"> Box 1 </div>
27   <div class="box"> Box 2 </div>
28   <div class="box"> Box 3 </div>
29 </div>
```



Gambar: Hasil penggunaan property justify-content pada x axis

Di dalam selector .container saya menambah property `flex-direction: row`. Dengan demikian, main axis flex container ada di sumbu x (berjejer dari kiri ke kanan).

Nilai `justify-content: flex-start` akan mendorong semua flex item ke bagian awal, ini merupakan nilai default jika property `justify-content` tidak ditulis.

Sedangkan nilai `justify-content: flex-end` akan mendorong semua flex item ke bagian ujung, serta nilai `justify-content: center` akan memposisikan semua flex item di tengah flex container.

Lanjut, kita akan lihat praktek dari 3 nilai `justify-content` selanjutnya:

08.justify-content-row-2.html

```

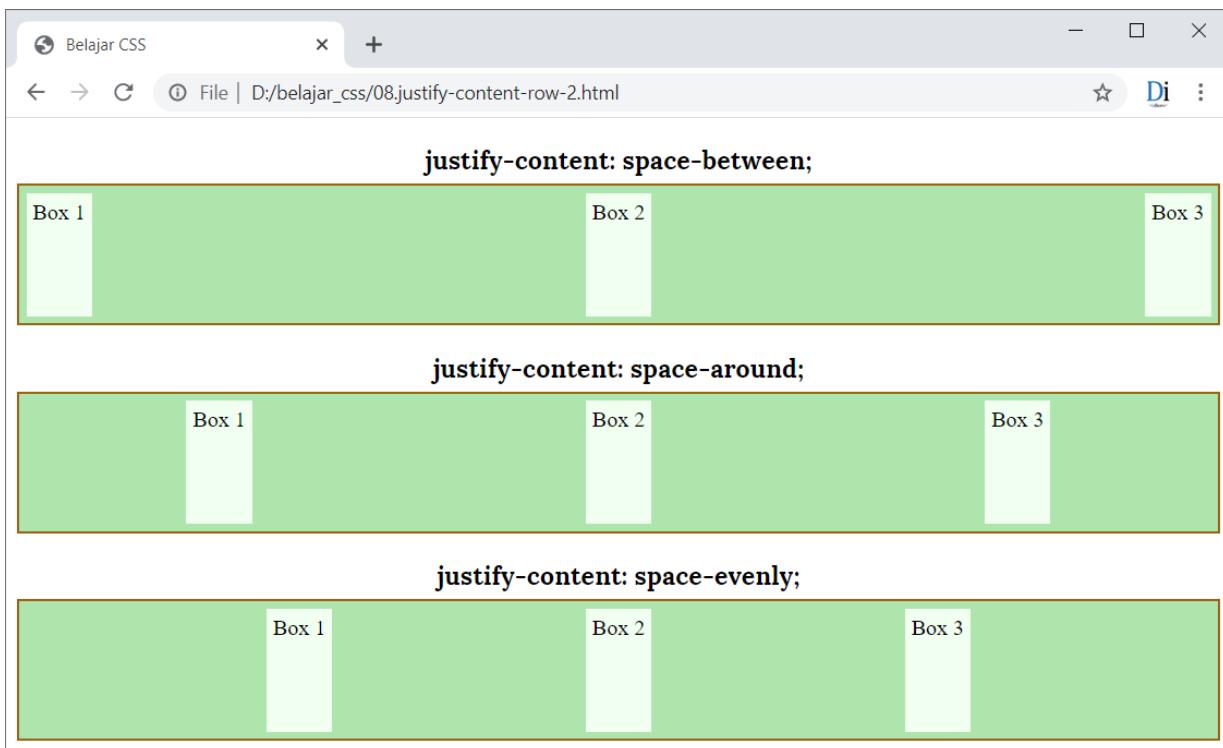
1 <style>
2   .container {
3     ...
4     display: flex;
5     flex-direction: row;
6   }
7   ...
8 </style>
9 ...
10
11 <h3>justify-content: space-between;</h3>
12 <div class="container" style="justify-content: space-between;">
13   <div class="box"> Box 1 </div>
14   <div class="box"> Box 2 </div>
15   <div class="box"> Box 3 </div>

```

```

16 </div>
17
18 <h3>justify-content: space-around;</h3>
19 <div class="container" style="justify-content: space-around;">
20   <div class="box"> Box 1 </div>
21   <div class="box"> Box 2 </div>
22   <div class="box"> Box 3 </div>
23 </div>
24
25 <h3>justify-content: space-evenly;</h3>
26 <div class="container" style="justify-content: space-evenly;">
27   <div class="box"> Box 1 </div>
28   <div class="box"> Box 2 </div>
29   <div class="box"> Box 3 </div>
30 </div>

```



Gambar: Hasil penggunaan property justify-content pada x axis

Nilai `justify-content: space-between` akan mendorong flex item pertama ke bagian awal, flex item terakhir ke bagian ujung, serta mengatur element lain diantaranya.

Nilai `justify-content: space-around` akan membagi spasi kosong sama besar untuk sisi kiri dan kanan setiap flex item. Perhatikan bahwa jarak spasi antara Box 1 dan Box 2 dua kali lebih besar daripada spasi di sisi kiri Box 1.

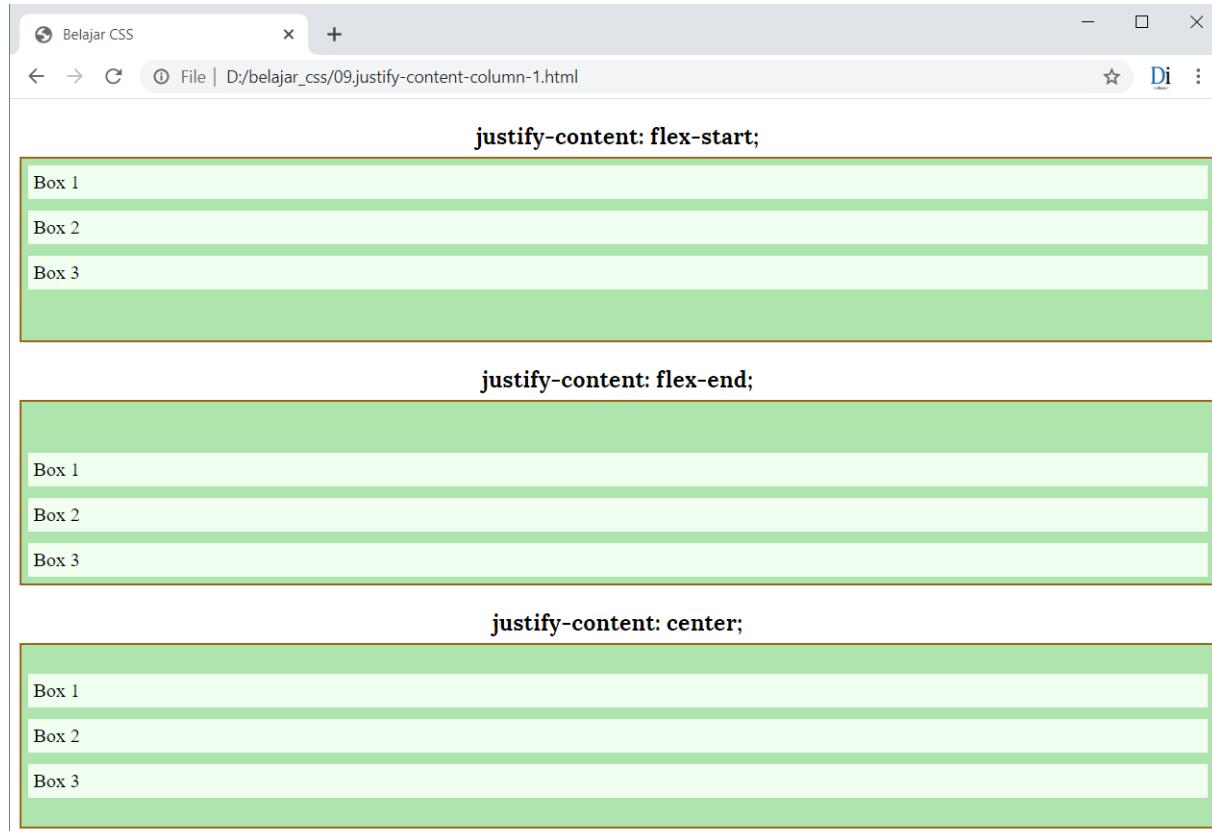
Nilai `justify-content: space-evenly` akan membagi spasi sama besar antar setiap flex item, termasuk dengan sisi dalam *flex container*. Sekarang ruang spasi antara Box 1 dan Box 2 sama besar dengan spasi di sisi kiri Box 1.

Praktek yang kita bahas ini baru untuk sisi sumbu x saja. Jika *flex container* di set dengan

property `flex-direction: column`, maka *main axis* akan berubah ke sumbu y, sehingga property `justify-content` akan mengatur urutan dari atas ke bawah, bukan lagi dari kiri ke kanan:

09.justify-content-column-1.html

```
1 <style>
2   .container {
3     ...
4     display: flex;
5     flex-direction: column;
6   }
7   ...
8 </style>
9 ...
10
11 <h3>justify-content: flex-start;</h3>
12 <div class="container" style="justify-content: flex-start;">
13   <div class="box"> Box 1 </div>
14   <div class="box"> Box 2 </div>
15   <div class="box"> Box 3 </div>
16 </div>
17
18 <h3>justify-content: flex-end;</h3>
19 <div class="container" style="justify-content: flex-end;">
20   <div class="box"> Box 1 </div>
21   <div class="box"> Box 2 </div>
22   <div class="box"> Box 3 </div>
23 </div>
24
25 <h3>justify-content: center;</h3>
26 <div class="container" style="justify-content: center;">
27   <div class="box"> Box 1 </div>
28   <div class="box"> Box 2 </div>
29   <div class="box"> Box 3 </div>
30 </div>
```



Gambar: Hasil penggunaan property justify-content pada y axis

Dengan menukar nilai property `flex-direction` pada selector `.container` menjadi `column`, maka main axis flexbox akan berubah ke arah sumbu y (dari atas ke bawah).

Penjelasan tentang nilai `justify-content: flex-start`, `justify-content: flex-end` dan `justify-content: center` tetap sama seperti sebelumnya, hanya saja sekarang posisi awal berada di baris atas, dan posisi akhir ada di sisi bawah.

Begin juga untuk nilai `space-between`, `space-around` dan `space-evenly`:

09.justify-content-column-2.html

```

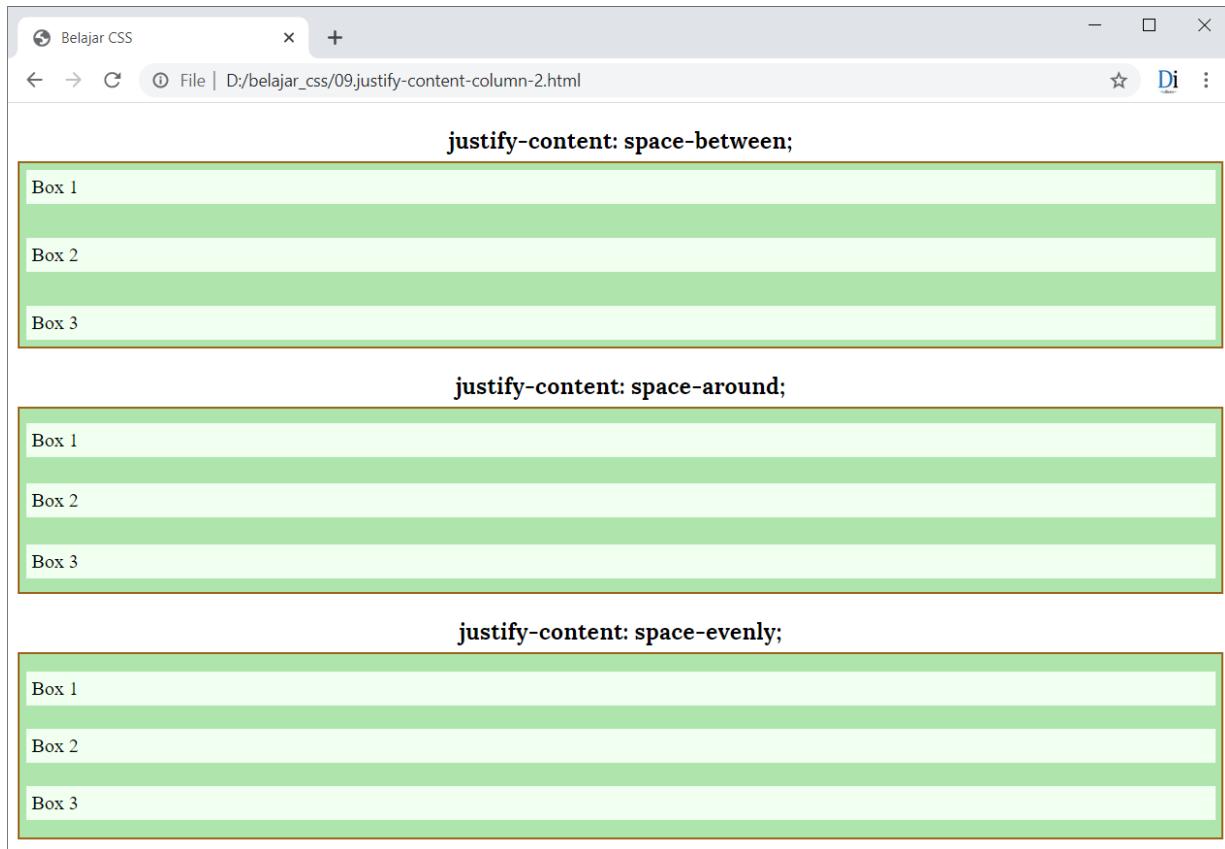
1 <style>
2   .container {
3     ...
4     display: flex;
5     flex-direction: column;
6   }
7   ...
8 </style>
9 ...
10
11 <h3>justify-content: space-between;</h3>
12 <div class="container" style="justify-content: space-between;">
13   <div class="box"> Box 1 </div>
14   <div class="box"> Box 2 </div>
15   <div class="box"> Box 3 </div>

```

```

16 </div>
17
18 <h3>justify-content: space-around;</h3>
19 <div class="container" style="justify-content: space-around;">
20   <div class="box"> Box 1 </div>
21   <div class="box"> Box 2 </div>
22   <div class="box"> Box 3 </div>
23 </div>
24
25 <h3>justify-content: space-evenly;</h3>
26 <div class="container" style="justify-content: space-evenly;">
27   <div class="box"> Box 1 </div>
28   <div class="box"> Box 2 </div>
29   <div class="box"> Box 3 </div>
30 </div>

```



Gambar: Hasil penggunaan property justify-content pada y axis

Efek penggunaan ketiga nilai ini juga sama seperti praktek pada sumbu x, hanya saja sekarang arah flex item berjejer dari atas ke bawah (sumbu y).

Perbedaan jarak spasi dalam contoh di atas mungkin kurang terlihat karena flex container saya set agak pendek (agar menghemat tempat). Untuk hasil yang lebih jelas, silahkan set nilai height flex container menjadi lebih besar, misalnya 500px.

18.6. Property align-items

Property **align-items** dipakai untuk mengatur posisi *flex item* pada **cross axis** (sumbu yang bersilangan dengan sumbu utama).

Maksudnya, jika sumbu utama *flex container* ada di arah sumbu x, maka property **align-items** dipakai untuk mengatur posisi *flex item* di sumbu y. Begitu juga sebaliknya, jika sumbu utama *flex container* ada di arah sumbu y, maka property **align-items** dipakai untuk mengatur posisi *flex item* pada sumbu x.

Definisi seperti ini membuat property **align-items** sering tertukar dengan **justify-content**, sebab fungsinya sangat mirip dan hanya bertukar peran tergantung sumbu utama yang dipakai.

Property **align-items** bisa diisi dengan 5 nilai, yakni:

- **stretch** (*default*)
- **flex-start**
- **flex-end**
- **center**
- **baseline**

Mari lihat contoh penggunaannya:

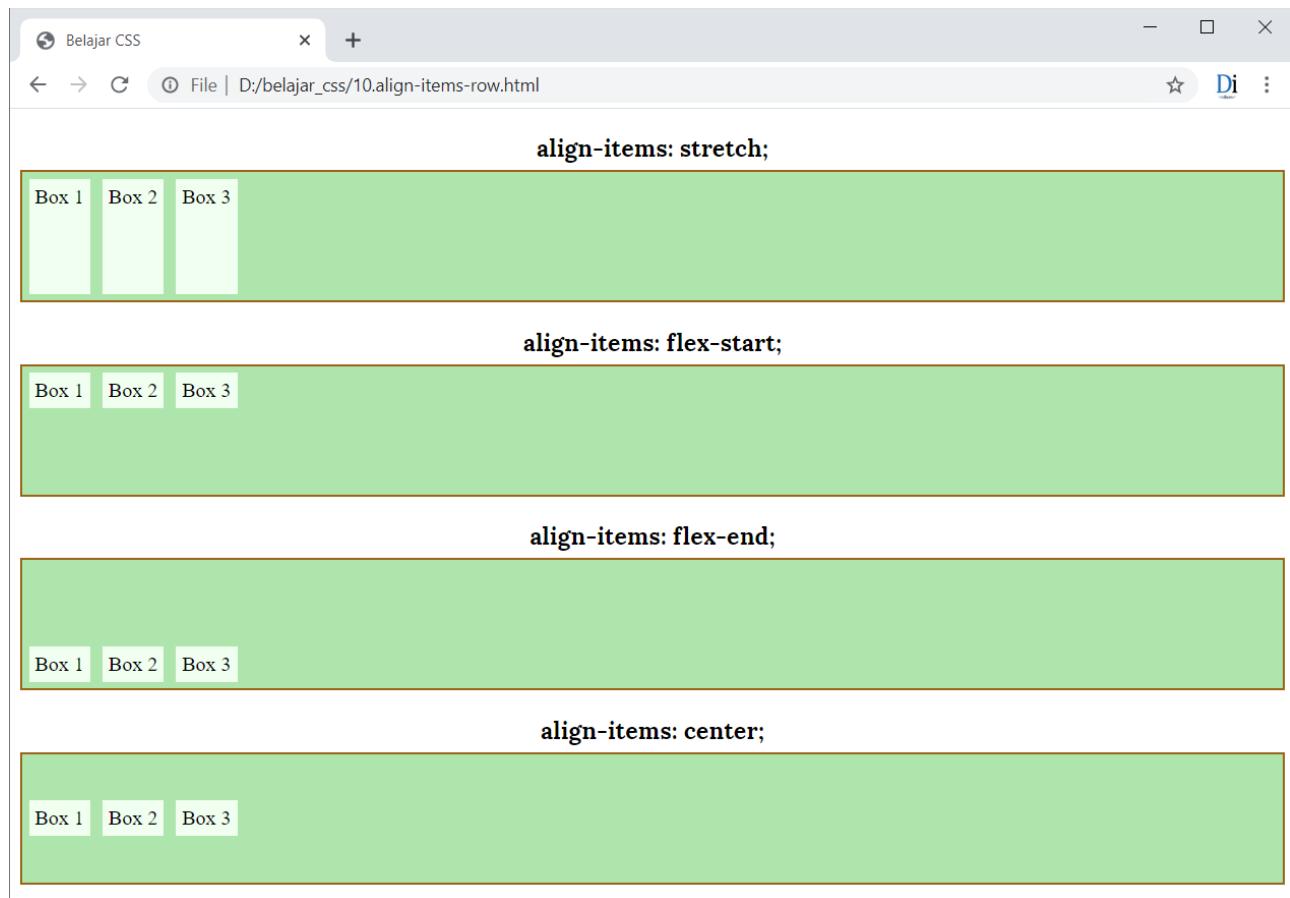
10.align-items-row.html

```

1 <style>
2   .container {
3     height: 100px;
4     border: 2px solid #965e15;
5     background-color: #AAE3A7;
6     display: flex;
7     flex-direction: row;
8   }
9   ...
10 </style>
11 ...
12
13 <h3>align-items: stretch;</h3>
14 <div class="container" style="align-items: stretch;">
15   <div class="box"> Box 1 </div>
16   <div class="box"> Box 2 </div>
17   <div class="box"> Box 3 </div>
18 </div>
19
20 <h3>align-items: flex-start;</h3>
21 <div class="container" style="align-items: flex-start;">
22   <div class="box"> Box 1 </div>
23   <div class="box"> Box 2 </div>
24   <div class="box"> Box 3 </div>
25 </div>
```

CSS3 FlexBox

```
26
27 <h3>align-items: flex-end;</h3>
28 <div class="container" style="align-items: flex-end;">
29   <div class="box"> Box 1 </div>
30   <div class="box"> Box 2 </div>
31   <div class="box"> Box 3 </div>
32 </div>
33
34 <h3>align-items: center;</h3>
35 <div class="container" style="align-items: center;">
36   <div class="box"> Box 1 </div>
37   <div class="box"> Box 2 </div>
38   <div class="box"> Box 3 </div>
39 </div>
```



Gambar: Hasil penggunaan property align-items pada x axis

Flex container saya set dengan `flex-direction: row`, sehingga main axis ada di sumbu x (arah kiri ke kanan). Dengan demikian property `align-items` akan mengatur posisi flex item di sumbu y (dari arah atas ke bawah).

Nilai `align-items: stretch` akan memperbesar semua flex item untuk memenuhi tinggi container, dengan syarat tidak ada property `height` yang mengatur tinggi dari flex item tersebut. Ini merupakan nilai default jika property `align-items` tidak ditulis.

Nilai `align-items: flex-start` akan mendorong semua flex item ke bagian awal (sisi atas).

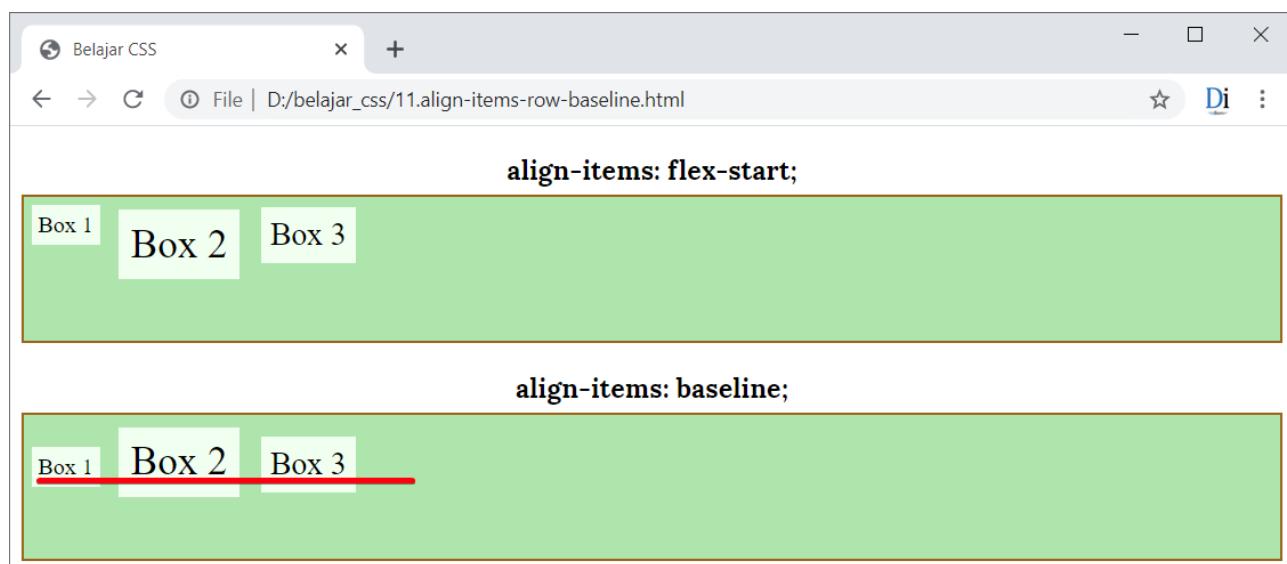
Begitu juga dengan `align-items: flex-end` yang akan mendorong flex item ke bagian akhir (sisi bawah), serta `align-items: center` akan mengatur semua flex item di bagian tengah.

Nilai terakhir untuk `align-items` adalah `baseline`. Ini dipakai untuk mengatur flex item berdasarkan garis dasar teks. Efeknya baru bisa terlihat jika flex item memiliki tinggi font yang berbeda-beda:

11.align-items-row-baseline.html

```

1 <style>
2   .container {
3     height: 100px;
4     border: 2px solid #965e15;
5     background-color: #AAE3A7;
6     display: flex;
7     flex-direction: row;
8   }
9   ...
10 </style>
11 ...
12
13 <h3>align-items: flex-start;</h3>
14 <div class="container" style="align-items:flex-start;">
15   <div class="box" style="font-size: 16px;"> Box 1 </div>
16   <div class="box" style="font-size: 28px;"> Box 2 </div>
17   <div class="box" style="font-size: 22px;"> Box 3 </div>
18 </div>
19
20 <h3>align-items: baseline;</h3>
21 <div class="container" style="align-items:baseline;">
22   <div class="box" style="font-size: 16px;"> Box 1 </div>
23   <div class="box" style="font-size: 28px;"> Box 2 </div>
24   <div class="box" style="font-size: 22px;"> Box 3 </div>
25 </div>
```



Gambar: Hasil penggunaan property align-items baseline pada x axis

Ukuran font setiap box saya buat berlainan dengan property `font-size`. Saat flex container di set dengan `align-items: flex-start`, setiap box disusun berdasarkan sisi paling atas.

Hasilnya terlihat agak berjenjang karena efek `padding: 0.3em` di selector `.box`. Jika anda masih ingat, satuan `em` ini bergantung ke ukuran `font-size`.

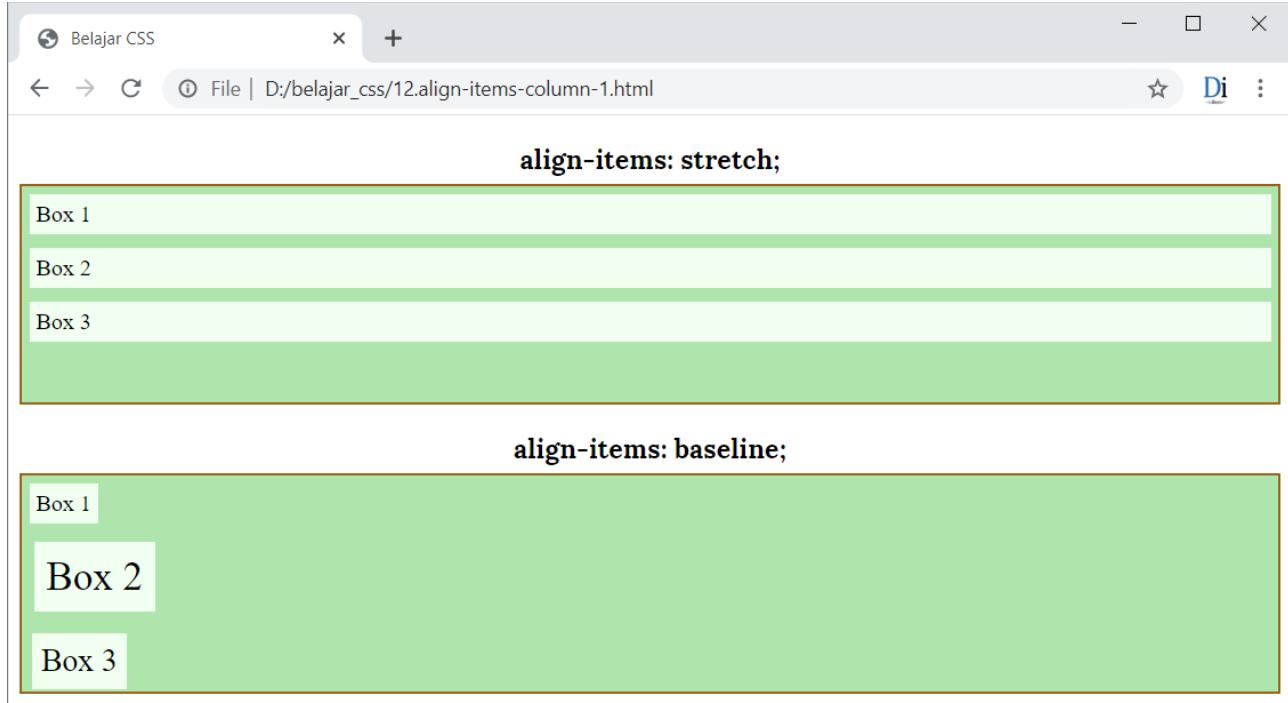
Ketika flex container di set dengan `align-items: baseline`, setiap box diatur berdasarkan `baseline`, yakni garis dasar setiap teks. Dalam gambar di atas, `baseline` ini terlihat pada garis bantu berwarna merah.

Lanjut, jika main axis atau sumbu utama flex container di ubah searah sumbu y, (melalui property `flex-direction: column`), maka efek property `align-item` akan berpengaruh ke `cross axis`, yakni arah kiri-kanan. Kita akan lihat efeknya untuk nilai `stretch` dan `baseline` terlebih dahulu:

12.align-items-column-1.html

```

1 <style>
2   .container {
3     height: 200px;
4     border: 2px solid #965e15;
5     background-color: #AAE3A7;
6     display: flex;
7     flex-direction: column;
8   }
9 ...
10 </style>
11 ...
12
13 <h3>align-items: stretch;</h3>
14 <div class="container" style="align-items: stretch;">
15   <div class="box"> Box 1 </div>
16   <div class="box"> Box 2 </div>
17   <div class="box"> Box 3 </div>
18 </div>
19
20 <h3>align-items: baseline;</h3>
21 <div class="container" style="align-items: baseline;">
22   <div class="box" style="font-size: 16px;"> Box 1 </div>
23   <div class="box" style="font-size: 28px;"> Box 2 </div>
24   <div class="box" style="font-size: 22px;"> Box 3 </div>
25 </div>
```



Gambar: Hasil penggunaan property align-items stretch dan baseline pada y axis

Untuk nilai `align-items: stretch`, flex item akan dipaksa memanjang memenuhi parent element selama flex item tersebut tidak memiliki property `width`. Ini merupakan nilai default jika property `align-items` tidak ditulis.

Sedangkan untuk nilai `align-items: baseline` tidak terlihat efek perubahan karena `baseline` ini bekerja di sumbu x, sehingga tidak berpengaruh.

Berikut contoh untuk nilai `flex-start`, `flex-end` dan `center`:

13.align-items-column-2.html

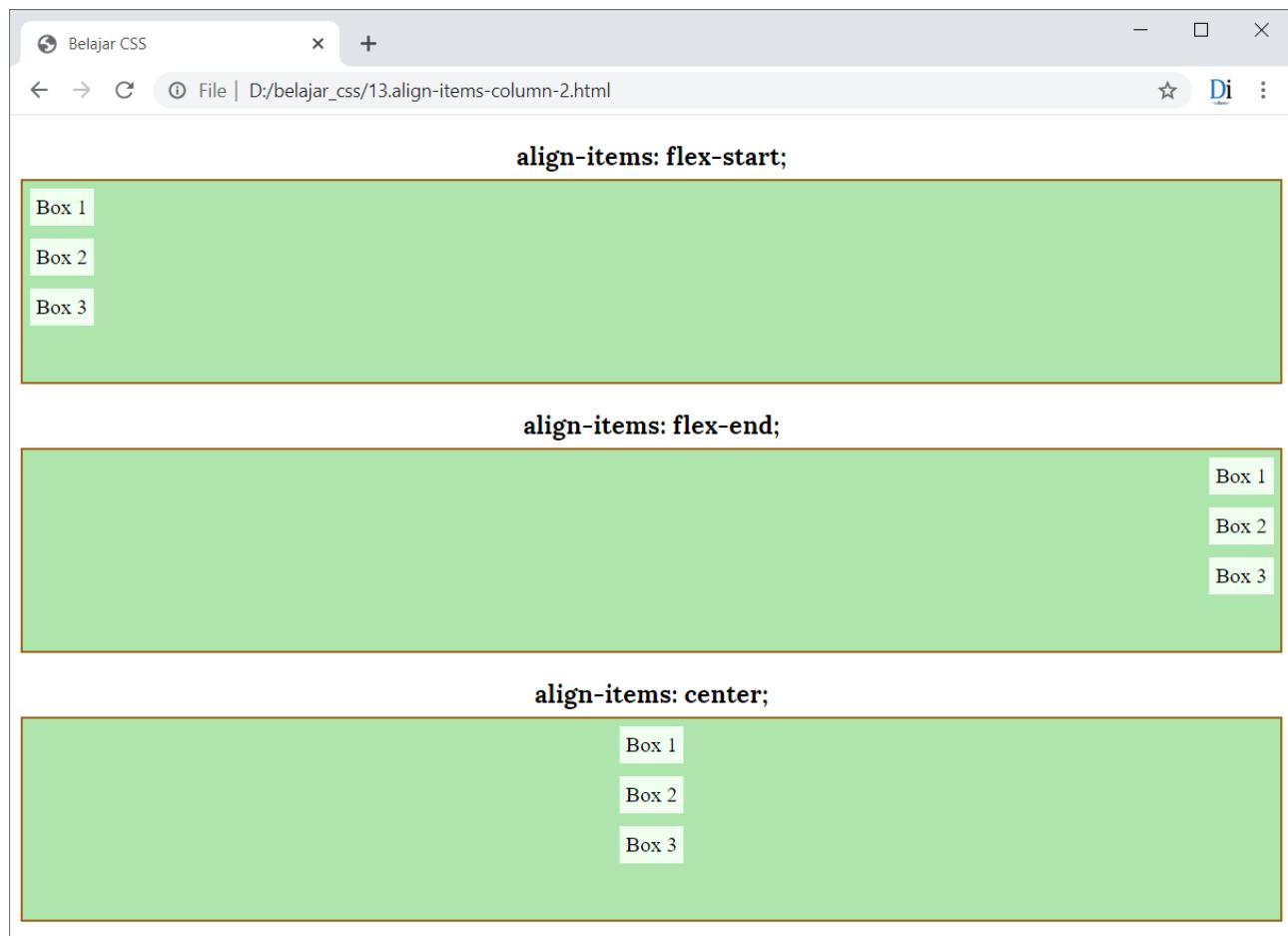
```

1 <style>
2   .container {
3     height: 150px;
4     border: 2px solid #965e15;
5     background-color: #AAE3A7;
6     display: flex;
7     flex-direction: column;
8   }
9   ...
10 </style>
11 ...
12
13 <h3>align-items: flex-start;</h3>
14 <div class="container" style="align-items: flex-start;">
15   <div class="box"> Box 1 </div>
16   <div class="box"> Box 2 </div>
17   <div class="box"> Box 3 </div>
18 </div>
19

```

CSS3 FlexBox

```
20 <h3>align-items: flex-end;</h3>
21 <div class="container" style="align-items: flex-end;">
22   <div class="box"> Box 1 </div>
23   <div class="box"> Box 2 </div>
24   <div class="box"> Box 3 </div>
25 </div>
26
27 <h3>align-items: center;</h3>
28 <div class="container" style="align-items: center;">
29   <div class="box"> Box 1 </div>
30   <div class="box"> Box 2 </div>
31   <div class="box"> Box 3 </div>
32 </div>
```



Gambar: Hasil penggunaan property align-items pada y axis

Seperti yang bisa di tebak, setiap nilai akan menggeser posisi box ke arah awal (kiri), akhir (kanan) dan tengah flex container.

Pasangan property `justify-content` dan `align-items` memang sangat mirip, dan saya sendiri masih sering tertukar saat ingin menggunakannya. Kunci perbedaan ada di **main axis** dan **cross axis**.

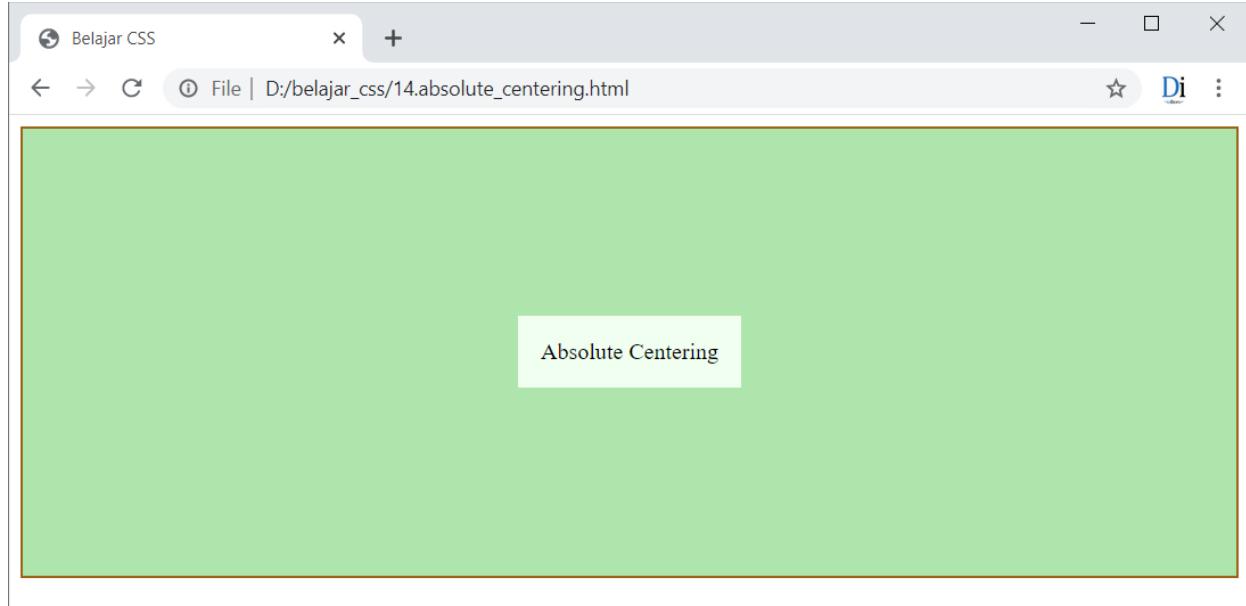
Membuat Efek Absolute Centering

Melalui property `justify-content` dan `align-items`, kita bisa membuat *absolute centering* dengan mudah.

Absolute centering adalah istilah dimana sebuah element persis di tengah parent element secara horizontal dan vertikal sekaligus. Ini cukup sulit dilakukan menddungkan float layout. Dengan flexbox, cukup isi nilai center ke dalam property `justify-content` dan `align-items`:

14.absolute_centering.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 90vh;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             display: flex;
12             flex-direction: row;
13             justify-content: center;
14             align-items: center;
15         }
16         .box {
17             background-color: honeydew;
18             margin: 1em;
19             padding: 1em;
20         }
21     </style>
22 </head>
23 <body>
24     <div class="container">
25         <div class="box"> Absolute Centering </div>
26     </div>
27 </body>
28 </html>
```



Gambar: Membuat absolute centering menggunakan flexbox

Inilah salah satu keunggulan penggunaan flexbox untuk mengatur posisi element.

18.7. Property align-content

Property **align-content** dipakai untuk mengatur posisi flex item di dalam *fleksibel flex container* (container yang di set dengan `flex-wrap: wrap`). Property **align-content** tidak berfungsi jika flex container di set sebagai `nowrap`.

Selain itu efeknya baru bisa terlihat jika jumlah flex item cukup banyak serta diatur pada sumbu x (menggunakan `flex-direction: row` atau `flex-direction: row-reverse`).

Terdapat 7 nilai yang bisa diisi untuk **align-content**:

- `flex-start`
- `flex-end`
- `center`
- `space-between`
- `space-around`
- `space-evenly`
- `stretch (default)`

Kita akan lihat hasil praktik dari semua nilai ini:

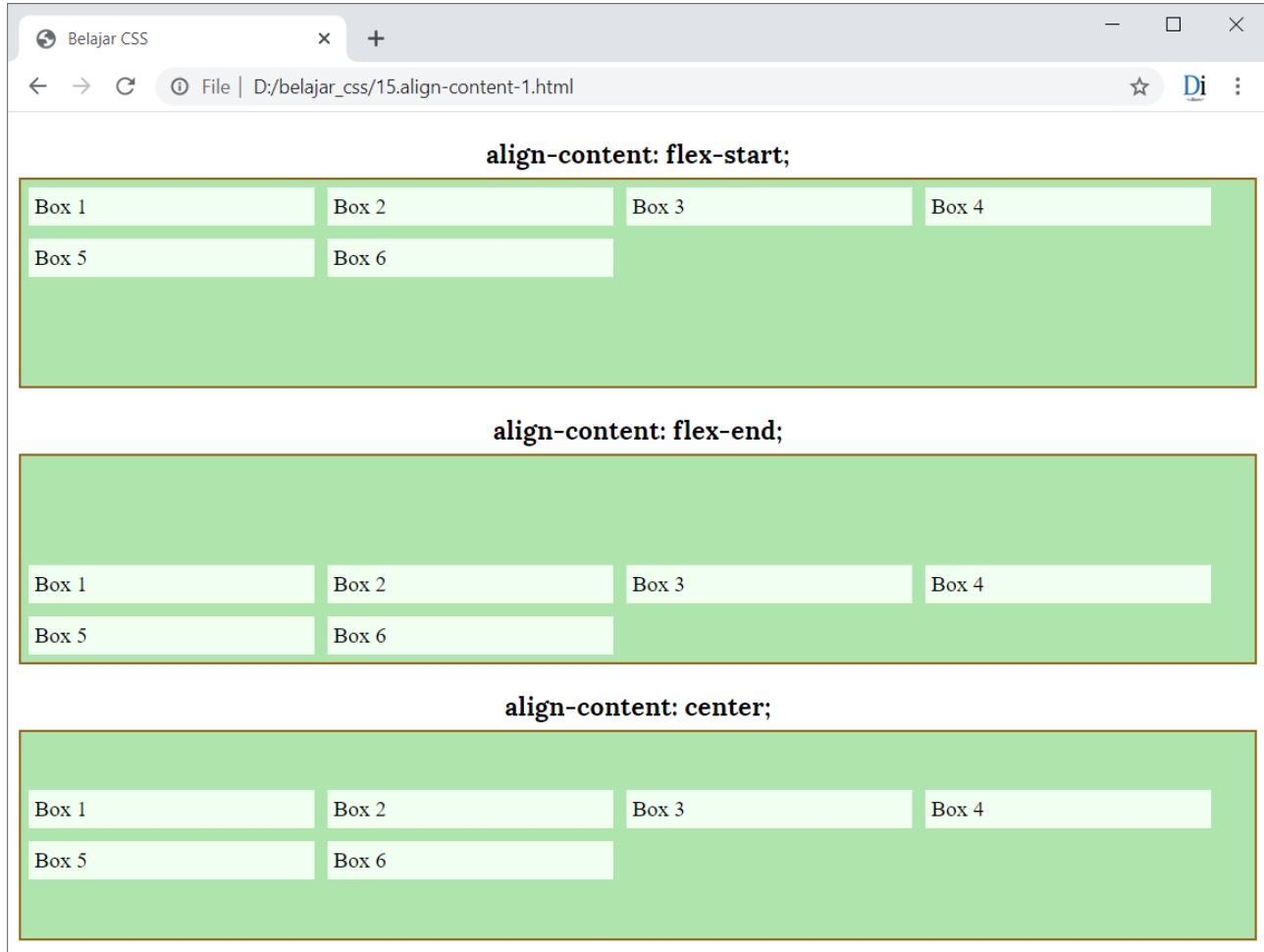
15.align-content-1.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
```

```
5   <title>Belajar CSS</title>
6   <style>
7     .container {
8       height: 150px;
9       border: 2px solid #965e15;
10      background-color: #AAE3A7;
11      display: flex;
12      flex-flow: row wrap;
13    }
14    .box {
15      width: 200px;
16      background-color: honeydew;
17      margin: 0.3em;
18      padding: 0.3em;
19    }
20    h3{
21      text-align: center;
22      font-family: lora,serif;
23      margin-bottom: 0.3em;
24    }
25  </style>
26 </head>
27 <body>
28
29 <h3>align-content: flex-start;</h3>
30 <div class="container" style="align-content: flex-start;">
31   <div class="box"> Box 1 </div>
32   <div class="box"> Box 2 </div>
33   <div class="box"> Box 3 </div>
34   <div class="box"> Box 4 </div>
35   <div class="box"> Box 5 </div>
36   <div class="box"> Box 6 </div>
37 </div>
38
39 <h3>align-content: flex-end;</h3>
40 <div class="container" style="align-content: flex-end;">
41   <div class="box"> Box 1 </div>
42   <div class="box"> Box 2 </div>
43   <div class="box"> Box 3 </div>
44   <div class="box"> Box 4 </div>
45   <div class="box"> Box 5 </div>
46   <div class="box"> Box 6 </div>
47 </div>
48
49 <h3>align-content: center;</h3>
50 <div class="container" style="align-content: center;">
51   <div class="box"> Box 1 </div>
52   <div class="box"> Box 2 </div>
53   <div class="box"> Box 3 </div>
54   <div class="box"> Box 4 </div>
55   <div class="box"> Box 5 </div>
56   <div class="box"> Box 6 </div>
57 </div>
58
59 </body>
```

60 </html>



Gambar: Hasil penggunaan property align-content

Kali ini setiap flex container memiliki 6 flex item, selain itu setiap box di set dengan `width: 200px`. Agar efek property `align-content` bisa terlihat, flex container saya set dengan `flex-flow: row wrap`, sehingga jika tidak muat dalam satu baris, flex item akan membentuk baris baru.

Sekilas, efek semua nilai `align-content` terlihat mirip seperti `justify-content`. Bedanya, `align-content` berefek untuk flex item yang terdiri dari 2 baris atau lebih.

Nilai `align-content: flex-start` akan mendorong setiap baris flex item ke bagian atas, `align-content: flex-end` akan mendorong setiap baris flex item ke bagian bawah, serta `align-content: center` untuk menempatkan baris flex item di bagian tengah flex container.

Lanjut, berikut hasil untuk nilai `space-between`, `space-around` dan `space-evenly`:

15.align-content-2.html

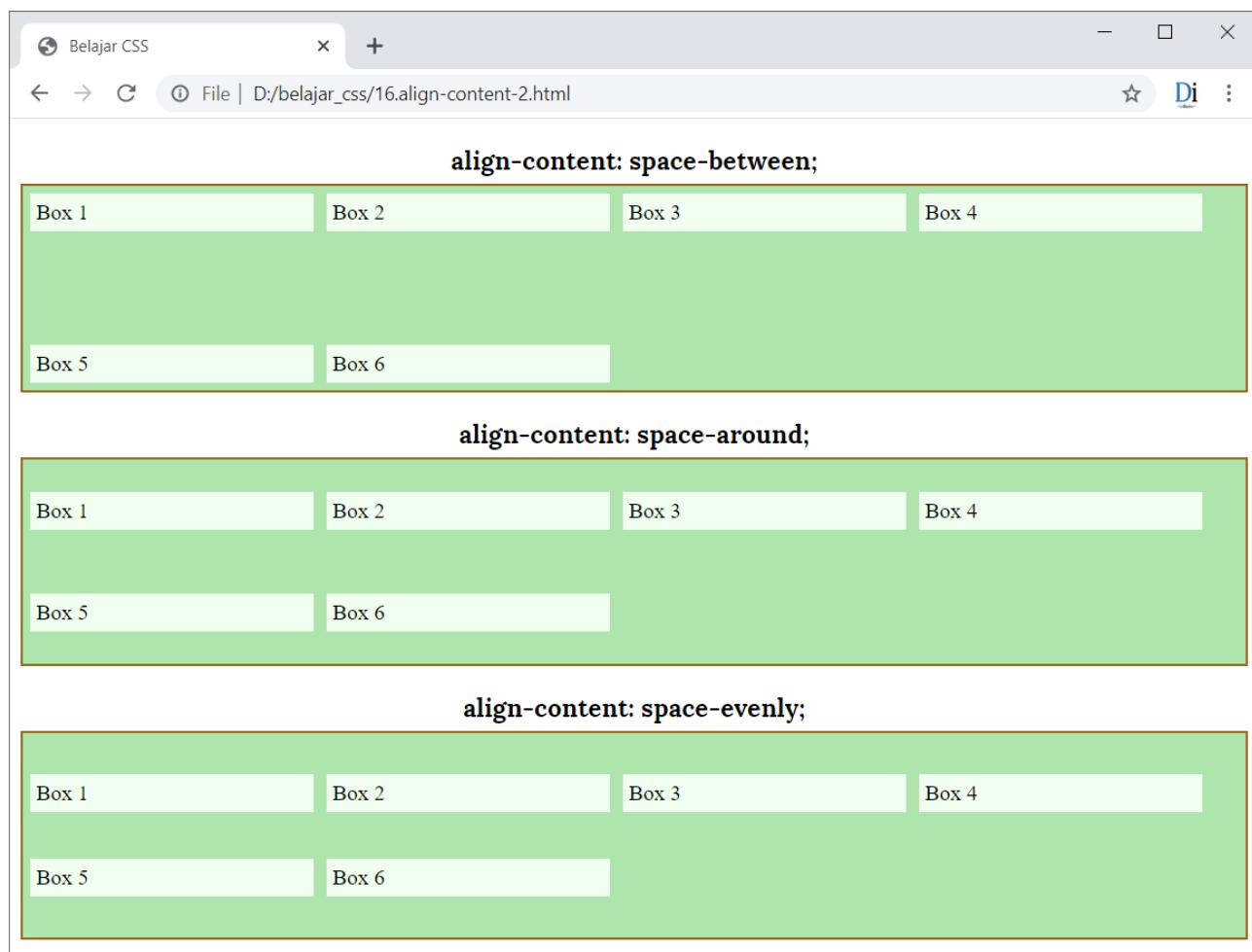
```

1 ...
2 <h3>align-content: space-between;</h3>
3 <div class="container" style="align-content: space-between;">

```

CSS3 FlexBox

```
4   <div class="box"> Box 1 </div>
5   ...
6   <div class="box"> Box 6 </div>
7 </div>
8
9 <h3>align-content: space-around;</h3>
10 <div class="container" style="align-content: space-around;">
11   <div class="box"> Box 1 </div>
12   ...
13   <div class="box"> Box 6 </div>
14 </div>
15
16 <h3>align-content: space-evenly;</h3>
17 <div class="container" style="align-content: space-evenly;">
18   <div class="box"> Box 1 </div>
19   ...
20   <div class="box"> Box 6 </div>
21 </div>
```



Gambar: Hasil penggunaan property align-content

Nilai `justify-content: space-between` akan mendorong flex item baris pertama ke bagian atas, flex item baris terakhir ke bagian bawah, serta mengatur element lain diantaranya (jika ada).

Nilai `justify-content: space-around` akan membagi spasi kosong sama besar untuk sisi atas dan bawah setiap baris flex item. Perhatikan bahwa jarak spasi antara baris pertama dan baris kedua lebih besar daripada spasi di atas baris pertama.

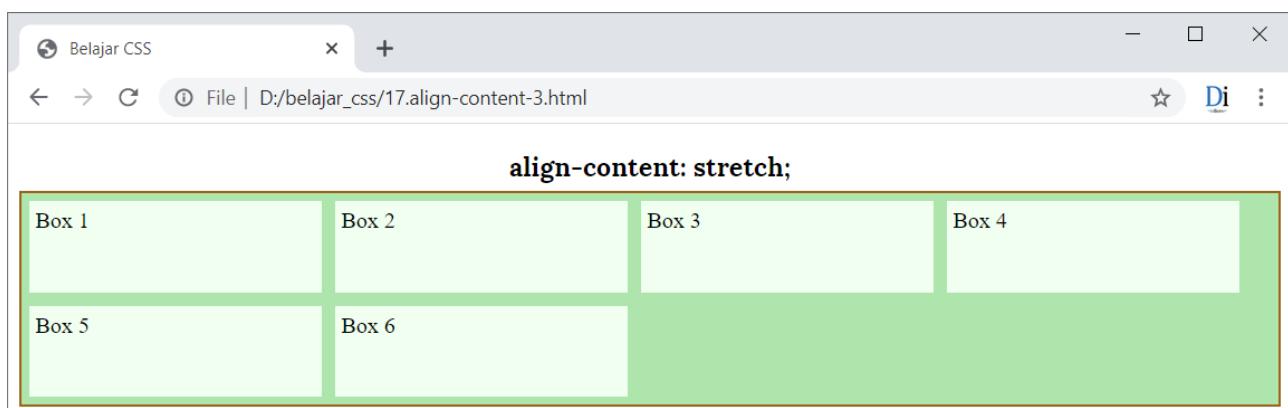
Nilai `justify-content: space-evenly` akan membagi spasi sama besar antar setiap baris element, termasuk dengan sisi dalam *flex container*. Sekarang ruang spasi antara baris pertama dan baris kedua akan sama besar dengan spasi di atas baris pertama.

Terakhir, kita lihat efek nilai `stretch`:

16.align-content-3.html

```

1 ...
2 <h3>align-content: stretch;</h3>
3 <div class="container" style="align-content: stretch;">
4   <div class="box"> Box 1 </div>
5 ...
6   <div class="box"> Box 6 </div>
7 </div>
```



Gambar: Hasil penggunaan property `align-content: stretch`

Nilai `align-content: stretch` akan memperbesar semua flex item untuk memenuhi tinggi *flex container*, selama tidak ada property `height` yang mengatur tinggi item tersebut. Ini merupakan nilai default jika property `align-content` tidak ditulis.

Bahasan tentang `align-content` ini menutup materi property untuk *flex container*, berikutnya kita akan masuk ke kelompok property yang akan mengatur *flex item* secara individu.

18.8. Property `flex-basis`

Property `flex-basis` dipakai untuk mengatur lebar atau tinggi dasar dari *flex item*. Nilai yang bisa diinput berupa satuan `length` seperti px, em, %, serta nilai keyword `auto` dan `content`.

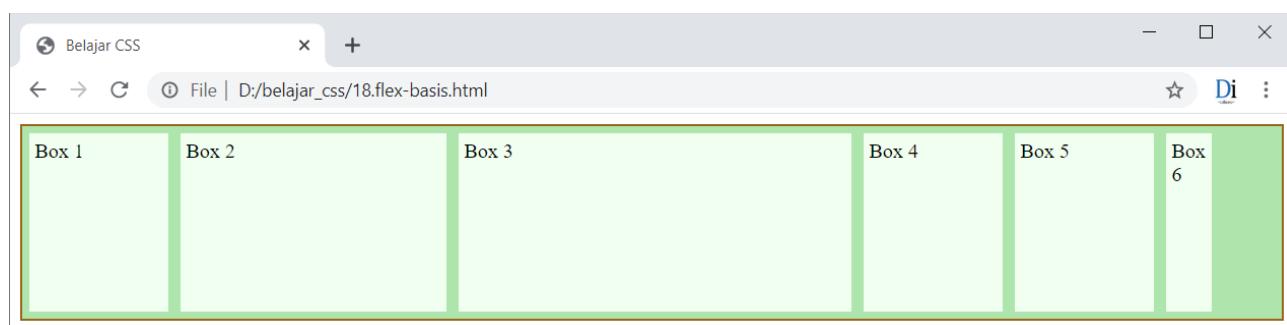
Jika element yang di set dengan `flex-basis` memiliki property `length` atau `height`, nilai milik `flex-basis` akan di menimpa nilai tersebut. Berikut contoh penggunaannya:

18.flex-basis.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             display: flex;
12         }
13         .box {
14             width: 100px;
15             background-color: honeydew;
16             margin: 0.3em;
17             padding: 0.3em;
18         }
19     </style>
20 </head>
21 <body>
22
23 <div class="container">
24     <div class="box"> Box 1 </div>
25     <div class="box" style="flex-basis: 200px;"> Box 2 </div>
26     <div class="box" style="flex-basis: 300px;"> Box 3 </div>
27     <div class="box" style="flex-basis: content;"> Box 4 </div>
28     <div class="box" style="flex-basis: auto;"> Box 5 </div>
29     <div class="box" style="flex-basis: 0;"> Box 6 </div>
30 </div>
31
32 </body>
33 </html>

```



Gambar: Hasil penggunaan property flex-basis

Di dalam tag `<style>`, selector `.box` memiliki property `width: 100px`, ini berarti lebar dari setiap box akan di set di 100px.

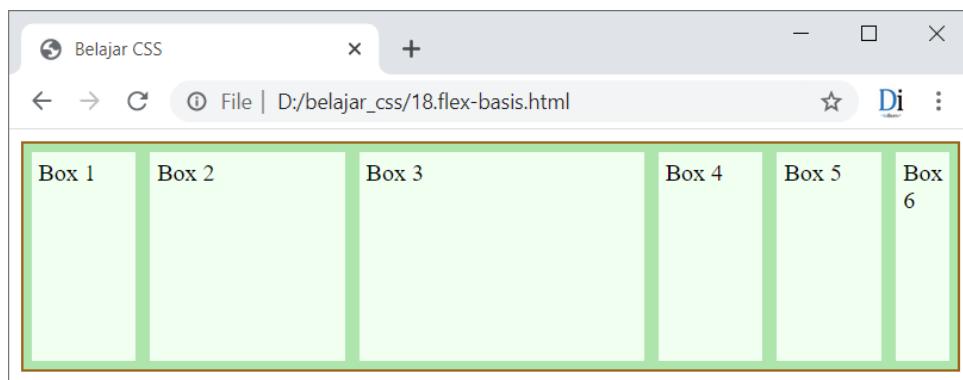
Akan tetapi untuk Box 2 dan Box 3 saya tambah dengan inline CSS `flex-basis: 200px` dan `flex-basis: 300px`. Efeknya, lebar Box 2 dan Box 3 akan menjadi 200px dan 300px. Property ini akan menimpa nilai `width: 100px`.

Box 4 di set dengan `flex-basis: content`. Nilai ini mengatur lebar element sesuai dengan konten yang ada di dalamnya, kecuali jika element tersebut memiliki atribut `width` atau `height`. Dalam contoh di atas, selector `.box` sudah memiliki property `width`, sehingga lebar Box 3 akan diambil dari property `width`.

Box 5 di set dengan `flex-basis: auto`. Nilai `auto` akan melihat apakah element ini memiliki `width` atau `height`. Jika ada, nilai itulah yang akan dipakai. Jika tidak ada, maka dianggap sama dengan `flex-basis: content`.

Box 6 di set dengan `flex-basis: 0`. Ini akan membuat lebar element sekecil mungkin. Jika berisi teks, maka lebar flex item ditentukan dari satu kata terpanjang.

Sama seperti property `width`, property `flex-basis` sebenarnya dianggap sebagai `max-width`, yakni jika lebar flex container tidak cukup besar, setiap element dipaksa mengecil dengan mempertahankan proporsinya:



Gambar: Flex item ikut mengecil saat flex container di perkecil

Property `flex-basis` akan lebih banyak dipakai untuk mengatur lebar element, dimana main axis dari flex container ada di sumbu x, namun kita juga memakai property `flex-basis` di sumbu y. Syaratnya, main axis flex container harus di set ke sumbu y menggunakan `flex-direction: column` atau `flex-direction: column-reverse`.

Berikut contoh prakteknya:

19.flex-basis-column.html

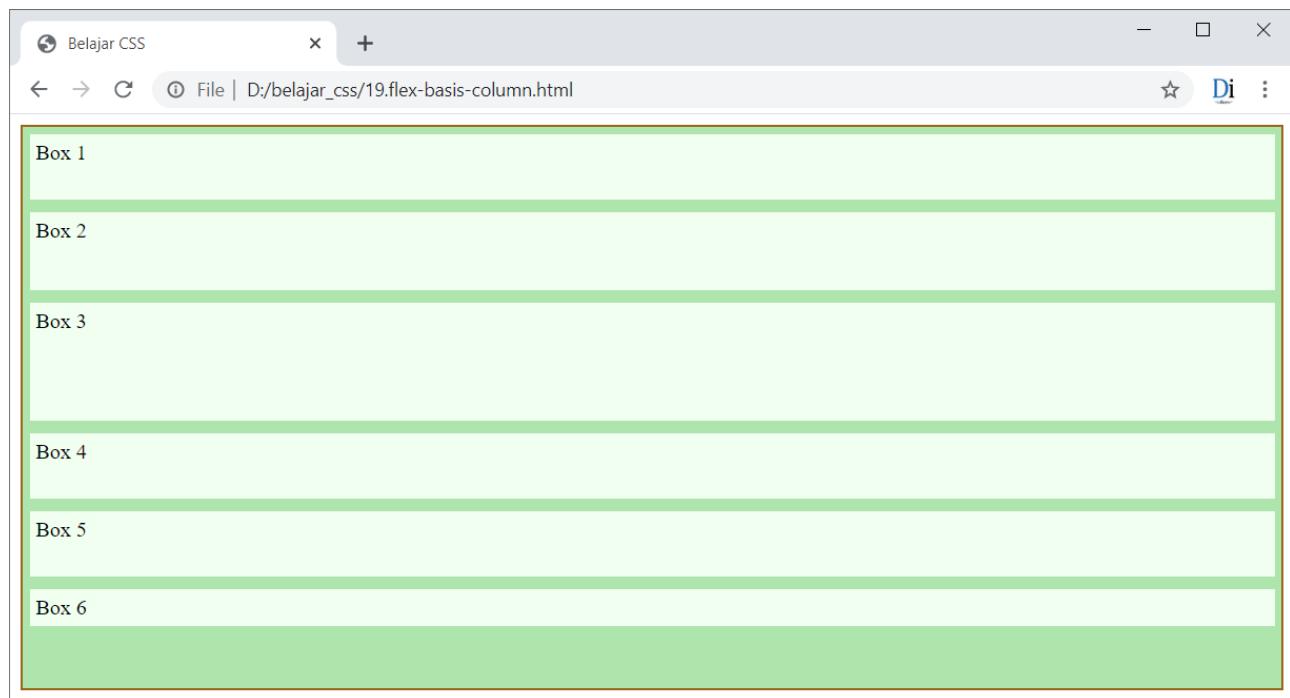
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 95vh;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             display: flex;
12             flex-direction: column;

```

CSS3 FlexBox

```
13     }
14     .box {
15         height: 40px;
16         background-color: honeydew;
17         margin: 0.3em;
18         padding: 0.3em;
19     }
20 </style>
21 </head>
22 <body>
23
24 <div class="container">
25     <div class="box"> Box 1 </div>
26     <div class="box" style="flex-basis: 50px;"> Box 2 </div>
27     <div class="box" style="flex-basis: 80px;"> Box 3 </div>
28     <div class="box" style="flex-basis: content;"> Box 4 </div>
29     <div class="box" style="flex-basis: auto;"> Box 5 </div>
30     <div class="box" style="flex-basis: 0;"> Box 6 </div>
31 </div>
32
33 </body>
34 </html>
```



Gambar: Property flex-basis dipakai untuk sumbu y

Di baris 12 saya menambah `flex-direction: column` ke dalam class `.container`. Ini akan membuat flex item berjejer dari atas ke bawah. Dengan pengaturan ini, property `flex-basis` akan berpengaruh ke tinggi element, yakni sebagai pengganti `height`.

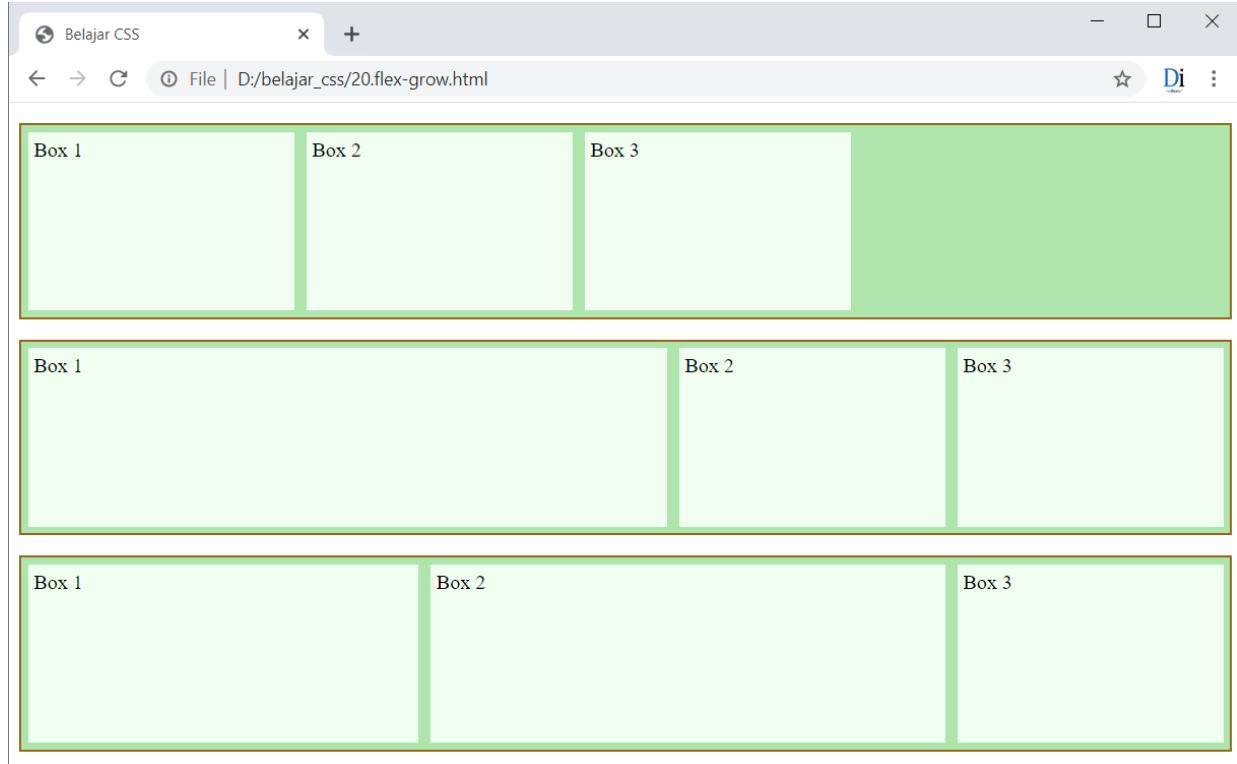
18.9. Property flex-grow

Property **flex-grow** berfungsi untuk menentukan skala pembesaran flex item jika lebar atau tinggi flex container mengizinkan element tersebut untuk diperbesar. Nilai yang bisa diisi berupa angka tanpa satuan seperti 1, 2, 5, dst.

Berikut contoh praktik penggunaan property ini:

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             display: flex;
12             margin-top: 1em;
13         }
14         .box {
15             width: 200px;
16             background-color: honeydew;
17             margin: 0.3em;
18             padding: 0.3em;
19         }
20     </style>
21 </head>
22 <body>
23
24     <div class="container">
25         <div class="box"> Box 1 </div>
26         <div class="box"> Box 2 </div>
27         <div class="box"> Box 3 </div>
28     </div>
29
30     <div class="container">
31         <div class="box" style="flex-grow: 1;"> Box 1 </div>
32         <div class="box"> Box 2 </div>
33         <div class="box"> Box 3 </div>
34     </div>
35
36     <div class="container">
37         <div class="box" style="flex-grow: 1;"> Box 1 </div>
38         <div class="box" style="flex-grow: 2;"> Box 2 </div>
39         <div class="box"> Box 3 </div>
40     </div>
41
42 </body>
43 </html>
```



Gambar: Hasil penggunaan property `flex-grow`

Tampilan flex container pertama tidak menggunakan property `flex-grow` sama sekali. Hasilnya, ketiga Box memiliki lebar sesuai dengan yang tertulis di property `width`, yakni 200px.

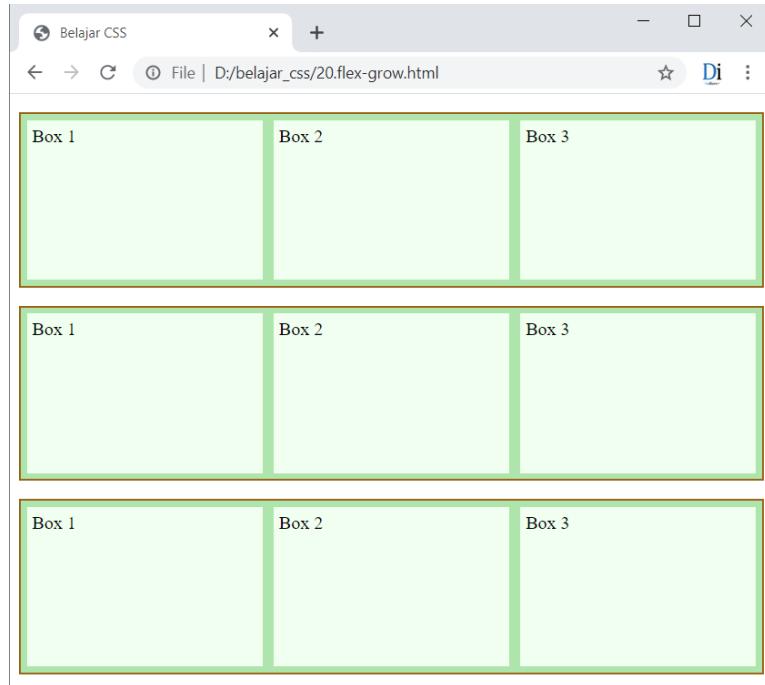
Untuk flex container kedua, saya menambah property `flex-grow: 1` di baris 31. Hasilnya, Box 1 langsung membesar untuk memenuhi sisa ruang yang tersedia. Efek ini **sangat penting** untuk dipahami bahwa begitu sebuah element memiliki property `flex-grow`, ia langsung membesar memenuhi semua sisa ruang yang ada.

Pada flex container ketiga, terdapat property `flex-grow: 1` untuk Box 1, serta `flex-grow: 2` untuk Box 2. Dengan tambahan ini, Box 1 dan Box 2 akan berbagi ruang sisa dengan skala 1:2.

Seandainya ruang flex container tersisa 600 pixel, maka Box 1 akan mendapat tambahan 200 pixel, dan Box 2 mendapat tambahan 400 pixel. Ini sesuai dengan perbandingan nilai `flex-grow` antar kedua element tersebut.

Sesuai dengan namanya, efek property `flex-grow` baru terlihat apabila ada "ruang tumbuh" di dalam flex container. Jika tidak ada ruang kosong, `flex-grow` tidak akan berefek dan lebar setiap element bergantung pada property `width`.

Berikut tampilan kode yang sama dengan mengecilkan ukuran web browser:



Gambar: Property flex-grow tidak berefek jika tidak ada ruang berlebih di dalam flex container

Ketiga flex container tampil sama karena tidak ada ruang untuk memperbesar diri.

Nilai default dari `flex-grow` adalah 0 dimana lebar setiap element sesuai dengan nilai property `width` dan tidak akan "bertumbuh".

Efek property `flex-grow` juga bisa dipakai untuk mengatur tinggi sebuah element, dengan syarat main axis flex container di ubah ke sumbu y.

18.10. Property `flex-shrink`

Property `flex-shrink` merupakan kebalikan dari `flex-grow`. Property ini berfungsi untuk menentukan skala pengecilan flex item jika lebar atau tinggi flex container memaksa element tersebut untuk mengecil. Nilai yang bisa diisi berupa angka tanpa satuan seperti 1, 2, 5, dst.

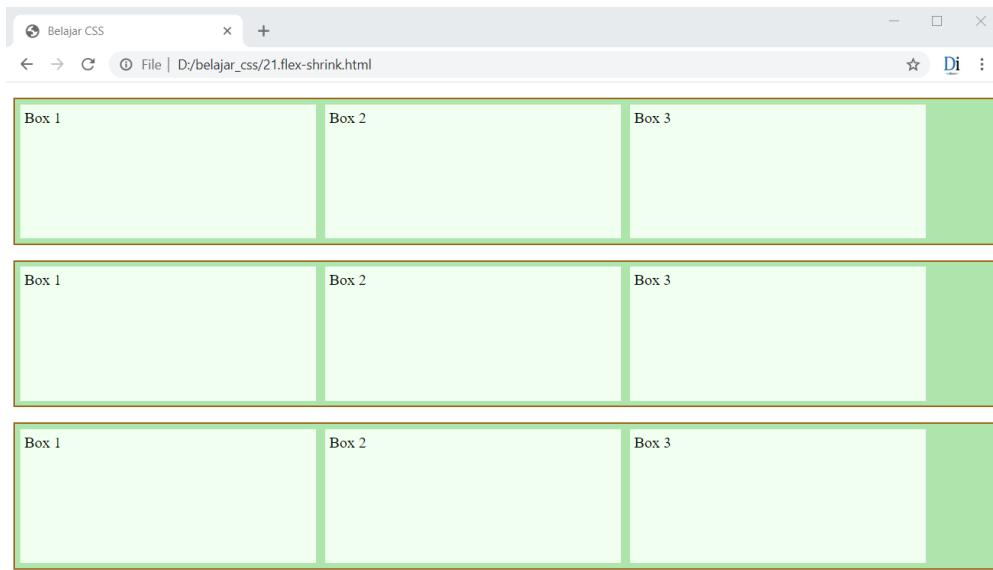
Berikut contoh praktik penggunaan property `flex-shrink`:

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             display: flex;
12             margin-top: 1em;
13      }
  
```

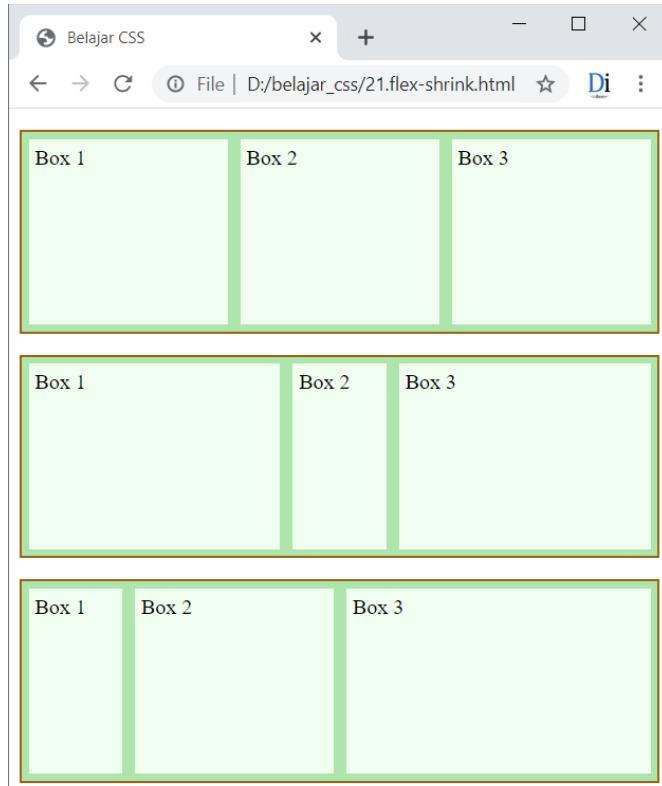
CSS3 FlexBox

```
14 .box {
15     width: 300px;
16     background-color: honeydew;
17     margin: 0.3em;
18     padding: 0.3em;
19 }
20 </style>
21 </head>
22 <body>
23
24 <div class="container">
25     <div class="box"> Box 1 </div>
26     <div class="box" style="flex-shrink: 1;"> Box 2 </div>
27     <div class="box"> Box 3 </div>
28 </div>
29
30 <div class="container">
31     <div class="box"> Box 1 </div>
32     <div class="box" style="flex-shrink: 2;"> Box 2 </div>
33     <div class="box"> Box 3 </div>
34 </div>
35
36 <div class="container">
37     <div class="box" style="flex-shrink: 3;"> Box 1 </div>
38     <div class="box" style="flex-shrink: 2;"> Box 2 </div>
39     <div class="box"> Box 3 </div>
40 </div>
41
42 </body>
43 </html>
```



Gambar: Property flex-shrink tidak berefek jika flex container cukup lebar

Jika flex container cukup lebar, maka nilai property `flex-shrink` tidak akan berefek. Kode di atas baru terlihat efeknya saat jendela web browser diperkecil:



Gambar: Hasil property flex-shrink saat flex container dibatasi

Ketika flex container tidak cukup menampung lebar normal dari setiap flex item, maka nilai **flex-shrink** akan dipakai untuk menentukan skala pengecilan.

Pada flex container pertama, Box 2 memiliki property **flex-shrink**: 1. Saat lebar flex container diperkecil, seluruh box juga ikut mengecil dengan proporsi yang sama. Ini karena **flex-shrink: 1** merupakan nilai default dari setiap flex item. Jika kita tidak ingin flex item ikut mengecil, bisa set sebagai **flex-shrink: 0**.

Di flex container kedua, nilai **flex-shrink** pada Box 2 saya naikkan menjadi 2. Efeknya, Box 2 diperkecil dua kali lebih cepat daripada element lain.

Dan pada flex container ketiga, proporsi skala pengecilan Box 1, Box 2 dan Box 3 adalah 3:2:1, sehingga box 1 akan mengecil paling cepat dibandingkan box lainnya.

Efek property **flex-shrink** juga bisa dipakai untuk mengatur tinggi flex item, dengan syarat main axis flex container di ubah ke sumbu y.

18.11. Property **flex**

Property **flex** merupakan penulisan singkat (*shorthand*) dari **flex-basis**, **flex-grow** dan **flex-shrink**. Property ini bisa diisi dengan 1, 2 atau 3 nilai.

Jika diisi dengan satu nilai, itu akan menjadi nilai untuk **flex-basis** atau **flex-grow** tergantung

apakah ada nilai satuan setelah angka.

Penulisan berikut akan men-set nilai untuk **flex-basis**:

- **flex: 100px;**
- **flex: 10em;**
- **flex: 30%;**

Sedangkan jika ditulis angka saja tanpa satuan, akan men-set nilai untuk **flex-grow**:

- **flex: 2;**
- **flex: 4;**

Jika property **flex** ditulis dengan 2 nilai, nilai pertama akan dipakai untuk **flex-grow**, sedangkan nilai kedua akan dipakai untuk **flex-basis** atau **flex-shrink** tergantung ada tidaknya satuan.

Penulisan berikut akan men-set nilai untuk **flex-grow** dan **flex-basis**:

- **flex: 2 100px;**
- **flex: 3 10em;**

Sedangkan penulisan berikut akan men-set nilai untuk **flex-grow** dan **flex-shrink**:

- **flex: 2 1;**
- **flex: 3 2;**

Terakhir, jika property **flex** ditulis dengan 3 nilai, urutannya adalah **flex-grow**, **flex-shrink**, dan **flex-basis**.

Property berikut:

flex: 3 2 100px;

Sama artinya dengan:

```
flex-grow: 3;
flex-shrink: 2;
flex-basis: 100px;
```

Nilai default dari property **flex** adalah **flex: 0 1 auto**, sehingga jika salah satu tidak diisi, nilai defaultnya yang akan dipakai.

Mari kita lihat contoh praktik dari property **flex** ini:

22.flex.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
```

CSS3 FlexBox

```
6  <style>
7      .container {
8          height: 150px;
9          border: 2px solid #965e15;
10         background-color: #AAE3A7;
11         display: flex;
12         margin-top: 1em;
13     }
14     .box {
15         width: 300px;
16         background-color: honeydew;
17         margin: 0.3em;
18         padding: 0.3em;
19     }
20 </style>
21 </head>
22 <body>
23
24     <div class="container">
25         <div class="box" style="flex: none;"> Box 1 </div>
26         <div class="box" style="flex: 0 1 auto ;"> Box 2 </div>
27         <div class="box" style="flex: 2 2;"> Box 3 </div>
28         <div class="box" style="flex: 0 100px ;"> Box 4 </div>
29         <div class="box" style="flex: 1 4 200px ;"> Box 4 </div>
30     </div>
31
32 </body>
33 </html>
```



Gambar: Hasil penggunaan property flex

Dalam kode ini saya menggunakan berbagai penulisan property flex. Efek yang ada akan lebih jelas jika dijalankan secara langsung lalu besar dan kecil lebar jendela web browser.

18.12. Property align-self

Property **align-self** berfungsi untuk mengatur posisi flex item secara individu. Property ini akan menimpa pengaturan **align-items** milik flex container. Nilai yang bisa diisi juga sama seperti **align-items**, yakni:

- stretch
- flex-start
- flex-end
- center
- baseline

Berikut contoh penggunaannya:

23.align-self-row.html

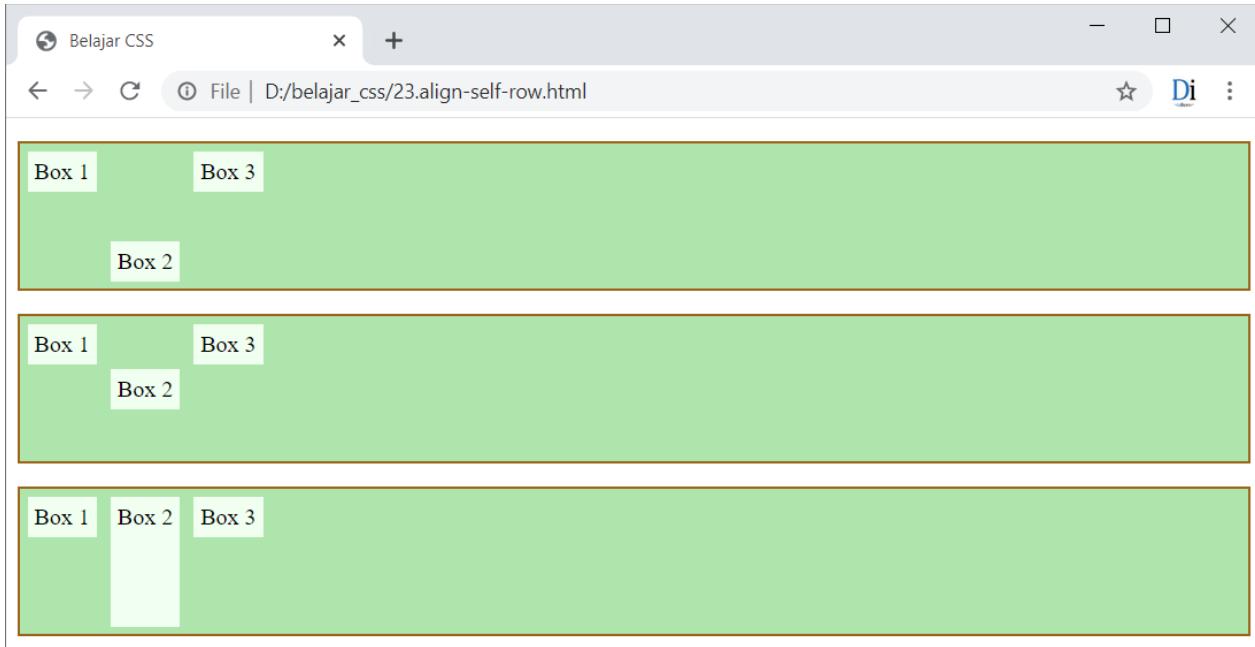
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             margin-top: 1em;
12             display: flex;
13             flex-direction: row;
14             align-items: flex-start;
15         }
16         .box {
17             background-color: honeydew;
18             margin: 0.3em;
19             padding: 0.3em;
20         }
21     </style>
22 </head>
23 <body>
24
25     <div class="container">
26         <div class="box"> Box 1 </div>
27         <div class="box" style="align-self: flex-end;"> Box 2 </div>
28         <div class="box"> Box 3 </div>
29     </div>
30
31     <div class="container">
32         <div class="box"> Box 1 </div>
33         <div class="box" style="align-self: center;"> Box 2 </div>
34         <div class="box"> Box 3 </div>
35     </div>
36

```

CSS3 FlexBox

```
37 <div class="container">
38   <div class="box"> Box 1 </div>
39   <div class="box" style="align-self:stretch;"> Box 2 </div>
40   <div class="box"> Box 3 </div>
41 </div>
42
43 </body>
44 </html>
```



Gambar: Hasil penggunaan property align-self

Di dalam class .container, terdapat property align-items: flex-start yang akan membuat semua flex item berkumpul di bagian awal.

Akan tetapi Box 2 saya set dengan property align-self:flex-end, align-self: center dan align-self:stretch. Ini membuat box tersebut memiliki posisi tersendiri.

Secara teknis, property align-self beroperasi pada **cross axis**. Jika kita mengubah main axis flex container menjadi searah sumbu y, maka pengaturan align-self akan berpengaruh ke sumbu x:

24.align-self-row-column.html

```
1 <style>
2   .container {
3     height: 120px;
4     ...
5     flex-direction: column;
6     align-items: flex-start;
7   }
8   ...
9 </style>
10
```

```

11 ...
12 <div class="container">
13   <div class="box"> Box 1 </div>
14   <div class="box" style="align-self: flex-end;"> Box 2 </div>
15   <div class="box"> Box 3 </div>
16 </div>
17
18 <div class="container">
19   <div class="box"> Box 1 </div>
20   <div class="box" style="align-self: center;"> Box 2 </div>
21   <div class="box"> Box 3 </div>
22 </div>
23
24 <div class="container">
25   <div class="box"> Box 1 </div>
26   <div class="box" style="align-self:stretch;"> Box 2 </div>
27   <div class="box"> Box 3 </div>
28 </div>

```



Gambar: Hasil penggunaan property align-self

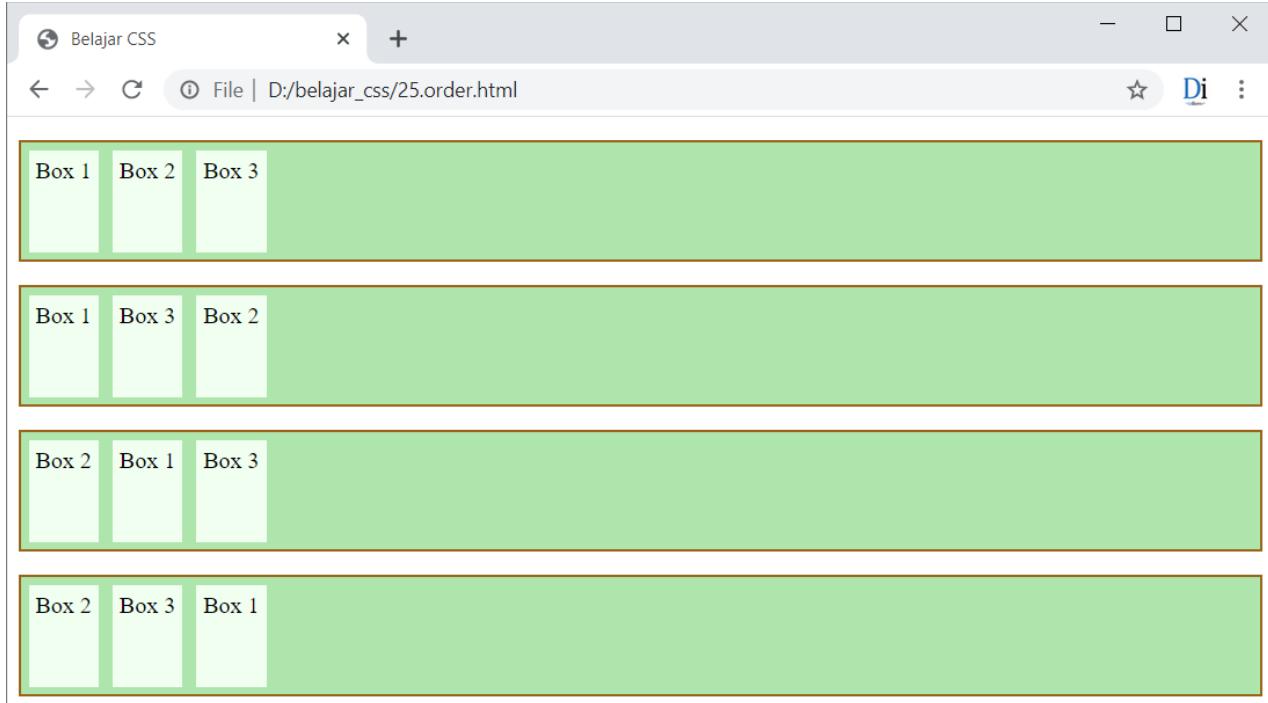
18.13. Property order

Property flexbox terakhir yang akan kita bahas adalah **order**. Property **order** dipakai untuk mengatur urutan flex item relatif ke flex item lainnya. Nilai yang bisa diisi berupa angka tanpa satuan dan bisa bernilai negatif seperti `order: 1`, `order: 2`, `order: -1` dst. Nilai paling kecil akan berada di bagian awal flex container.

Berikut contoh penggunaan property **order**:

25.order.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 80px;
9              border: 2px solid #965e15;
10             background-color: #AAE3A7;
11             margin-top: 1em;
12             display: flex;
13         }
14         .box {
15             background-color: honeydew;
16             margin: 0.3em;
17             padding: 0.3em;
18         }
19     </style>
20 </head>
21 <body>
22
23     <div class="container">
24         <div class="box"> Box 1 </div>
25         <div class="box" style="order: 0;"> Box 2 </div>
26         <div class="box"> Box 3 </div>
27     </div>
28
29     <div class="container">
30         <div class="box"> Box 1 </div>
31         <div class="box" style="order: 1;"> Box 2 </div>
32         <div class="box"> Box 3 </div>
33     </div>
34
35     <div class="container">
36         <div class="box"> Box 1 </div>
37         <div class="box" style="order: -1;"> Box 2 </div>
38         <div class="box"> Box 3 </div>
39     </div>
40
41     <div class="container">
42         <div class="box" style="order: 3;"> Box 1 </div>
43         <div class="box" style="order: 1;"> Box 2 </div>
44         <div class="box" style="order: 2;"> Box 3 </div>
45     </div>
46
47 </body>
48 </html>
```



Gambar: Penggunaan property order untuk mengurutkan flex item

Di dalam flex container pertama, saya menambah `order: 0` ke dalam box 2. Hasilnya tidak ada perubahan apapun karena 0 merupakan angka default dari setiap flex item.

Untuk flex container kedua, nilai property order Box 2 saya ubah menjadi 1. Maka Box 2 ini akan langsung pindah ke bagian paling akhir (kanan). Ini karena meskipun Box 1 dan Box 3 tidak memiliki property order, tapi secara default dianggap order: 0 sehingga lebih kecil daripada order: 1.

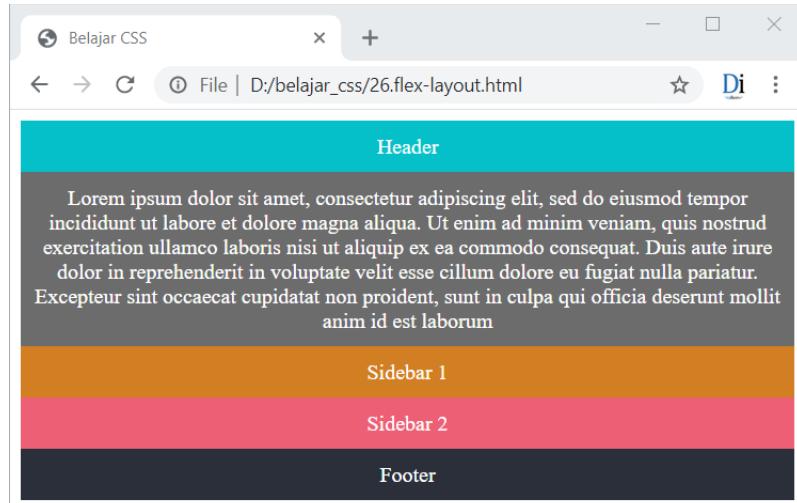
Pada flex container ketiga, nilai property order dari Box 2 saya tukar lagi menjadi -1. Efeknya, Box 2 akan bergeser ke posisi paling awal (kiri) karena lebih kecil dari 0.

Untuk flex container terakhir, saya memberikan nilai yang berbeda-beda untuk setiap box. Prinsip dasarnya, box dengan nilai order paling kecil akan berada di sisi awal terlebih dahulu, baru kemudian diikuti dengan nilai-nilai lain secara bertahap.

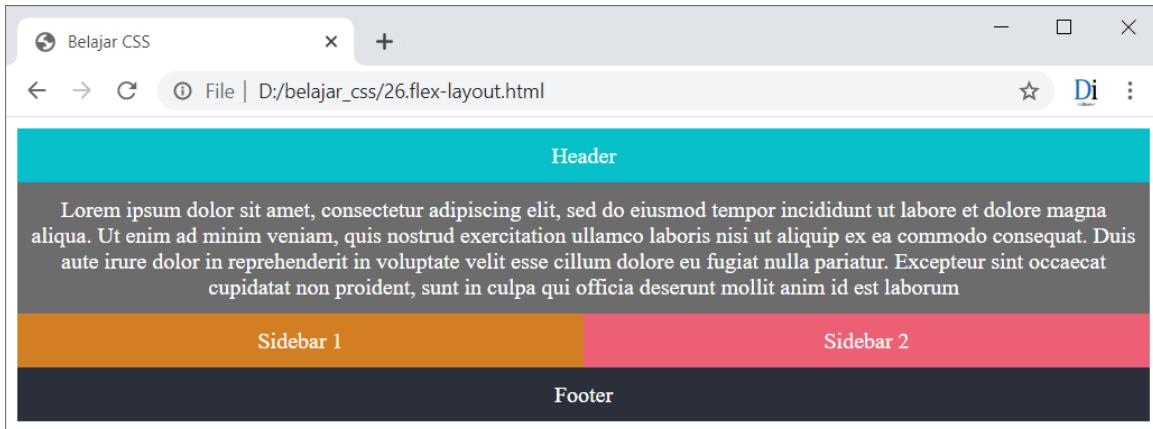
Property order ini sangat bermanfaat untuk membuat responsive layout, karena kita bisa dengan mudah mengatur urutan element pada media query tertentu.

18.14. Membuat Layout Responsive Sederhana

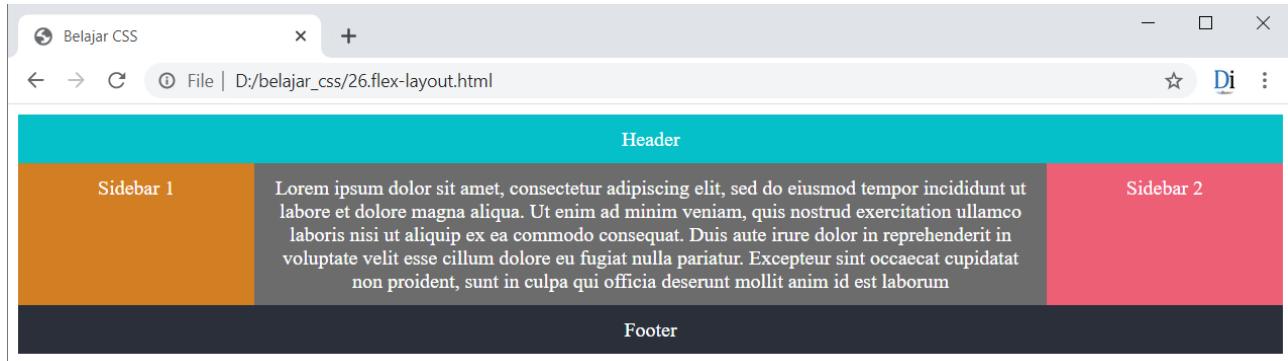
Setelah mempelajari semua property flexbox, saatnya menerapkan konsep tersebut untuk membuat layout responsive sederhana. Berikut tampilan yang akan kita buat:



Gambar: Tampilan layout untuk versi smartphone (lebar kurang dari 600px)



Gambar: Tampilan layout untuk versi tablet (lebar kurang dari 900px)



Gambar: Tampilan layout untuk versi desktop (lebar lebih dari 901px)

Layout ini terdiri dari 5 komponen: **header**, **main content**, **sidebar 1**, **sidebar 2** dan **footer**. Untuk tampilan smartphone, saya ingin semua komponen berjejer dari atas ke bawah. Lalu untuk tampilan tablet, sidebar akan saling bersebelahan dalam satu baris. Terakhir pada tampilan desktop, main content berada di antara kedua sidebar.

Prinsip dasar pembuatan layout ini seperti tetap butuh media query, hanya saja sekarang kita akan pakai konsep flexbox pada setiap breakpoint.

Sebagai sarana latihan, silahkan coba rancang sebentar bentuk layout seperti di atas. Tidak masalah jika anda butuh waktu yang cukup lama karena ini sekaligus menguji pemahaman tentang materi flexbox.

Baik, untuk membuat layout di atas saya akan mulai dari perancangan struktur HTML terlebih dahulu:

```

1 <div class="container">
2   <header class="header">Header</header>
3   <main class="main">
4     Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
5     do eiusmod tempor incididunt ut labore et dolore magna aliqua.
6     Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
7     nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
8     reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
9     pariatur. Excepteur sint occaecat cupidatat non proident,
10    sunt in culpa qui officia deserunt mollit anim id est laborum
11  </main>
12  <aside class="sidebar-1">Sidebar 1</aside>
13  <aside class="sidebar-2">Sidebar 2</aside>
14  <footer class="footer">Footer</footer>
15 </div>
```

Seperti biasa, pada bagian luar terdapat `<div class="container">` yang bertindak sebagai flex container. Semua komponen halaman akan berada di dalam tag ini.

Saya juga menggunakan semantic tag HTML5, yakni tag `<header>` untuk bagian header, tag `<main>` untuk main content, 2 buah tag `<aside>` untuk sidebar, serta tag `<footer>` untuk bagian footer. Setiap tag memiliki atribut `class` yang akan di pakai sebagai selector CSS.

Layout Untuk Layar Smartphone

Di akhir bab CSS3 Media Query kita membuat layout dengan konsep *desktop first*, dimana perancangan layout dimulai untuk versi desktop, baru kemudian membuat layout versi tablet dan versi smartphone.

Pada praktik kali ini, saya ingin memakai konsep **mobile first**. Kita akan rancang tampilan default untuk versi smartphone, baru kemudian masuk ke layout versi tablet dan desktop menggunakan media query.

Untuk versi smartphone, saya ingin setiap element berjejer membentuk kolom dari atas ke bawah (berurutan pada sumbu y). Berikut kode CSS yang diperlukan:

```

1 .container {
2   color: white;
3   text-align: center;
4   display: flex;
5   flex-flow: row wrap;
6 }
7
```

```

8 .container > * {
9   padding: 10px;
10  flex-basis: 100%;
11 }
12
13 .header    { background-color: #04bcc5; }
14 .main      { background-color: #666666; }
15 .sidebar-1 { background-color: #cf781e; }
16 .sidebar-2 { background-color: #eb59ee; }
17 .footer    { background-color: #252a34; }

```

Setelah menulis property `display: flex` di baris 5, saya juga menambah property `flex-flow: row wrap` ke dalam selector `.container`.

Sekilas ini tampak aneh karena jika kita ingin membuat tampilan layout berjejer dari atas ke bawah, seharusnya nilai yang dipakai adalah `flex-flow: column nowrap` atau bisa juga `flex-direction: column`.

Namun dalam kasus ini saya merasa lebih mudah jika main axis `flex container` tetap berada pada sumbu x. Untuk membuat tampilan berjejer dari atas ke bawah, semua flex item akan di set dengan `flex-basis: 100%`. Property ini membuat flex item melebar secara horizontal, dan karena flex container juga di set sebagai `wrap`, maka akan terus membuat baris baru.

Penambahan property `flex-basis: 100%` di lakukan pada selector `.container > *` di baris 10. Jika anda masih ingat, tanda "`>`" dipakai untuk membuat `child selector`. Di sini `child` yang di cari menggunakan tanda universal "`*`", maka ini akan cocok dengan semua element yang merupakan anak (`child`) dari class `.container`.

Property di baris 13 – 17 dipakai untuk membuat warna background dari setiap komponen halaman. Sebenarnya property `flex-basis: 100%` juga bisa kita tulis terpisah ke setiap selector ini. Saya menulisnya ke dalam selector `.container > *` agar lebih singkat saja, selain itu kita juga punya tempat untuk menulis property yang bisa berefek ke semua flex item.

Dengan tambahan kode ini, tampilan halaman menjadi sebagai berikut:



Gambar: Membuat layout dengan flexbox

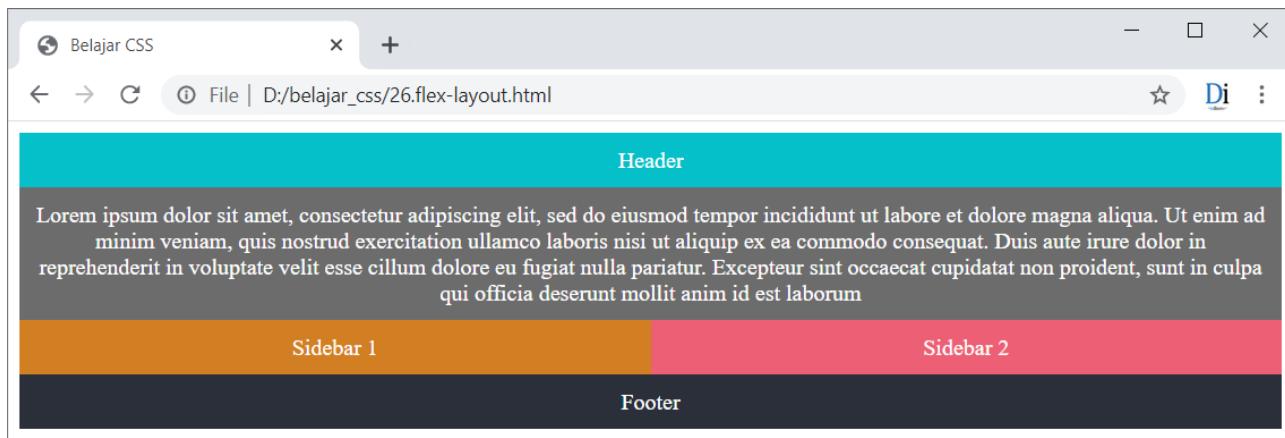
Layout Untuk Layar Tablet

Sekarang kita akan buat tampilan untuk versi tablet. Di sini saya ingin kedua sidebar saling berbagi ruang. Ini bisa didapat dengan mengubah property `flex-basis` milik kedua sidebar menjadi `auto`, lalu tambah property `flex-grow: 1`. Berikut kode yang diperlukan:

```
1 @media all and (min-width: 600px) {
2     .sidebar-1 {flex-basis:auto; flex-grow: 1;}
3     .sidebar-2 {flex-basis:auto; flex-grow: 1;}
4 }
```

Media query `@media all and (min-width: 600px)` akan efektif jika lebar jendela minimal 600px.

Sebelumnya, semua flex item di set dengan `flex-basis: 100%`. Jika nilai `flex-basis` kita tukar menjadi `auto`, maka lebar element akan bergantung kepada property `width` atau jika tidak ada, bergantung ke lebar konten yang ada di dalamnya. Ini membuat kedua sidebar mengecil dan berderet dalam satu baris. Kemudian dengan tambahan `flex-grow: 1`, kedua sidebar akan melebar memenuhi panjang flex container.



Gambar: Membuat layout dengan flexbox

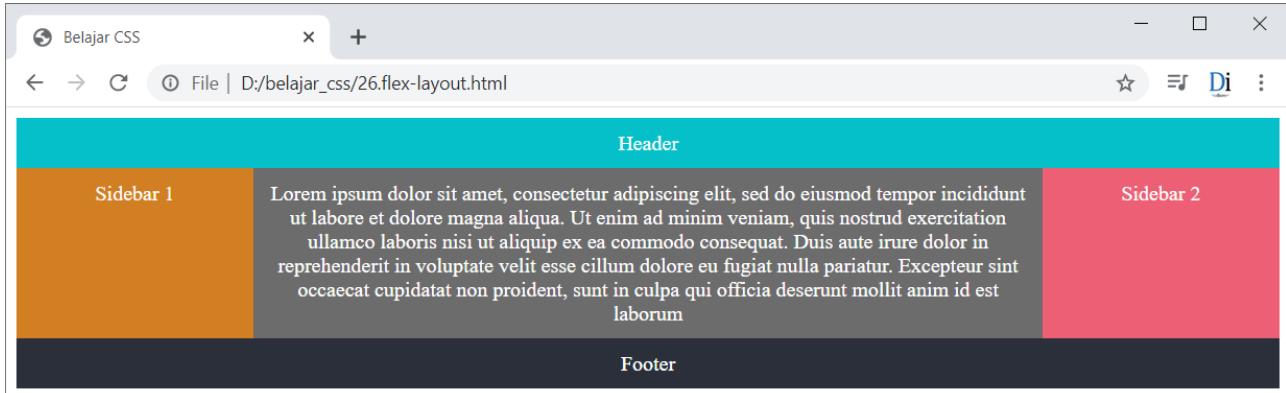
Layout Untuk Layar Desktop

Terakhir, masuk ke pembuatan layout untuk layar desktop. Kali ini saya ingin main content berada di antara kedua sidebar. Untuk membuatnya, kita perlu bantuan property `order` supaya posisi `.sidebar-1` bisa berada sebelum `.main`.

Selain itu agar sidebar dan main content ada di satu baris, property `flex-basis` milik `.main` juga perlu di set menjadi `50%`. Berikut kode yang diperlukan:

```
1 @media all and (min-width: 900px) {
2     .header      { order: 1; }
3     .sidebar-1  { order: 2; }
4     .main        { order: 3; flex-basis:50%; flex-grow: 1; }
5     .sidebar-2  { order: 4; }
6     .footer      { order: 5; }
```

7 }



Gambar: Membuat layout dengan flexbox

Property `order` perlu ditulis untuk semua element karena jika terdapat satu saja element yang memiliki `order` lebih dari 0, maka akan mempengaruhi urutan semua item. Alternatif lain, bisa menggunakan nilai negatif sebagai berikut:

```

1 @media all and (min-width: 900px) {
2   .header    { order: -2; }
3   .sidebar-1 { order: -1; }
4   .main      { flex-basis:50%; flex-grow: 1; }
5 }
```

Karena secara default semua element memiliki nilai `order: 0`, maka trik dengan nilai negatif ini bisa dipakai untuk menggeser beberapa element saja.

Berikut kode lengkap dari proses pembuatan layout dengan flexbox:

26.flex-layout.html

```

1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .container {
8       color: white;
9       text-align: center;
10      display: flex;
11      flex-flow: row wrap;
12    }
13
14    .container > * {
15      padding: 10px;
16      flex-basis: 100%;
17    }
18
19    .header    { background-color: #04bcc5; }
20    .main      { background-color: #666666; }
```

```

21     .sidebar-1 { background-color: #cf781e; }
22     .sidebar-2 { background-color: #eb596e; }
23     .footer      { background-color: #252a34; }
24
25     @media all and (min-width: 600px) {
26         .sidebar-1 {flex-basis:auto; flex-grow: 1;}
27         .sidebar-2 {flex-basis:auto; flex-grow: 1;}
28     }
29
30     @media all and (min-width: 900px) {
31         .header      { order: 1;}
32         .sidebar-1 { order: 2;}
33         .main        { order: 3; flex-basis:50%; flex-grow: 1;}
34         .sidebar-2 { order: 4;}
35         .footer      { order: 5;}
36     }
37
38
39     </style>
40 </head>
41 <body>
42
43     <div class="container">
44         <header class="header">Header</header>
45         <article class="main">
46             Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
47             do eiusmod tempor incididunt ut labore et dolore magna aliqua.
48             Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
49             nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
50             reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
51             pariatur. Excepteur sint occaecat cupidatat non proident,
52             sunt in culpa qui officia deserunt mollit anim id est laborum
53         </article>
54         <aside class="sidebar-1">Sidebar 1</aside>
55         <aside class="sidebar-2">Sidebar 2</aside>
56         <footer class="footer">Footer</footer>
57     </div>
58
59 </body>
60 </html>

```

Silahkan berkreasi lebih lanjut dengan membuat berbagai bentuk layout lain. Menggunakan flexbox dan media query, layout responsive bisa dibuat dengan lebih mudah daripada menggunakan float layout.

Dalam bab ini kita telah membahas cukup detail tentang CSS3 Flexbox, berikutnya akan masuk ke materi yang tidak kalah menarik, yakni tentang **CSS3 Grid**.

19. CSS3 Grid

CSS3 Grid di rilis hampir bersamaan dengan CSS3 flexbox. Sama seperti flexbox, CSS grid juga ditujukan untuk perancangan layout.

Sedikit berbeda dengan flexbox, CSS grid bisa mengatur posisi element pada 2 dimensi sekaligus, yakni sumbu x (horizontal), serta sumbu y (vertikal). Ini membuatnya sangat pas untuk perancangan layout yang lebih kompleks. Namun dalam banyak situasi, grid sering dikombinasikan dengan flexbox untuk membuat layout CSS modern.

Kata "grid" merupakan istilah umum dalam dunia desain. Biasanya grid merujuk ke garis bantu yang dipakai saat pembuatan sketsa. Di dalam dunia CSS sendiri, istilah "grid" juga sudah cukup populer sebelum dirilisnya CSS3 Grid.

Sekitar tahun 2008, terdapat [960 grid system](#)²⁸ yang kala itu banyak dipakai sebagai alat bantu perancangan layout web. Angka 960 merujuk ke lebar layar 960 pixel yang bisa dipecah ke dalam 12 atau 16 kolom. Berbagai framework CSS juga memiliki sistem grid sendiri, seperti di Bootstrap, Materialize atau Bulma.

Maka jika kita mendengar istilah "grid", bisa cek terlebih dahulu apakah yang dimaksud itu "CSS3 grid", "960 grid system", atau "grid system" kepunyaan framework CSS". Dalam buku ini, istilah grid merujuk ke CSS3 Grid.

19.1. Grid Container

Agar bisa menggunakan CSS grid, suatu element harus berada di dalam "grid container". Ini dibuat dengan cara menambah property `display:grid` atau `display:inline-grid` ke dalam parent element. Berikut contoh penggunaannya:

01.grid-container.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              height: 150px;

```

²⁸ <https://960.gs>

```

9      border: 2px solid #965e15;
10     margin-top: 1em ;
11   }
12   .box {
13     background-color: aquamarine;
14     text-align: center;
15     padding: 0.3em;
16   }
17   .box:nth-child(even) {
18     background-color: honeydew;
19   }
20 </style>
21 </head>
22 <body>
23
24   <div class="container">
25     <div class="box"> Box A </div>
26     <div class="box"> Box B </div>
27     <div class="box"> Box C </div>
28   </div>
29
30   <div class="container" style="display: grid;">
31     <div class="box"> Box A </div>
32     <div class="box"> Box B </div>
33     <div class="box"> Box C </div>
34   </div>
35
36   <div class="container" style="display: inline-grid;">
37     <div class="box"> Box A </div>
38     <div class="box"> Box B </div>
39     <div class="box"> Box C </div>
40   </div>
41
42 </body>
43 </html>

```

Di dalam tag `<style>`, saya menulis 3 buah selector: `.container`, `.box`, dan `.box:nth-child(even)`.

Selector pertama, `.container` ditujukan untuk men-style grid container, diantaranya mengatur tinggi element dengan `height: 150px`, membuat efek `border: 2px solid #965e15`, serta memberikan `margin-top: 1em`.

Selector kedua, `.box` dipakai untuk *grid item*, yakni element yang nantinya berada di dalam flex container. Saya memberikan warna `background-color: aquamarine`, mengatur teks agar berada di tengah dengan `text-align: center`, serta menambah `padding: 0.3em` agar teks tidak terlalu menempel ke sisi tepi box.

Selector ketiga memanfaatkan *pseudo class selectors*, yakni `.box:nth-child(even)`. Selector ini akan mencari semua box dengan urutan genap, lalu mengubah warna `background-color: honeydew`. Ini saya lakukan agar warna box bisa berselang seling untuk mempermudah kita

dalam melihat urutan box.

Masuk ke kode HTML, saya membuat 3 buah container yang masing-masingnya berisi 3 box. Container pertama tidak memiliki property tambahan (baris 24), container kedua dengan property `display: grid` (baris 30), serta container ketiga dengan property `display: inline-grid` (baris 36).

Berikut hasil dari kode diatas:



Gambar: Hasil penggunaan `display: grid` dan `display: inline-grid`

Efek yang terlihat ketika ditambah property `display: grid` atau `display: inline-grid` adalah: tinggi setiap box di-stretch untuk memenuhi grid container. Ini hanya bisa terjadi jika tinggi grid container di set dari property `height`, namun tidak untuk grid item (tinggi grid item tidak di set).

Perbedaan antara `display: grid` serta `display: inline-grid` mirip seperti beda *block level element* dan *inline level element*, dimana `display: grid` akan membuat container berada di satu baris baru dan secara default melebar memenuhi lebar parent element. Sedangkan `display: inline-grid` akan menempatkan container di baris yang sudah ada serta secara default mengecil sesuai ukuran child element.

Jenis-jenis Grid Property

Setelah selesai membuat *grid container*, langkah berikutnya adalah menambah beberapa property grid lain untuk mengatur tampilan element. Sama seperti flexbox, property untuk

grid bisa dipecah menjadi 2 kelompok: property untuk **grid container**, dan property untuk **grid item**.

Pengertian *grid container* sudah kita bahas sebelumnya, yakni element yang menampung semua grid item. Property yang ditulis ke grid container akan berpengaruh secara global.

Sedangkan *grid item* adalah sebutan dari setiap *child element* yang berada di dalam *grid container*. Dalam contoh sebelumnya, Box A, Box B dan Box C adalah flex item.

Berikut daftar property untuk **grid container**:

- `grid-template-columns`
- `grid-template-rows`
- `grid-template-areas`
- `grid-column-gap / column-gap`
- `grid-row-gap / row-gap`
- `grid-gap / gap (shorthand)`
- `justify-items`
- `align-items`
- `place-items (shorthand)`

Dan berikut property untuk **grid item**:

- `grid-column-start`
- `grid-column-end`
- `grid-row-start`
- `grid-row-end`
- `grid-column (shorthand)`
- `grid-row (shorthand)`
- `grid-area`
- `justify-self`
- `align-self`
- `place-self (shorthand)`

Semua property ini akan kita bahas secara bertahap.

19.2. Property `grid-template-columns`

Property **`grid-template-columns`** dipakai untuk menentukan jumlah dan lebar kolom grid. Setelah property ini ditulis, semua grid item akan "patuh" dan berderet sesuai jumlah kolom yang didefinisikan. Jika jumlah grid item melebihi jumlah kolom, maka akan pindah membentuk baris baru (mirip seperti efek *wrap* pada flexbox).

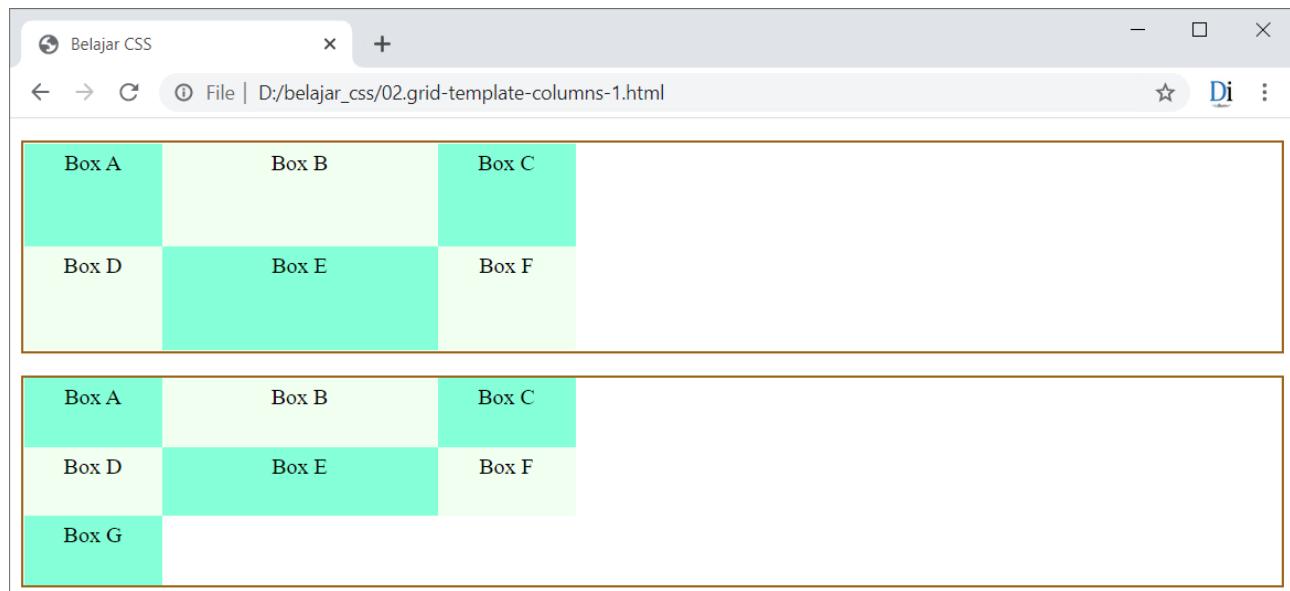
Berikut contoh penggunaannya:

02.grid-template-columns-1.html

```

1 <style>
2   .container {
3     ...
4     display: grid;
5   }
6   ...
7 </style>
8
9 <div class="container" style="grid-template-columns: 100px 200px 100px;">
10  <div class="box"> Box A </div>
11  <div class="box"> Box B </div>
12  <div class="box"> Box C </div>
13  <div class="box"> Box D </div>
14  <div class="box"> Box E </div>
15  <div class="box"> Box F </div>
16 </div>
17
18 <div class="container" style="grid-template-columns: 100px 200px 100px;">
19  <div class="box"> Box A </div>
20  <div class="box"> Box B </div>
21  <div class="box"> Box C </div>
22  <div class="box"> Box D </div>
23  <div class="box"> Box E </div>
24  <div class="box"> Box F </div>
25  <div class="box"> Box G </div>
26 </div>

```



Gambar: Hasil penggunaan property grid-template-columns

Untuk menghemat tempat, mulai dari contoh ini sampai akhir bab nanti saya tidak menulis lengkap kode CSS dan HTML yang ada (kecuali diperlukan). Kita akan fokus ke property yang sedang dibahas saja. Kode lengkap tersedia di file `belajar_css.zip`.

Kali ini saya sudah memindahkan property `display: grid` ke dalam selector `.container` di baris 4. Dengan demikian, semua tag HTML yang memiliki atribut `class="container"` secara otomatis akan menjadi *grid container*.

Di baris 9 dan 18, terdapat tambahan property `grid-template-columns: 100px 200px 100px` ke dalam container. Ini artinya grid memiliki 3 buah kolom dengan lebar 100px untuk kolom pertama, 200px untuk kolom kedua, dan 100px untuk kolom ketiga.

Semua grid item akan mengikuti pola ini, tidak peduli berapa banyak jumlahnya. Container pertama memiliki 6 item, sedangkan container kedua memiliki 7 item. Item-item ini akan terus membuat baris baru jika sudah melebihi 3 kolom dalam 1 baris.

Perhatikan juga di sisi kanan grid container masih terdapat ruang kosong, namun karena setiap kolom di set dengan nilai yang tetap, yakni 100px, 200px, dan 100px, sisa ruang ini tidak dipakai.

Setiap penambahan angka ke dalam property `grid-template-columns` akan menambah kolom baru. Misalnya jika ditulis:

```
grid-template-columns: 100px 200px 100px 150px 300px;
```

Maka jumlah kolom grid menjadi 5 buah, dengan lebar 100px, 200px, 100px, 150px, dan 300px.

Kita juga bisa menulis semua nilai length untuk lebar kolom grid, termasuk %, em, rem, vw, dll. Misalnya property berikut:

```
grid-template-columns: 20% 6rem 15vw;
```

Akan membuat 3 buah kolom grid dengan lebar 20% untuk kolom pertama, 6rem untuk kolom kedua dan 15vw untuk kolom ketiga.

Nilai Length auto dan fr

Selain nilai length biasa, juga terdapat nilai khusus untuk lebar kolom grid, yakni **auto** dan **fr**.

Nilai kolom **auto** artinya kita mengizinkan kolom grid untuk membesar memenuhi sisa ruang yang ada. Misalnya jika ditulis:

```
grid-template-columns: 100px 200px 100px auto;
```

Maka akan terbentuk 4 kolom dengan lebar 100px untuk kolom pertama, 200px untuk kolom kedua, 100px untuk kolom ketiga, serta sisanya akan menjadi lebar untuk kolom keempat.

Yang paling menarik dan banyak dipakai untuk grid adalah **fr** (singkatan dari *fraction*). Nilai **fr** dipakai untuk membagi lebar grid container dengan skala tertentu. Misalnya jika ditulis property berikut:

```
grid-template-columns: 1fr 2fr 1fr;
```

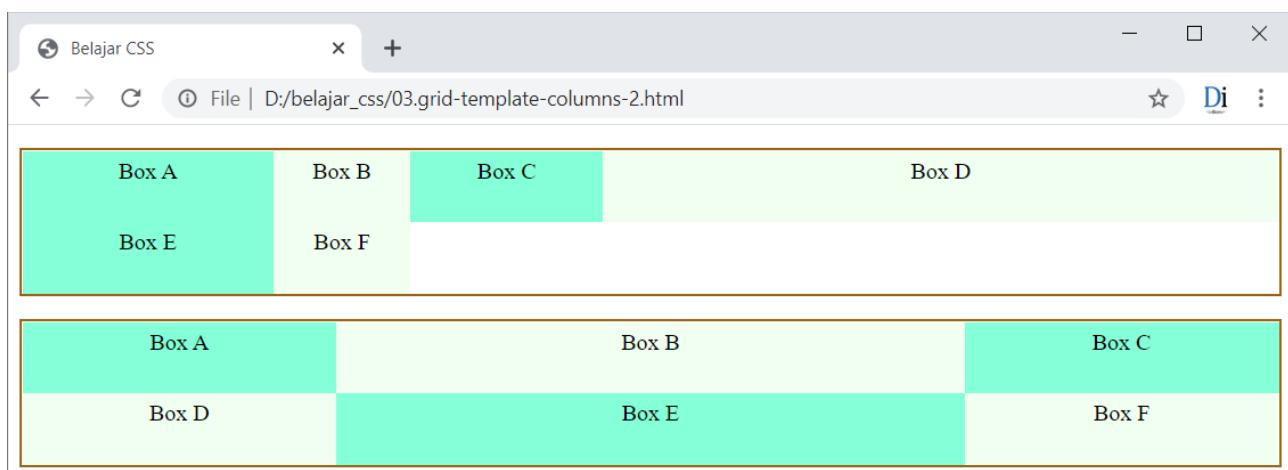
Maka grid container akan membagi 3 kolom dengan perbandingan lebar 1 : 2 : 1. Saat lebar container diperbesar atau diperkecil, skala perbandingan ini tetap dipertahankan.

Berikut contoh praktik dari nilai length auto dan fr:

03.grid-template-columns-2.html

```

1 <div class="container" style="grid-template-columns: 20% 6rem 15vw auto">
2   <div class="box"> Box A </div>
3   <div class="box"> Box B </div>
4   <div class="box"> Box C </div>
5   <div class="box"> Box D </div>
6   <div class="box"> Box E </div>
7   <div class="box"> Box F </div>
8 </div>
9
10 <div class="container" style="grid-template-columns: 1fr 2fr 1fr">
11   <div class="box"> Box A </div>
12   <div class="box"> Box B </div>
13   <div class="box"> Box C </div>
14   <div class="box"> Box D </div>
15   <div class="box"> Box E </div>
16   <div class="box"> Box F </div>
17 </div>
```



Gambar: Hasil penggunaan nilai auto dan fr untuk grid-template-columns

Container pertama memiliki 4 kolom dengan lebar 20% untuk kolom pertama, 6rem untuk kolom kedua, 15vw untuk kolom ketiga, dan auto untuk kolom ke empat.

Sedangkan container kedua memiliki 3 kolom dengan skala perbandingan 1 : 2 : 1. Ketika lebar web browser diperbesar atau diperkecil, ketiga kolom ikut menyesuaikan diri dengan skala yang ada. Efek ini sangat menarik dipakai untuk membuat web responsive.

Mengulang Nilai dengan Fungsi repeat()

Agar memudahkan penulisan kolom dengan nilai berulang, kita bisa menggunakan fungsi **repeat()** ke dalam `grid-template-columns`. Fungsi ini butuh 2 buah nilai input, yakni jumlah

perulangan dan lebar kolom yang akan diulang.

Sebagai contoh, penulisan property berikut:

```
grid-template-columns: repeat(5, 1fr)
```

Sama artinya dengan:

```
grid-template-columns: 1fr 1fr 1fr 1fr 1fr;
```

Yakni membuat 5 kolom dengan lebar setiap kolom sebesar 1fr. Kita juga bisa mengulang beberapa kolom sekaligus (tidak hanya satu kolom saja). Misalnya property berikut:

```
grid-template-columns: repeat(3, 200px 1fr)
```

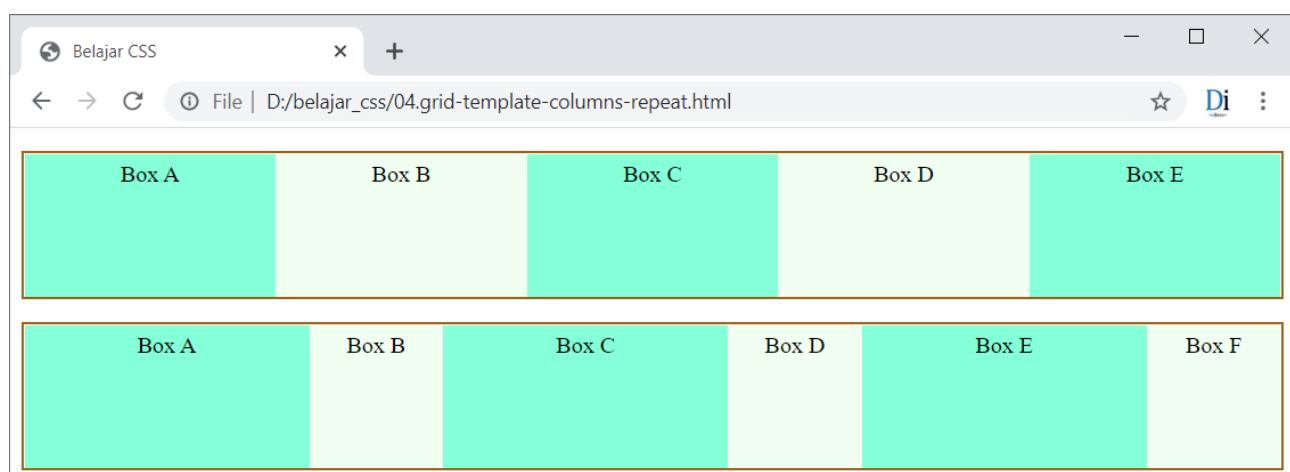
Sama artinya dengan:

```
grid-template-columns: 200px 1fr 200px 1fr 200px 1fr;
```

Berikut contoh praktik dari penggunaan fungsi repeat():

04.grid-template-columns-repeat.html

```
1 <div class="container" style="grid-template-columns: repeat(5, 1fr);">
2   <div class="box"> Box A </div>
3   <div class="box"> Box B </div>
4   <div class="box"> Box C </div>
5   <div class="box"> Box D </div>
6   <div class="box"> Box E </div>
7 </div>
8
9 <div class="container" style="grid-template-columns: repeat(3, 200px 1fr);">
10  <div class="box"> Box A </div>
11  <div class="box"> Box B </div>
12  <div class="box"> Box C </div>
13  <div class="box"> Box D </div>
14  <div class="box"> Box E </div>
15  <div class="box"> Box F </div>
16 </div>
```



Gambar: Hasil penggunaan fungsi repeat()

Mengatur Lebar Kolom dengan Fungsi minmax()

Untuk pengaturan lebar kolom yang lebih fleksibel lagi, CSS menyediakan fungsi **minmax()**.

Fungsi ini butuh dua buah nilai input yang berguna untuk menentukan lebar kolom minimum dan maksimum. Sebagai contoh, property berikut:

```
grid-template-columns: minmax(200px, 300px) minmax(200px, 300px) minmax(200px, 300px);
```

Akan membuat 3 buah kolom grid dimana masing-masing kolom memiliki lebar minimum 200px dan lebar maksimum 300px.

Jika grid container diperkecil, lebar kolom tidak bisa menjadi lebih kecil dari nilai minimum ini. Begitu juga jika lebar container diperbesar, lebar kolom tidak bisa lebih besar daripada nilai maksimum (meskipun masih ada ruang yang bisa dipakai).

Berikut contoh prakteknya:

05.grid-template-columns-minmax.html

```
1 <div class="container" style="grid-template-columns: minmax(200px, 300px)
2 minmax(200px, 300px) minmax(200px, 300px);">
3
4   <div class="box"> Box A </div>
5   <div class="box"> Box B </div>
6   <div class="box"> Box C </div>
7   <div class="box"> Box D </div>
8   <div class="box"> Box E </div>
9   <div class="box"> Box F </div>
10 </div>
```



Gambar: Hasil penggunaan fungsi minmax()

Ketika jendela web browser diperkecil (yang secara tidak langsung juga memperkecil grid container), akan terjadi overflow karena setiap kolom harus memiliki lebar 200px. Sebaliknya ketika jendela web browser diperbesar, lebar kolom hanya bisa membesar sampai 300px saja.

Membuat Jumlah Kolom Dinamis (auto-fill dan auto-fit)

Fungsi `repeat()` yang kita bahas sebelumnya memiliki keyword khusus untuk membuat jumlah kolom dinamis, yakni **auto-fill** dan **auto-fit**. Jika nilai pertama dari fungsi `repeat()` diisi dengan salah satu nilai ini, maka jumlah kolom ditentukan dari seberapa banyak yang bisa ditampung oleh grid container.

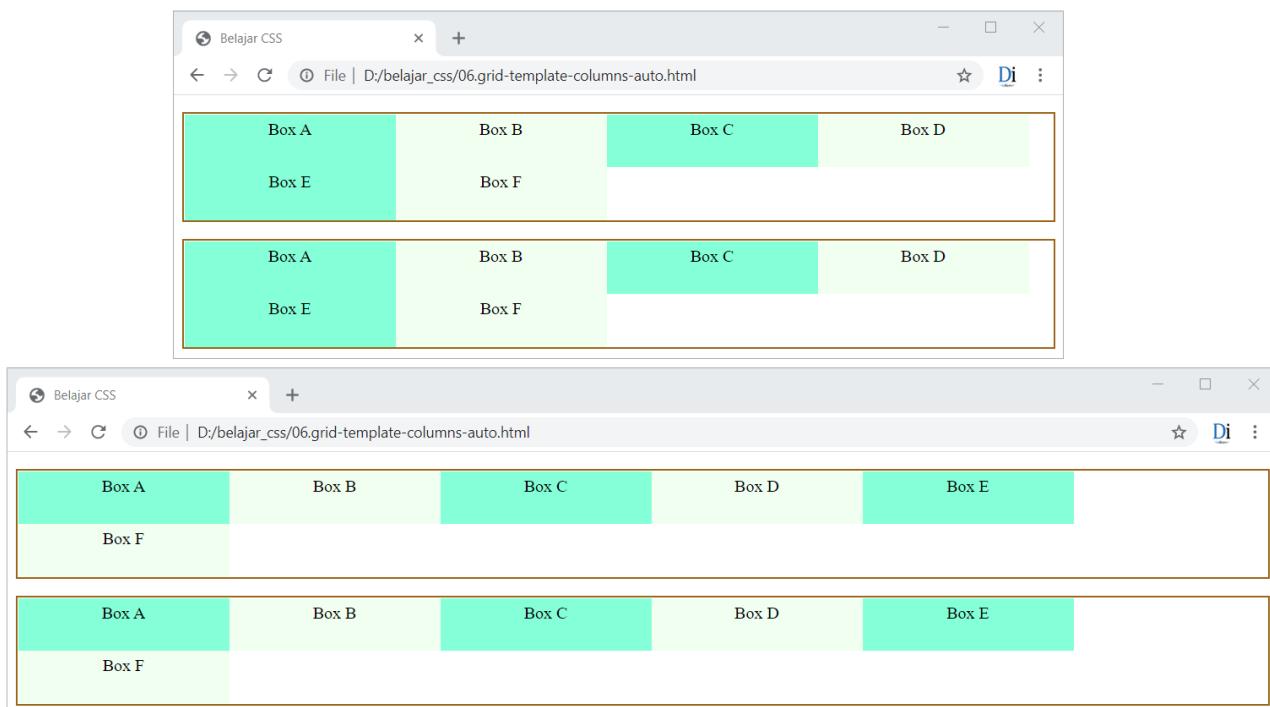
Sebagai contoh, jika ditulis salah satu property berikut:

```
grid-template-columns: repeat(auto-fill, 200px);
grid-template-columns: repeat(auto-fit, 200px);
```

Maka jumlah kolom grid akan bertambah atau berkurang setiap kelipatan 200px. Mari kita lihat contoh prakteknya:

06.grid-template-columns-auto.html

```
1 <div class="container" style="grid-template-columns: repeat(auto-fill,200px);">
2   <div class="box"> Box A </div>
3   ...
4   <div class="box"> Box F </div>
5 </div>
6
7 <div class="container" style="grid-template-columns: repeat(auto-fit,200px);">
8   <div class="box"> Box A </div>
9   ...
10  <div class="box"> Box F </div>
11 </div>
```



Gambar: Hasil penggunaan nilai auto-fill dan auto-fit untuk fungsi repeat()

Hasil dari kode di atas akan lebih jelas ketika dijalankan langsung dan ubah-ubah ukuran jendela web browser. Jumlah kolom akan bertambah dan berkurang selama ada ruang kosong sebesar 200px di sisi kanan grid container.

Perbedaan antara nilai `repeat(auto-fill)` dan `repeat(auto-fit)` baru akan terlihat jika lebar maksimum kolom di set dengan nilai relatif seperti % atau fr. Contoh prakteknya akan kita lihat sesaat lagi.

Dalam praktek di atas, efek `auto-fill` dan `auto-fit` akan sama karena lebar kolom grid di set fix pada 200px.

Agar lebih dinamis, kita bisa men-set lebar maksimum dan minimum kolom grid menggunakan fungsi `minmax()`. Dengan memberikan angka relatif seperti `1fr` sebagai nilai maksimum, lebar kolom akan menjadi responsive, yakni ikut membesar / mengecil sampai ada ruang untuk kolom baru. Efek yang dihasilkan menjadi sangat menarik. Berikut contoh yang dimaksud:

07.grid-template-columns-auto-fr.html

```

1 <div class="container"
2   style="grid-template-columns: repeat(auto-fill, minmax(200px,1fr));">
3     <div class="box"> Box A </div>
4     <div class="box"> Box B </div>
5     <div class="box"> Box C </div>
6     <div class="box"> Box D </div>
7     <div class="box"> Box E </div>
8   </div>
9
10 <div class="container"
11   style="grid-template-columns: repeat(auto-fit, minmax(200px,1fr));">
12   <div class="box"> Box A </div>
13   <div class="box"> Box B </div>
14   <div class="box"> Box C </div>
15   <div class="box"> Box D </div>
16   <div class="box"> Box E </div>
17 </div>
```

Property yang ditulis memang cukup panjang, yakni :

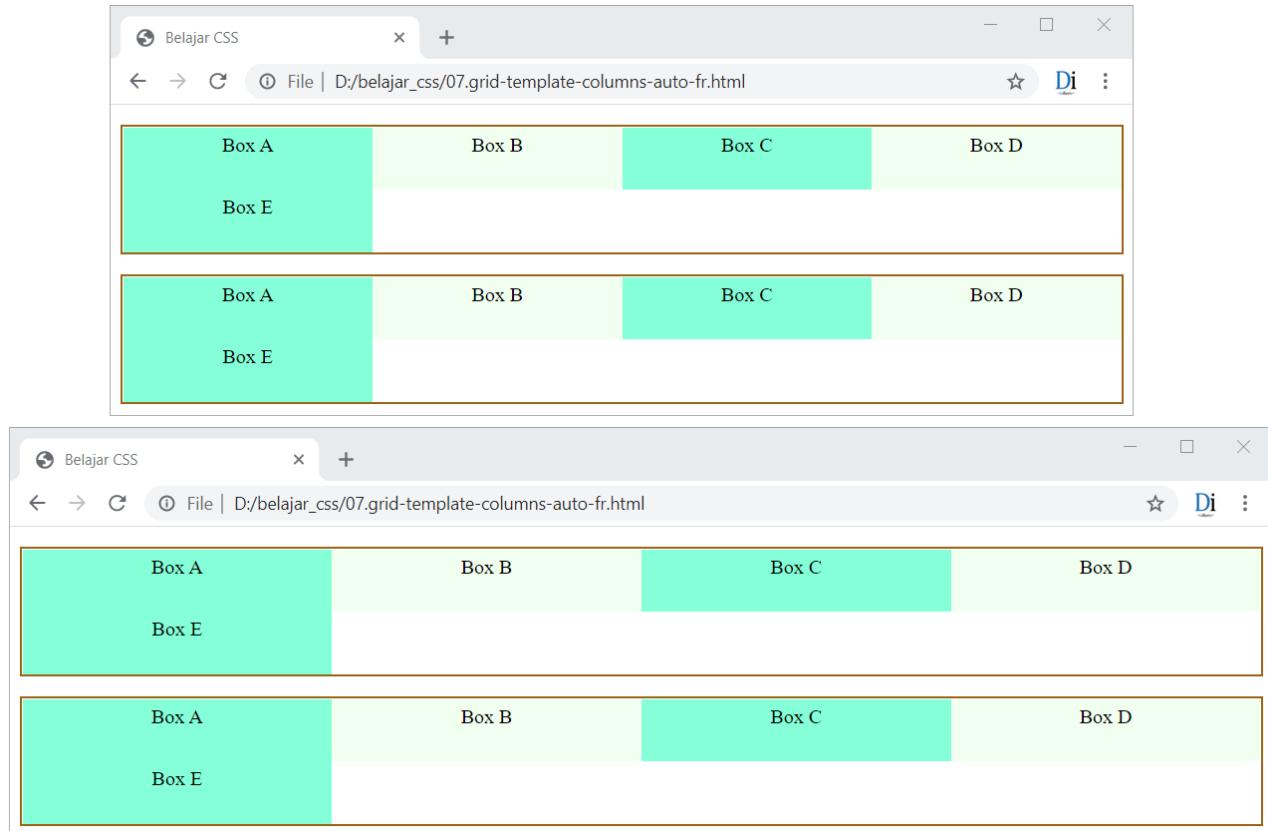
`grid-template-columns: repeat(auto-fill, minmax(200px,1fr))`

Kode di atas bisa dibaca: "Buat grid dengan jumlah kolom dinamis setiap 200px. Namun jika terdapat sisa ruang kurang dari 200px, lebarkan kolom untuk memenuhi grid container".

Beda dengan contoh kita sebelumnya adalah, jika lebar kolom fix di 200px saja, maka pada lebar tertentu akan terdapat ruang kosong di sisi kiri grid, dimana ruang itu tidak cukup 200px untuk membuat kolom baru.

Dengan menambahkan `minmax(200px,1fr)`, sisa ruang itu ditutupi dengan cara melebarkan setiap kolom. Berikut salah satu hasilnya:

CSS3 Grid



Gambar: Hasil penggunaan nilai minmax(200px,1fr)

Kedua jendela di atas menampilkan kode yang sama. Perhatikan sekarang tidak ada ruang kosong di sisi kanan grid container karena grid item bisa melebar untuk mengisi ruang tersebut. Syaratnya, lebar ruang kosong itu kurang dari 200px, karena jika lebih dari 200px akan diisi oleh kolom baru.

Sekarang kita bisa masuk ke perbedaan antara nilai `auto-fill` dan `auto-fit`. Masih dengan kode yang sama dengan praktik di atas, berikut tampilan saat jendela web browser diperlebar hingga semua grid item berada di baris yang sama:



Gambar: Beda antara auto-fit dan auto-fill

Container pertama di set dengan property `grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))`. Nilai **auto-fit** akan terus membentuk kolom baru meskipun tidak ada grid item yang bisa mengisi kolom tersebut. Inilah yang membuat ada spasi putih di sisi

kanan grid container.

Container kedua di set dengan property `grid-template-columns: repeat(auto-fit, minmax(200px, 1fr))`. Nilai **auto-fit** akan memaksa setiap grid item untuk terus melebar mengisi ruang kosong yang sebenarnya cukup untuk kolom baru, namun tidak ada grid item lain yang bisa mengisinya.

Perbedaan antara nilai `auto-fill` dan `auto-fit` memang hanya untuk kasus dimana terdapat ruang lebih seperti ini saja. Dalam banyak situasi, hasil dari kedua nilai akan sama persis.

19.3. Property `grid-template-rows`

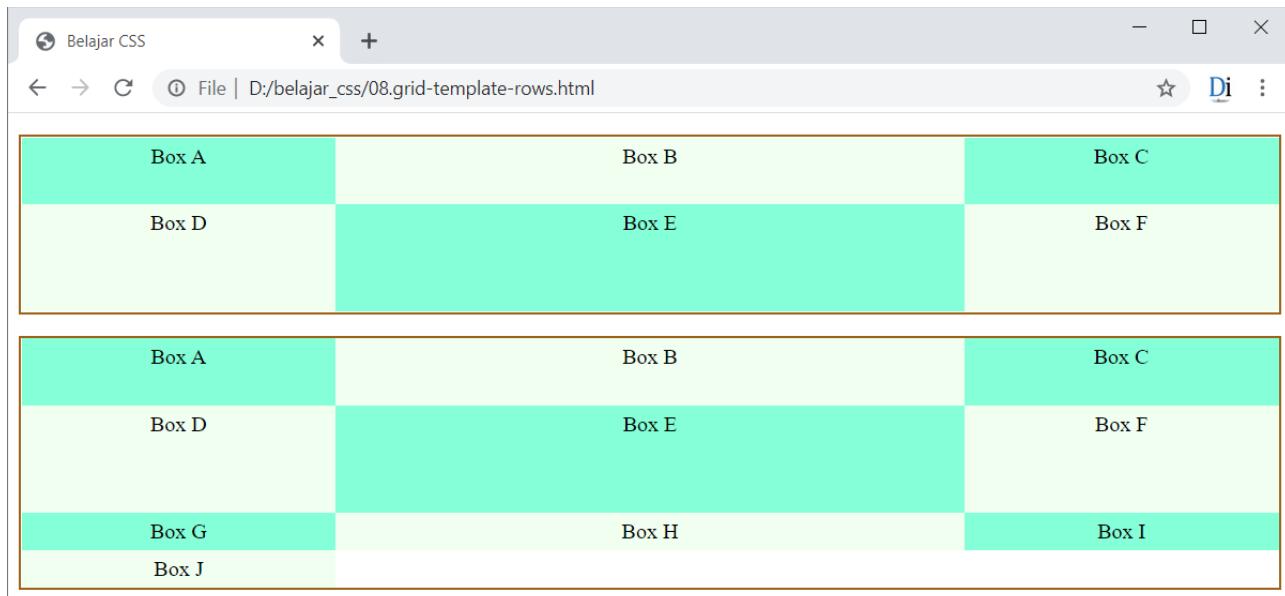
Property `grid-template-rows` dipakai untuk menentukan jumlah dan lebar baris grid. Cara penulisan nilai untuk property ini sama seperti `grid-template-columns`. Misalnya jika kita ingin membuat grid yang terdiri dari 3 baris dengan tinggi 100px untuk baris pertama, 200px untuk baris kedua dan 300px untuk baris ketiga, bisa menggunakan property berikut:

```
grid-template-rows: 100px 200px 300px;
```

Perpaduan antara `grid-template-columns` dengan `grid-template-rows` akan membuat sebuah sistem grid yang terdiri dari kolom dan baris. Berikut contoh penggunaannya:

08.grid-template-rows.html

```
1 <style>
2   .container {
3     border: 2px solid #965e15;
4     margin-top: 1em ;
5     display: grid;
6     grid-template-columns: 1fr 2fr 1fr;
7     grid-template-rows: 50px 80px;
8   }
9   ...
10 </style>
11
12 <div class="container">
13   <div class="box"> Box A </div>
14   ...
15   <div class="box"> Box F </div>
16 </div>
17
18 <div class="container">
19   <div class="box"> Box A </div>
20   ...
21   <div class="box"> Box J </div>
22 </div>
```



Gambar: Hasil penggunaan property grid-template-columns dengan grid-template-rows

Di dalam selector .container, saya menulis 2 property berikut:

```
grid-template-columns: 1fr 2fr 1fr;
grid-template-rows: 50px 80px;
```

Artinya, grid container akan memiliki 3 kolom (dengan lebar 1fr, 2fr dan 1fr) serta 2 baris (dengan tinggi 50px dan 80px). Struktur grid ini bisa menampung $3 \times 2 = 6$ item. Contoh ideal ada di container pertama, dimana baris kedua terlihat lebih tinggi karena di set dengan 80px.

Apabila kita terus menambah grid item lain, aturan 3 kolom tetap berlaku, namun tinggi baris hanya berefek untuk baris pertama dan kedua saja (sesuai dengan yang ditulis pada property `grid-template-rows`). Ini bisa terlihat pada container kedua yang terdiri dari 10 item. Tinggi baris ketiga dan keempat tidak dipengaruhi oleh `grid-template-rows`.

Dalam banyak situasi, property `grid-template-rows` tidak diperlukan karena umumnya kita ingin tinggi element bersifat fleksibel dan berubah sesuai konten yang ada. Terlebih jika jumlah grid item sudah melebihi total kolom, secara otomatis akan pindah ke bawah membuat garis baru.

19.4. Property row-gap, column-gap dan gap

Property **row-gap**, **column-gap** dan **gap** dipakai untuk membuat spasi di antar baris dan kolom grid. Nilai yang bisa diisi berupa satuan length seperti px, em, rem, dll.

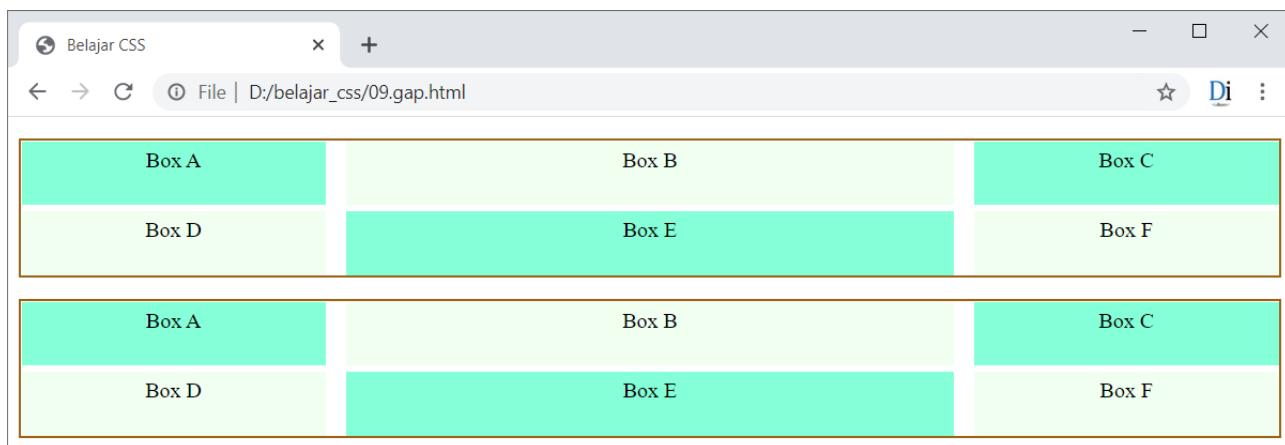
Sesuai dengan namanya, `row-gap` dipakai untuk mengatur spasi antar baris, dan `column-gap` dipakai untuk men-set spasi antar kolom. Sedangkan property `gap` merupakan shorthand untuk mengatur gap antar baris dan kolom sekaligus.

Berikut contoh penggunaan:

09.gap.html

```

1 <div class="container" style="row-gap: 5px; column-gap: 15px;">
2   <div class="box"> Box A </div>
3   ..
4   <div class="box"> Box F </div>
5 </div>
6
7 <div class="container" style="gap: 5px 15px;">
8   <div class="box"> Box A </div>
9   ...
10  <div class="box"> Box F </div>
11 </div>
```



Gambar: Hasil penggunaan property row-gap, column-gap dan gap

Untuk property `gap`, nilai pertama dipakai untuk mengatur `row-gap`, dan nilai kedua untuk `column-gap`. Dalam contoh di atas, property `row-gap: 5px; column-gap: 15px` bisa disingkat menjadi `gap: 5px 15px`.

Jika property `gap` diisi dengan satu nilai saja, maka itu akan dipakai untuk `row-gap` dan `column-gap` sekaligus.

Property `row-gap`, `column-gap` dan `gap` adalah penamaan baru dalam standar CSS. Sebelumnya ketiga property ini memiliki awal "grid-", yakni `grid-row-gap`, `grid-column-gap` dan `grid-gap`. Penulisan lama ini masih di dukung oleh mayoritas web browser dan tetap bisa dipakai.

19.5. Property grid-column-start dan grid-column-end

Kali ini kita masuk ke materi tentang penggabungan kolom dan baris grid. Terdapat 2 cara yang bisa dipakai, yakni proses penggabungan dengan menghitung urutan baris / kolom, atau bisa juga menggunakan **grid-area**.

Sebenarnya dengan grid-area akan lebih mudah, namun kita akan lihat konsep dasar penggabungan grid dengan menghitung urutan baris / kolom terlebih dahulu.

Sebagai bahan praktek, silahkan pelajari sejenak kode berikut:

10.grid-template.html

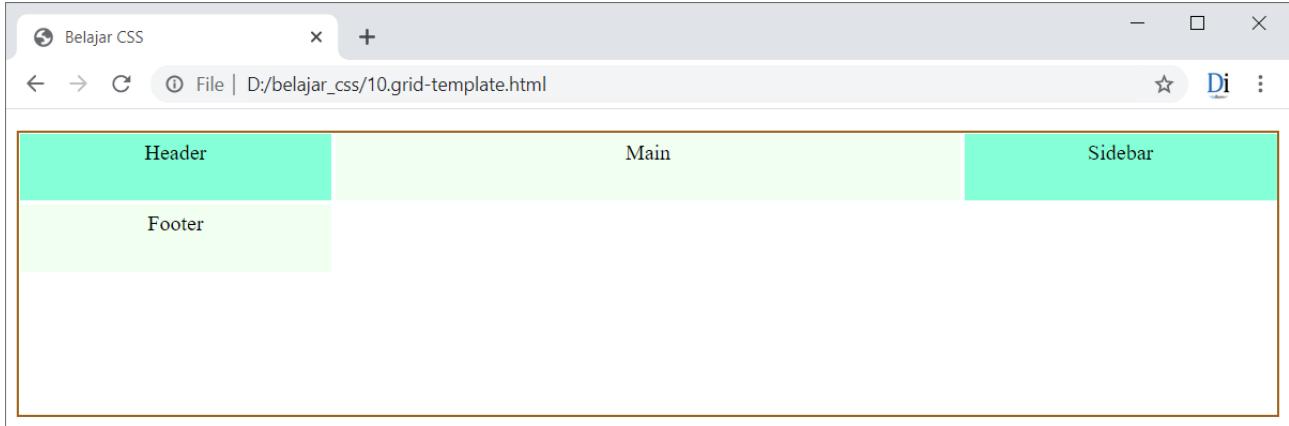
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              border: 2px solid #965e15;
9              margin-top: 1em ;
10             display: grid;
11             grid-template-columns: 1fr 2fr 1fr;
12             grid-template-rows: repeat(4, 50px);
13             gap: 0.2rem;
14         }
15         .box {
16             background-color:aquamarine;
17             text-align: center;
18             padding: 0.3em;
19         }
20         .box:nth-child(even) {
21             background-color:honeydew;
22         }
23     </style>
24 </head>
25 <body>
26
27     <div class="container">
28         <div class="box header"> Header </div>
29         <div class="box main"> Main </div>
30         <div class="box sidebar"> Sidebar </div>
31         <div class="box footer"> Footer </div>
32     </div>
33
34 </body>
35 </html>
```

Semua property ini sudah kita bahas sebelumnya. Isi dari selector .container dipakai untuk membuat struktur grid yang terdiri dari 3 kolom dan 4 baris, yakni hasil dari grid-template-columns: 1f 2fr 1fr serta grid-template-rows: repeat(4, 50px)). Di antara baris dan kolom ini terdapat gap sebesar 0.2rem hasil dari property gap: 0.2rem.

Jumlah grid item saya modifikasi sedikit yang kali ini terdiri dari 4 buah tag <div>. Setiap tag <div> memiliki atribut class="box [nama posisi]". Nama posisi mencerminkan fungsi dari item tersebut, apakah untuk header, main, sidebar atau footer.

Berikut tampilan dari kode di atas:

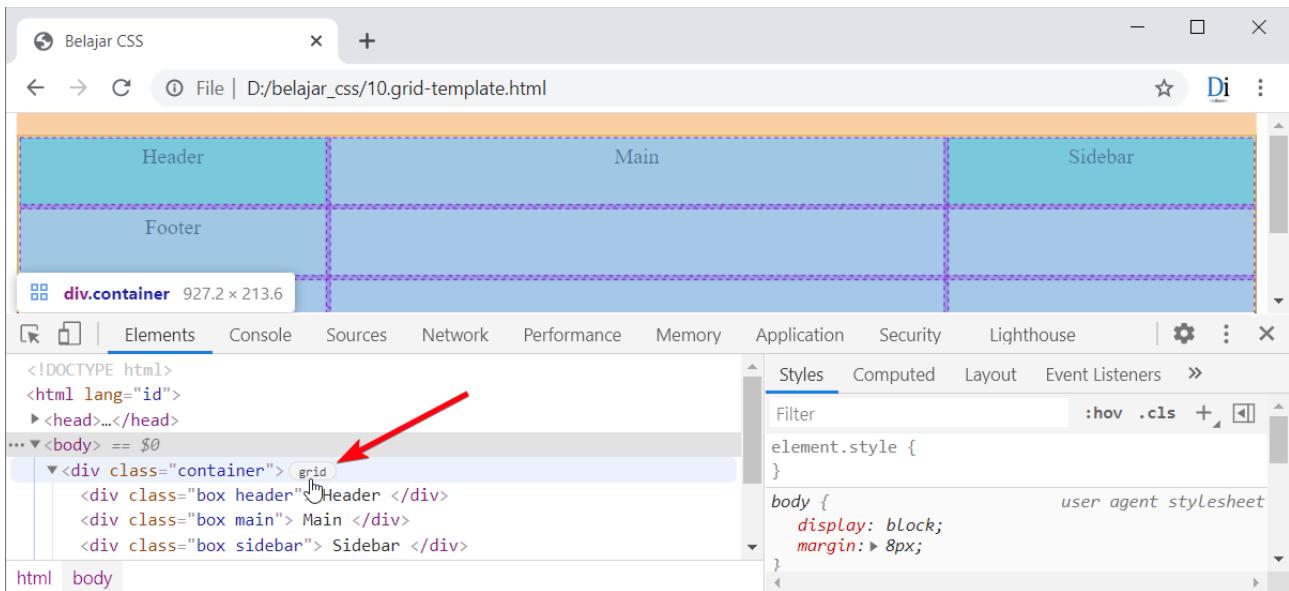


Gambar: Hasil tampilan file 10.grid-template.html

Meskipun kita hanya memiliki 4 item, tapi struktur grid ini terdiri dari 12 sel (hasil dari 3 kolom x 4 baris).

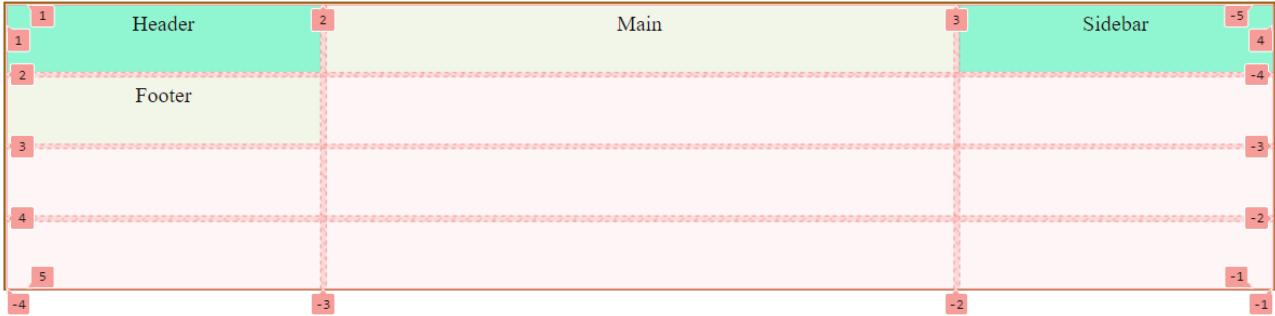
Property **grid-column-start** dan **grid-column-end** bisa dipakai untuk memperbesar ukuran sebuah grid item agar bisa mengambil tempat beberapa kolom sekaligus. Nilai untuk kedua property berupa nomor urut awal dan akhir dari garis grid.

Fitur developer tools bawaan web browser bisa membantu kita untuk melihat nomor urut garis dan juga kolom grid. Silahkan tekan CTRL + SHIFT + i, lalu inspect tag <div> yang menjadi grid container, kemudian klik tombol **grid** yang ada disebelahnya:



Gambar: Klik tombol grid yang ada di inspect element

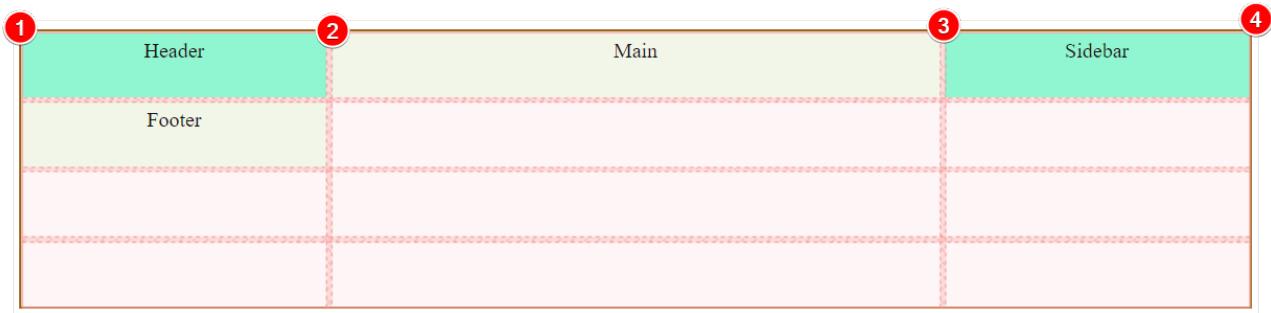
Sesaat kemudian, akan muncul garis bantu disertai nomor. Nomor-nomor inilah yang akan menjadi nilai input dari property **grid-column-start** dan **grid-column-end**.



Gambar: Nomor urut baris dan kolom grid

Angka yang ada cukup banyak karena Google Chrome ikut menampilkan angka positif dan negatif. Jika anda sedang merancang layout grid, fitur dari developer tools ini akan sangat membantu.

Kali ini kita akan fokus di nomor urut kolom positif yang diperlihatkan dengan lebih jelas pada ilustrasi berikut:



Gambar: Nomor urut kolom grid positif

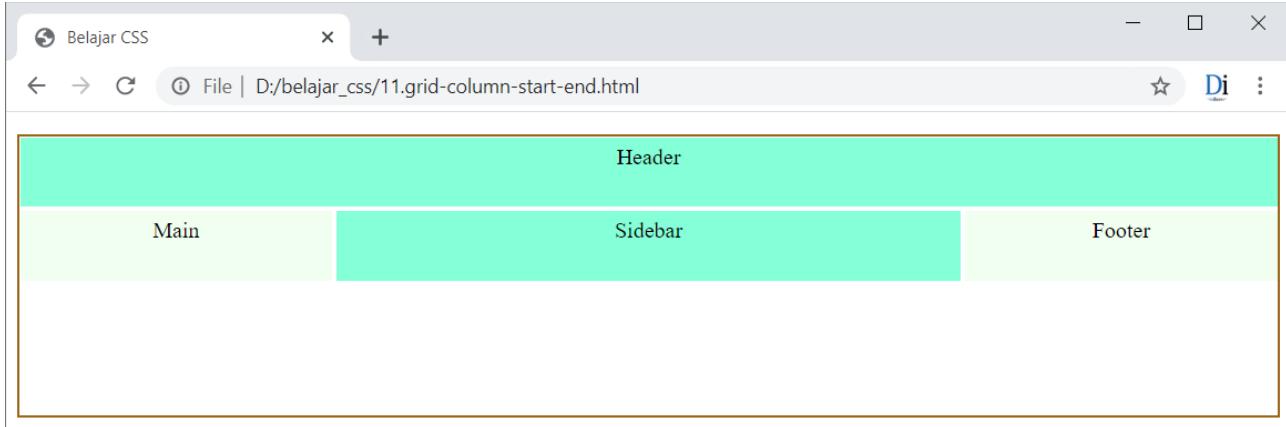
Property `grid-column-start` berfungsi untuk menentukan garis awal kolom grid dan akan terus melebar hingga garis akhir yang diisi pada `grid-column-end`. Sebagai contoh, jika saya ingin element "Header" melebar mulai dari garis kolom 1 hingga garis kolom 4, bisa menggunakan property berikut:

11.grid-column-start-end.html

```

1  ...
2  .header {
3    grid-column-start: 1;
4    grid-column-end: 4;
5  }

```



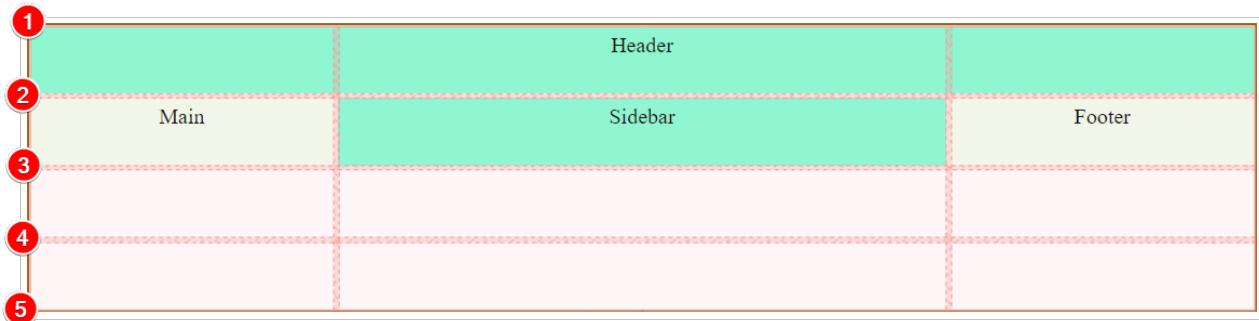
Gambar: Grid item "Header" melebar dari nomor kolom 1 hingga nomor kolom 4

Hasilnya, tag `<div class="box header"> Header </div>` akan melebar mengambil tempat 1 baris penuh. Karena baris pertama sudah diambil untuk `.header`, maka grid item lain akan pindah ke baris kedua.

19.6. Property grid-row-start dan grid-row-end

Property `grid-row-start` dan `grid-row-end` dipakai untuk proses penggabungan baris grid. Konsep dasar penggunaannya sangat mirip seperti `grid-column-start` dan `grid-column-end`, hanya saja sekarang akan berefek pada baris, bukan lagi kolom.

Melanjutkan contoh yang sama, berikut ilustrasi nomor baris grid:



Gambar: Nomor urut baris grid positif

Sebagai contoh praktek, kali ini saya ingin element "Main" membesar dan mengambil 2 kolom dan 2 baris, yakni mulai dari garis kolom 1 hingga 3, serta garis baris 2 hingga 4. Berikut property yang diperlukan:

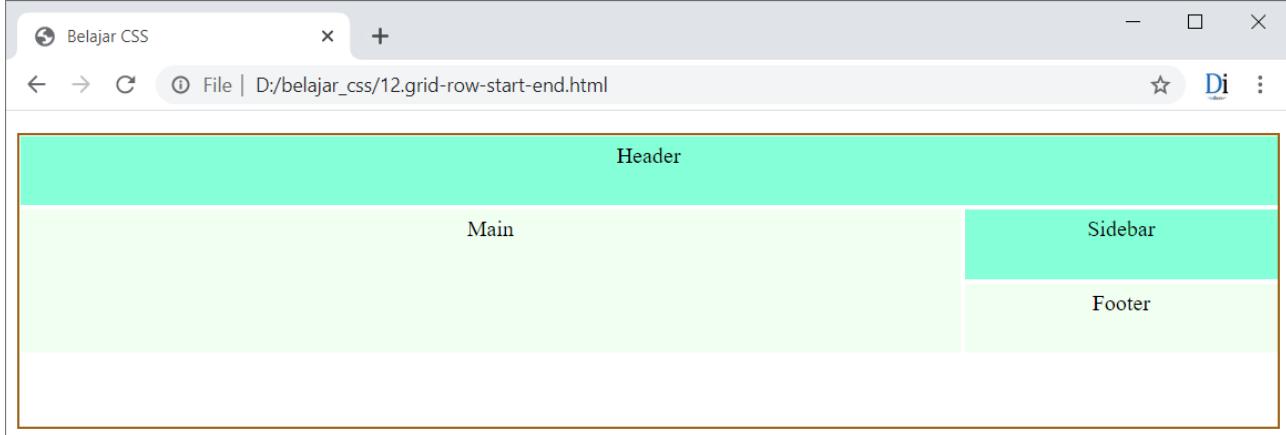
`12.grid-row-start-end.html`

```

1 ...
2 .main {
3   grid-column-start: 1;
4   grid-column-end: 3;
5   grid-row-start: 2;

```

```
6     grid-row-end: 4;
7 }
```



Gambar: Grid item "Main" melebar dari nomor kolom 1 hingga 2 serta nomor baris 2 hingga 4

Hasilnya, tag `<div class="box main"> Main </div>` akan mengambil 2 baris di bawah header, dan melebar mengambil tempat sebanyak 3 kolom.

19.7. Property **grid-column** dan **grid-row**

Sebelumnya kita menulis nomor garis awal dan garis akhir menggunakan 2 property terpisah. Selain itu terdapat juga penulisan singkat (*shorthand*) dengan property **grid-column** dan **grid-row**. Format penulisan nilai kedua property ini adalah sebagai berikut:

`<nomor_garis_awal> / <nomor_garis_akhir>`

Sebagai contoh, property `grid-column: 3 / 4` merupakan penulisan singkat dari:

`grid-column-start: 3;
grid-column-end: 4;`

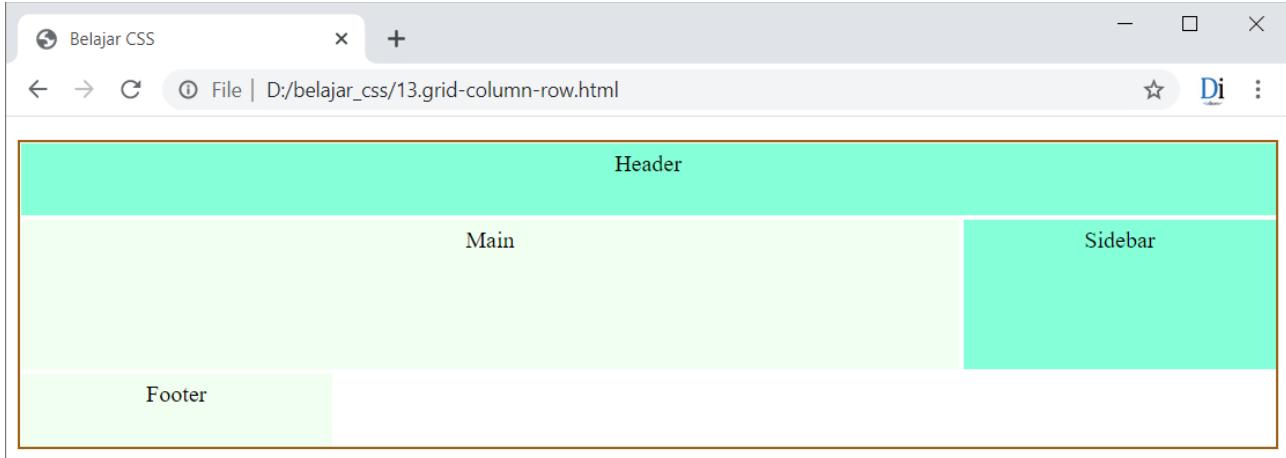
Atau property `grid-row: 2 / 4` sama artinya dengan:

`grid-row-start: 2;
grid-row-end: 4;`

Berikut praktek penggunaan property **grid-column** dan **grid-row**:

13.grid-column-row.html

```
1 ...
2 .sidebar {
3   grid-column: 3 / 4;
4   grid-row: 2 / 4;
5 }
```



Gambar: Hasil penggunaan property grid-column dan grid-row

Selector `.sidebar` di atas artinya saya ingin tag `<div class="box sidebar"> Sidebar </div>` mengambil tempat mulai dari garis kolom ke-3 sampai ke-4, serta baris dari garis 2 sampai 4. Hasilnya, sidebar akan berada di sisi kanan grid container.

Penulisan nilai untuk property `grid-column` dan `grid-row` juga bisa menggunakan keyword `span`. Caranya, tulis jumlah kolom atau baris pembesaran setelah kata "span". Berikut contoh yang dimaksud:

```

1 .footer {
2   grid-column: 1 / span 3;
3   grid-row: 4 / span 1;
4 }
```

Kode di atas berarti saya ingin selector `.footer` mengambil tempat mulai dari kolom garis 1, lalu melebar sebanyak 3 kolom, serta mulai dari garis baris 4 dan melebar sebanyak 1 baris.

Berikut kode program lengkap dari praktek pembuatan grid layout yang kita bahas sejauh ini:

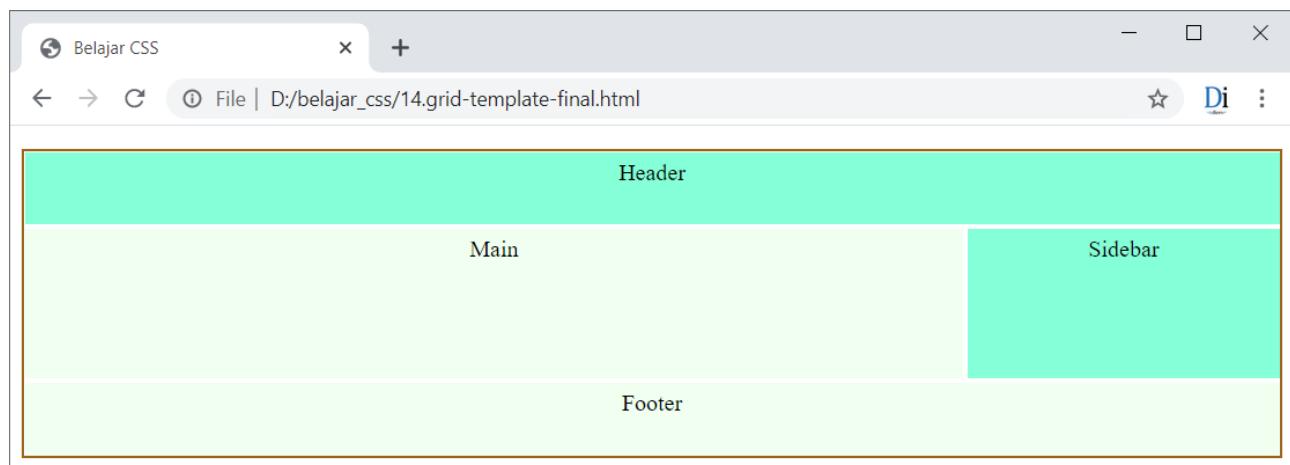
`14.grid-template-final.html`

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
7      .container {
8        border: 2px solid #965e15;
9        margin-top: 1em ;
10       display: grid;
11       grid-template-columns: 1fr 2fr 1fr;
12       grid-template-rows: repeat(4, 50px);
13       gap: 0.2rem;
14     }
15     .box {
16       background-color:aquamarine;
```

CSS3 Grid

```
17     text-align: center;
18     padding: 0.3em;
19 }
20 .box:nth-child(even) {
21     background-color:honeydew;
22 }
23 .header {
24     grid-column-start: 1;
25     grid-column-end: 4;
26 }
27 .main {
28     grid-column-start: 1;
29     grid-column-end: 3;
30     grid-row-start: 2;
31     grid-row-end: 4;
32 }
33 .sidebar {
34     grid-column: 3 / 4;
35     grid-row: 2 / 4;
36 }
37 .footer {
38     grid-column: 1 / span 3;
39     grid-row: 4 / span 1;
40 }
41 </style>
42 </head>
43 <body>
44
45 <div class="container">
46     <div class="box header"> Header </div>
47     <div class="box main"> Main </div>
48     <div class="box sidebar"> Sidebar </div>
49     <div class="box footer"> Footer </div>
50 </div>
51
52 </body>
53 </html>
```

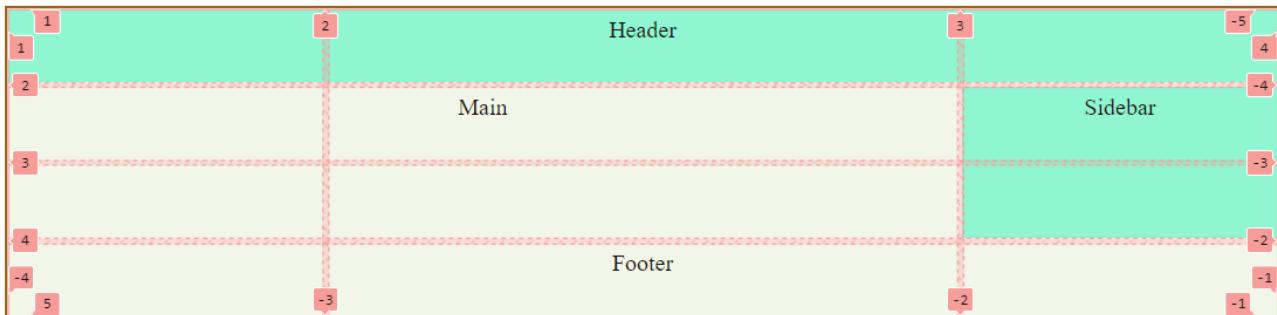


Gambar: Hasil pembuatan layout dengan nomor urut baris / kolom

Fokus bahasan kita ada di antara baris 23 – 40, yakni berbagai property untuk membuat grid layout menggunakan nomor urut kolom dan nomor urut baris.

Sebagai tambahan, kita juga bisa mengisi angka negatif sebagai nomor kolom / baris.

Perhitungan angka negatif ini dimulai dari sisi kanan grid container. Garis grid bantu dari developer tools akan mempermudah kita dalam menghitung urutan garis:



Gambar: Nomor urut baris dan kolom grid

Angka di sisi atas dan sisi kiri menampilkan nomor urut positif untuk kolom dan baris.

Sedangkan angka di sisi bawah dan sisi kanan menampilkan nomor urut negatif untuk kolom dan baris.

Sebagai contoh, property untuk grid item `.main` bisa ditulis sebagai berikut:

```

1 .main {
2   grid-column: 1 / 3;
3   grid-row: 2 / 4;
4 }
```

Atau bisa juga ditulis sebagai:

```

1 .main {
2   grid-column: -4 / -2;
3   grid-row: -4 / -2;
4 }
```

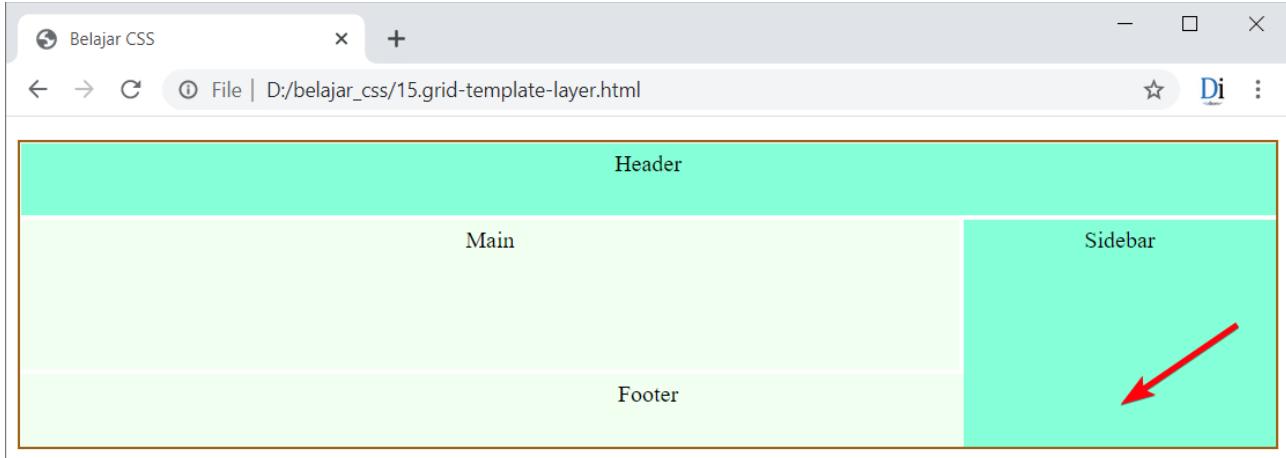
Fitur lain dari penggabungan grid ini adalah, bisa berada di atas grid item lain (membuat efek layer). Sebagai contoh, jika selector `.sidebar` saya ubah menjadi berikut:

15.grid-template-layer.html

```

1 .sidebar {
2   grid-column: 3 / 4;
3   grid-row: 2 / 5;
4   z-index: 1;
5 }
```

Maka bagian bawah sidebar akan berada di atas bagian footer:



Gambar: Bagian bawah sidebar berada di atas footer

Efeknya memang agak susah dilihat, tapi silahkan hapus tambahan property `z-index: 1`, maka sekarang bagian footer yang akan menimpa sisi bawah sidebar. Ketika terjadi penumpukan element seperti ini, property `z-index` bisa dipakai untuk mengatur element mana yang berada di layer / lapisan pertama.

19.8. Property `grid-template-areas` dan `grid-area`

Selain menggunakan nomor urut baris / kolom, terdapat cara yang lebih mudah untuk menggabungkan grid item, yakni melalui property `grid-template-areas` dan property `grid-area`. Menggunakan kedua property ini, kita bisa "menggambar" pola grid layout tanpa harus menghitung urutan baris.

Sebagai bahan praktek, kita masih menggunakan struktur HTML seperti contoh sebelumnya, yakni:

```

1 <div class="container">
2   <div class="box header"> Header </div>
3   <div class="box main"> Main </div>
4   <div class="box sidebar"> Sidebar </div>
5   <div class="box footer"> Footer </div>
6 </div>
```

Langkah pertama adalah, memberi nama grid area untuk setiap item. Ini dilakukan melalui property `grid-area` seperti contoh berikut:

```

1 .header {
2   grid-area: header;
3 }
4 .main {
5   grid-area: main;
6 }
7 .sidebar {
8   grid-area: sidebar;
```

```

9  }
10 .footer {
11   grid-area: footer;
12 }
```

Saya memberi nama grid item yang sama dengan nama selector. Akan tetapi ini tidak wajib, anda bebas memberi nama lain sesuai keinginan.

Langkah selanjutnya, buat pola gambar grid menggunakan selector `grid-template-areas` ke dalam grid container:

```

1  .container {
2    border: 2px solid #965e15;
3    margin-top: 1em ;
4    display: grid;
5    grid-template-columns: 1fr 2fr 1fr;
6    grid-template-rows: repeat(4, 50px);
7    gap: 0.2rem;
8    grid-template-areas:
9      " header header header "
10     " main main sidebar "
11     " main main sidebar "
12     " footer footer footer ";
13 }
```

Semua property di baris 2 sampai 7 sama seperti praktek kita sebelumnya. Kode tersebut dipakai untuk membuat struktur grid yang terdiri dari 12 sel (hasil dari 3 kolom x 4 baris).

Fokus kita ada di cara penulisan property `grid-template-areas`. Teks yang ditulis di sini bersesuaian dengan jumlah baris dan kolom struktur grid. Di baris pertama terdapat teks "`header header header`", artinya saya ingin 3 kolom baris pertama diisi oleh header. Teks "header" berasal dari nilai `grid-area: header` yang sebelumnya ditulis dalam selector `.header`.

Lanjut ke baris kedua, terdapat teks "`main main sidebar`". Artinya, saya ingin 2 kolom baris kedua diambil oleh `grid-area: main`, lalu kolom ketiga baris kedua diisi oleh `grid-area: sidebar`.

Teks di baris ketiga sama seperti baris kedua, yang berarti saya ingin main dan sidebar mengambil tempat di 2 baris.

Terakhir di baris 4 terdapat "`footer footer footer`" yang akan mengisi seluruh kolom dengan grid item footer. Berikut kode program lengkap dari latihan kita:

16.grid-area.html

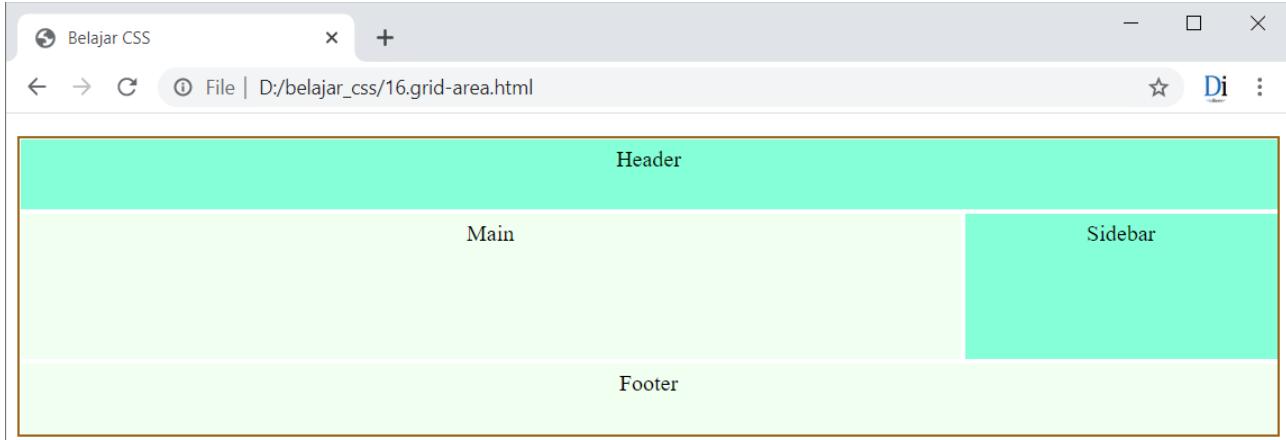
```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4    <meta charset="UTF-8">
5    <title>Belajar CSS</title>
6    <style>
```

CSS3 Grid

```
7   .container {
8     border: 2px solid #965e15;
9     margin-top: 1em ;
10    display: grid;
11    grid-template-columns: 1fr 2fr 1fr;
12    grid-template-rows: repeat(4, 50px);
13    gap: 0.2rem;
14    grid-template-areas:
15      " header header header "
16      " main main sidebar "
17      " main main sidebar "
18      " footer footer footer ";
19  }
20  .box {
21    background-color:aquamarine;
22    text-align: center;
23    padding: 0.3em;
24  }
25  .box:nth-child(even) {
26    background-color:honeydew;
27  }
28  .header {
29    grid-area: header;
30  }
31  .main {
32    grid-area: main;
33  }
34  .sidebar {
35    grid-area: sidebar;
36  }
37  .footer {
38    grid-area: footer;
39  }
40  </style>
41 </head>
42 <body>
43
44  <div class="container">
45    <div class="box header"> Header </div>
46    <div class="box main"> Main </div>
47    <div class="box sidebar"> Sidebar </div>
48    <div class="box footer"> Footer </div>
49  </div>
50
51 </body>
52 </html>
```

CSS3 Grid



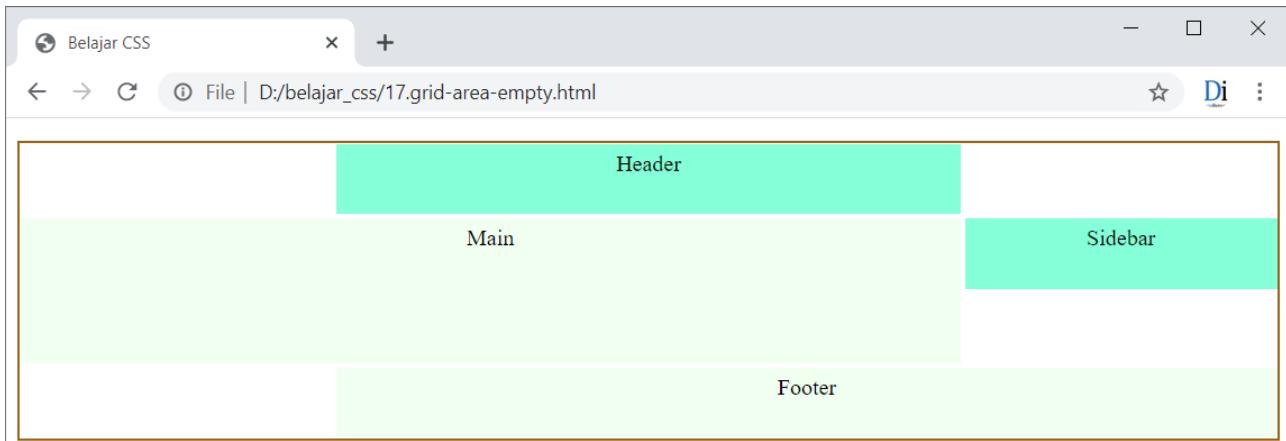
Gambar: Membuat layout dengan grid-template-areas

Hasilnya sama persis seperti praktik dengan garis kolom / baris grid, namun dengan cara yang lebih mudah.

Jika kita ingin memberikan ruang kosong pada sel grid tertentu, bisa diisi dengan karakter titik sebagai berikut:

17.grid-area-empty.html

```
1 .container {  
2   ...  
3   grid-template-areas:  
4     ". header ."  
5     " main main sidebar "  
6     " main main ."  
7     ". footer footer ";  
8 }
```



Gambar: Membuat layout dengan ruang kosong

Di dalam gambar pola grid-template-areas, saya mengganti nama area dengan tanda titik di beberapa tempat. Hasilnya, terdapat ruang kosong di tempat pola titik berada.

Agar gambar pola menjadi lebih jelas, karakter titik ini boleh ditulis beberapa kali seperti

contoh berikut:

```

1 .container {
2 ...
3
4   grid-template-areas:
5     " ..... header .... "
6     " main main sidebar "
7     " main main ..... "
8     " ..... footer footer ";
9 }
```

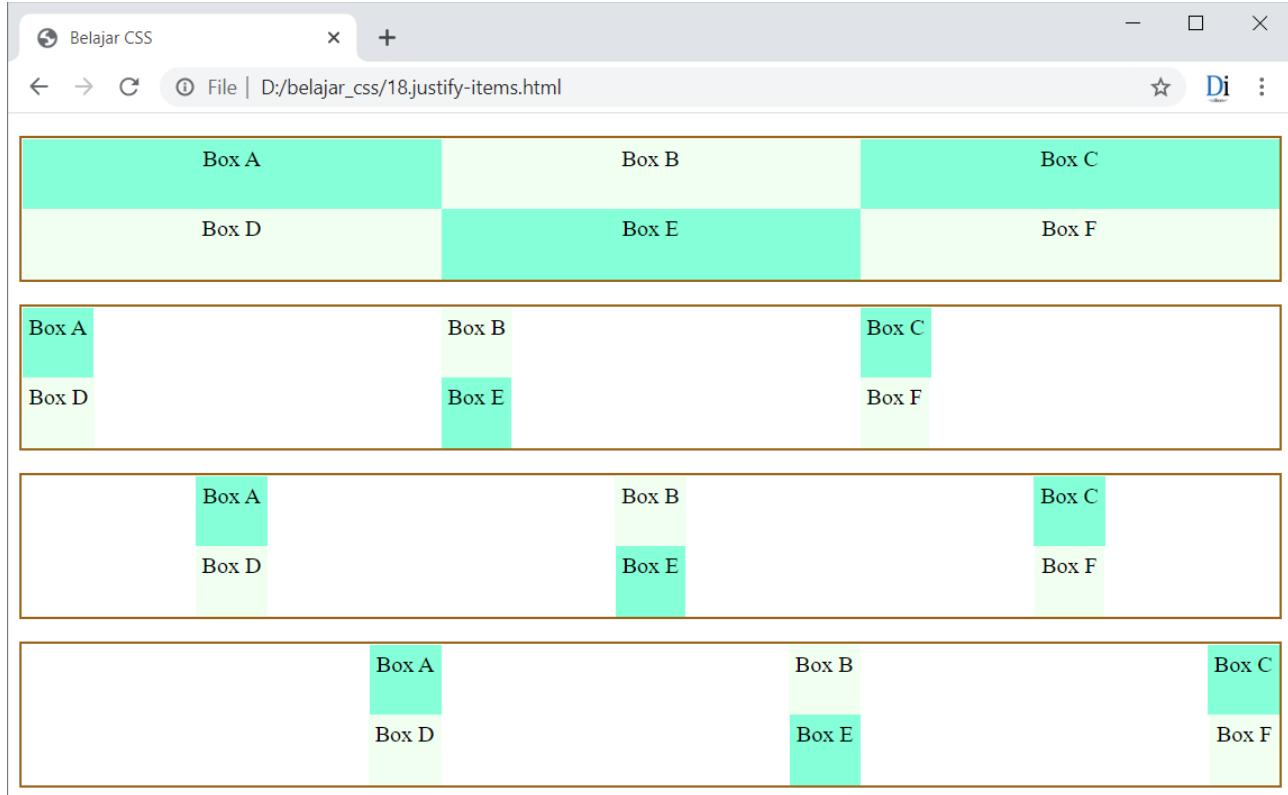
19.9. Property justify-items

Property **justify-items** berfungsi untuk mengatur posisi semua grid item pada sumbu x. Nilai yang bisa diisi adalah stretch (default), start, center, dan end. Berikut contoh penggunaannya:

18.justify-items.html

```

1 <style>
2   .container {
3     height: 100px;
4     border: 2px solid #965e15;
5     margin-top: 1em ;
6     display: grid;
7     grid-template-columns: repeat(3, 1fr);
8   }
9 ...
10 </style>
11
12 <div class="container" style="justify-items: stretch;">
13   <div class="box"> Box A </div>
14   ...
15   <div class="box"> Box F </div>
16 </div>
17
18 <div class="container" style="justify-items: start;">
19   ...
20 </div>
21
22 <div class="container" style="justify-items: center;">
23   ...
24 </div>
25
26 <div class="container" style="justify-items: end;">
27   ...
28 </div>
```



Gambar: Hasil penggunaan property justify-items

Struktur grid container terdiri dari 3 kolom hasil dari `grid-template-columns: repeat(3, 1fr)`. Jika property `width` setiap flex item tidak di set, secara default akan melebar (`stretch`) memenuhi 1fr.

Namun begitu diberikan nilai `grid-template-columns: start, center, atau end`, setiap flex item akan mengecil sesuai lebar konten yang ada di dalamnya.

19.10. Property align-items

Property **align-items** berfungsi untuk mengatur posisi semua grid item pada sumbu y. Nilai yang bisa diisi adalah `stretch` (default), `start`, `center`, dan `end`. Berikut contoh penggunaannya:

19.align-items.html

```

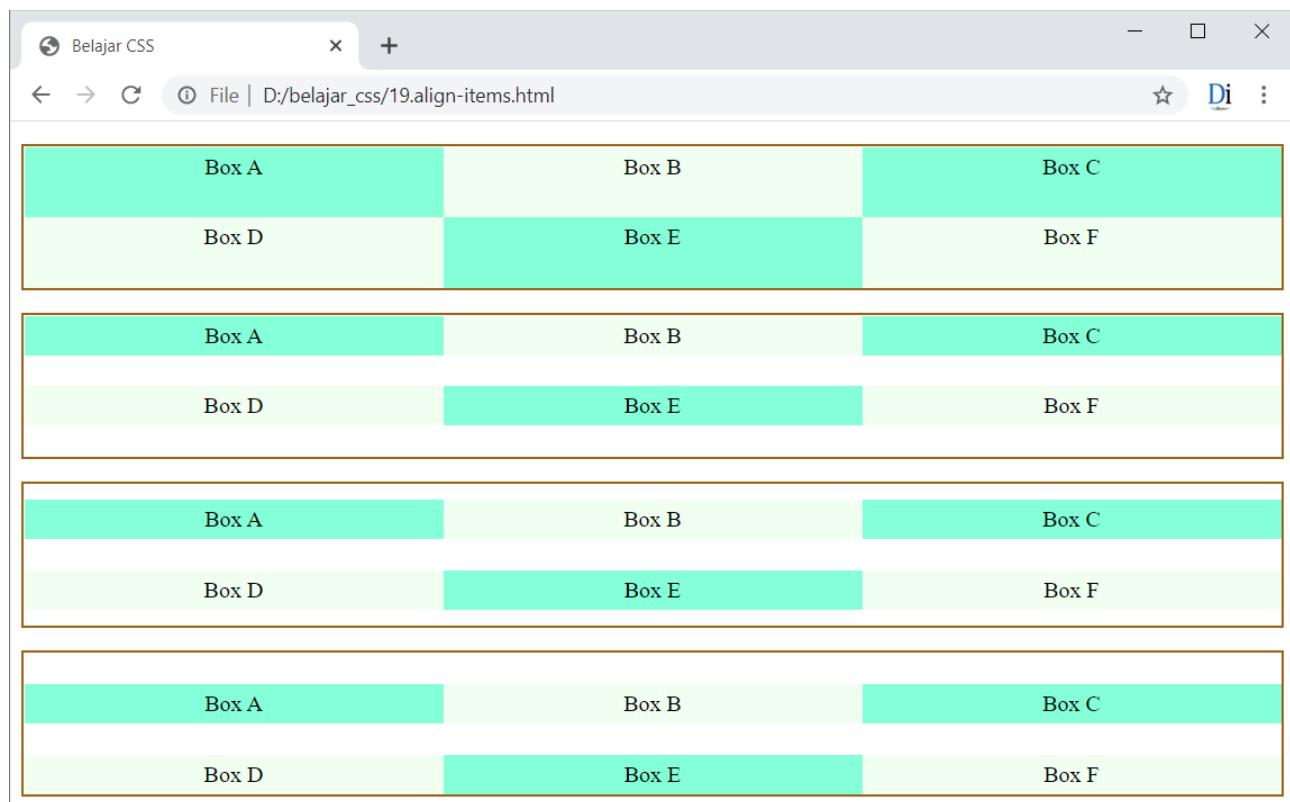
1 <style>
2   .container {
3     height: 100px;
4     border: 2px solid #965e15;
5     margin-top: 1em ;
6     display: grid;
7     grid-template-columns: repeat(3, 1fr);
8   }
9 ...

```

```

10 </style>
11
12 <div class="container" style="align-items: stretch;">
13   <div class="box"> Box A </div>
14   ...
15   <div class="box"> Box F </div>
16 </div>
17
18 <div class="container" style="align-items: start;">
19   ...
20 </div>
21
22 <div class="container" style="align-items: center;">
23   ...
24 </div>
25
26 <div class="container" style="align-items: end;">
27   ...
28 </div>

```



Gambar: Hasil penggunaan property align-items

Struktur grid container masih sama seperti contoh sebelum ini, yaitu terdiri dari 3 kolom hasil `grid-template-columns: repeat(3, 1fr)`.

Jika property `height` setiap flex item tidak di set, secara default akan melebar (`stretch`) memenuhi tinggi. Syarat lain, `width` dari grid container juga harus ditentukan, jika tidak tinggi grid item hanya akan sebatas tinggi konten yang ada di dalamnya saja.

Begitu diberikan nilai `grid-align-items: start, center atau end`, tinggi setiap flex item akan mengecil sesuai lebar konten yang ada di dalamnya.

19.11. Property place-items

Property `place-items` merupakan penulisan singkat (*shorthand*) dari `justify-items` dan `align-items`. Nilai pertama dari `place-items` adalah untuk `justify-items`, kemudian diikuti nilai untuk `align-items`.

Sebagai contoh, property `place-items: start end` artinya sama dengan:

```
justify-items: start;
align-items: end;
```

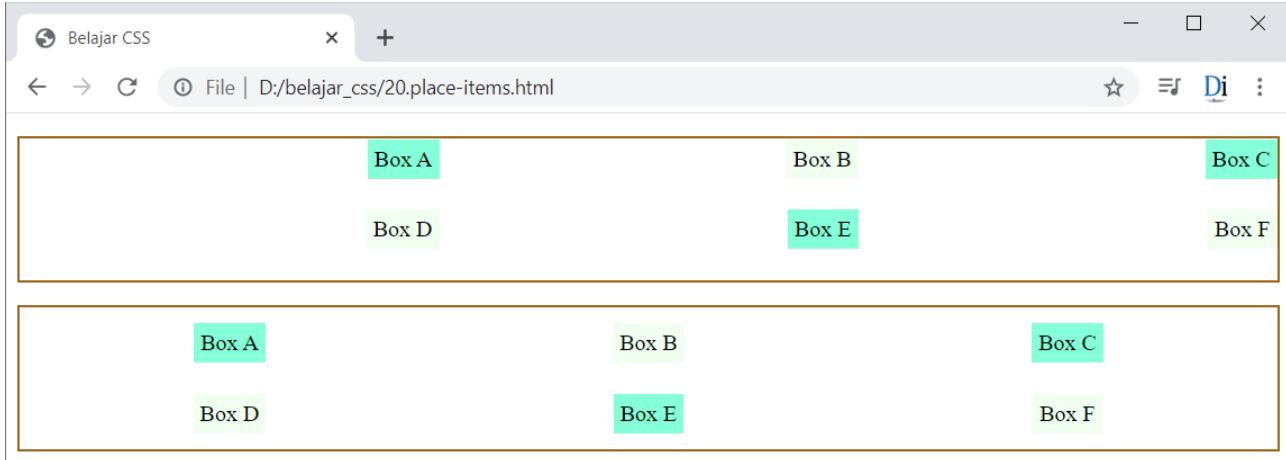
Jika nilai yang diisi untuk `place-items` hanya satu, maka nilai tersebut akan dipakai untuk `justify-items` dan `align-items` sekaligus. Sebagai contoh, `place-items: center` merupakan penulisan singkat dari:

```
justify-items: center;
align-items: center;
```

Berikut contoh prakteknya:

20.place-items.html

```
1 <style>
2   .container {
3     ...
4     display: grid;
5     grid-template-columns: repeat(3, 1fr);
6   }
7   ...
8 </style>
9
10 <div class="container" style="place-items: start end;">
11   <div class="box"> Box A </div>
12   ...
13   <div class="box"> Box F </div>
14 </div>
15
16 <div class="container" style="place-items: center;">
17   <div class="box"> Box A </div>
18   ...
19   <div class="box"> Box F </div>
20 </div>
```



Gambar: Hasil penggunaan property place-items

19.12. Property justify-self, align-self, dan place-self

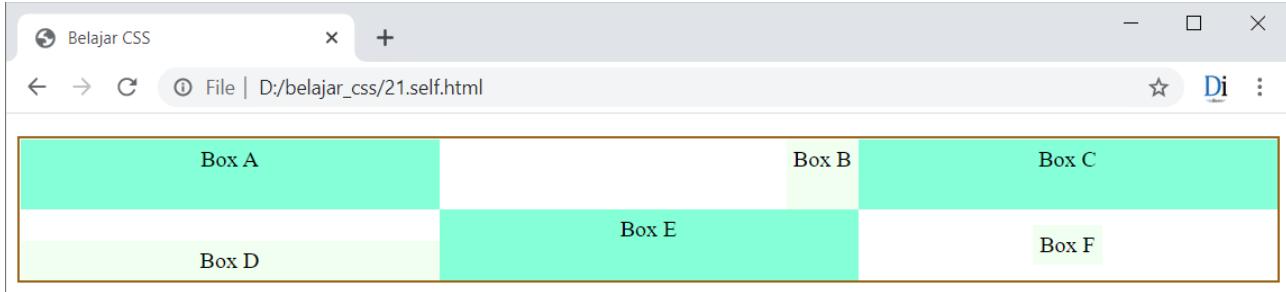
Ketiga property ini merupakan variasi dari property `justify-items`, `align-items` dan `place-items` yang ditempatkan ke dalam grid item, bukan lagi ke grid container. Dengan property **justify-self**, **align-self**, dan **place-self**, kita bisa mengatur posisi setiap grid item secara individu.

Nilai yang bisa diisi ke dalam ketiga property ini sama seperti "kakaknya", yakni `stretch`, `start`, `center`, dan `end` seperti contoh berikut:

21.self.html

```

1 <style>
2   .container {
3     ...
4     display: grid;
5     grid-template-columns: repeat(3, 1fr);
6   }
7   ...
8 </style>
9
10 <div class="container">
11   <div class="box"> Box A </div>
12   <div class="box" style="justify-self: end;"> Box B </div>
13   <div class="box"> Box C </div>
14   <div class="box" style="align-self: end;"> Box D </div>
15   <div class="box"> Box E </div>
16   <div class="box" style="place-self: center;"> Box F </div>
17 </div>
```



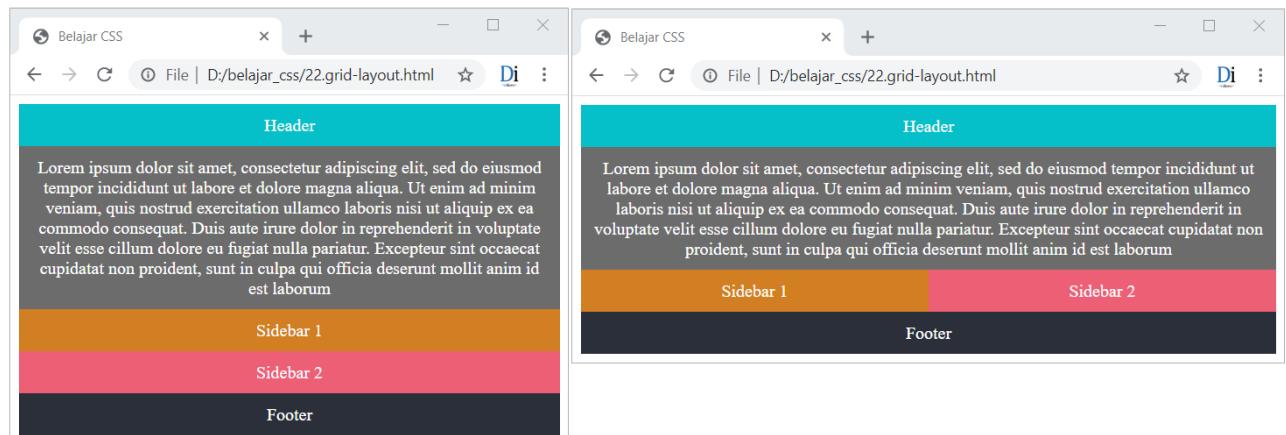
Gambar: Hasil penggunaan property justify-self, align-self, dan place-self

Dalam contoh ini, tampilan box B, box D dan box F akan berubah sesuai nilai yang diinput ke dalam property `justify-self`, `align-self`, dan `place-self`.

19.13. Membuat Layout Responsive Sederhana

Sekarang saatnya kita masuk ke praktik implementasi pembuatan layout responsive menggunakan CSS3 grid.

Agar bisa dibandingkan, saya akan membuat tampilan layout yang sama persis seperti praktik di akhir bab CSS3 flexbox. Berikut tampilan yang dimaksud:



Gambar: Tampilan layout untuk versi smartphone dan tablet



Gambar: Tampilan layout untuk versi desktop

Dibandingkan versi flexbox, membuat layout dengan grid akan lebih mudah, apalagi jika memakai property `grid-template-area` dimana kita cukup menulis gambar pola untuk setiap breakpoint.

Struktur HTML yang diperlukan adalah sebagai berikut:

```

1 <div class="container">
2   <header class="header">Header</header>
3   <article class="main">
4     Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
5       do eiusmod tempor incididunt ut labore et dolore magna aliqua.
6       Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
7         nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
8           reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
9             pariatur. Excepteur sint occaecat cupidatat non proident,
10            sunt in culpa qui officia deserunt mollit anim id est laborum
11        </article>
12        <aside class="sidebar-1">Sidebar 1</aside>
13        <aside class="sidebar-2">Sidebar 2</aside>
14        <footer class="footer">Footer</footer>
15      </div>
```

Layout Untuk Layar Smartphone

Melihat dari tiga jenis tampilan yang akan kita rancang, saya memutuskan untuk memakai struktur grid yang terdiri dari 4 kolom dengan perbandingan 1fr, 2fr, 2fr, 1fr. Dengan demikian kolom di sisi tengah halaman akan terlihat lebih besar dibandingkan sisi kiri.

Tampilan untuk smartphone juga sangat *simple*, yakni setiap grid item mengambil tempat penuh untuk setiap baris. Berikut kode CSS yang diperlukan:

```

1 .container {
2   color: white;
3   text-align: center;
4   display: grid;
5   grid-template-columns: 1fr 1.5fr 1.5fr 1fr;
6   grid-template-areas:
7     " header header header header "
8     " main main main main "
9     " sidebar1 sidebar1 sidebar1 sidebar1 "
10    " sidebar2 sidebar2 sidebar2 sidebar2 "
11    " footer footer footer footer ";
12 }
13
14 .container > * {
15   padding: 10px;
16 }
17
18 .header { background-color: #04bcc5; grid-area: header; }
19 .main { background-color: #666666; grid-area: main; }
20 .sidebar-1 { background-color: #cf781e; grid-area: sidebar1; }
21 .sidebar-2 { background-color: #eb596e; grid-area: sidebar2; }
```

```
22 .footer { background-color: #252a34; grid-area: footer;}
```

Struktur grid dibuat oleh property `grid-template-columns: 1fr 1.5fr 1.5fr 1fr` di baris 5, yang langsung diikuti property `grid-template-areas` untuk "menggambar" grid layout.

Nama grid area menggunakan nama yang sama dengan atribut class dari setiap element. Ini didefinisikan dari property `grid-area` di baris 18 – 22.

Hasilnya, kita sudah selesai membuat tampilan layout versi smartphone.

Layout Untuk Layar Tablet

Untuk tampilan layar tablet, sidebar akan saling berjejer pada satu baris yang sama. Caranya, cukup ubah gambar grid layout seperti kode berikut:

```
1 @media all and (min-width: 600px) {
2   .container {
3     grid-template-areas:
4       " header header header header "
5       " main main main main "
6       " sidebar1 sidebar1 sidebar2 sidebar2 "
7       " footer footer footer footer ";
8   }
9 }
```

Perubahan pola grid ada di baris 6. Sekarang pada baris tersebut area sidebar1 mengambil 2 kolom dan diikuti oleh area sidebar2 juga sebanyak 2 kolom.

Layout Untuk Layar Desktop

Untuk tampilan layar desktop, kedua sidebar akan mengapit main konten pada baris yang sama. Ini sangat mudah dilakukan dengan mengatur ulang pola grid milik property `grid-template-areas`:

```
1 @media all and (min-width: 900px) {
2   .container {
3     grid-template-areas:
4       " header header header header "
5       " sidebar1 main main sidebar2 "
6       " footer footer footer footer ";
7   }
8 }
```

Sekarang pola untuk baris setelah header adalah "`sidebar1 main main sidebar2`", sehingga main konten akan berada di tengah-tengah kedua sidebar.

Berikut kode program lengkap dari latihan membuat layout dengan CSS3 grid:

`22.grid-layout.html`

```
1 <!DOCTYPE html>
```

```
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     .container {
8       color: white;
9       text-align: center;
10      display: grid;
11      grid-template-columns: 1fr 1.5fr 1.5fr 1fr;
12      grid-template-areas:
13        " header header header header "
14        " main main main main "
15        " sidebar1 sidebar1 sidebar1 sidebar1 "
16        " sidebar2 sidebar2 sidebar2 sidebar2 "
17        " footer footer footer footer ";
18    }
19
20   .container > * {
21     padding: 10px;
22   }
23
24   .header { background-color: #04bcc5; grid-area: header; }
25   .main { background-color: #666666; grid-area: main; }
26   .sidebar-1 { background-color: #cf781e; grid-area: sidebar1; }
27   .sidebar-2 { background-color: #eb596e; grid-area: sidebar2; }
28   .footer { background-color: #252a34; grid-area: footer; }
29
30 @media all and (min-width: 600px) {
31   .container {
32     grid-template-areas:
33       " header header header header "
34       " main main main main "
35       " sidebar1 sidebar1 sidebar2 sidebar2 "
36       " footer footer footer footer ";
37   }
38 }
39
40 @media all and (min-width: 900px) {
41   .container {
42     grid-template-areas:
43       " header header header header "
44       " sidebar1 main main sidebar2 "
45       " footer footer footer footer ";
46   }
47 }
48 </style>
49 </head>
50 <body>
51
52   <div class="container">
53     <header class="header">Header</header>
54     <article class="main">
55       Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed
56       do eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

```
57 Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
58 nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
59 reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla
60 pariatur. Excepteur sint occaecat cupidatat non proident,
61 sunt in culpa qui officia deserunt mollit anim id est laborum
62 </article>
63 <aside class="sidebar-1">Sidebar 1</aside>
64 <aside class="sidebar-2">Sidebar 2</aside>
65 <footer class="footer">Footer</footer>
66 </div>
67
68 </body>
69 </html>
```

Seperti yang kita lihat, pembuatan layout menggunakan CSS grid jauh lebih mudah dibandingkan CSS flexbox. Terlebih dari pola `grid-template-areas` bisa ditebak seperti apa bentuk akhir layout tanpa perlu menjalankan file HTML. Ini tidak bisa dilakukan (atau tepatnya jauh lebih sulit) jika ditulis menggunakan flexbox.

Meskipun begitu, flexbox tetap punya keunggulan tersendiri. Pengaturan layout pada 1 baris lebih mudah diatur dengan flexbox, misalnya mengatur posisi logo di dalam sidebar, atau mengatur konten yang ada di dalam footer.

Jadi kita tidak harus memilih ingin menggunakan grid atau flexbox, lebih bagus jika keduanya saling dikombinasikan.

Dalam bab ini kita telah membahas tentang cara penggunaan CSS3 Grid. Bab berikutnya akan disambung dengan materi "gado-gado" yang membahas berbagai hal yang dirasa perlu terkait CSS tapi tidak terlalu besar untuk bisa menjadi bab tersendiri.

20. Materi CSS Lainnya

Sampai di sini kita sudah membahas hampir semua property dasar CSS. Selain itu masih ada beberapa materi lain yang sangat menarik namun kurang pas menjadi bab tersendiri (karena cukup singkat). Oleh karena itu saya kumpulkan saja menjadi satu bab gado-gado dengan judul "Materi CSS Lainnya".

20.1. CSS Variable (custom property)

CSS variable sudah lama diimpikan banyak web developer, yakni sebuah fitur dimana kita bisa menyimpan nilai pada satu tempat, lalu mengaksesnya dari berbagai tempat lain. Contoh kasus yang sering ditemui adalah kebutuhan mereferensi warna seperti praktek berikut:

01.tanpa_variabel.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          .container {
8              width: 400px;
9              border: 5px solid #965e15;
10             margin: 0 auto;
11             padding: 1em;
12             font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
13         }
14         h1 {
15             margin-top: 0;
16             text-align: center;
17             color:#965e15;
18         }
19         span{
20             color:#965e15;
21         }
22     </style>
23 </head>
24 <body>
25
26     <div class="container">
27         <h1>Belajar CSS</h1>
```

```
28 <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iusto non  
29 tenetur voluptatibus dolor amet error rerum? Laboriosam, velit impedit.  
30 Tempore at <span>necessitatibus quo laboriosam!</span> Quo vitae  
31 excepturi recusandae facilis assumenda?</p>  
32 </div>  
33  
34 </body>  
35 </html>
```



Gambar: Sebuah paragraf dalam box

Di sini saya membuat kode HTML yang terdiri dari tag `<h1>`, sebuah paragraf `<p>` yang di dalamnya terdapat beberapa kata dalam tag ``. Menggunakan CSS, tampilan web menjadi lebih menarik dengan tambahan warna teks, serta pengaturan padding dan border.

Salah satu prinsip dasar desain UX adalah menggunakan warna yang konsisten. Sebuah web biasanya memiliki satu atau dua warna dasar yang dipakai pada berbagai element. Dalam contoh di atas, saya menggunakan warna cokelat `#965e15` untuk bagian judul `<h1>`, warna tag ``, serta warna border.

Jika terdapat komponen web lain, warna cokelat ini seharusnya terus di pakai. Akan tetapi, kode warna heksadesimal cukup sulit di hapal (biasanya harus selalu di copy paste), kemudian jika suatu saat kita ingin mengubah tema warna misalnya menjadi merah, maka kode warna itu harus diubah satu per satu. Kendala inilah yang bisa diselesaikan dengan **CSS variable**.

Di dalam dunia programming, *variable* (atau variabel) adalah "penanda identitas yang digunakan untuk menampung suatu nilai." Jika di awal kode program kita mendefinisikan variabel "warna" dan mengisinya dengan nilai "`#965e15`", maka sepanjang kode program kata "warna" bisa dipakai sebagai pengganti nilai "`#965e15`".

CSS3 menyediakan fitur yang disebut sebagai **custom property**, karena nantinya kita memang membuat sebuah property baru, namun fitur ini lebih populer dikenal sebagai **CSS variable**.

Berikut format dasar pembuatan CSS Variable atau CSS custom property:

```
--nama-variabel: nilai;
```

Nama setiap variabel harus diawali dengan tanda minus 2 kali, yakni " -- ". Sebagai contoh, jika kita ingin membuat variabel bernama --warna-utama dengan nilai #965e15, bisa menggunakan kode berikut:

```
1 .container {  
2   --warna-utama : #965e15;  
3   ...  
4 }
```

Kemudian untuk mengakses variabel, gunakan fungsi var(--nama-variabel), misalnya:

```
1 .container {  
2   --warna-utama : #965e15;  
3   border: 5px solid var(--warna-utama);  
4   ...  
5 }
```

Di baris 3, penulisan warna border menggunakan var(--warna-utama), sehingga secara tidak langsung akan berganti menjadi kode warna #965e15.

Berikut hasil modifikasi kode CSS dari contoh kita sebelumnya yang kali ini menggunakan CSS variabel:

02.css_variabel.html

```
1 <style>  
2   .container {  
3     --warna-utama : #965e15;  
4  
5     width: 400px;  
6     border: 5px solid var(--warna-utama);  
7     margin: 0 auto;  
8     padding: 1em;  
9     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
10    }  
11  h1 {  
12    margin-top: 0;  
13    text-align: center;  
14    color: var(--warna-utama);  
15  }  
16  span{  
17    color: var(--warna-utama);  
18  }  
19 </style>
```

Setelah variabel --warna-utama di definisikan di baris 3, maka semua selector yang berada di dalam container bisa mengakses variabel tersebut, termasuk di baris 14 dan 17.

Penulisan warna dengan variabel ini menjadikan kode lebih fleksibel. Jika nanti saya ingin mengubah warna menjadi #197419, cukup tukar di baris 3, maka semua nilai warna yang memakai variabel --warna-utama, akan ikut berubah.

Nilai yang diisi untuk variabel juga tidak hanya warna, tapi bisa juga nilai lain seperti pixel, em, dll, misalnya:

```
1 .container {  
2   --lebar-padding : 1em;  
3   padding: var(--lebar-padding);  
4   ...  
5 }
```

Selector :root

CSS variabel tetap mengikuti konsep "cascading", yakni hanya bisa diakses oleh property yang berada di dalam ruang lingkup selector yang mendefinisikannya.

Misalkan saya memiliki struktur HTML berikut:

```
1 <div class="container">  
2   <h1>Belajar CSS</h1>  
3   <p>Lorem ipsum dolor sit amet,...</p>  
4 </div>  
5 <h2>Belajar HTML</h2>
```

Jika sebuah variabel di definisikan dalam selector `.container`, maka selector `h2` tidak bisa mengakses nilai tersebut, karena sudah berada di luar selector `.container`.

Pada umumnya, kita ingin variabel ini bersifat global agar bisa diakses dari mana saja. Salah satu solusi adalah mendefinisikan variabel di dalam selector `html`, karena pastinya semua element ada di dalam tag `<html>`.

Namun terdapat selector khusus yang sering dipakai untuk tempat pendefinisian variabel, yakni pseudo selector `:root`. Berikut contoh penggunaannya:

03.variabel_root.html

```
1 <style>  
2   :root{  
3     --warna-utama : #197419;  
4     --lebar-padding : 1em;  
5   }  
6   .container {  
7     width: 400px;  
8     border: 5px solid var(--warna-utama);  
9     margin: 0 auto;  
10    padding: var(--lebar-padding);  
11    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
12  }  
13  h1 {  
14    margin-top: 0;  
15    text-align: center;  
16    color: var(--warna-utama);  
17  }  
18  span{
```

```
19      color: var(--warna-utama);  
20  }  
21 </style>
```

Dengan menuliskan semua variabel di dalam selector `:root`, maka variabel bisa diakses dari seluruh property.

Namun kita juga tidak harus mendefinisikan variabel di dalam `:root`, karena seperti praktek sebelumnya, CSS variabel bisa ditulis dari mana saja, termasuk selector `html` atau `body` yang dalam banyak kasus tidak akan berbeda dengan `:root` (tetapi bisa diakses secara global).

Salah satu alasan menggunakan `:root` adalah agar kita memiliki satu tempat khusus untuk mendefinisikan variabel saja, tidak bercampur dengan perintah lain.

Secara teknis, pseudo selector `:root` berada pada tingkat yang sama dengan selector `html`, yakni melingkupi semua struktur HTML. Namun karena CSS juga bisa dipakai untuk bahasa lain seperti XML atau SVG, maka `:root` bersifat lebih umum.

20.2. Fungsi calc()

CSS menyediakan fungsi `calc()` yang mengizinkan kita melakukan perhitungan nilai property. Perhitungan tersebut ditulis sebagai argumen di dalam tanda kurung. Berikut contoh penggunaannya:

```
width: calc(100px + 50px);
```

Ketika dijalankan, web browser akan memproses penambahan tersebut dan sama artinya dengan `width: 150px`.

Operasi yang di dukung adalah "kabataku", yakni penambahan " + ", pengurangan " - ", perkalian " * " dan pembagian " / ". Selain itu kita juga bisa menggabung berbagai nilai satuan, seperti:

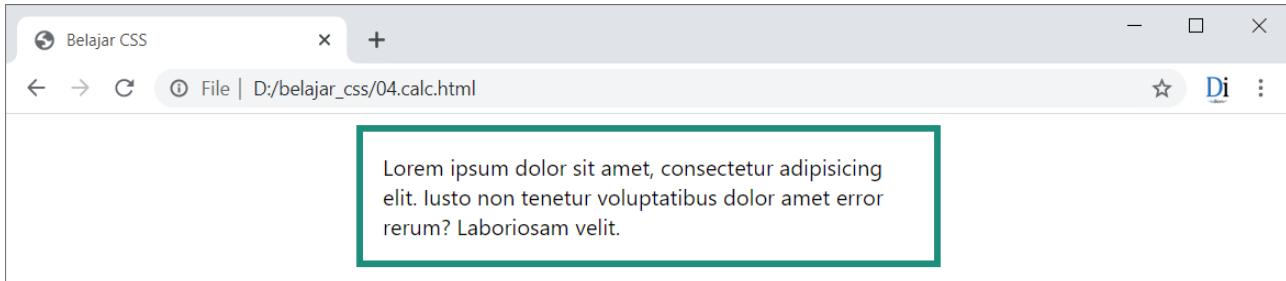
```
width: calc(100px * 2 + 40vw / 2);
```

Web browser secara otomatis akan menghitung semua nilai yang ada. Berikut contoh praktek dari fungsi `calc()`:

04.calc.html

```
1  <!DOCTYPE html>  
2  <html lang="id">  
3  <head>  
4    <meta charset="UTF-8">  
5    <title>Belajar CSS</title>  
6    <style>  
7      div {  
8        width: calc(100px * 2 + 40vw / 2);  
9        border: 5px solid rgb(25, 136, 117);  
10       margin: 0 auto;
```

```
11     padding: calc(30px - 1em);  
12     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;  
13   }  
14 </style>  
15 </head>  
16 <body>  
17   <div>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Iusto non  
18     tenetur voluptatibus dolor amet error rerum? Laboriosam velit. </div>  
19 </body>  
20 </html>
```



Gambar: Hasil penggunaan fungsi calc()

Pada contoh ini saya menggunakan fungsi `calc()` untuk menghitung nilai width di baris 8, serta menghitung nilai padding di baris 11. Fungsi `calc()` mungkin tidak terlalu sering kita pakai, tapi bisa menjadi solusi dalam situasi tertentu.

20.3. Property Filter

Fitur-fitur baru dari CSS3 sudah banyak yang bisa menggantikan peranan aplikasi pengolah gambar. Diantaranya kita bisa membuat efek bayangan dari `box-shadow`, membuat sudut melengkung dengan `border-radius`, mengubah bentuk element dengan `transform`, dan yang tidak kalah menarik adalah menambah efek gambar dengan property **filter**.

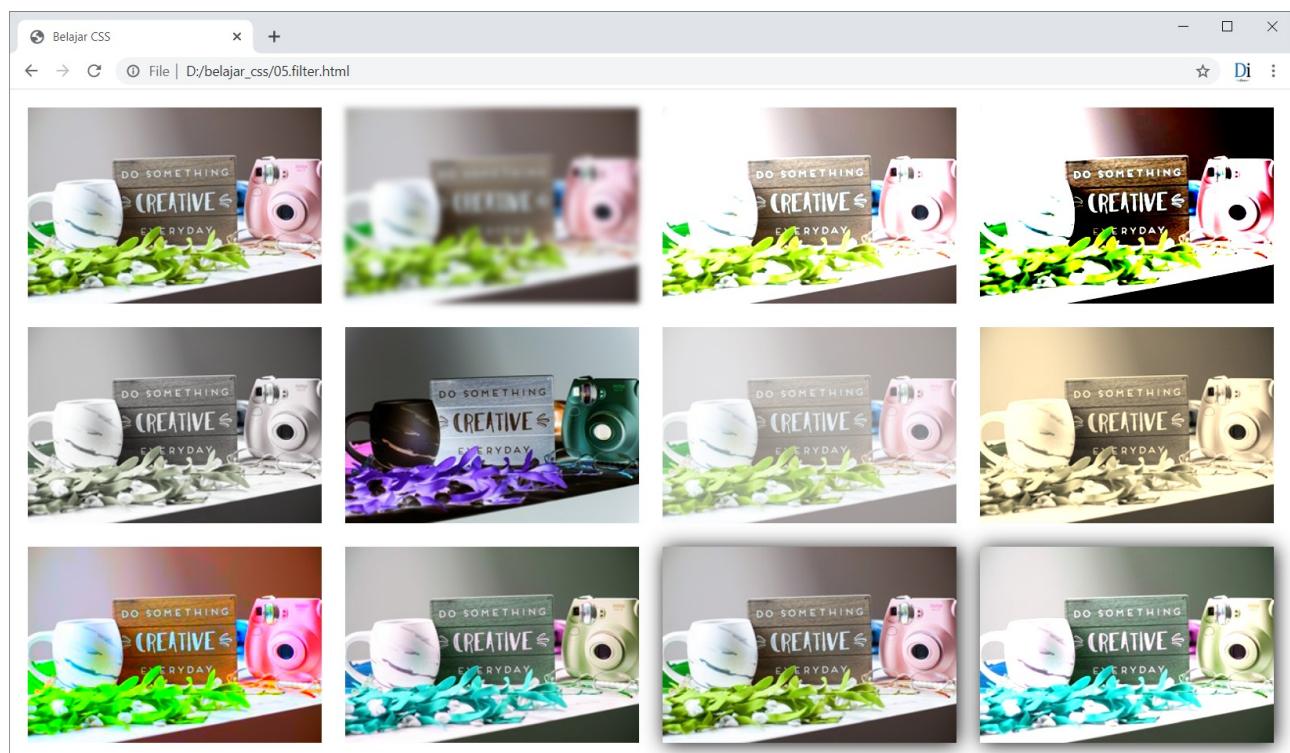
Bagi yang sering utak-atik foto, tentu tidak asing dengan istilah *blur*, *brightness*, *contrass*, hingga efek *sepia*. Semua ini juga bisa dilakukan dengan property CSS3 `filter`. Total terdapat 10 efek filter yang saat ini didukung, yakni:

- blur
- brightness
- contrast
- grayscale
- invert
- opacity
- sepia
- saturate
- hue-rotate
- drop-shadow

Umumnya property filter dipakai ke dalam gambar, yakni tag seperti contoh berikut:

05.filter.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          img { margin: 10px; }
8      </style>
9  </head>
10 <body>
11     
12     
13     
14     
15     
16     
17     
18     
19     
20     
21     
22     
23 </body>
24 </html>
```



Gambar: Berbagai efek dari penggunaan property filter

Gambar pertama di baris 11 tidak menggunakan efek filter apapun, sehingga bisa dipakai sebagai gambar pembanding.

Gambar kedua di baris 12 memiliki property `filter: blur(3px)`. Angka 3px di sini menjadi nilai input dari fungsi *gaussian blur*. Semakin tinggi nilainya, gambar akan semakin kabur. Jika ditulis `filter: blur(0px)`, artinya tidak ada efek blur sama sekali.

Gambar ketiga di baris 13 memiliki property `filter: brightness(150%)`. Brightness adalah efek untuk mengatur tingkat kecerahan gambar. Semakin tinggi nilainya, gambar akan semakin terang. Nilai input 150% berarti tingkat kecerahan gambar di-set menjadi 1,5 kali gambar asli.

Gambar keempat di baris 14 memiliki property `filter: contrast(350%)`. Contrast adalah efek untuk mengatur tingkat perbandingan warna. Semakin tinggi nilai contrast, gambar akan semakin tajam karena perbedaan warna menjadi lebih kuat. Nilai input 350% artinya tingkat kontras gambar di-set menjadi 3,5 kali gambar asli.

Gambar kelima di baris 15 memiliki property `filter: grayscale(85%)`. Grayscale adalah filter untuk "menghapus" warna gambar menjadi abu-abu. Nilai yang berefek antara 0% (gambar normal) hingga 100% (full grayscale). Ketika di set sebagai `grayscale(85%)`, kita masih bisa melihat sedikit warna.

Gambar keenam di baris 16 memiliki property `filter: invert(100%)`. Invert adalah filter untuk "membalik warna". Nilai yang berefek antara 0% (normal) hingga 100% (full invert).

Gambar ketujuh di baris 17 memiliki property `filter: opacity(50%)`. Opacity adalah filter untuk mengatur tingkat ke-transparan gambar. Nilai yang berefek antara 0% (gambar normal) hingga 100% (full transparan), atau bisa juga antara angka 0 hingga 1. Efek yang dihasilkan dari `filter: opacity()` sama persis seperti efek property `opacity`.

Gambar kedelapan di baris 18 memiliki property `filter: sepia(90%)`. Sepia adalah filter untuk membuat gambar "jadul". Nilai yang berefek antara 0% (normal) hingga 100% (full sepia).

Gambar kesembilan di baris 19 memiliki property `filter: saturate(500%)`. Saturate adalah filter untuk mengatur kekuatan warna. Penjelasan lebih lanjut tentang saturation pernah kita bahas di materi nilai warna hsl. Nilai `saturate(0%)` akan membuat efek warna abu-abu, `saturate(100%)` menjadi gambar normal, dan `saturate(500%)` untuk saturasi sebesar 5 kali.

Gambar kesepuluh di baris 20 memiliki property `filter: hue-rotate(90deg)`. Hue rotate adalah filter untuk mengatur color wheel. Penjelasan lebih lanjut tentang hue dan color wheel juga pernah kita bahas di materi nilai warna hsl. Nilai input yakni satuan derajat yang mencerminkan nilai putaran color wheel.

Gambar kesebelas di baris 21 memiliki property `filter: drop-shadow(2px 2px 10px)`. Filter ini dipakai untuk membuat efek bayangan. Nilai yang diisi sama persis seperti property box-shadow, yakni dalam format `drop-shadow: inset offset-x offset-y blur-radius spread`

color. Dalam contoh ini, property filter tersebut akan menghasilkan efek yang sama jika ditulis dengan property box-shadow: 2px 2px 10px.

Gambar terakhir di baris 22 – 23 memperlihatkan kalau kita bisa menambah banyak efek filter untuk satu gambar. Di antara setiap filter dipisah dengan karakter spasi.

Filter drop-shadow vs box-shadow

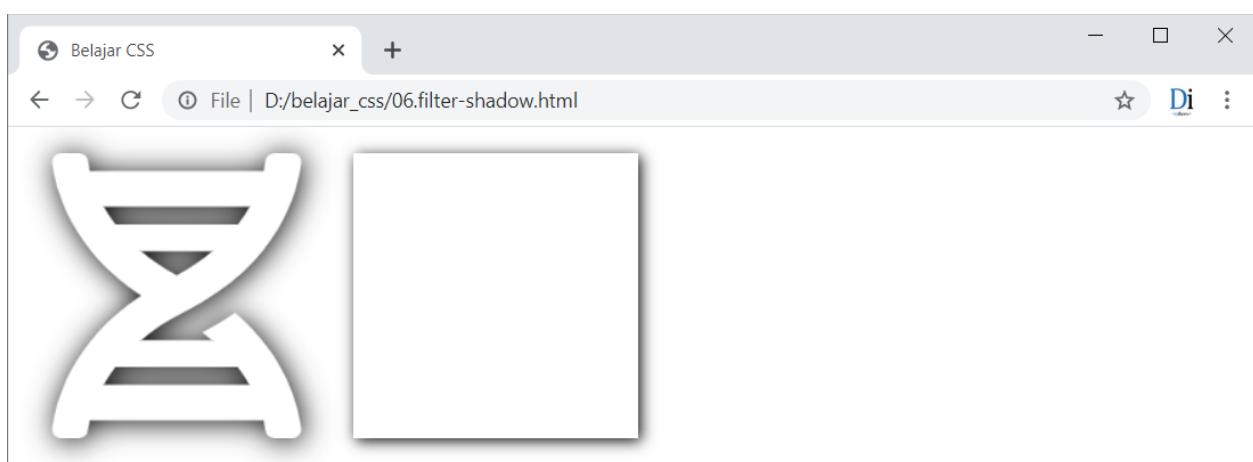
Dalam contoh sebelumnya saya menulis bahwa property filter: drop-shadow() memiliki efek yang sama dengan property box-shadow. Tapi sebenarnya ada perbedaan jika gambar yang dipakai mendukung efek transparasi atau dikenal juga dengan istilah *alpha channel*.

Tidak semua gambar mendukung *alpha channel*. Format gambar .jpg tidak mendukung fitur transparasi, tapi format gambar .png, .gif atau .svg mendukungnya. Biasanya gambar dengan *alpha channel* harus kita buat menggunakan aplikasi pengolah gambar seperti Photoshop atau tools tertentu.

Berikut contoh hasil perbedaan antara property filter: drop-shadow() dengan box-shadow untuk gambar yang memiliki *alpha channel*:

06.filter-shadow.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     img { margin: 10px; }
8   </style>
9 </head>
10 <body>
11   
12   
13 </body>
14 </html>
```



Gambar: Contoh perbedaan drop-shadow vs box-shadow

Untuk gambar dengan *alpha channel*, property `filter: drop-shadow()` membuat efek bayangan yang mengikuti alur gambar dna.png. Sedangkan property `box-shadow` hanya mengikuti bentuk box dari element HTML biasa.

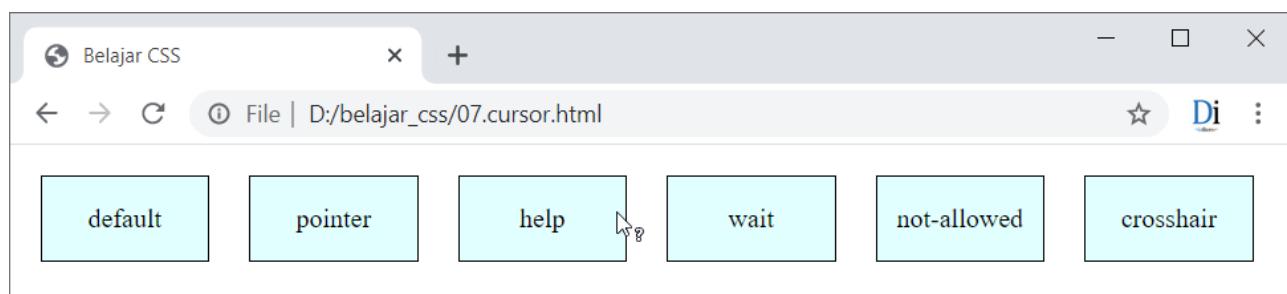
Fitur seperti ini bisa kita pakai untuk membuat berbagai efek menarik menggunakan CSS.

20.4. Property cursor

Property **cursor** berfungsi untuk mengubah tampilan cursor mouse saat berada di atas sebuah element. Nilai yang bisa diisi cukup beragam, diantaranya `default`, `pointer`, `help`, `wait`, `not-allowed` dan `crosshair`. Berikut contoh penggunaan dari property **cursor**:

07.cursor.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              display: inline-block;
9              text-align: center;
10             line-height: 50px;
11             border: 1px solid black;
12             background-color: lightcyan;
13             margin: 10px;
14             width: 100px;
15             height: 50px;
16         }
17     </style>
18 </head>
19 <body>
20     <div style="cursor: default;">default</div>
21     <div style="cursor: pointer;">pointer</div>
22     <div style="cursor: help;">help</div>
23     <div style="cursor: wait;">wait</div>
24     <div style="cursor: not-allowed;">not-allowed</div>
25     <div style="cursor: crosshair;">crosshair</div>
26 </body>
27 </html>
```



Gambar: Hasil penggunaan property cursor

Dalam kode ini saya membuat 6 tag <div> yang masing-masingnya ditambah property cursor. Agar bisa terlihat, arahkan cursor mouse ke atas setiap box, maka gambar icon cursor akan berubah.

Berikut daftar lengkap dari nilai property cursor CSS:

CSS Custom Cursors			
I auto	↗ move	↙ no-drop	↔ col-resize
↖ all-scroll	↘ pointer	ⓧ not-allowed	↑ row-resize
+ crosshair	↖ progress	↔ e-resize	↗ ne-resize
↖ default	I text	↑ n-resize	↖ nw-resize
↖? help	↖ vertical-text	↓ s-resize	↖ se-resize
I inherit	↖ wait	↔ w-resize	↗ sw-resize

Gambar: Berbagai nilai property cursor CSS (sumber: webcodegeeks.com)

Salah satu nilai yang sering dipakai adalah `cursor: pointer`, karena menampilkan gambar "tangan" untuk membuat efek "clickable". Misalnya jika kita sedang men-style sebuah tag <div> menjadi bentuk tombol, tag tersebut perlu tambahan property `cursor: pointer` agar pengunjung web bisa paham kalau itu bisa di-klik.

Beberapa element HTML sudah memiliki nilai cursor bawaan, misalnya tag <a> secara default tampil sebagai `cursor: pointer`.

20.5. Property resize

Property **resize** berfungsi untuk membuat element bisa di-resize atau diatur ukurannya menggunakan mouse. Nilai yang didukung adalah `horizontal`, `vertical` dan `both`.

Agar bisa efektif, property `resize` harus berpasangan dengan property `overflow` dengan nilai `auto`, `hidden`, atau `scroll`, tapi tidak dengan `overflow: visible`.

Berikut contoh penggunaan dari property `resize`:

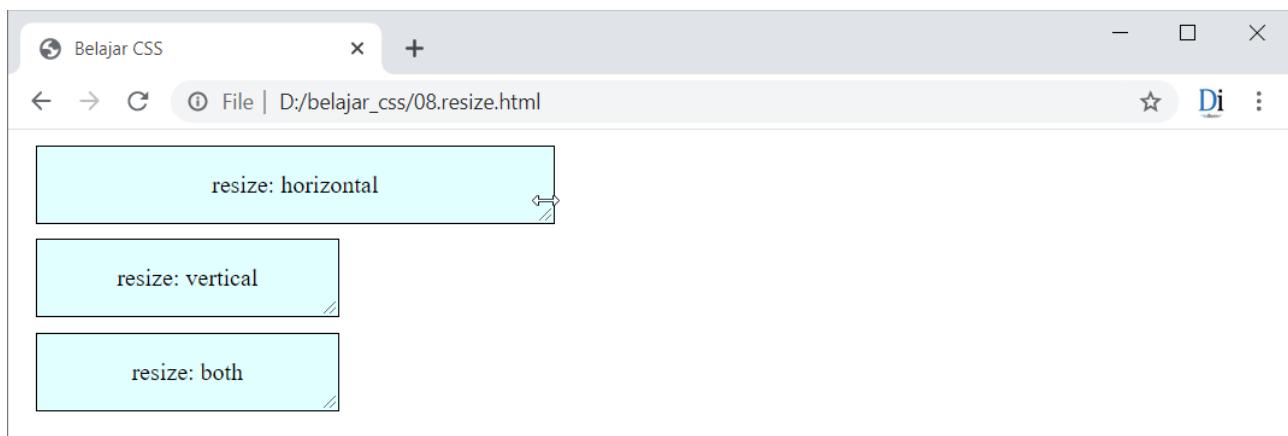
08.resize.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              text-align: center;
9              line-height: 50px;
10             border: 1px solid black;
```

```

11     background-color: lightcyan;
12     margin: 10px;
13     width: 200px;
14     height: 50px;
15     overflow: auto;
16   }
17 </style>
18 </head>
19 <body>
20   <div style="resize: horizontal">resize: horizontal</div>
21   <div style="resize: vertical">resize: vertical</div>
22   <div style="resize: both">resize: both</div>
23 </body>
24 </html>

```



Gambar: Hasil penggunaan property resize

Ketika ditambahkan property `resize`, di ujung kanan bawah element akan muncul garis penanda bahwa ukuran element bisa diperbesar. Sesuai dengan nilainya, `resize: horizontal` akan membatasi pembesaran secara horizontal saja (sumbu x), kemudian nilai `resize: vertical` membatasi pembesaran element secara vertikal (sumbu y), dan nilai `resize: both` mengizinkan pembesaran secara horizontal dan vertikal sekaligus.

Property `resize` ini relatif jarang dipakai, namun cukup berguna jika kita membuat efek-efek interaktif menggunakan JavaScript.

20.6. Property list-style

Di dalam HTML, tag `` dan `` dipakai untuk membuat list yang terdiri dari kumpulan tag ``. Secara default, setiap list memiliki nomor urut atau karakter penanda (*bullet*) di sisi kiri. Menggunakan CSS, tampilan penanda list ini bisa diatur dari 4 property berikut:

- `list-style-type`
- `list-style-image`
- `list-style-position`
- `list-style (shorthand)`

Property list-style-type

Property `list-style-type` berfungsi untuk mengatur jenis karakter *bullet*. Nilai yang bisa diisi cukup beragam dan akan berbeda tergantung apakah kita memakainya pada ordered list tag `` atau unordered list tag ``.

Untuk ordered list atau list terurut, `list-style-type` bisa diisi dengan salah satu nilai berikut:

- `decimal`
- `decimal-leading-zero`
- `lower-roman`
- `upper-roman`
- `lower-greek`
- `lower-latin`
- `upper-latin`
- `armenian`
- `georgian`
- `lower-alpha`
- `upper-alpha`
- `none`

Mari masuk ke contoh prakteknya:

09.list-style-type-ol.html

```
1 ...  
2 <style>  
3   ol { display: inline-block; }  
4 </style>  
5 ...  
6 <body>  
7   <ol style="list-style-type: decimal;">  
8     <li>List Item</li>  
9     <li>List Item</li>  
10    <li>List Item</li>  
11    <li>List Item</li>  
12  </ol>  
13  <ol style="list-style-type: decimal-leading-zero;">  
14    ...  
15  </ol>  
16  <ol style="list-style-type: lower-alpha;">  
17    ...  
18  </ol>  
19  <ol style="list-style-type: upper-alpha;">  
20    ...  
21  </ol>  
22  <ol style="list-style-type: lower-roman;">  
23    ...  
24  </ol>  
25 </body>
```

26 </html>



Gambar: Hasil penggunaan property list-style-type untuk ordered list

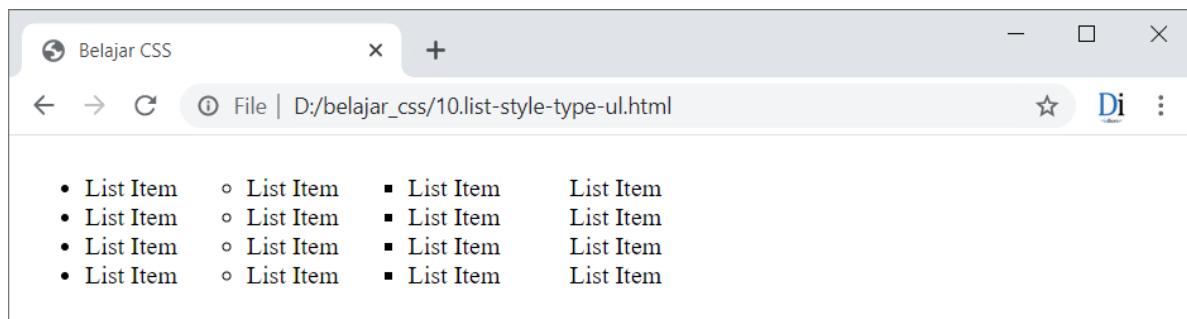
Sedangkan untuk unordered list atau list yang tidak terurut, bisa diisi salah satu nilai:

- disc
- circle
- square
- none

Berikut contoh prakteknya:

10.list-style-type-ul.html

```
1 <body>
2   <ul style="list-style-type: disc;">
3     <li>List Item</li>
4     <li>List Item</li>
5     <li>List Item</li>
6     <li>List Item</li>
7   </ul>
8   <ul style="list-style-type: circle;">
9     ...
10    ...
11   <ul style="list-style-type: square;">
12     ...
13   <ul style="list-style-type: none;">
14     ...
15   </ul>
16 </body>
```



Gambar: Hasil penggunaan property list-style-type untuk unordered list

Sebenarnya kita bisa memakai nilai `list-style-type` apa saja ke dalam tag `` maupun tag ``. Namun tentu paling ideal jika tag `` dipakai untuk list yang terurut dan tag `` untuk list yang tidak terurut.

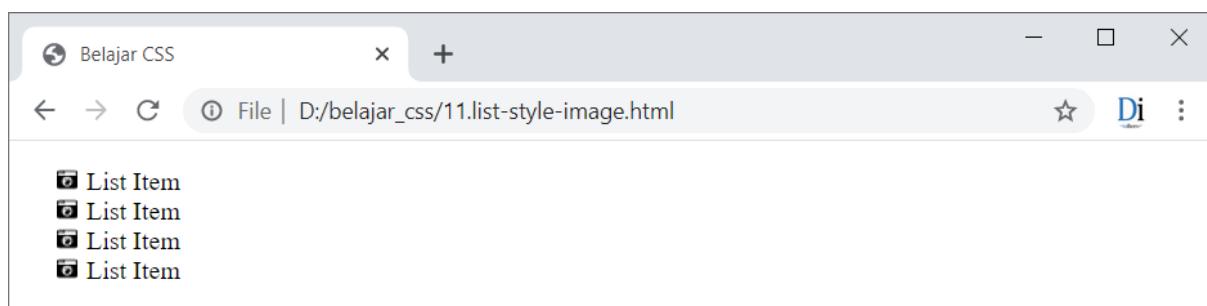
Nilai `list-style-type: none` akan menghapus karakter penanda list. Ini cukup sering dipakai saat membuat komponen menu navigasi.

Property `list-style-image`

Property `list-style-image` berfungsi untuk mengubah karakter penanda list menjadi gambar. Nilai dari property ini berupa alamat gambar dalam format `url(path/ke/gambar)`. Berikut contoh penggunaannya:

11.list-style-image.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6 </head>
7 <body>
8   <ul style="list-style-image:url(img/camera-retro-solid.png);">
9     <li>List Item</li>
10    <li>List Item</li>
11    <li>List Item</li>
12    <li>List Item</li>
13  </ul>
14 </body>
15 </html>
```



Gambar: Hasil penggunaan property `list-style-image`

Untuk contoh ini saya sudah menyiapkan gambar kecil bernama `camera-retro-solid.png` di folder `img`, sehingga ketika ditambahkan property `list-style-image:url(img/camera-retro-solid.png)`, maka karakter bullet akan diganti menjadi gambar ini.

Dalam prakteknya, menambah gambar list dengan cara ini kurang fleksibel. Ukuran gambar harus pas dengan tinggi font, jika font diperbesar maka juga harus memakai versi gambar yang lebih besar pula.

Untuk web modern, akan lebih praktis jika gambar list dibuat menggunakan font khusus icon

seperti **font awesome** (kita bahas pada bab selanjutnya).

Property list-style-position

Seperti yang bisa di tebak, property **list-style-position** dipakai untuk mengatur posisi **bullet**. Namun nilai yang didukung hanya ada 2, yakni **inside** dan **outside** (**default**). Berikut hasil perbedaan antara kedua nilai ini:

12.list-style-position.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     ul { display: inline-block; }
8     li { border: 1px solid black; }
9   </style>
10 </head>
11 <body>
12   <ul style="list-style-position: inside;">
13     <li>List Item</li>
14     <li>List Item</li>
15     <li>List Item</li>
16     <li>List Item</li>
17   </ul>
18   <ul style="list-style-position: outside;">
19     <li>List Item</li>
20     <li>List Item</li>
21     <li>List Item</li>
22     <li>List Item</li>
23   </ul>
24 </body>
25 </html>
```



Gambar: Hasil penggunaan property list-style-position

Pada contoh ini setiap list saya tambah property `border: 1px solid black` agar perbedaan antara nilai **inside** dan **outside** bisa terlihat lebih jelas. Nilai **inside** akan menempatkan bullet di dalam border, sedangkan nilai **outside** akan menempatkannya di luar border.

Property **list-style-position** relatif jarang dipakai karena jika kita ingin mengatur jarak

bullet dengan teks, akan lebih mudah dengan menambahkan padding-left ke dalam tag .

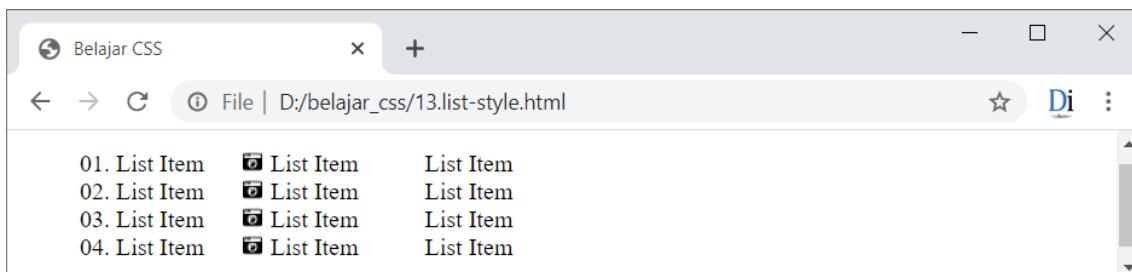
Property list-style

Property list-style adalah penulisan singkat (*shorthand*) dari gabungan list-style-type, list-style-image dan list-style-position. Property ini bisa diisi dengan 1, 2 atau 3 nilai dengan urutan bebas, sesuai dengan nilai yang bisa diisi ke dalam tiga property tersebut.

Berikut contoh penggunaan dari property list-style:

13.list-style.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          ol { display: inline-block; }
8      </style>
9  </head>
10 <body>
11     <ol style="list-style: decimal-leading-zero inside;">
12         <li>List Item</li>
13         <li>List Item</li>
14         <li>List Item</li>
15         <li>List Item</li>
16     </ol>
17     <ol style="list-style: url(img/camera-retro-solid.png) outside;">
18         ...
19     </ol>
20     <ol style="list-style: none;">
21         ...
22     </ol>
23 </body>
24 </html>
```



Gambar: Hasil penggunaan property list-style-position

Property list-style ini lebih sering dipakai dan memang lebih praktis daripada menulis 3 property list secara terpisah.

20.7. Property scroll-behavior

Property `scroll-behavior` berfungsi untuk mengatur proses pemindahan tampilan layar dari suatu link internal. Secara default, jika kita men-klik link internal, layar web browser akan "lompat" menuju tempat yang dituju. Dengan menambah property `scroll-behavior`, proses ini bisa dibuat menjadi efek `scrolling` yang berjalan secara perlahan.

Agar lebih mudah dipahami, kita akan lihat efek default dari link internal HTML terlebih dahulu:

14.normal_link_internal.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     div { height: 2000px; }
8   </style>
9 </head>
10 <body>
11   <h1 id="judul">Belajar CSS Uncover</h1>
12   <div></div>
13   <a href="#judul">Up</a>
14 </body>
15 </html>
```



Gambar: Efek default dari link internal

Saya membuat struktur halaman HTML yang terdiri dari sebuah judul `<h1 id="judul">`, satu tag `<div>` dan satu internal link yang dibuat dari `Up`.

Tinggi dari tag `<div>` di-set menjadi 2000px agar judul `<h1>` terpisah cukup jauh dengan link

dari tag <a>. Ini sebagai ilustrasi dari konten web yang cukup panjang. Ketika kita men-klik link "Up" di bagian bawah halaman, layar web browser langsung lompat ke bagian atas. Inilah yang dimaksud dengan efek default dari link internal.

Dalam HTML, nilai atribut href="#judul1" dari tag <a> dipakai untuk membuat link internal yang akan mencari sebuah element dengan id="judul1". Penjelasan lebih lanjut tentang link internal ini tersedia di buku **HTML Uncover**.

Dengan menambahkan property scroll-behavior: smooth ke dalam selector html, proses pemindahan akan menjadi lebih smooth dengan efek scrolling ke atas, tidak langsung lompat:

15.scroll-behavior.html

```
1 ...  
2 <style>  
3   html { scroll-behavior: smooth; }  
4   div { height: 2000px; }  
5 </style>  
6 ...
```

Efek yang dimaksud akan lebih jelas saat dijalankan langsung. Namun saya yakin anda sudah sering menemuinya di berbagai web, yakni icon tanda panah ke atas saat kita sampai di bagian bawah artikel.



Gambar: Tombol "Up" di web duniaIlkom

Selain nilai scroll-behavior: smooth, tersedia juga nilai scroll-behavior: auto yang merupakan nilai default (efeknya sama seperti jika property scroll-behavior tidak ditulis).

20.8. Property border-collapse

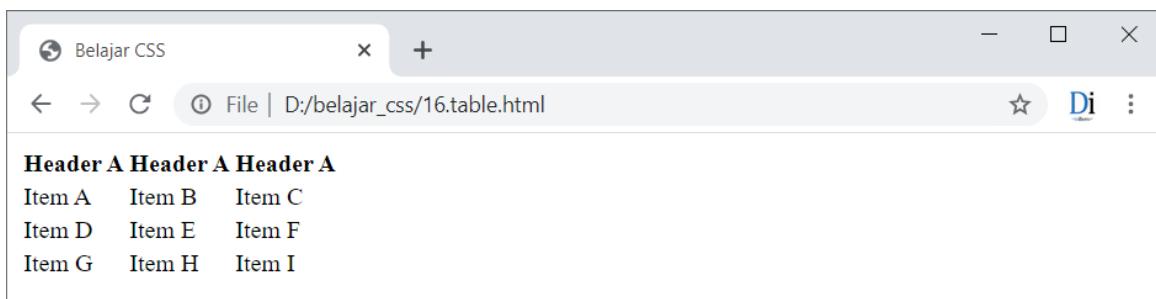
Property border-collapse berfungsi untuk menyatukan border yang saling bersentuhan. Ini umumnya dipakai ketika kita men-style tabel HTML menggunakan property border.

Bawaan web browser, tabel HTML ditampilkan tanpa bingkai seperti contoh berikut:

16.table.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6  </head>
7  <body>
8      <table>
9          <tr>
10         <th>Header A</th>
11         <th>Header A</th>
12         <th>Header A</th>
13     </tr>
14     <tr>
15         <td>Item A</td>
16         <td>Item B</td>
17         <td>Item C</td>
18     </tr>
19     <tr>
20         <td>Item D</td>
21         <td>Item E</td>
22         <td>Item F</td>
23     </tr>
24     <tr>
25         <td>Item G</td>
26         <td>Item H</td>
27         <td>Item I</td>
28     </tr>
29 </table>
30 </body>
31 </html>
```



Gambar: Tampilan default tabel HTML

Property border bisa digunakan untuk membuat efek bingkai. Akan tetapi hasilnya bisa berbeda tergantung selector yang dipakai. Jika kita menambah border ke selector `table` saja, hasilnya bingkai akan ada di sisi luar tabel:

17.table-border-1.html

```

1  table {
2      border: 1px solid black;
3  }
```

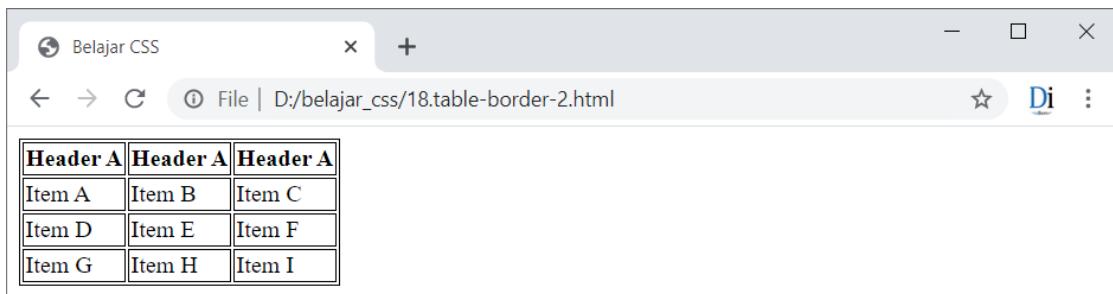


Gambar: Border pada selector table saja

Jika kita ingin menambah garis di antara setiap sel tabel, tambah property `border` ke dalam selector `td` dan `th`:

18.table-border-2.html

```
1 table, th, td {  
2   border: 1px solid black;  
3 }
```



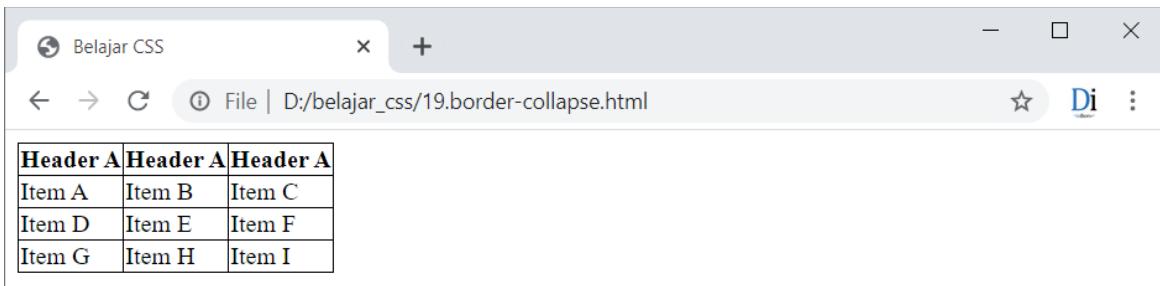
Gambar: Border pada selector table, th dan td

Sekarang semua sel tabel sudah memiliki border. Namun jika diperhatikan, terdapat 2 garis di antara setiap sel tabel. Ini terjadi karena `border-right` dari satu sel tabel berdempet dengan `border-left` dari sel tabel di sebelahnya. Begitu juga dengan `border-bottom` dari satu sel tabel akan berdempet dengan `border-top` dari sel di bawahnya.

Untuk beberapa situasi, tampilan seperti ini tidak masalah. Akan tetapi jika kita ingin menghapus border yang saling berdempet tersebut, bisa menambah property `border-collapse: collapse;`:

19.border-collapse.html

```
1 table, th, td {  
2   border: 1px solid black;  
3   border-collapse: collapse;  
4 }
```



Gambar: Efek penambahan property border-collapse

Hasilnya, border di antara setiap sel tabel dipaksa untuk menyatu atau *collapse*, menyisakan satu garis saja.

Nilai lain untuk property `border-collapse` adalah `separate`, yang akan memisahkan setiap border. Ini merupakan nilai default jika property `border-collapse` tidak ditulis.

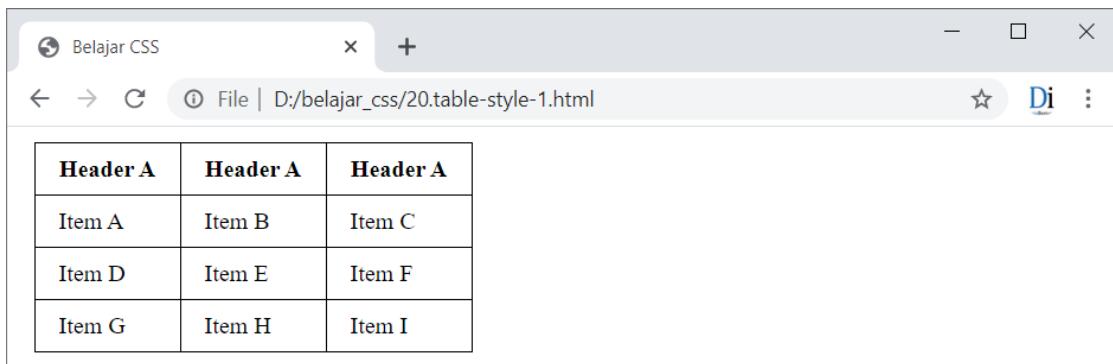
Merapikan Tampilan Tabel

Property `border-collapse` yang baru saja kita bahas setidaknya membuat tampilan tabel menjadi lebih rapi. Agar lebih menarik lagi, selector `table`, `th`, `td` dan `tr` bisa ditambah berbagai property box model terutama `padding` dan `margin`, misalnya sebagai berikut:

20.table-style-1.html

```

1  table {
2      margin: 10px;
3  }
4  table, th, td {
5      border: 1px solid black;
6      border-collapse: collapse;
7  }
8  th, td {
9      padding: 0.5em 1em;
10 }
```



Gambar: Merapikan tabel dengan margin dan padding

Property `margin: 10px` saya tambahkan ke dalam selector `table` agar tampilan tabel tidak terlalu menempel ke bingkai web browser. Selain itu penambahan `padding: 0.5em 1em` ke

dalam selector `th, td` membuat teks di dalam sel tabel juga tidak terlalu dengan dengan border.

Untuk hasil yang lebih menarik lagi, kita bisa membuat efek belang-belang (*zebra stripes*) menggunakan *pseudo class selector* `:nth-child`. Berikut contoh kode yang dibutuhkan:

21.table-style-2.html

```
1  table {
2    margin: 10px;
3    border-collapse: collapse;
4  }
5  th, td {
6    padding: 0.5em 1em;
7    border-bottom: 1px solid black;
8  }
9  th {
10   border-width: 3px;
11 }
12 tr:nth-child(even) {
13   background-color: lightgray;
14 }
```

The screenshot shows a web browser window titled "Belajar CSS". The address bar indicates the file is located at "D:/belajar_css/21.table-style-2.html". The browser interface includes standard controls like minimize, maximize, and close buttons, as well as a back/forward button, a refresh button, and a search/address bar. Below the header, there is a toolbar with icons for star, Di, and more. The main content area displays a table with three columns and three rows. The first row has a header with the text "Header A" repeated three times. The second row contains the text "Item A", "Item B", and "Item C". The third row contains "Item D", "Item E", and "Item F". The fourth row contains "Item G", "Item H", and "Item I". The rows alternate in background color, creating a zebra-striped effect where the even-numbered rows are light gray and the odd-numbered rows are white.

Header A	Header A	Header A
Item A	Item B	Item C
Item D	Item E	Item F
Item G	Item H	Item I

Gambar: Membuat efek zebra stripes

Efek belang-belang pada tabel didapat dari penggunaan selector `tr:nth-child(even)`. Selector ini akan cocok dengan setiap tag `<tr>` genap. Untuk semua tag `<tr>` tersebut, ubah warna `background-color` menjadi `lightgray`.

Selain itu saya juga berkreasi dengan border, yakni menggunakan `border-bottom` untuk tag `th` dan `td`. Hasilnya, tidak terdapat garis border sebagai pemisah kolom, hanya di antara setiap baris saja.

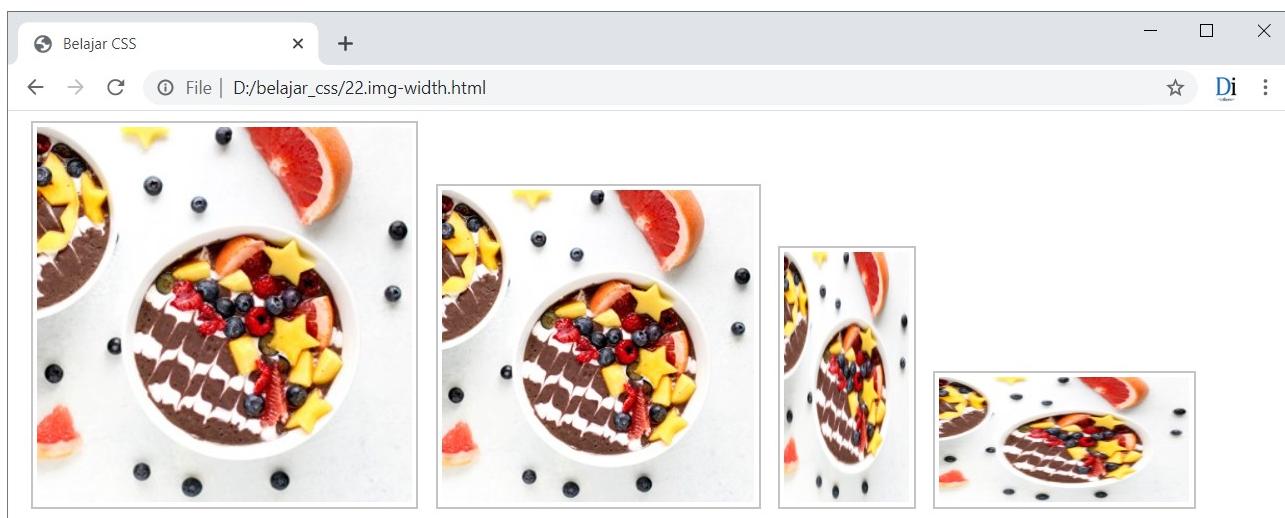
20.9. Property `object-fit`

Property `object-fit` berguna untuk mengatur apa yang terjadi ketika sebuah gambar dipaksa mengecil. Nilai yang bisa diisi adalah `fill`, `contain`, `cover`, `none`, dan `scale-down`.

Sebelum membahas perbedaan nilai-nilai ini, mari kita lihat efek default bawaan web browser:

22.img-width.html

```
1 <!DOCTYPE html>
2 <html lang="id">
3 <head>
4   <meta charset="UTF-8">
5   <title>Belajar CSS</title>
6   <style>
7     img {
8       margin: 10px;
9       border: 2px solid silver;
10      padding: 3px;
11    }
12  </style>
13 </head>
14 <body>
15   <div>
16     
17     
18     
19     
20   </div>
21 </body>
22 </html>
```



Gambar: Hasil pengecilan ukuran gambar

Dalam contoh ini saya menampilkan gambar `img/photo02.jpg` dengan berbagai ukuran. Gambar `photo02.jpg` memiliki dimensi asli 300×300 pixel yang bisa dilihat pada tag `` pertama (paling kiri).

Untuk gambar kedua di baris 17, terdapat tambahan property `width: 250px`. Jika sebuah gambar mendapat tambahan salah satu dari property `width` atau `height`, maka gambar tetap tampil proporsional. Dimensi yang tidak di-set akan di kalkulasi secara otomatis.

Akan tetapi jika property `width` dan `height` ditulis sekaligus, gambar akan dipaksa "meregang" (stretch) seperti pada gambar ketiga dan keempat. Property `object-fit` bisa dipakai dalam kasus seperti ini. Berikut contoh penggunaannya:

23.object-fit.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          img {
8              margin-left: 10px;
9              border: 2px solid silver;
10             padding: 3px;
11         }
12         .crop {
13             width: 100px;
14             height: 200px;
15         }
16     </style>
17 </head>
18 <body>
19     <div>
20         
21         
22         
23         
24         
25         
26     </div>
27 </body>
28 </html>
```



Gambar: Hasil penggunaan property `object-fit`

Gambar pertama (paling kiri) menampilkan gambar asli dengan dimensi 300 x 300 pixel. Sisa

gambar lain menggunakan class `.crop` dengan property `width: 100px` dan `height: 200px`. Kedua nilai ini akan memotong ukuran gambar asli menjadi tidak proporsional.

Property `object-fit: fill` pada gambar kedua akan memaksa gambar meregang untuk memenuhi dimensi baru. Ini merupakan nilai default jika property `object-fit` tidak ditulis.

Property `object-fit: contain` pada gambar ketiga akan berusaha menampilkan satu gambar utuh pada dimensi yang baru. Hasilnya, bisa akan terdapat ruang kosong karena tidak pas dengan proporsi gambar (sisi atas dan sisi bawah pada contoh ini).

Property `object-fit: cover` pada gambar keempat akan berusaha memenuhi dimensi baru tanpa meregangkan gambar. Hasilnya, ada kemungkinan bagian gambar yang tidak terlihat.

Property `object-fit: none` pada gambar kelima mirip seperti `cover`, namun tetap mempertahankan ukuran gambar asli.

Property `object-fit: scale-down` pada gambar keenam akan menghasilkan efek antara nilai `contain` atau `none`, tergantung mana ukuran yang paling kecil.

Dari semua property `object-fit` ini, nilai `cover` cukup sering dipakai untuk menghindari efek `stretch`, selama kita tidak masalah dengan sedikit bagian gambar yang tidak kelihatan.

20.10. Property clip-path

Property **clip-path** berfungsi untuk memodifikasi bentuk element HTML. Efek yang dihasilkan sangat menarik karena kita bisa mengubah dimensi element menjadi lingkaran, segitiga, serta berbagai bentuk lain yang lebih kompleks.

Nilai untuk property `clip-path` berupa fungsi, yakni `circle()`, `ellipse()`, `inset()`, dan `polygon()`.

Nilai `clip-path: circle()`

Sesuai dengan namanya, property `clip-path: circle()` dipakai untuk membuat bentuk lingkaran. Fungsi `circle()` butuh 1 nilai input berupa besar jari-jari. Selain itu kita juga bisa menentukan titik koordinat lingkaran yang secara default berada di tengah-tengah element.

Berikut contoh penggunaan dari nilai `clip-path circle()`:

24.clip-path-circle.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 200px;
```

```
9      height: 200px;
10     margin: 10px;
11     background-color: aquamarine;
12     display: inline-block;
13   }
14 </style>
15 </head>
16 <body>
17   <div></div>
18   <div style="clip-path: circle(50%)"></div>
19   <div style="clip-path: circle(50px)"></div>
20   <div style="clip-path: circle(100px at top)"></div>
21   <div style="clip-path: circle(200px at 0 0)"></div>
22 </body>
23 </html>
```



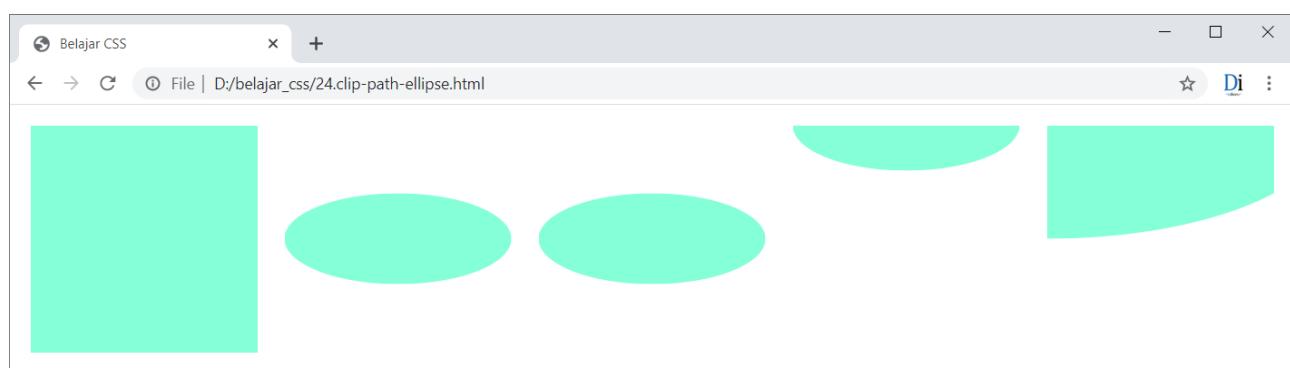
Nilai clip-path: ellipse()

Nilai `clip-path: ellipse()` dipakai untuk membuat bentuk elips. Nilai input fungsi `ellipse()` berupa dua buah nilai yang dipisah dengan tanda spasi. Nilai ini mewakili jari-jari elips untuk sumbu x dan sumbu y. Selain itu kita juga bisa menentukan titik pusat elips.

Berikut contoh penggunaan nilai `clip-path ellipse()`:

24.clip-path-ellipse.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 200px;
9              height: 200px;
10             margin: 10px;
11             background-color: aquamarine;
12             display: inline-block;
13         }
14     </style>
15 </head>
16 <body>
17     <div></div>
18     <div style="clip-path: ellipse(50% 20%)"></div>
19     <div style="clip-path: ellipse(100px 40px)"></div>
20     <div style="clip-path: ellipse(100px 40px at top)"></div>
21     <div style="clip-path: ellipse(250px 100px at 0 0)"></div>
22 </body>
23 </html>
```



Gambar: Hasil dari `clip-path: ellipse()`

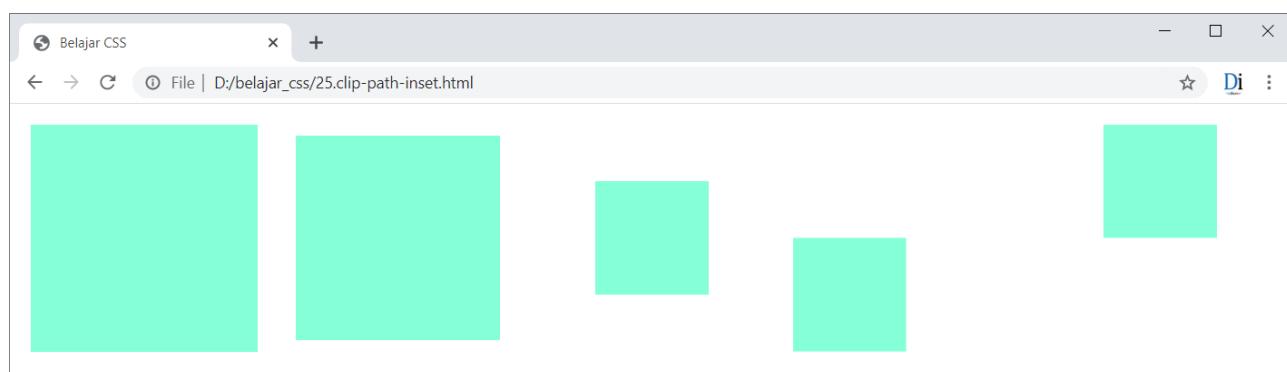
Secara umum, penulisan perintah `ellipse()` mirip seperti `circle()`, hanya saja kita perlu menuliskan 2 ukuran jari-jari. Titik pusat elips juga bisa diubah menggunakan keyword maupun titik koordinat.

Nilai clip-path: inset()

Nilai `clip-path: ellipse()` dipakai untuk membuat bentuk persegi dengan cara memotong sisi dalam element. Nilai yang bisa diisi berupa jarak untuk ke-4 sisi element. Pengertian ini akan kita bahas dengan contoh kode berikut:

25.clip-path-inset.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 200px;
9              height: 200px;
10             margin: 10px;
11             background-color: aquamarine;
12             display: inline-block;
13         }
14     </style>
15 </head>
16 <body>
17     <div></div>
18     <div style="clip-path: inset(10px 10px 10px 10px);"></div>
19     <div style="clip-path: inset(50px 50px 50px 50px)"></div>
20     <div style="clip-path: inset(100px 100px 0 0)"></div>
21     <div style="clip-path: inset(0 50px 100px 50px)"></div>
22 </body>
23 </html>
```



Gambar: Hasil dari `clip-path: inset()`

Untuk tag `<div>` kedua, terdapat tambahan property `clip-path: inset(10px 10px 10px 10px)`. Property ini bisa dibaca: potong element dengan jarak 10px dari sisi atas, 10px dari sisi kanan, 10px dari sisi bawah, dan 10px dari sisi kanan. Hasilnya, kotak menjadi sedikit lebih kecil dari tag `<div>` pertama.

Prinsip yang sama juga diterapkan untuk property inset untuk tag `<div>` ketiga, keempat dan kelima. Urutan sisi sama seperti penulisan border atau padding, yakni searah sumbu jam atau

TRouBLE (Top, Right, Bottom, dan Left).

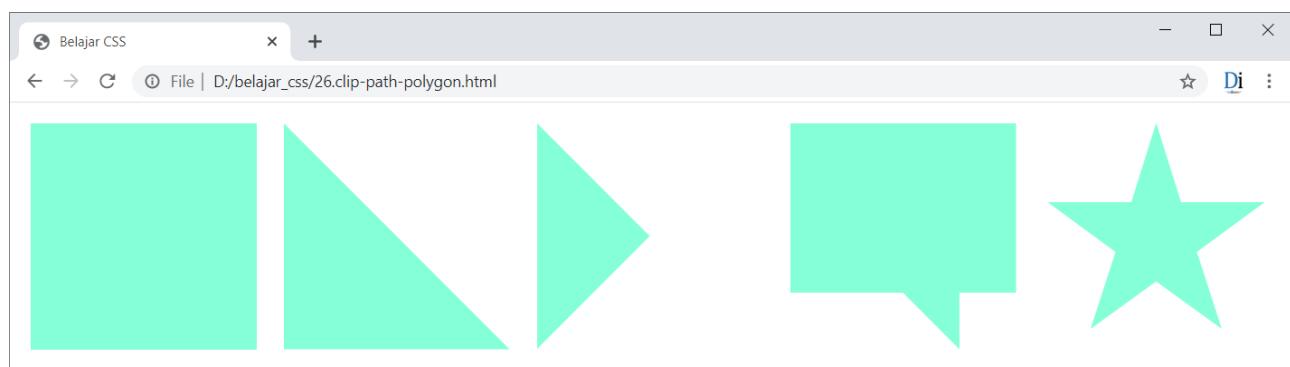
Nilai clip-path: polygon()

Nilai `clip-path: polygon()` menjadi yang paling kompleks sekaligus sangat menarik. Fungsi `polygon()` mengizinkan kita memotong element dalam bentuk apa saja. Pemotongan dilakukan dengan cara menulis pasangan titik koordinat dalam bentuk `x1 y1, x2 y2, x3 y3, ... dst.` Di antara setiap titik koordinat akan ditarik garis lurus sampai membentuk sebuah gambar 2 dimensi.

Berikut contoh penggunaan nilai `polygon()` untuk property `clip-path`:

26.clip-path-polygon.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          div {
8              width: 200px;
9              height: 200px;
10             margin: 10px;
11             background-color: aquamarine;
12             display: inline-block;
13         }
14     </style>
15 </head>
16 <body>
17     <div></div>
18     <div style="clip-path: polygon(0 0, 100% 100%, 0 100%);"></div>
19     <div style="clip-path: polygon(0 0, 100px 100px, 0 200px)"></div>
20     <div style="clip-path: polygon(0% 0%, 100% 0%, 100% 75%, 75% 75%, 75% 100%, 50% 75%, 0% 75%)"></div>
21     <div style="clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%, 50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%)"></div>
22   </body>
23 </html>
```



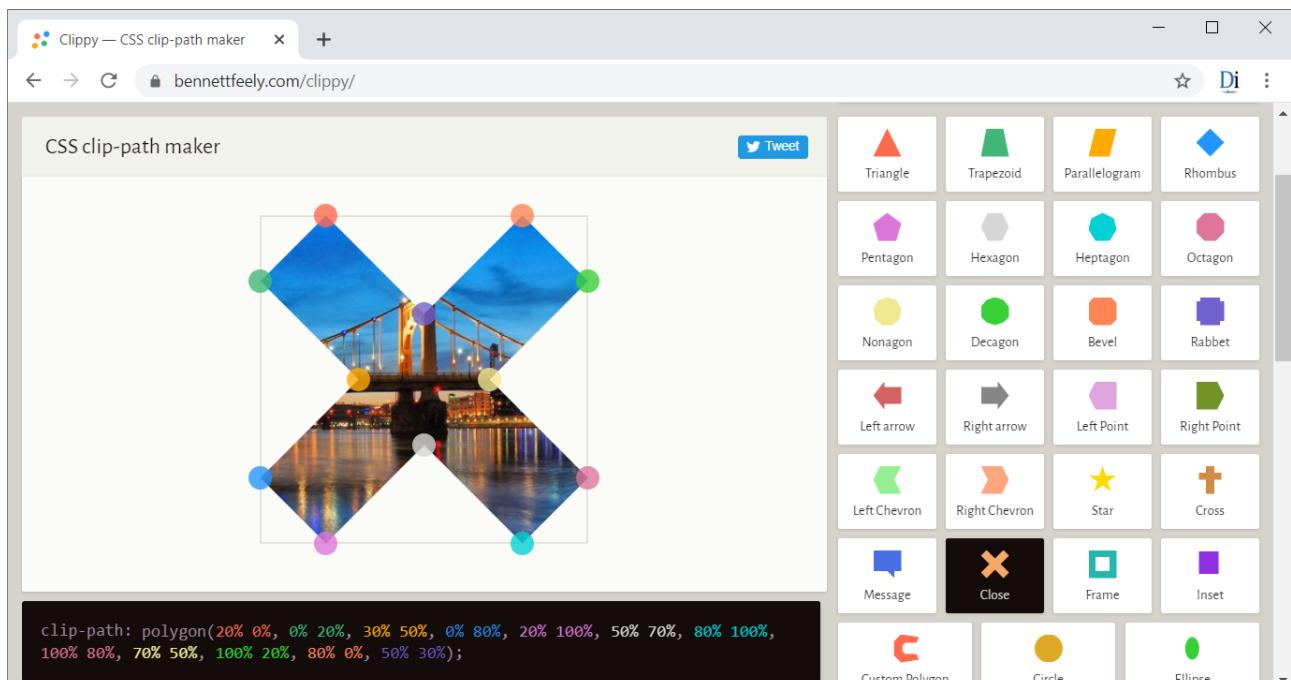
Gambar: Contoh penggunaan `clip-path: polygon()`

Untuk tag `<div>` kedua, terdapat tambahan property `clip-path: polygon(0 0, 100% 100%, 0 100%)`. Property ini bisa dibaca: potong element mulai dari koordinat 0 0 (sudut kiri atas), lalu lanjut ke koordinat 100% 100% di sudut kanan bawah, dan terakhir koordinat 0 100% di sudut kiri bawah. Setelah sampai di koordinat terakhir, garis akan ditarik kembali ke koordinat pertama.

Menggunakan satuan persen untuk nilai `polygon()` terasa lebih mudah karena kita bisa langsung menebak kira-kira dimana posisi titik tersebut. Akan tetapi jika bisa menulis koordinat dengan satuan pixel seperti tag `<div>` ketiga.

Koordinat yang bisa ditulis tidak dibatasi, sehingga kita bisa "menggambar" pemotongan element menjadi bentuk yang lebih kompleks seperti karakter bintang pada gambar terakhir.

Untuk memudahkan penulisan titik koordinat `polygon()`, kita bisa menggunakan tools online yang salah satunya bisa ke bennettfeely.com/clippy.



Gambar: Membuat nilai polygon dari bennettfeely.com/clippy

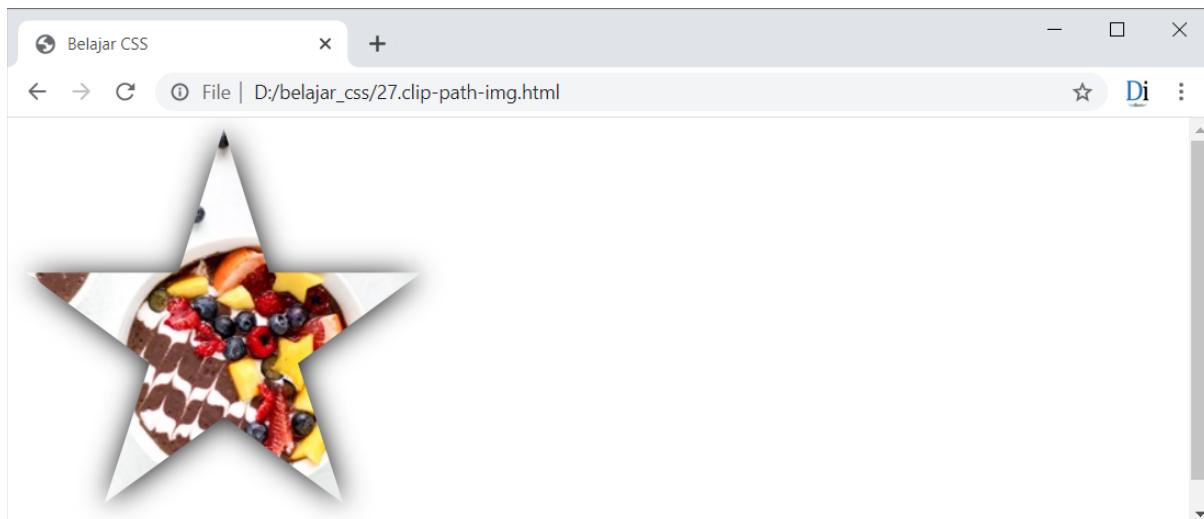
Dalam halaman ini tersedia berbagai bentuk polygon. Jika ingin dipakai, tinggal copy paste property `clip-path` yang tersedia di bagian bawah. Dua polygon terakhir dari contoh sebelum ini juga saya ambil dari web bennettfeely.com/clippy. Selain itu kita juga bisa membuat bentuk polygon baru dengan menggeser titik koordinat yang ada.

Memakai `clip-path` Untuk Gambar

Property `clip-path` juga bisa dipakai untuk memotong gambar. Caranya, tempatkan `clip-path` ke dalam tag `` seperti contoh berikut:

27.clip-path-img.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          img {
8              clip-path: polygon(50% 0%, 61% 35%, 98% 35%, 68% 57%, 79% 91%,
9                  50% 70%, 21% 91%, 32% 57%, 2% 35%, 39% 35%);
10         }
11         div {
12             filter: drop-shadow(1px 1px 10px);
13         }
14     </style>
15 </head>
16 <body>
17     <div>
18         
19     </div>
20 </body>
21 </html>
```



Gambar: Hasil penggunaan property clip-path ke dalam gambar

Di sini saya menggabung clip-path polygon dengan filter drop-shadow. Efek yang dihasilkan sangat menarik, gambar terpotong sesuai dengan nilai clip-path, serta bayangan drop-shadow berada di belakang bentuk polygon.

Agar filter drop-shadow berada di belakang polygon, property `filter` harus ditempatkan ke parent element, yang dalam hal ini ke dalam tag `<div>`, bukan ke tag ``.

Property `clip-path` memang relatif baru, sebelumnya property ini bernama "`clip`" namun sudah berstatus *deprecated* (usang).

20.11. Property user-select

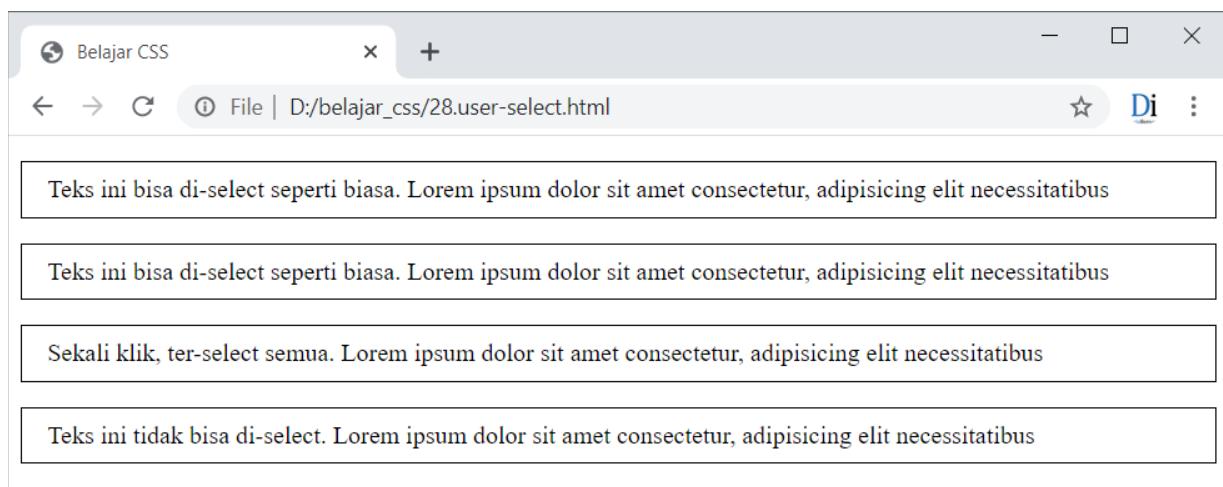
Property **user-select** berfungsi untuk mengatur cara user men-seleksi konten yang ada di dalam sebuah element. Nilai yang bisa diisi antara lain: **auto** (*default*), **text**, **all** dan **none**.

Berikut contoh penggunaannya:

28.user-select.html

```

1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <style>
7          p {
8              border: 1px solid black;
9              padding: 0.5rem 1rem;
10         }
11     </style>
12 </head>
13 <body>
14     <p style="user-select: auto;">Teks ini bisa di-select seperti biasa.
15         Lorem ipsum dolor sit amet consectetur, adipisicing elit necessitatibus</p>
16     <p style="user-select: text;">Teks ini bisa di-select seperti biasa.
17         Lorem ipsum dolor sit amet consectetur, adipisicing elit necessitatibus</p>
18     <p style="user-select: all;">Sekali klik, ter-select semua.
19         Lorem ipsum dolor sit amet consectetur, adipisicing elit necessitatibus</p>
20     <p style="user-select: none;">Teks ini tidak bisa di-select.
21         Lorem ipsum dolor sit amet consectetur, adipisicing elit necessitatibus</p>
22 </body>
23 </html>
```



Gambar: Hasil penggunaan property user-select

Paragraf pertama memiliki property **user-select: auto**. Ini merupakan nilai default dimana user bisa menyeleksi konten seperti biasa. Pengecualian adalah untuk konten hasil *pseudo selector* `::before` dan `::after` yang secara default memang tidak bisa di-select.

Paragraf kedua memiliki property `user-select: text`. Ini berarti user hanya bisa men-select konten teks yang ada di dalam paragraf. Namun dalam prakteknya, saya tetap bisa men-select konten lain seperti gambar dari tag ``.

Paragraf ketiga memiliki property `user-select: all`. Nilai `all` ini cukup unik, dimana begitu teks di klik, akan langsung men-select semua paragraf. Fitur ini sering kita temukan pada web yang menyediakan kode untuk di copy.

Paragraf terakhir dengan property `user-select: none` akan mencegah user untuk men-select konten yang ada. Efek ini bisa dipakai jika kita tidak ingin teks di copy secara langsung. Namun dengan sedikit "kreatifitas", seseorang tetap bisa mencopy teks memakai `inspect element` bawaan developer tools atau mencopynya dari source code HTML.

Dalam bab ini kita sudah membahas berbagai property dan fitur CSS yang sayang untuk dilewatkan. Beberapa property memang masih relatif baru seperti CSS `variable`, `filter`, `object-fit` dan `clip-path`, namun seharusnya sudah bisa berjalan di mayoritas web browser.

Berikutnya kita akan bahas cara menambah logo icon ke dalam halaman HTML menggunakan font awesome.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

=====

21. Font Awesome

Ketika masuk ke perancangan template, kadang kita butuh gambar icon sebagai pengganti teks. Salah satu cara untuk membuatnya adalah menggunakan **font awesome**. Dalam bab ini akan dibahas tentang apa itu font awesome dan bagaimana cara menggunakannya.

Font awesome bukanlah bagian dari CSS, tapi karena cukup sering dipakai, saya ingin membahasnya di buku CSS Uncover ini.

21.1. Pengertian Font Awesome

Font awesome (fontawesome.com) adalah font khusus yang berisi kumpulan icon dan logo. Icon ini bisa kita pakai untuk mempercantik tampilan web. Total terdapat ribuan icon yang tersedia di dalam *font awesome*, mulai dari icon checklist, icon cart, hingga logo populer seperti icon facebook dan twitter.

Meskipun ada alternatif lain, *font awesome* sangat populer dan menjadi "standar tidak resmi" dari sebuah font icon. Secara berkala *font awesome* terus di update dan memperbanyak jumlah icon. Saat buku ini ditulis, terdapat sekitar 1609 icon gratis di dalam *font awesome* versi 5.15 dan jumlahnya terus bertambah.

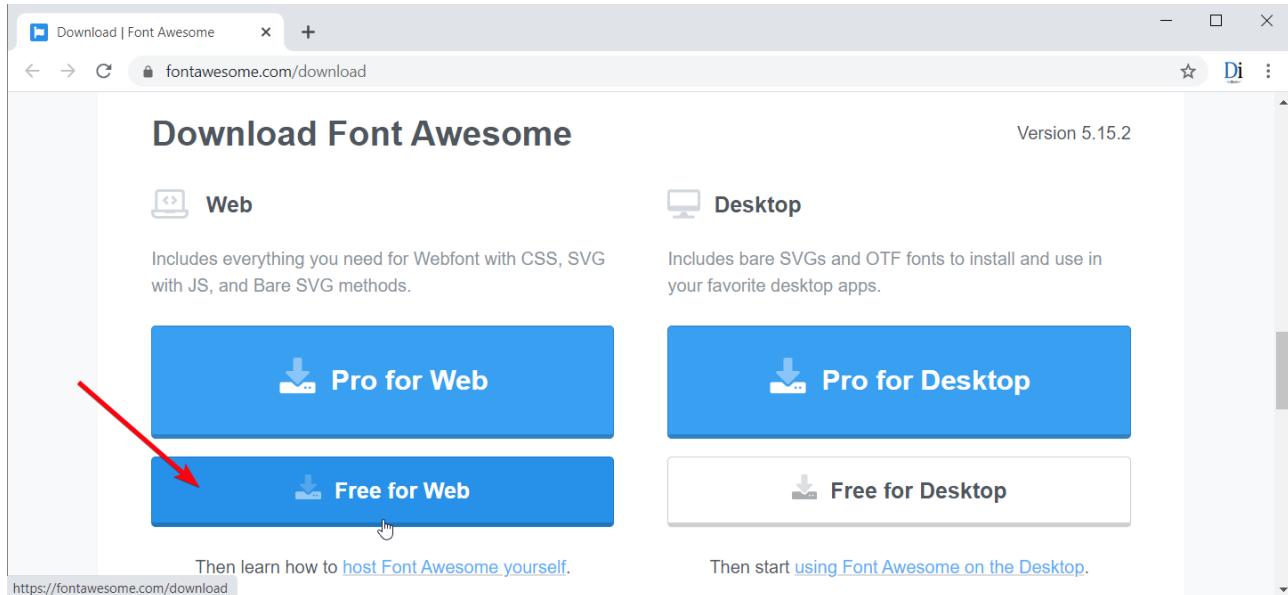
Mulai versi 5, *font awesome* hadir dalam versi gratis dan versi berbayar (sekitar US \$60 per tahun). Namun versi gratisnya sudah sangat lengkap, lebih dari cukup untuk keperluan pembuatan web standar.

21.2. Mendownload Font Awesome

Agar bisa menggunakan font awesome, kita harus menghubungkannya dengan file HTML saat ini. Caranya, download file *font awesome*, simpan ke dalam sebuah folder lalu akses menggunakan tag <link> sebagai external CSS.

File font awesome bisa di download dari fontawesome.com/download²⁹. Silahkan buka link tersebut lalu klik tombol "**Free for Web**" di bagian "Download Font Awesome" seperti gambar berikut:

²⁹ <https://fontawesome.com/download>



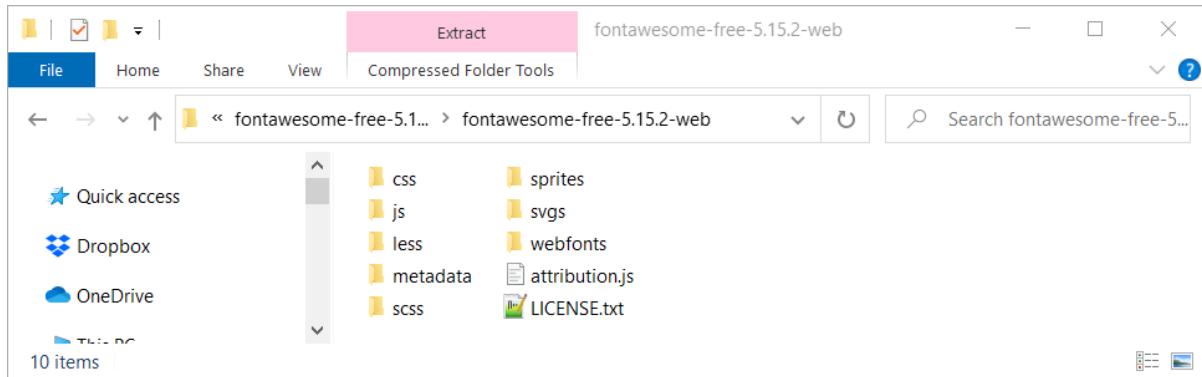
Gambar: Tampilan halaman home website font awesome

Proses download akan berlangsung beberapa saat. File yang akan didapat nanti adalah `fontawesome-free-5.15.2-web.zip` yang berukuran sekitar 5,2 MB. Besar kemungkinan versi yang akan anda dapat lebih baru dari ini.

Tampilan web font awesome akan berubah dari waktu ke waktu terutama jika terdapat update. Apabila anda tidak menemukan link download atau ada perubahan versi, saya juga menyiapkan file `fontawesome-free-5.15.2-web.zip` di **Folder Aplikasi dan Installer Google Drive** (link yang di sertakan pada saat pembelian eBook ini).

21.3. Menghubungkan File Font Awesome

Silahkan double klik file `fontawesome-free-5.15.2-web.zip` atau buka menggunakan aplikasi unzip. Dalam file ini terdapat folder lagi yang bernama `fontawesome-free-5.15.2-web.zip`, kembali double klik untuk melihat isinya:

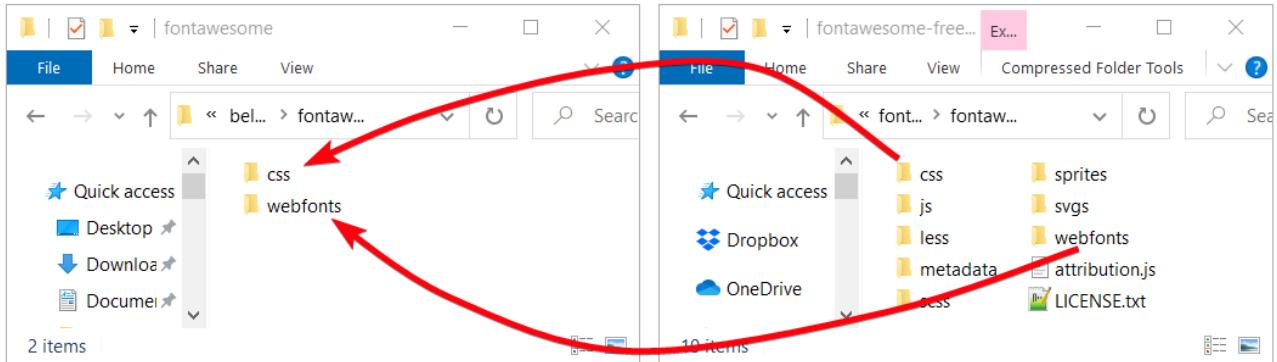


Gambar: dari file `fontawesome-free-5.15.2-web.zip`

Font Awesome

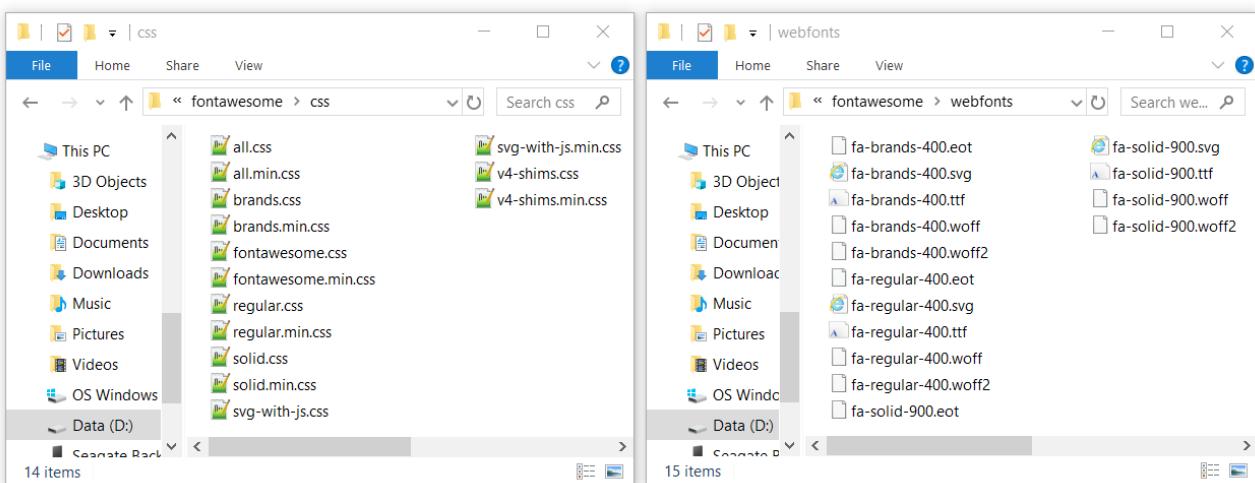
Terdapat 8 folder dan 2 file teks. Dari semua file ini yang kita perlukan hanya 2 folder, yakni folder **css** dan folder **webfonts**.

Sebelum itu, buka folder latihan kita, yakni **belajar_css** (atau folder lain yang anda pakai), lalu buat folder baru bernama **fontawesome**. Kemudian extract dan copy folder **css** dan folder **webfonts** dari **fontawesome-free-5.15.2-web.zip** ke folder **fontawesome** ini:



Gambar: Copy folder css dan webfonts ke dalam folder fontawesome

Hasil akhirnya folder **fontawesome** akan berisi 2 buah folder: **css** dan **webfonts**. Berikut isi kedua folder tersebut:



Isi folder css (kiri) dan webfonts (kanan) bawaan dari font awesome

Folder **css** dari font awesome berisi berbagai file CSS yang nantinya harus kita akses dari file HTML. Sedangkan folder **webfonts** berisi file font yang dipakai internal oleh *font awesome*.

File CSS dari *font awesome* terdiri dari 2 variasi: versi normal dan versi *minified*. Versi *minified* memiliki tambahan **.min** di belakang nama file. Versi *minified* berukuran lebih kecil karena di *compress* dengan cara menghilangkan seluruh spasi dan baris komentar.

Semua file CSS di folder **fontawesome/css** berisi file dengan pembagian yang berbeda-beda. Jika hanya ingin memakai icon logo produk seperti *facebook* atau *twitter* saja, maka cukup akses file **brand.css**. Jika ingin menggunakan icon biasa, bisa akses file **regular.css** dan

`solid.css`. Namun jika ingin keduanya (berisi seluruh icon), akses file `all.css`.

Dalam struktur folder `belajar_css` yang sudah kita siapkan, file `all.css` ini berada di alamat `fontawesome/css/all.css`. Sehingga tag `<link>` yang akan dipakai adalah:

```
<link href="fontawesome/css/all.css" rel="stylesheet">
```

Atau untuk versi *minified*:

```
<link href="fontawesome/css/all.min.css" rel="stylesheet">
```

Jika anda memiliki struktur folder berbeda, tinggal sesuaikan alamat *path* untuk atribut `href`. Dengan demikian, berikut kode HTML yang berisi link ke file CSS **font awesome**:

01.font-awesome_link.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <link href="fontawesome/css/all.min.css" rel="stylesheet">
7      <style>
8          /* Kode CSS disini... */
9      </style>
10     </head>
11     <body>
12         <!-- Kode HTML disini -->
13     </body>
14 </html>
```

Tambahannya ada di baris 6, yakni tag `<link>` untuk mengakses file `fontawesome/css/all.min.css`.

File Font Awesome di CDN

Cara akses alternatif file *font awesome* adalah dari CDN (Content Delivery Network). Salah satunya disediakan oleh `cloudflare.com`:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.2/css/all.min.css">
```

Cara ini memang lebih praktis karena kita tidak perlu repot mendownload file *font awesome*. Tapi agar bisa berjalan, harus selalu terhubung ke internet.

21.4. Cara Penggunaan Font Awesome

Setelah file **font awesome** bisa diakses (baik yang di download maupun dari CDN) kita bisa masuk ke cara penggunaan font awesome, yakni dengan menulis beberapa class khusus ke dalam tag `<i>` atau tag ``.

Sebagai contoh, untuk menampilkan icon rumah (home), bisa menggunakan kode berikut:

```
<i class="fas fa-home"></i>
```

Di sini terdapat 2 buah penulisan class, yakni .fas dan .fa-home. Penulisan ini merupakan aturan dari **font awesome versi 5** yang sedikit berbeda dengan versi font awesome 4.

Untuk font awesome 5, format dasar penulisan nama class adalah sebagai berikut:

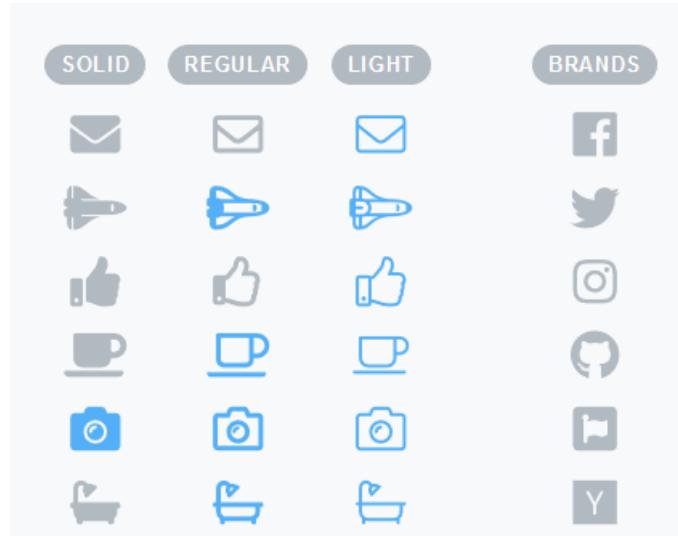
```
<i class="jenis_kelompok_icon fa-<kode_icon>"></i>
```

Tag yang dipakai untuk membuat icon font awesome tidak harus `<i>`, tapi juga bisa inline element lain seperti ``. Namun tag `<i>` adalah yang paling banyak dipakai, termasuk di dokumentasi resmi font awesome.

Selain class .fas, terdapat juga class .fab, .far, dan .fal sebagai jenis kelompok icon. Berikut perbedaan dari keempat class ini:

Nama Jenis Icon	Nama class	Contoh	Ketersediaan
Brands	fab	fab fa-facebook	Free
Solid	fas atau fa	fas fa-home	Free
Regular	far	far fa-check-square	Free / Pro
Light	fal	fal fa-address-card	Pro

Dalam tabel di atas terlihat ada 4 kolompok icon di **font awesome 5**. Berikut perbandingan tampilan dari keempat jenis icon tersebut:



Perbandingan jenis icon font awesome

Di sini bisa terlihat perbedaan dari variasi icon **solid**, **regular**, **light** dan **brands**. Khusus untuk

kelompok icon **light**, hanya tersedia dalam versi berbayar.

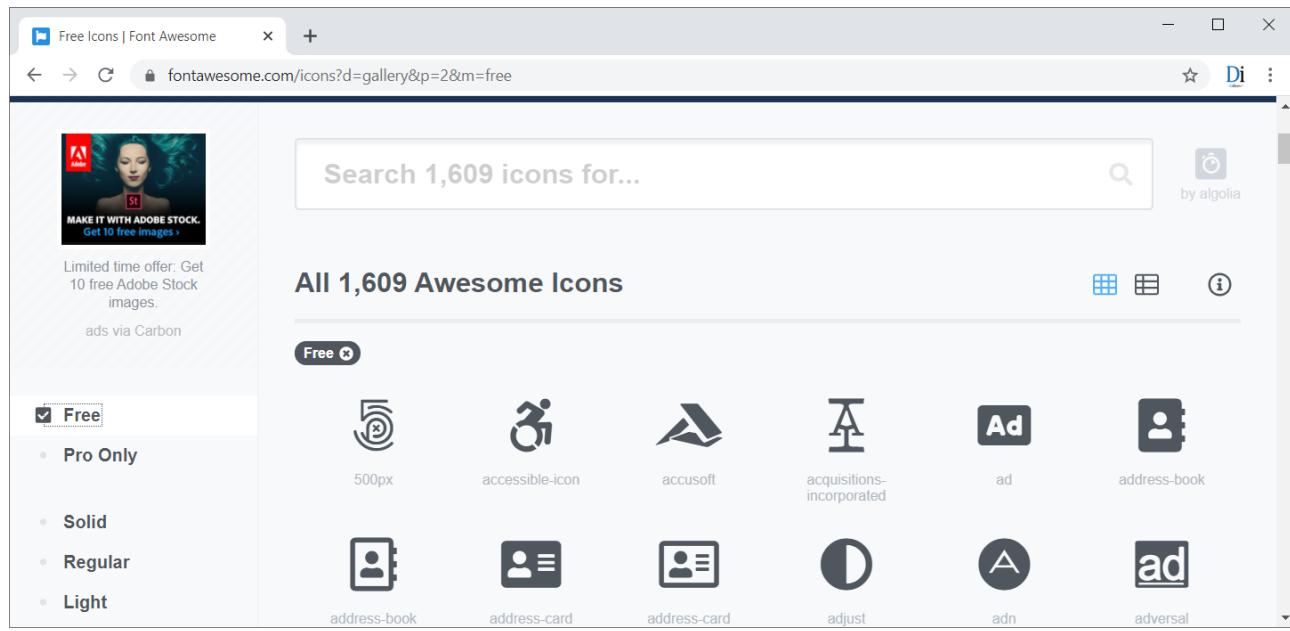
- **Solid** adalah kelompok icon dengan tampilan mayoritas hitam.
- **Regular** adalah kelompok icon dengan tampilan yang lebih dominan putih.
- **Light** adalah kelompok icon dengan font yang lebih tipis.
- **Brands** adalah kelompok icon merk atau logo seperti facebook, twitter dan instagram.

Update: font-awesome juga menambah variasi icon baru dalam konsep 2 warna (*duotone*), namun ini juga hanya bisa diakses untuk versi pro.

Menebak class untuk icon cukup mudah, yakni fa + <awalan huruf pertama dari jenis icon>. Misalnya untuk **brand** menggunakan class **.fab**, untuk **solid** menggunakan class **.fas**, dst. Sehingga jika kita ingin memakai icon facebook yang termasuk kelompok **brand**, penulisan classnya adalah **.fab**:

```
<i class="fab fa-facebook"></i>
```

Daftar dari seluruh icon font-awesome bisa di lihat ke fontawesome.com/icons³⁰.

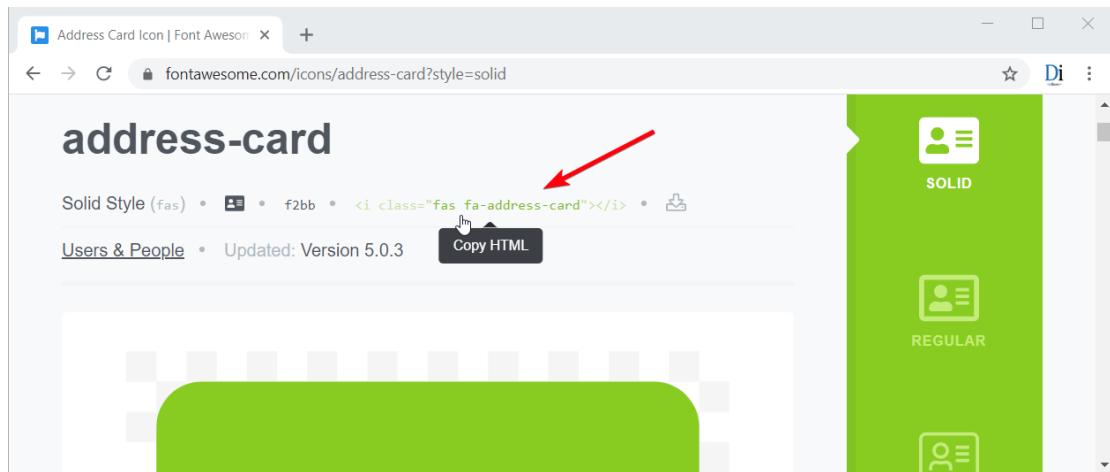


Di sidebar kiri terdapat berbagai pilihan untuk men-filter jenis icon. Karena kita menggunakan versi free, sebaiknya aktifkan filter **Free**.

Klik salah satu icon yang ingin dipakai, kemudian akan tampil halaman review dari icon tersebut lengkap dengan kode yang diperlukan:

30 <https://fontawesome.com/icons?d=gallery&p=2&m=free>

Font Awesome



Halaman review icon, lengkap dengan kode yang diperlukan

Alternatif lain yang lebih praktis, bisa ke **Font Awesome Cheatsheet**: fontawesome.com/cheatsheet³¹. Di halaman ini bisa terlihat semua icon beserta nama classnya:

A screenshot of a web browser displaying the Font Awesome Cheatsheet at fontawesome.com/cheatsheet. The page title is "Cheatsheet | Font Awesome". The main content is titled "Solid Icons". It lists various icons with their names, class names, and hex codes. For example, "ad" has class "fas fa-ad" and hex code f641. "align-right" has class "fas fa-align-right" and hex code f038. "angle-double-up" has class "fas fa-angle-double-up" and hex code f102.

Icon	Name	Class	Hex	Icon	Name	Class	Hex	Icon	Name	Class	Hex			
ad	ad	f641		address-book	address-book	f2b9		address-card	address-card	f2bb		adjust	adjust	f042
air-freshener	air-freshener	f5d0		align-center	align-center	f037		align-justify	align-justify	f039		align-left	align-left	f036
align-right	align-right	f038		allergies	allergies	f461		ambulance	ambulance	f0f9		american-sign-language-interpreting	american-sign-language-interpreting	f2a3
anchor	anchor	f13d		angle-double-down	angle-double-down	f103		angle-double-left	angle-double-left	f100		angle-double-right	angle-double-right	f101
angle-double-up	angle-double-up	f102		angle-down	angle-down	f107		angle-left	angle-left	f104		angle-right	angle-right	f105

Tampilan halaman Font Awesome Free's Cheatsheet

Berikut contoh cara penggunaan icon dari font awesome:

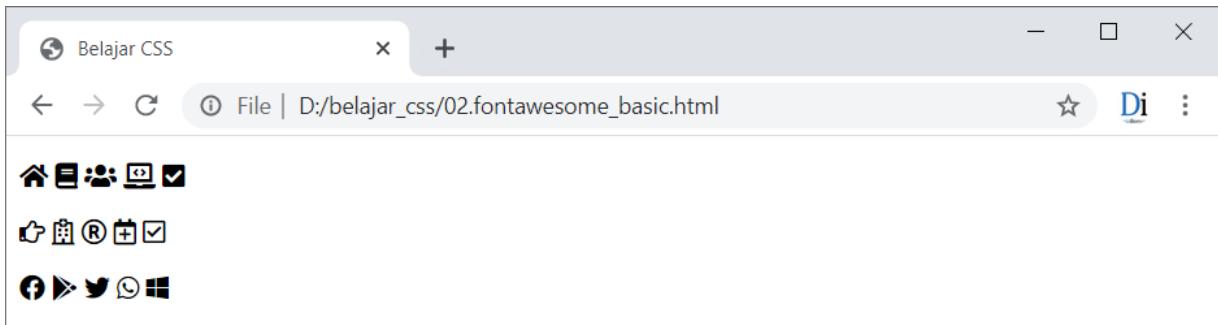
02. fontawesome_basic.html

```
1  <!DOCTYPE html>
2  <html lang="id">
3  <head>
4      <meta charset="UTF-8">
5      <title>Belajar CSS</title>
6      <link href="fontawesome/css/all.min.css" rel="stylesheet">
7  </head>
8  <body>
9      <p>
10         <i class="fas fa-home"></i>
11         <i class="fas fa-book"></i>
12         <i class="fas fa-users"></i>
13         <i class="fas fa-laptop-code"></i>
```

³¹ <https://fontawesome.com/cheatsheet>

Font Awesome

```
14    <i class="fas fa-check-square"></i>
15  </p>
16  <p>
17    <i class="far fa-hand-point-right"></i>
18    <i class="far fa-hospital"></i>
19    <i class="far fa-registered"></i>
20    <i class="far fa-calendar-plus"></i>
21    <i class="far fa-check-square"></i>
22  </p>
23  <p>
24    <i class="fab fa-facebook"></i>
25    <i class="fab fa-google-play"></i>
26    <i class="fab fa-twitter"></i>
27    <i class="fab fa-whatsapp"></i>
28    <i class="fab fa-windows"></i>
29  </p>
30 </body>
31 </html>
```



Gambar: Tampilan berbagai icon dari font awesome

Menggunakan tag `<i>`, saya menampilkan berbagai jenis icon font awesome. Jika icon ini tidak tampil, besar kemungkinan file CSS font awesome tidak terhubung ke file HTML.

21.5. Manipulasi Icon Font Awesome

Sebelum era font external, icon website umumnya dibuat dari gambar. Jika kita ingin menampilkan logo facebook, maka harus membuat sebuah gambar `logo_facebook.jpg` atau `logo_facebook.png`.

Meskipun masih bisa, cara ini kurang fleksibel. Jika kita ingin memakai lebih dari 1 ukuran icon, terpaksa membuat beberapa gambar. Atau bisa juga menyiapkan 1 gambar dengan resolusi yang cukup besar, lalu atur dari properti `width` dan `height` CSS.

Alternatif lain adalah membuat icon berbasis **vector** seperti format `.svg`. Gambar yang dibuat menggunakan `svg` tidak akan pecah saat diperbesar.

Berita baiknya, icon *font awesome* semuanya berbasis vector. Dengan demikian kita bisa memperbesar icon ini tanpa kehilangan detail. Selain itu karena icon di dalam font awesome berbentuk **font**, maka bisa diwarnai menggunakan property `color` CSS seperti teks pada

umumnya.

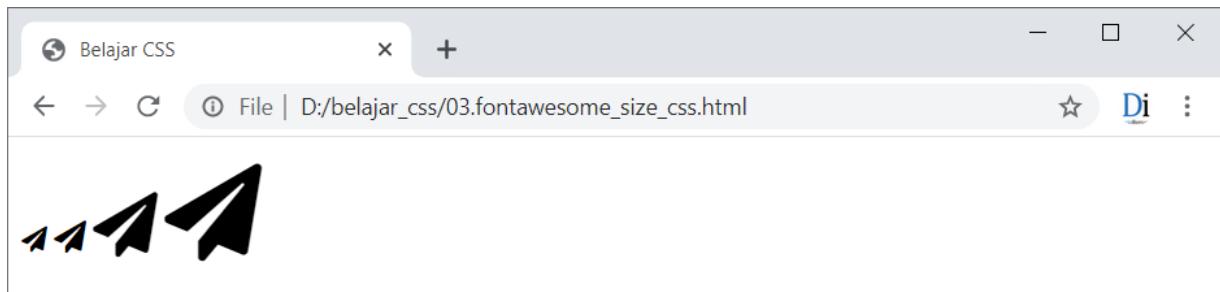
Lebih jauh lagi, font awesome menyediakan class khusus untuk membuat efek tambahan seperti mengatur arah icon (*rotate*), membuat animasi berputar (*spinning*), hingga menggabung beberapa icon menjadi icon baru.

Memperbesar Ukuran Icon

Icon di dalam font awesome adalah sebuah font, sehingga bisa diperbesar dengan mengatur nilai property `font-size`. Berikut contoh prakteknya:

03.fontawesome_size_css.html

```
1 <p>
2   <i class="fas fa-paper-plane" style="font-size: 16px"></i>
3   <i class="fas fa-paper-plane" style="font-size: 20px"></i>
4   <i class="fas fa-paper-plane" style="font-size: 40px"></i>
5   <i class="fas fa-paper-plane" style="font-size: 60px"></i>
6 </p>
```



Berbagai ukuran icon font awesome dari property `font-size`

Bisa terlihat meskipun ukuran font diperbesar, gambar icon tetap terlihat tajam. Inilah keunggulan icon berbasis vector seperti font awesome.

Selain menggunakan kode CSS langsung, font awesome juga menyediakan 13 class bawaan untuk mengatur ukuran icon, yakni class `.fa-xs`, `.fa-sm`, `.fa-lg`, serta class `.fa-1x`, `.fa-2x`, `.fa-3x`, dst hingga `.fa-10x`:

04.fontawesome_size.html

```
1 <p>
2   <i class="fas fa-laptop-code fa-xs"></i>
3   <i class="fas fa-laptop-code fa-sm"></i>
4   <i class="fas fa-laptop-code fa-lg"></i>
5   <i class="fas fa-laptop-code fa-2x"></i>
6   <i class="fas fa-laptop-code fa-3x"></i>
7   <i class="fas fa-laptop-code fa-5x"></i>
8   <i class="fas fa-laptop-code fa-7x"></i>
9   <i class="fas fa-laptop-code fa-10x"></i>
10 </p>
```



Berbagai ukuran icon font awesome dari class bawaan

Class bawaan font awesome ini menggunakan satuan `em` sebagai dasar, dengan perhitungan berikut:

```
.fa-xs { font-size: .75em; }
.fa-sm { font-size: .875em; }
.fa-lg { font-size: 1.33333em; }
.fa-1x { font-size: 1em; }
.fa-2x { font-size: 2em; }
.fa-3x { font-size: 3em; }
.fa-4x { font-size: 4em; }
.fa-5x { font-size: 5em; }
.fa-6x { font-size: 6em; }
.fa-7x { font-size: 7em; }
.fa-8x { font-size: 8em; }
.fa-9x { font-size: 9em; }
.fa-10x { font-size: 10em; }
```

Kita bebas apakah ingin men-set ukuran icon secara langsung menggunakan property `font-size` CSS atau memakai salah satu class bawaan font awesome di atas.

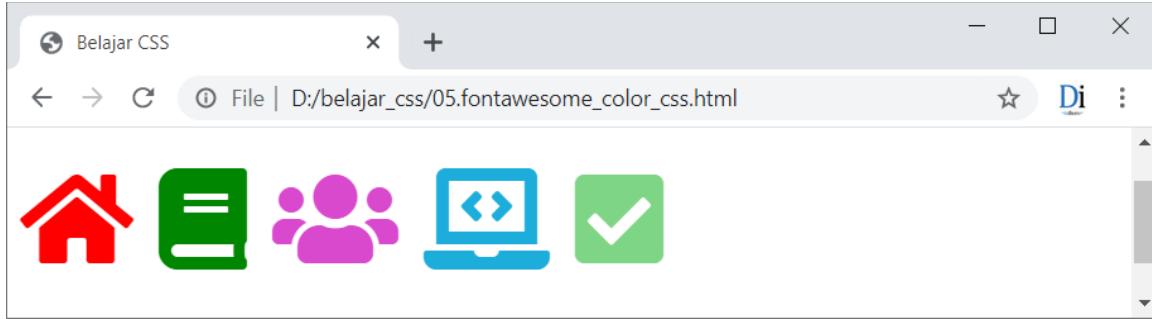
Mengubah Warna Icon

Kembali karena font awesome adalah sebuah font, warna icon ini bisa diubah menggunakan property `color` CSS:

05.fontawesome_color_css.html

```
1 <p class="fa-4x">
2   <i class="fas fa-home" style="color:red"></i>
3   <i class="fas fa-book" style="color:green"></i>
4   <i class="fas fa-users" style="color:#d743ca"></i>
5   <i class="fas fa-laptop-code" style="color:rgb(24, 168, 216)"></i>
6   <i class="fas fa-check-square" style="color:rgba(86, 199, 95, 0.8)"></i>
7 </p>
```

Font Awesome



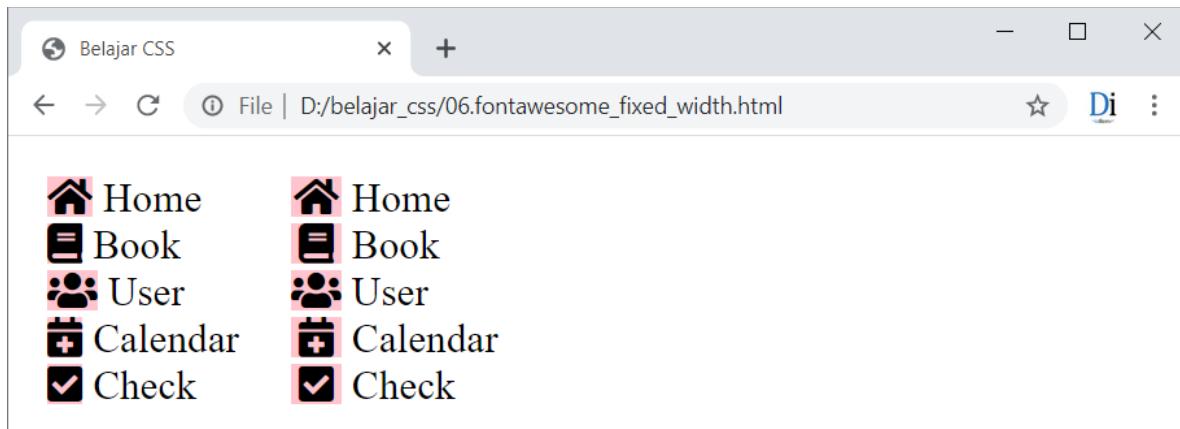
Berbagai warna icon font awesome dari property color CSS

Fixed Width Icon

Font awesome terdiri dari berbagai icon dengan lebar dan tinggi bervariasi. Akibatnya, jika kita menampilkan icon secara vertical, variasi lebar ini membuat tampilan icon menjadi tidak sejajar. Oleh karena itu font awesome menyediakan class `.fa-fw` untuk mengatur agar lebar setiap icon menjadi sama besar (*fixed width*). Berikut perbedaannya:

06.fontawesome_fixed_width.html

```
1 <style>
2   i { background-color: pink }
3   .container {
4     font-size: 25px;
5     float: left;
6     padding: 1rem;
7   }
8 </style>
9 ...
10
11 <div class="container">
12   <div><i class="fas fa-home"></i> Home </div>
13   <div><i class="fas fa-book"></i> Book </div>
14   <div><i class="fas fa-users"></i> User </div>
15   <div><i class="fas fa-calendar-plus"></i> Calendar </div>
16   <div><i class="fas fa-check-square"></i> Check </div>
17 </div>
18 <div class="container">
19   <div><i class="fas fa-home fa-fw"></i> Home </div>
20   <div><i class="fas fa-book fa-fw"></i> Book </div>
21   <div><i class="fas fa-users fa-fw"></i> User </div>
22   <div><i class="fas fa-calendar-plus fa-fw"></i> Calendar </div>
23   <div><i class="fas fa-check-square fa-fw"></i> Check </div>
24 </div>
```



Tampilan icon secara vertikal

Saya menambah property `background-color:pink` ke dalam semua tag `<i>` agar bisa terlihat lebar setiap icon.

Hasil di sebelah kiri adalah tampilan default dari font awesome. Terlihat lebar setiap icon berbeda-beda sehingga teks menjadi tidak sejajar. Tampilan di sebelah kanan adalah hasil dari tambahan class `.fa-fw` ke semua tag `<i>`. Sekarang setiap icon memiliki lebar yang sama sehingga tampak lebih rapi.

Secara internal, isi dari class `.fa-fw` ini adalah sebagai berikut:

```
1 .fa-fw {
2   text-align: center;
3   width: 1.25em;
4 }
```

Trik yang dipakai font awesome adalah menyamakan lebar tag `<i>` menjadi `1.25em` lalu mengatur agar icon tampil dengan rata tengah menggunakan property `text-align:center`.

Icon List

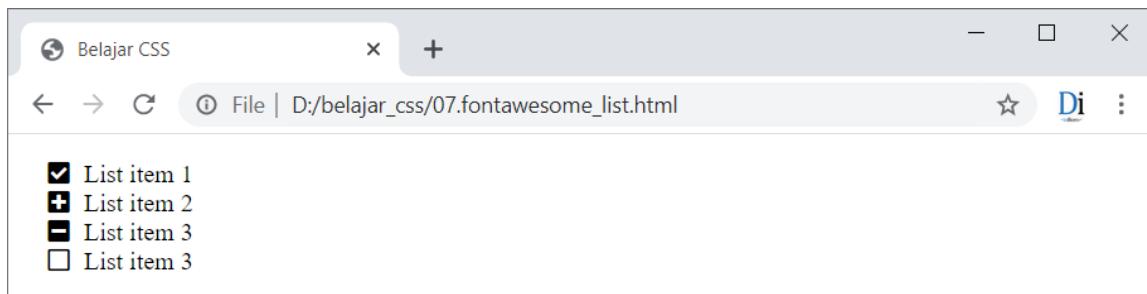
Icon font awesome cukup sering dipakai menggantikan tombol list bawaan HTML. Agar lebih mudah, font awesome menyediakan class `.fa-ul` dan `.fa-li` untuk keperluan ini. Berikut contoh penggunaannya:

07.fontawesome_list.html

```
1 <div>
2   <ul class="fa-ul">
3     <li>
4       <span class="fa-li"><i class="fas fa-check-square"></i></span>List item 1
5     </li>
6     <li>
7       <span class="fa-li"><i class="fas fa-plus-square"></i></span>List item 2
8     </li>
9     <li>
10    <span class="fa-li"><i class="fas fa-minus-square"></i></span>List item 3
```

Font Awesome

```
11  </li>
12  <li>
13    <span class="fa-li"><i class="far fa-square"></i></span>List item 3
14  </li>
15 </ul>
16 </div>
```



Penggunaan icon dari font awesome untuk list

Class `.fa-ul` harus ditempatkan ke dalam parent element list, yakni tag `` (baris 2). Kemudian class `.fa-ul` ditempatkan ke tag `` yang menjadi container dari tag `<i>`. Tag `` ini juga nantinya berada di dalam setiap tag ``.

Kedua class ini, `.fa-ul` dan `.fa-li` memudahkan kita mengganti icon list standar dengan icon font awesome tanpa perlu membuat kode CSS sendiri.

Rotating Icon

Font awesome juga menyediakan class khusus untuk memutar icon (rotating icon). Caranya, tambah salah satu dari class berikut ke dalam tag `<i>`:

- `.fa-rotate-90`
- `.fa-rotate-180`
- `.fa-rotate-270`
- `.fa-flip-horizontal`
- `.fa-flip-vertical`

Sesuai namanya, setiap class akan memutar icon sesuai sudut derajat, atau dibalik (flip) secara horizontal atau vertical. Berikut contoh penggunaannya:

08.fontawesome_rotate.html

```
1 <div class="fa-3x">
2   <i class="fas fa-truck"></i>
3   <i class="fas fa-truck fa-rotate-90"></i>
4   <i class="fas fa-truck fa-rotate-180"></i>
5   <i class="fas fa-truck fa-rotate-270"></i>
6   <i class="fas fa-truck fa-flip-horizontal"></i>
7   <i class="fas fa-truck fa-flip-vertical"></i>
8 </div>
```

Font Awesome



Memutar icon dengan class bawaan font awesome

Dengan ini, kita bisa memutar semua icon bawaan font awesome.

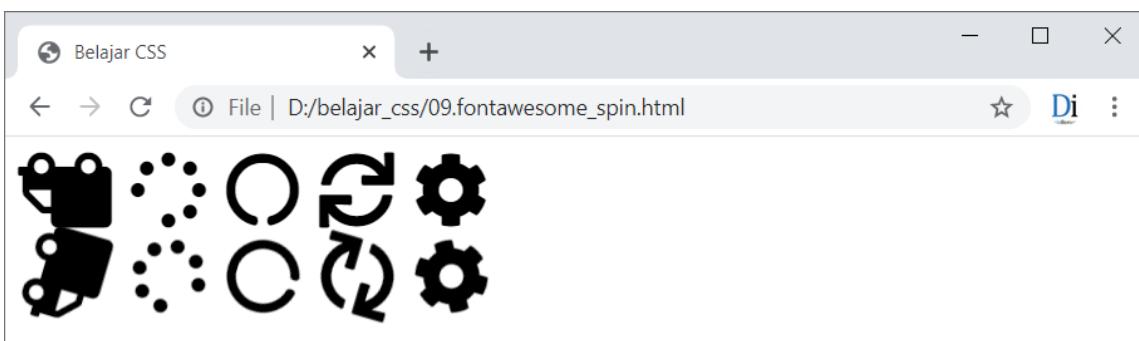
Spinning Icon

Jika memutar icon masih kurang menarik, bagaimana jika memutarnya dengan animasi (*spinning*)? Font awesome menyediakan class `.fa-pulse` dan `.fa-spin` untuk keperluan ini. Class `.fa-pulse` akan memutar icon per-posisi setiap 45 derajat (efek patah-patah), sedangkan class `.fa-spin` akan memutar icon secara halus.

Berikut contoh kode programnya:

09.fontawesome_spin.html

```
1 <div class="fa-3x">
2   <i class="fas fa-truck fa-pulse"></i>
3   <i class="fas fa-spinner fa-pulse"></i>
4   <i class="fas fa-circle-notch fa-pulse"></i>
5   <i class="fas fa-sync fa-pulse"></i>
6   <i class="fas fa-cog fa-pulse"></i>
7 </div>
8 <div class="fa-3x">
9   <i class="fas fa-truck fa-spin"></i>
10  <i class="fas fa-spinner fa-spin"></i>
11  <i class="fas fa-circle-notch fa-spin"></i>
12  <i class="fas fa-sync fa-spin"></i>
13  <i class="fas fa-cog fa-spin"></i>
14 </div>
```



Animasi putaran icon dengan class bawaan font awesome

Karena keterbatasan media, efek animasi ini tidak bisa terlihat di buku. Silahkan jalankan

langsung kode program di atas.

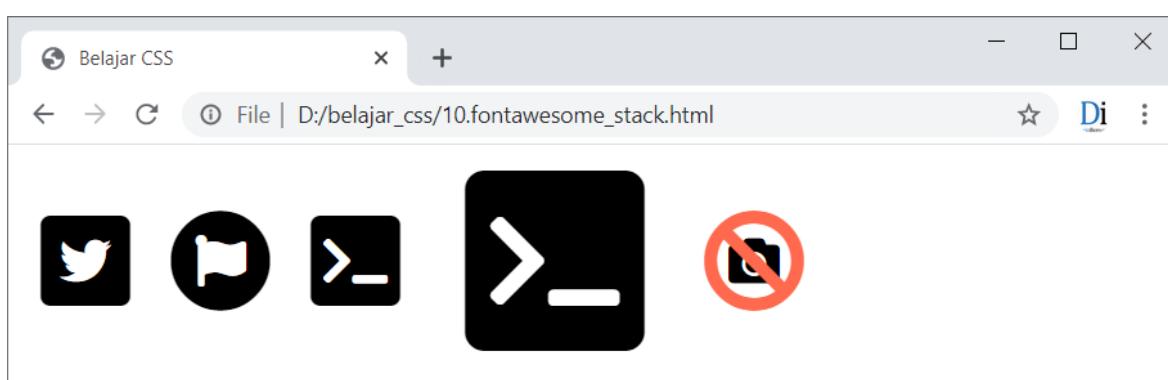
Stacked Icon

Stacked icon adalah sebutan dari menumpuk 2 buah icon untuk menghasilkan icon baru. Font awesome menyediakan 4 class untuk membuat efek ini, yaitu class .fa-stack, .fa-stack-1x, .fa-stack-2x dan .fa-inverse.

Mari kita bahas menggunakan contoh kode program:

10.fontawesome_stack.html

```
1 <div>
2   <span class="fa-stack fa-2x">
3     <i class="fas fa-square fa-stack-2x"></i>
4     <i class="fab fa-twitter fa-stack-1x fa-inverse"></i>
5   </span>
6   <span class="fa-stack fa-2x">
7     <i class="fas fa-circle fa-stack-2x"></i>
8     <i class="fas fa-flag fa-stack-1x fa-inverse"></i>
9   </span>
10  <span class="fa-stack fa-2x">
11    <i class="fas fa-square fa-stack-2x"></i>
12    <i class="fas fa-terminal fa-stack-1x fa-inverse"></i>
13  </span>
14  <span class="fa-stack fa-4x">
15    <i class="fas fa-square fa-stack-2x"></i>
16    <i class="fas fa-terminal fa-stack-1x fa-inverse"></i>
17  </span>
18  <span class="fa-stack fa-2x">
19    <i class="fas fa-camera fa-stack-1x"></i>
20    <i class="fas fa-ban fa-stack-2x" style="color:Tomato"></i>
21  </span>
22 </div>
```



Hasil penggabungan 2 icon (stacked icon)

Untuk membuat **stacked icon**, kita perlu sebuah *parent element* dengan class .fa-stack. Dalam contoh di atas, parent element tersebut berupa tag .

Di dalam tag inilah tulis dua buah icon. Class .fa-stack-1x dan .fa-

`stack-2x` bisa dipakai untuk menentukan besar masing-masing icon. Serta class `.fa-inverse` bisa dipakai untuk membalik warna icon, yakni membuat efek negatif dimana warna hitam menjadi putih dan putih menjadi hitam.

Sebagai contoh, gambar paling kiri adalah gabungan dari icon `fa-square` (kotak persegi) yang di perbesar 2 kali menggunakan class `.fa-stack-2x` (baris 3) dan icon `fa-twitter` (logo twitter) dengan class `.fa-stack-2x` yang warnanya dibalik menggunakan class `.fa-inverse` (baris 4).

Lebih jauh lagi, stacked icon juga bisa dikombinasikan dengan efek font awesome lain seperti warna dan rotasi. Gambar paling kanan adalah kombinasi dari icon warna merah (`fa-ban`) dengan icon camera (`fa-camera`). Perhatikan bahwa urutan penulisan font juga berpengaruh. Tanda silang berada di atas gambar camera karena icon silang ini ditulis setelah icon camera.

Font awesome memang bukan bagian dari CSS, tapi cukup banyak dipakai dalam pembuatan template.

Pada bab selanjutnya kita akan masuk ke contoh kasus pembuatan berbagai komponen halaman menggunakan CSS.

Terimakasih sudah membeli eBook / buku asli dari DuniaIlkom

Saya menyadari menulis eBook sangat beresiko karena mudah di bajak dan digandakan. Namun ini saya lakukan agar teman-teman (terutama yang di daerah) bisa mendapat materi belajar programming berkualitas dengan harga yang relatif terjangkau.

Proses penulisan buku ini juga tidak sebentar, butuh waktu berbulan-bulan. Mohon kerja sama teman-teman semua untuk tidak menggandakan dan menyebarkan eBook ini. Hak cipta eBook sudah terdaftar di Depkumham RI.

~ Semoga ilmu yang didapat berkah dan bermanfaat. Terimakasih ~

22. CSS Case Study

Sepanjang buku ini kita telah membahas beragam selector dan property CSS. Mulai dari konsep dasar, aturan penulisan, cascade, inheritance, specificity, hingga berbagai fitur terbaru dari CSS3.

Namun ini semua baru dasar dari CSS. Untuk mampu membuat sebuah website utuh, kita harus menyatukan semuanya menjadi kode yang harmonis. Sebagai contoh, di dalam setiap website hampir selalu terdapat horizontal menu, bagaimana cara membuatnya?

Analogi ini mirip seperti belajar gitar. Sebelum bisa membawakan sebuah lagu, kita harus belajar dahulu teknik dasar bermain gitar seperti posisi jari, kunci nada, dsb. Setelah itu barulah belajar cara menyatukannya menjadi melodi yang harmonis.

Dalam bab terakhir buku CSS Uncover ini kita akan praktik merancang berbagai komponen website. Diantaranya membuat menu navigasi, bagian header, gambar display, hingga footer. Setiap komponen nantinya akan disatukan menjadi sebuah design web utuh.

Era responsive web design juga menghadirkan tantangan tersendiri. Setiap komponen web harus responsive untuk mengakomodasi semua ukuran layar. Materi media query serta CSS Flexbox akan mempermudah perancangan web responsive.

Untuk menghemat tempat, saya tidak menampilkan kode program utuh, tapi hanya sebagian sesuai pembahasan materi. Kode program full tersedia di file **belajar_css.zip**.

22.1. Membuat Menu Navigasi Vertikal

Salah satu komponen web yang selalu dibutuhkan adalah menu navigasi. Menu navigasi dalam bentuk horizontal dari kiri ke kanan, atau secara vertikal dari atas ke bawah. Kita akan mulai dengan menu navigasi vertikal terlebih dahulu karena lebih sederhana. Menu jenis ini sering dipakai di dalam bagian sidebar.

Berikut tampilan menu yang akan kita buat:



Gambar: Menu navigasi vertikal

Struktur menu navigasi bisa dibuat dari tag apa saja, yang paling sering dipakai adalah dari unordered list tag ``. Berikut kode HTML yang diperlukan:

01.nav-vertical.html

```

1 <nav>
2   <ul>
3     <li><a href="#" class="active">Home</a></li>
4     <li><a href="#">Service</a></li>
5     <li><a href="#">Project</a></li>
6     <li><a href="#">About</a></li>
7   </ul>
8 </nav>

```

Teks menu dibuat dari tag `<a>` agar langsung berfungsi sebagai link. Setiap link berada di dalam tag `` yang menjadi anggota dari unordered list tag ``. Tag `<nav>` di sisi luar berfungsi sebagai container dari menu kita.

Setelah struktur HTML selesai, saatnya masuk ke kode CSS:

01.nav-vertical.html

```

1 nav{
2   width: 200px;
3 }
4 ul {
5   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
6   list-style-type: none;
7   margin: 0;
8   padding: 0;
9 }
10 ul li {
11   background-color: #f1f1f1;
12 }
13 li a {
14   display: block;
15   color: #515151;
16   padding: 0.5em 1em;
17   text-decoration: none;

```

```

18 }
19 li a.active {
20   background-color: #44b5b5;
21   color: white;
22 }
23 li a:hover {
24   background-color: #515151;
25   color: white;
26 }

```

Kode di baris 1 – 3 dipakai untuk men-set lebar tag `<aside>` menjadi 200px. Karena tag `<aside>` berperan sebagai container, maka lebar ini akan membatasi lebar keseluruhan menu.

Kode di baris 4 – 9 berfungsi untuk mengatur jenis font menjadi 'Segoe UI', serta menghapus efek bawaan dari tag `` dengan property `list-style-type: none`, `margin: 0` serta `padding: 0`. Dengan tambahan kode ini, list tidak lagi memiliki bullet.

Di baris 10, selector `ul li` berisi kode untuk memberi warna background menu menjadi `#f1f1f1`, yakni warna abu-abu terang.

Property dari selector `li a` antara baris 13 – 18 akan menentukan bentuk menu. Yang paling penting di sini adalah `display: block` untuk mengubah tampilan tag `<a>` dari inline level element menjadi block level element. Dengan demikian, tag `<a>` akan memanjang memenuhi parent element.

Kemudian terdapat beberapa property tambahan untuk merapikan tampilan tag `<a>`. Property `color: #515151` membuat warna teks menjadi abu-abu tua, kemudian `padding: 0.5em 1em` berfungsi mengatur besar box berdasarkan padding, serta menghapus efek garis bawah bawaan tag `<a>` dengan property `text-decoration: none`.

Selector `li a.active` akan cocok dengan tag `<a>` yang memiliki tambahan class "active". Ini merupakan cara untuk menandakan posisi user di dalam sebuah menu. Misalnya jika user saat ini ada di halaman Home, maka warna menu home ini akan sedikit berbeda. Mengenai cara memindahkan class "active" dari satu menu ke menu lain, bisa dilakukan dari bahasa pemrograman server seperti PHP.

Selector terakhir, `li a:hover` dipakai untuk membuat efek perubahan warna background dan warna teks saat cursor mouse ada di atas menu.

Dengan kode ini, kita sudah berhasil membuat menu navigasi vertikal. Silahkan modifikasi dengan menambah beberapa efek lain, misalnya memberikan warna border left serta efek transisi seperti berikut:

02.nav-vertical-style.html

```

1 ...
2 li a {
3   display: block;
4   color: #515151;

```

```

5     padding: 0.5em 1em;
6     text-decoration: none;
7     border-left: 8px solid #515151;
8     text-transform: uppercase;
9   }
10    li a.active {
11       background-color: #44b5b5;
12       color: white;
13     }
14    li a:hover {
15       background-color: #515151;
16       color: white;
17       transition: all 0.3s;
18     }
19 ...

```



Gambar: Menu navigasi vertikal dengan efek transisi

Kali ini saya menambah property `border-left: 8px solid #515151` ke dalam selector `li a`, sehingga akan muncul garis hitam di sisi kiri menu. Property `text-transform: uppercase` berfungsi untuk mengubah teks menu menjadi huruf besar.

Tambahan lain ada di property `transition: all 0.3s` milik selector `li a:hover`. Ini akan membuat efek transisi warna secara bertahap saat cursor mouse berada di atas menu.

22.2. Membuat Menu Navigasi Horizontal

Menu vertikal sudah selesai, saatnya merancang menu navigasi horizontal. CSS3 Flexbox sangat membantu kita dalam hal ini, cukup tambah property `display: flex` ke dalam selector `ul` dan hapus `width` dari selector `nav`. Hasilnya, menu akan berjejer secara horizontal dari kiri ke kanan.

Berikut modifikasi dari kode CSS kita sebelumnya:

`03.nav-horizontal.html`

```

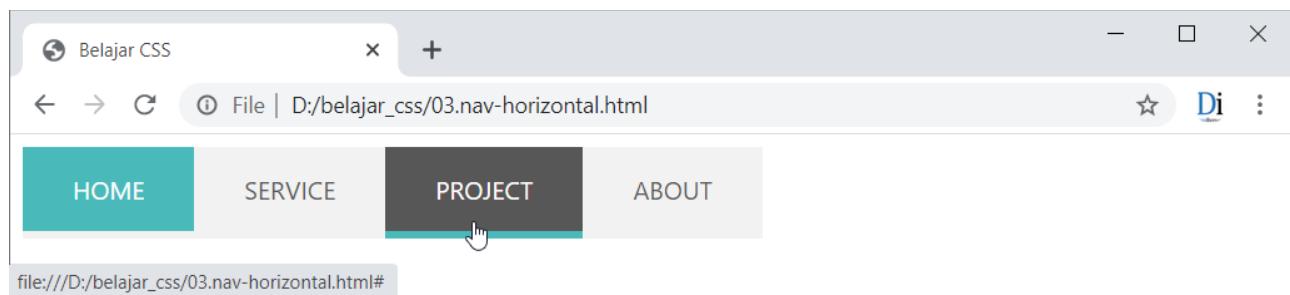
1  ul {
2     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
3     list-style-type: none;

```

```

4     margin: 0;
5     padding: 0;
6     display: flex;
7 }
8 ul li {
9     background-color: #f1f1f1;
10}
11 li a {
12     display: block;
13     color: #515151;
14     padding: 1em 2em;
15     text-decoration: none;
16     text-transform: uppercase;
17     border-bottom: 5px solid #f1f1f1;
18 }
19 li a.active {
20     background-color: #44b5b5;
21     color: white;
22 }
23 li a:hover {
24     background-color: #515151;
25     color: white;
26     transition: background-color 0.3s;
27     border-bottom: 5px solid #44b5b5;
28 }

```



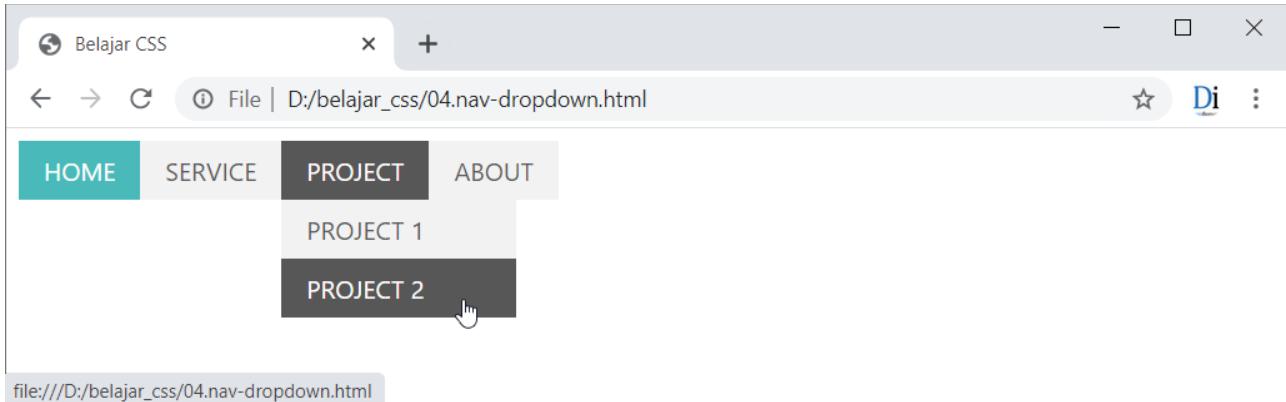
Gambar: Menu navigasi horizontal

Dengan penambahan property `display: flex` di baris 6, semua tag `` akan berjejer secara horizontal. Perubahan lain adalah mengatur ulang padding tag `<a>` di baris 14 serta menambah `border-bottom` di baris 17.

Ketika di hover, saya ingin efek transisi berjalan hanya untuk `background-color` saja, maka property yang diperlukan adalah `transition: background-color 0.3s` seperti di baris 16. Selain itu ketika di hover, ubah juga warna `border-bottom`.

22.3. Membuat Menu Navigasi Dropdown

Menu dropdown yang dimaksud di sini adalah menu horizontal yang memiliki "anak" atau submenu. Submenu baru akan tampil ketika menu utama di hover:



Gambar: Menu navigasi dropdown

Menu seperti ini cukup susah dibuat karena perlu trik tersendiri. Trik tersebut berkaitan dengan bagaimana menyembunyikan submenu ketika halaman load, dan baru tampil saat menu utama di hover.

Untuk membuatnya, kita berangkat dari struktur HTML berikut:

04.nav-dropdown.html

```

1 <nav>
2   <ul>
3     <li><a href="#" class="active">Home</a></li>
4     <li><a href="#">Service</a>
5       <ul class="submenu">
6         <li><a href="">Service 1</a></li>
7         <li><a href="">Service 2</a></li>
8         <li><a href="">Service 3</a></li>
9       </ul>
10    </li>
11    <li><a href="#">Project</a>
12      <ul class="submenu">
13        <li><a href="">Project 1</a></li>
14        <li><a href="">Project 2</a></li>
15      </ul>
16    </li>
17    <li><a href="#">About</a></li>
18  </ul>
19 </nav>
```

Ini pengembangan dari menu kita sebelumnya. Struktur submenu butuh nested list, yakni list di dalam list. Menu "Service" memiliki 3 submenu di antara baris 5 – 9, serta menu "Project" memiliki 2 submenu di baris 12 – 15. Submenu ditandai dengan atribut `class="submenu"` yang nantinya di perlukan sebagai selector CSS.

Berikut kode CSS yang diperlukan untuk membuat menu dropdown:

04.nav-dropdown.html

```
1 ul {
```

```

2   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
3   list-style-type: none;
4   margin: 0;
5   padding: 0;
6   display: flex;
7 }
8 ul li {
9   background-color: #f1f1f1;
10  position: relative;
11 }
12 li a {
13   display: block;
14   color: #515151;
15   padding: 0.5em 1em;
16   text-decoration: none;
17   text-transform: uppercase;
18 }
19 li a.active {
20   background-color: #44b5b5;
21   color: white;
22 }
23 li a:hover {
24   background-color: #515151;
25   color: white;
26 }
27 .submenu{
28   flex-flow: column;
29   position: absolute;
30   display: none;
31 }
32 nav li:hover>a {
33   background-color: #515151;
34   color: white;
35 }
36 nav li:hover ul{
37   display:block;
38   min-width: 150px;
39 }

```

Beberapa property masih sama seperti yang dipakai saat membuat menu horizontal, sehingga tidak saya bahas lagi. Kita akan fokus ke teknik pembuatan dropdown saja.

Pertama, selector `ul li` memiliki tambahan property `position: relative` di baris 10. Property ini dipakai sebagai patokan dari property `position: absolute` pada selector `.submenu` di baris 29. Dengan kombinasi dua property ini, posisi `.submenu` akan berada tepat di bawah menu utama.

Selector `.submenu` juga di set dengan `flex-flow: column` di baris 28 agar tampilan `.submenu` tersusun dari atas ke bawah. Property `display: none` di baris 30 akan menyembunyikan semua isi `.submenu` ketika halaman di load.

Dua selector terakhir memang cukup rumit. Selector `nav li:hover>a` di baris 32 – 34 bisa

dibaca "ubah warna background dan teks dari tag yang langsung berada setelah tag , hanya ketika tag dikenai :hover (mouse berada di atas tag), selain itu tag ini juga harus berada di dalam tag <nav>". Selector ini berfungsi agar ketika kita sedang memilih .submenu, warna dari menu utama juga ikut berubah.

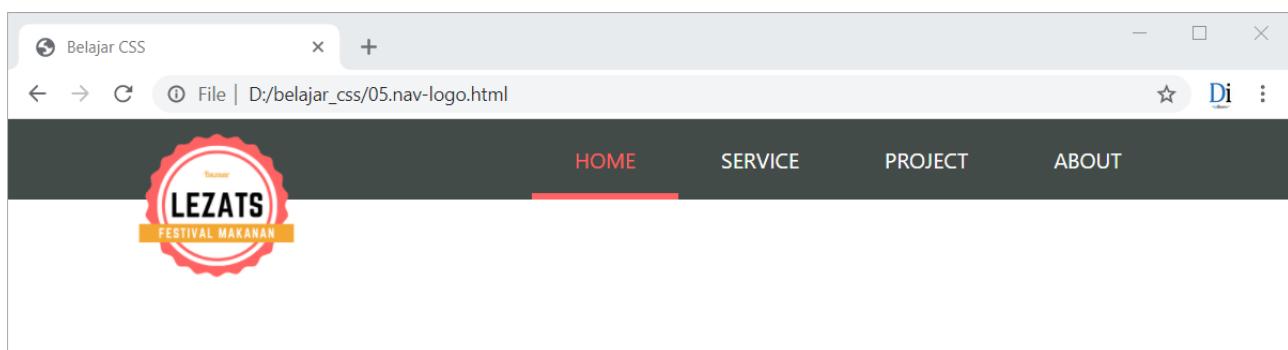
Kemudian, selector `nav li:hover ul` bisa dibaca: "Cari semua tag yang berada di dalam tag <nav>, lalu ketika cursor mouse berada di atas tag itu (hover), jalankan property CSS hanya untuk tag yang berada di dalam tag tersebut (target di sini adalah submenu)".

Property yang akan dijalankan oleh `nav li:hover ul` adalah `display:block` untuk menampilkan kembali .submenu yang sebelumnya masih hidden, serta `min-width: 150px` untuk mengatur lebar dari .submenu.

Dua property inilah yang saya sebut sebagai "trik" karena sangat sulit jika dipikirkan sendiri tanpa melihat referensi yang sudah ada. Pada prakteknya, saya juga masih sering copy-paste kode CSS yang didapat dari internet. Namun pengetahuan teknis CSS tetap diperlukan karena kode tersebut tetap harus dipahami apa fungsinya dan dimana harus ditempatkan (tidak bisa asal copas).

22.4. Membuat Header (Logo + Navigasi)

Selain menu navigasi, bagian header sebuah web biasanya juga memiliki gambar logo. Kali ini kita akan coba rancang tampilan dimana logo web ada di sisi kiri, dan menu navigasi ada di sisi kanan header:



Gambar: Header yang terdiri dari logo dan menu navigasi

Seerti biasa, kita akan siapkan struktur HTML yang diperlukan:

05.nav-logo.html

```
1 <div class="container">
2   <header>
3     <div class="logo">
4       
5     </div>
6     <nav>
7       <ul>
```

```

8      <li><a href="#" class="active">Home</a></li>
9      <li><a href="#">Service</a></li>
10     <li><a href="#">Project</a></li>
11     <li><a href="#">About</a></li>
12   </ul>
13 </nav>
14 </header>
15 </div>
```

Sebagai container luar terdapat tag `<div class="container">`, lalu di dalamnya juga ada tag `<header>`. Struktur berlapis seperti ini saya perlukan karena ingin membuat warna background header yang memanjang memenuhi lebar web browser.

Tag `<header>` memiliki 2 buah child element, yakni tag `<div class="logo">` yang berisi gambar logo, serta tag `<nav>` yang berisi menu navigasi. Agar kodennya tidak terlalu panjang, saya kembali menggunakan menu navigasi biasa, bukan versi dropdown.

Boleh bandingkan sejenak struktur HTML ini dengan tampilan menu sebelumnya, lalu perkirakan apa saja property CSS kita perlukan.

Baik, berikut kode CSS yang saya pakai:

05.nav-logo.html

```

1  *{
2    box-sizing: border-box;
3  }
4  body, html {
5    padding: 0;
6    margin: 0;
7  }
8  .container {
9    background-color: #3c4542;
10 }
11 header {
12   display: flex;
13   flex-direction: row;
14   justify-content: space-between;
15   width: 80vw;
16   height: 60px;
17   margin: 0 auto;
18 }
19 .logo img {
20   width: 120px;
21   padding-top: 10px;
22 }
23 ul {
24   font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
25   list-style-type: none;
26   margin: 0;
27   padding: 0;
28   display: flex;
29 }
```

```

30 li a {
31   display: block;
32   color: white;
33   padding: 0 2em;
34   line-height: 60px;
35   text-decoration: none;
36   text-transform: uppercase;
37   height: 60px;
38 }
39 li a.active {
40   color: #FF5C5C;
41   border-bottom: 5px solid #FF5C5C;
42 }
43 li a:hover {
44   color: #FF5C5C;
45   transition: all 0.3s;
46   border-bottom: 5px solid #FF5C5C;
47 }

```

Selector pertama, * {box-sizing: border-box} dipakai untuk mengubah efek default box model dari content-box menjadi border-box. Dengan perubahan ini maka nilai property width dan height semua element sudah termasuk dengan border serta padding. Ini sebenarnya tidak wajib diubah, lebih ke menunjukkan teknik umum yang dipakai oleh sebagian web desainer.

Selector body, html di baris 4 - 7 berguna untuk menghapus padding dan margin bawaan web browser. Jika ini tidak dilakukan, akan terdapat sedikit ruang margin di sekeliling web browser.

Selector ketiga, .container {background-color: #3c4542} akan memberikan warna background gelap untuk container. Karena lebar container tidak ditulis, maka warna ini akan memenuhi lebar web browser.

Proses mengatur posisi gambar dan menu navigasi dilakukan oleh selector header di antara baris 11 – 18. Prinsip dasarnya hanya menerapkan display: flex, flex-direction: row, serta justify-content: space-between. Ini akan memisahkan gambar logo ke sisi kanan header, dan menu navigasi ke sisi kiri header.

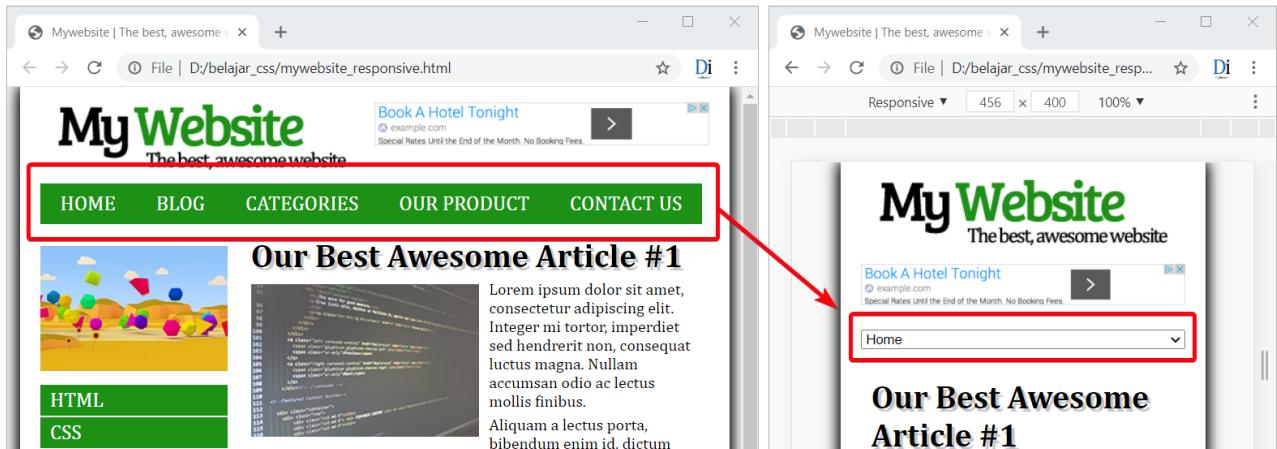
Tiga property milik header lain, yakni width: 80vw, height: 60px dan margin: 0 auto dipakai untuk menentukan lebar, tinggi serta agar posisi header agar berada di tengah parent element.

Selector .logo img di baris 19 – 22 berisi kode untuk mengatur ukuran gambar logo dengan width: 120px, serta mendorong gambar ke arah bawah dengan padding-top: 10px. Ukuran dan posisi logo ini lebih ke selera saja, apakah ingin tampil sejajar dengan menu, atau di dorong sedikit ke bawah seperti dalam kode ini.

Sisa selector lain dipakai untuk membuat menu navigasi horizontal yang mirip seperti sebelumnya. Sedikit variasi hanya dari segi pemilihan warna.

22.5. Membuat Header Responsive (Hamburger Menu)

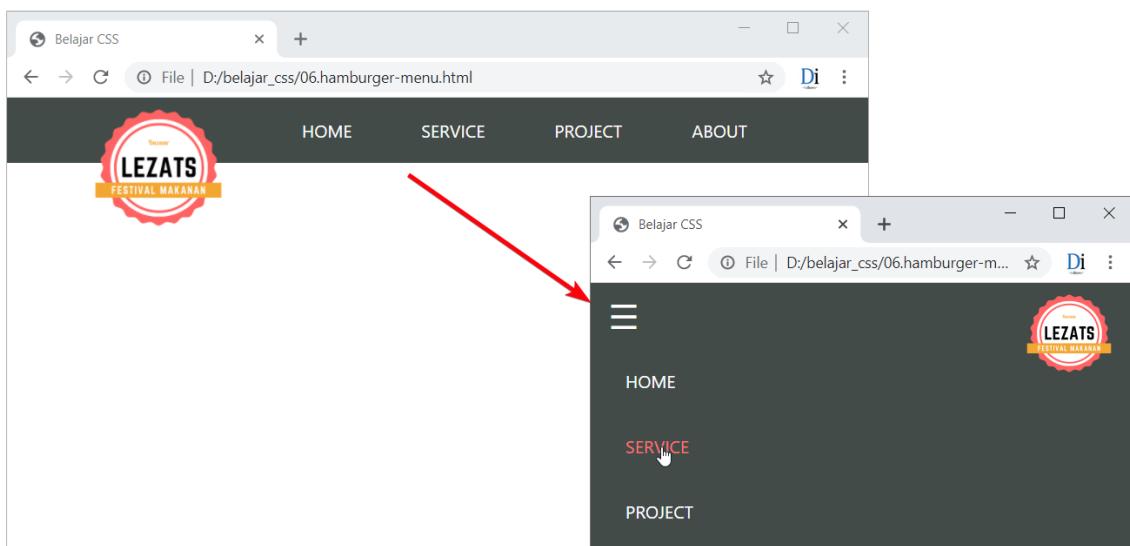
Salah satu efek paling sulit ketika membuat tampilan responsive adalah di bagian menu navigasi. Di studi kasus buku CSS Uncover versi sebelumnya, saya mengakali ini dengan menyembunyikan menu navigasi di layar smartphone, lalu menggantinya dengan tag <select>. Cara ini cukup praktis meskipun terkesan sedikit jadul.



Gambar: Mengganti menu navigasi dengan tag <select>

Untuk tampilan yang lebih modern, kebanyakan website memakai **hamburger menu** di versi smartphone. Istilah "hamburger" merujuk ke tiga garis berdempet yang mirip dengan hamburger. Ketika icon hamburger ini di klik, akan tampil menu vertikal yang tersusun dari atas ke bawah.

Setelah melihat beberapa referensi, saya menemukan cara membuat hamburger menu dari CSS saja tanpa JavaScript, meskipun kodenya juga tidak bisa dibilang sederhana. Berikut tampilan hamburger menu yang akan kita rancang:



Gambar: Tampilan hamburger menu pada versi smartphone

Trik yang dipakai adalah dengan menyamarkan sebuah checkbox, yakni tag `<input type="checkbox">` sebagai icon hamburger, lalu memanfaatkan selector `input:checked` untuk memeriksa apakah checkbox tersebut sudah diklik atau tidak.

Berikut struktur kode HTML yang diperlukan:

06.hamburger-menu.html

```

1 <div class="container">
2   <header>
3     <label for="hamburger">☰</label>
4     <input type="checkbox" id="hamburger"/>
5     <div class="logo">
6       
7     </div>
8     <nav>
9       <ul>
10      <li><a href="#" class="active">Home</a></li>
11      <li><a href="#">Service</a></li>
12      <li><a href="#">Project</a></li>
13      <li><a href="#">About</a></li>
14    </ul>
15  </nav>
16 </header>
17 </div>

```

Struktur ini mirip seperti header kita sebelumnya, hanya saja sekarang terdapat tambahan kode untuk membuat icon hamburger di baris 3, serta sebuah `<input type="checkbox">` di baris 4. Kode `☰`; adalah HTML entity yang akan ditampilkan sebagai garis tiga.

Di buku HTML Uncover saya pernah bahas bahwa tag `<label>` berfungsi sebagai penanda dari tag `<input>`. Untuk menghubungkan keduanya, atribut `for` milik `<label>`, diisi dengan nama yang sama dengan atribut `id` milik tag `<input>`.

Pada contoh ini, tag `<label for="hamburger">` akan berpasangan dengan tag `<input type="checkbox" id="hamburger"/>`. Maka ketika teks yang ada di dalam label diklik, secara otomatis checkbox juga akan terpilih meskipun ada jarak antara keduanya. Jalankan kode HTML di atas untuk melihat efek ini:



Gambar: Menguji klik checkbox hamburger menu

Tampilan halaman memang masih berantakan karena kita memiliki kode CSS apapun. Akan

tetapi icon hamburger, checkbox, logo serta struktur menu navigasi sudah tampil. Ketika icon hamburger di klik, checkbox juga ikut terpilih sebagai hasil dari penggunaan tag <label>.

Dan berikut kode CSS lengkap untuk mengaktifkan hamburger menu di layar smartphone:

07.hamburger-menu-full.html

```

1  *{
2      box-sizing: border-box;
3  }
4  html, body {
5      padding: 0;
6      margin: 0;
7  }
8  .container {
9      background-color: #3c4542;
10     position: relative;
11 }
12 header {
13     display: flex;
14     flex-direction: row;
15     justify-content: space-between;
16     width: 80vw;
17     height: 60px;
18     margin: 0 auto;
19 }
20 .logo img {
21     width: 120px;
22     padding-top: 10px;
23 }
24 ul {
25     font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
26     list-style-type: none;
27     margin: 0;
28     padding: 0;
29     display: flex;
30 }
31 li a {
32     display: block;
33     color: white;
34     padding: 0 2em;
35     line-height: 60px;
36     text-decoration: none;
37     text-transform: uppercase;
38     height: 60px;
39 }
40 li a:hover {
41     color: #FF5C5C;
42     transition: all 0.3s;
43     border-bottom: 5px solid #FF5C5C;
44 }
45
46 /* Sembunyikan hamburger icon */
47 header label, #hamburger {

```

```
48     display: none;
49 }
50
51 @media screen and (max-width: 768px){
52     /* Tampilkan hamburger icon */
53     header label {
54         display: inline-block;
55         color: white;
56         font-style: normal;
57         font-size: 2rem;
58         padding: 0.5rem;
59         padding-left: 1rem;
60     }
61
62     /* Susun ulang menu menjadi vertikal */
63     header {
64         width: 100%;
65     }
66     nav ul {
67         flex-direction: column;
68         width: 100%;
69         position: absolute;
70         top: 60px;
71         left: 0;
72     }
73     li a {
74         display: block;
75         color: white;
76         background-color: #3c4542;
77         padding: 0 2em;
78         text-decoration: none;
79         text-transform: uppercase;
80         height: auto;
81     }
82     li a:hover {
83         color: #FF5C5C;
84         border-bottom: none;
85     }
86
87     /* Pindahkan posisi logo ke kanan, dan perkecil */
88     .logo {
89         position: absolute;
90         right: 20px;
91         z-index: 1000;
92     }
93     .logo img {
94         width: 80px;
95     }
96
97     /* Untuk menampilkan / menyembunyikan menu */
98     nav { display: none; }
99     header input:checked ~ nav {
100         display: flex;
101     }
102 }
```

Kode yang diperlukan memang sangat panjang. Di sini kita sedang membuat 2 jenis tampilan, yakni horizontal menu untuk versi layar lebar (baris 1 - 44), serta hamburger menu untuk versi layar smartphone (baris 51 - 102).

Perintah untuk membuat horizontal menu sama persis seperti praktek membuat header sebelumnya, oleh karena itu tidak akan saya bahas lagi.

Tambahannya ada di baris 47, yakni selector `header label, #hamburger {display: none;}` yang berfungsi untuk menyembunyikan tag `<label>` dan tag `<input id="hamburger">`. Kedua tag ini tidak diperlukan untuk menu versi desktop.

Di baris 51, terdapat perintah media query yang akan aktif ketika lebar web browser sudah lebih kecil dari 768 pixel, inilah tampilan untuk versi smartphone.

Selector `header label` pada baris 53 – 60 dipakai untuk mengatur posisi icon hamburger. Icon ini sebelumnya sudah kita `hidden`, maka perlu ditampilkan kembali dengan `display: inline-block`. Saya juga memperbesar icon hamburger dengan property `font-size: 2rem`, mengubahnya ke warna putih, serta mengatur padding.

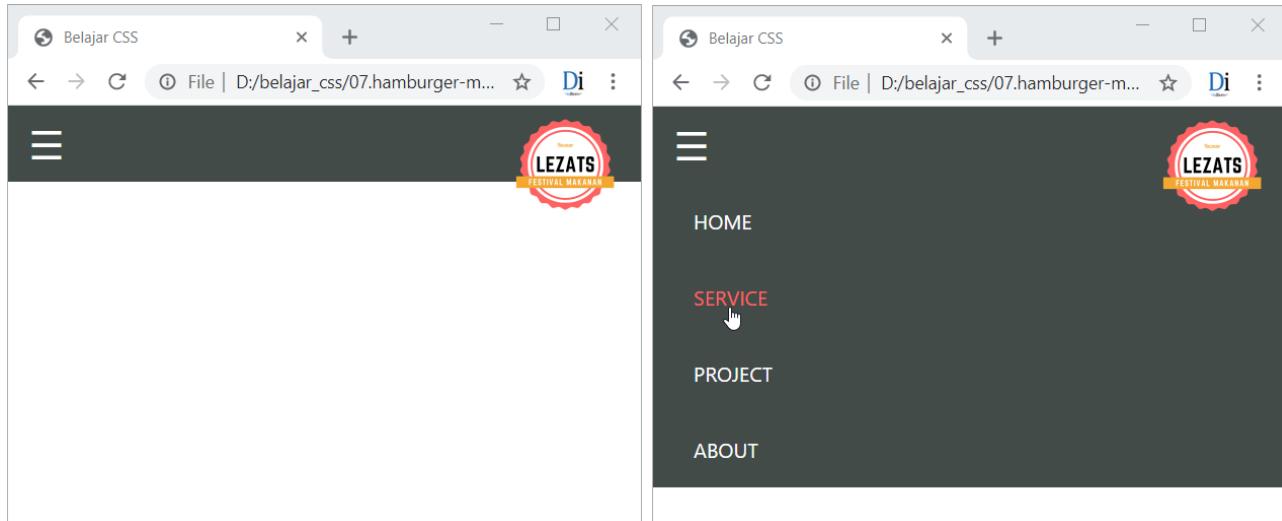
Kode antara baris 63 – 85 berfungsi untuk menyusun ulang menu navigasi yang sebelumnya tampil secara horizontal menjadi vertikal. Prinsip yang dipakai adalah menukar nilai property `flex-direction` milik `nav ul` dari sebelumnya `row`, menjadi `column`. Posisi menu ini juga di geser sedikit ke bawah dengan property `position: absolute` dan `top: 60px` karena dibagian atas sudah ditempati oleh icon hamburger.

Ketika tampil dalam bentuk hamburger menu, saya juga ingin gambar logo pindah ke sebelah kanan, sebab posisi sebelumnya di bagian kiri sudah ditempati icon hamburger. Perubahan posisi ini dilakukan dengan tambahan property `position: absolute` dan `right: 20px` untuk selector `.logo`. Yang berarti gambar logo akan berada sekitar 20px dari sisi kanan header. Property `z-index: 1000` berfungsi agar gambar logo berada di lapisan paling atas, tidak tertimpa oleh header.

Proses menampilkan dan menyembunyikan hamburger menu dilakukan pada baris 98 – 101. Di awal, tag `<nav>` akan di `hidden` terlebih dahulu menggunakan selector `nav {display: none;}`.

Kemudian saya ingin tag `<nav>` tampil kembali ketika icon hamburger di klik. Ini lah fungsi dari pseudo selector `header input:checked ~ nav {display: flex;}`. Selector ini bisa dibaca: "ubah property `display` milik tag `<nav>` menjadi `flex` jika tag `<input>` yang berada di dalam tag `<header>` di ceklist".

Dengan kode CSS ini, kita sudah sukses membuat hamburger menu:



Gambar: Tampilan hamburger menu saat hidden dan tampil

22.6. Membuat Hero Image

Selesai dengan menu navigasi, kita berangkat ke komponen web yang biasa disebut sebagai **hero image**. **Hero image** adalah gambar besar yang tampil di halaman home sebuah website. Biasanya gambar ini dipakai untuk membuat efek "wah" atau menampilkan info terbaru. Berikut tampilan yang akan dibuat:



Gambar: Tampilan hero image

Konsep dasar pembuatan hero image adalah menampilkan gambar background full size, lalu tambah pesan teks di atasnya. Pemilihan bentuk gambar dan jenis font yang tepat sangat berpengaruh ke segi estetika supaya tampil menarik.

Berikut kode HTML yang diperlukan:

08.hero.html

```

1 <div class="hero-container">
2   <div class="hero-text">
3     <h1>Baazar LEZATS</h1>
4     <p>Festival Makanan Terbesar di Indonesia</p>
5     <a href="">Daftar Sekarang</a>
6   </div>
7 </div>

```

Struktur yang cukup sederhana. Tag `<div class="hero-container">` di sisi luar berperan sebagai container, dan di sinilah gambar background akan ditambah. Kemudian tag `<div class="hero-text">` berfungsi untuk menampung teks yang tampil di atas hero image.

Berikut kode CSS yang diperlukan untuk membuatnya:

08.hero.html

```

1 * {
2   box-sizing: border-box;
3 }
4 html, body {
5   padding: 0;
6   margin: 0;
7   font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
8 }
9 .hero-container {
10   background-image: linear-gradient(rgba(0, 0, 0, 0.1),
11                                     rgba(0, 0, 0, 0.3)), url("img/hero.jpg");
12   height: 100vh;
13   background-position: center;
14   background-repeat: no-repeat;
15   background-size: cover;
16   display: flex;
17   align-items: center;
18   justify-content: center;
19 }
20 .hero-text {
21   text-align: center;
22   color: white;
23 }
24 .hero-text h1{
25   font-size: 4em;
26   margin: 0;
27   text-shadow: 4px 4px black;
28 }
29 .hero-text p{
30   font-size: 2em;
31   margin: 0;
32   text-shadow: 2px 2px black;
33 }
34 .hero-text a {
35   text-decoration: none;
36   display: inline-block;

```

```

37 margin-top: 1.3em;
38 padding: 0.7em 1.2em;
39 background-color: orangered;
40 color: white;
41 box-shadow: 5px 5px black;
42 font-size: 1.1em;
43 }
44 .hero-text a:hover {
45   background-color: #ff6933;
46 }

```

Kode di baris 1 – 8 sudah sering kita pakai, yakni men-set semua element sebagai `box-sizing: border-box`, serta menghapus padding dan margin bawaan tag `<html>` dan tag `<body>`.

Proses input gambar dilakukan dari selector `.hero-container` pada baris 9 – 19. Property `background-image` saya isi dengan gradient abu-abu transparan dari kode warna `rgba(0, 0, 0, 0.1)` sampai `rgba(0, 0, 0, 0.3)`. Perhatikan bahwa kode warna `rgb(0, 0, 0)` sama-sama mewakili warna hitam, perbedaan hanya di angka *alpha channel* untuk membuat transparasi warna gradient dari 0.1 ke 0.3.

Efek gradient ini diperlukan agar kita bisa mengatur tingkat kecerahan dari `img/hero.jpg`. Ketika sebuah teks berada di atas gambar, teks tersebut harus terlihat kontras. Dengan mengatur angka *alpha channel*, gambar hero bisa kita buat sedikit lebih gelap. Sebelum hadirnya fitur gradient di CSS3, transparansi gambar terpaksa diatur menggunakan photoshop, kemudian baru dipakai sebagai hero image.

Tinggi hero image saya set dengan `height: 100vh`, sehingga akan memenuhi seluruh tinggi layar. Anda bebas ingin menggantinya dengan angka lain jika dirasa terlalu besar.

Tiga property selanjutnya berfungsi untuk mengatur posisi gambar background, yakni `background-position: center`, `background-repeat: no-repeat` dan `background-size: cover`. Fungsi dari ketiga property ini sudah kita bahas pada bab tentang background. Karena gambar hero image lumayan besar, kita juga harus menyediakan gambar beresolusi tinggi agar tidak pecah.

Lanjut, tiga property flexbox dipakai untuk mengatur teks agar berada pas di tengah-tengah hero image, yakni `display: flex`, `align-items: center` dan `justify-content: center`.

Setelah itu sisa selector lain dipakai untuk mengatur tampilan `.hero-text`, seperti mengubah warnanya menjadi putih, membesarkan ukuran font, mengatur padding dan margin serta membuat efek text-shadow. Karena hanya berisi property dasar, tidak perlu kita bahas lagi.

Selain menggelapkan hero image agar teks bisa terlihat jelas, cara alternatif adalah dengan memberikan warna dasar gelap hanya di bagian teks saja, misalnya seperti tampilan berikut:



Gambar: Tampilan alternatif hero image

Untuk membuatnya, hapus efek gradient dari hero image, lalu tambah property `background-color` berwarna hitam transparan ke dalam `.hero-text`. Padding milik `.hero-text` juga perlu sedikit penyesuaian agar lebih rapi. Berikut bagian kode program yang perlu kita modifikasi:

09.hero-2.html

```
1 .hero-container {  
2   background-image: url("img/hero.jpg");  
3   ...  
4 }  
5 .hero-text {  
6   background-color: rgba(0, 0, 0, 0.5);  
7   padding: 2em;  
8   ...  
9 }
```

Beberapa web menggunakan slideshow sebagai hero-image. Untuk membuatnya tentu butuh kode CSS yang lebih kompleks lagi, termasuk tambahan perintah JavaScript. Umumnya web desainer lebih memilih menggunakan library atau framework seperti Bootstrap untuk membuatnya.

22.7. Membuat Showcase

Tidak ada istilah khusus untuk menyebut komponen yang akan kita buat ini. Saya tulis sebagai **showcase** karena biasanya diisi dengan penjelasan dari keunggulan produk atau jasa yang ada di suatu website. Berikut tampilan yang dimaksud:



Gambar: Tampilan komponen showcase

Dengan melihat sekilas tampilan ini, semoga anda bisa memperkirakan apa kode yang harus kita pakai. Pembagian gambar di kiri dan teks di kanan bisa dibuat menggunakan flexbox, selain itu gambar perlu tambahan bingkai serta sedikit efek bayangan.

Berikut kode HTML yang diperlukan:

10.showcase.html

```
1 <section class="showcase">
2   <div class="showcase-image">
3     
4   </div>
5   <div class="showcase-text">
6     <h1>Lorem ipsum dolor sit amet consectetur</h1>
7     <p>Perferendis doloribus adipisci maxime rerum iure, ex ad natus
8       consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum
9       voluptates quis? Quam nihil sequi dicta neque ullam ab tempore
10      aspernatur, culpa maxime labore dolore!</p>
11      <p>Molestiae, sequi illo optio vitae error enim earum?</p>
12    </div>
13 </section>
```

Tag `<section class="showcase">` berperan sebagai container dengan 2 buah child, yakni tag `<div class="showcase-image">` tempat gambar berada, serta tag `<div class="showcase-text">` yang berisi teks penjelasan.

Dan berikut kode CSS untuk mengatur tampilannya:

10.showcase.html

```
1 * {
2   box-sizing: border-box;
3 }
4 html, body, h1 {
```

```

5   padding: 0;
6   margin: 0;
7   font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
8 }
9 .showcase {
10  display: flex;
11  justify-content: space-between;
12  max-width: 1100px;
13  margin: 0 auto;
14  padding: 2em;
15 }
16 .showcase-image, .showcase-text{
17  width: 50%;
18  margin: 1em;
19 }
20 .showcase-image img{
21  width: 100%;
22  border: 1px solid silver;
23  padding: 0.3rem;
24  box-shadow: 2px 2px 5px black;
25 }
26
27 @media screen and (max-width: 768px){
28  .showcase {
29   flex-direction: column;
30   padding: 1em 2em ;
31  }
32  .showcase-image, .showcase-text{
33   width: auto;
34  }
35 }

```

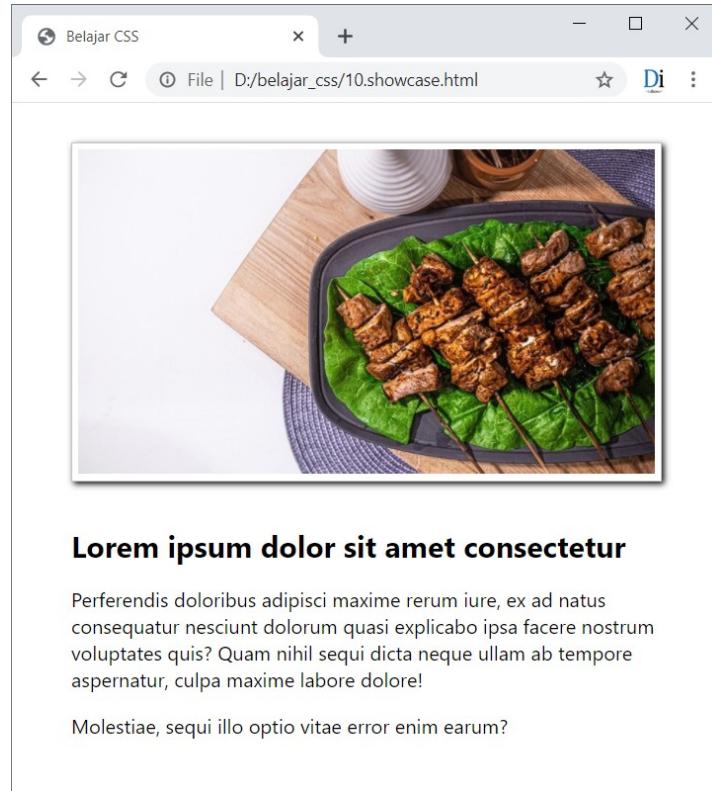
Di baris 9 – 15 saya menambah property `display: flex` dan `justify-content: space-between` ke dalam selector `.showcase`, tujuannya agar gambar dan teks saling bersebelahan satu sama lain.

Lebar `.showcase` diatur dengan property `max-width: 1100px` agar responsive, dimana lebar maksimal showcase ada di 1100px, namun tetap bisa mengecil. Property `margin: 0 auto` berguna supaya posisi `.showcase` berada di tengah-tengah halaman, serta mengatur padding sebesar 2em.

Melihat kembali struktur HTML, tag `<section class="showcase">` punya 2 buah child element, yakni `<div class="showcase-image">` dan `<div class="showcase-text">`. Lebar keduanya saya set 50% menggunakan property `width: 50%` di baris 17. Ini berarti container `.showcase` akan berbagi ruang sama besar untuk gambar dan teks.

Selector `.showcase-image img` di baris 20 – 25 berisi style untuk mempercantik tampilan gambar, yakni berupa tambahan `border`, `padding` dan `box-shadow`. Property `width: 100%` memegang peranan penting untuk membuat gambar menjadi responsive, yakni ikut membesar / mengecil sesuai lebar parent element.

Di baris 27 – 35, terdapat perintah media query untuk breakpoint smartphone. Jika lebar web browser kurang dari 768px, set property `flex-direction` milik `.showcase` menjadi `column`, serta lebar `.showcase-image`, `.showcase-text` menjadi `width: auto`. Ini akan membuat tampilan showcase dengan gambar di sisi atas, serta teks di sisi bawah:



Gambar: Tampilan showcase di breakpoint smartphone

Kode CSS yang sudah dibuat sebenarnya juga saya siapkan jika kita punya 2, 3 atau lebih struktur showcase. Sebagai contoh dalam kode ini saya membuat 2 buah showcase:

11.showcase-2.html

```
1 <section class="showcase">
2   <div class="showcase-image">
3     
4   </div>
5   <div class="showcase-text">
6     <h1>Lorem ipsum dolor sit amet consectetur</h1>
7     <p>Perferendis doloribus adipisci maxime rerum iure, ex ad natus
8       consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum
9       voluptates quis? Quam nihil sequi dicta neque ullam ab tempore
10      aspernatur, culpa maxime labore dolore!</p>
11      <p>Molestiae, sequi illo optio vitae error enim earum?</p>
12    </div>
13  </section>
14
15 <section class="showcase">
16   <div class="showcase-text">
17     <h1>Lorem ipsum dolor sit amet consectetur</h1>
```

```

18 <p>Perferendis doloribus adipisci maxime rerum iure, ex ad natus  

19    consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum  

20    voluptates quis? Quam nihil sequi dicta neque ullam ab tempore  

21    aspernatur, culpa maxime labore dolore!</p>  

22 <p>Molestiae, sequi illo optio vitae error enim earum?</p>  

23 </div>  

24 <div class="showcase-image">  

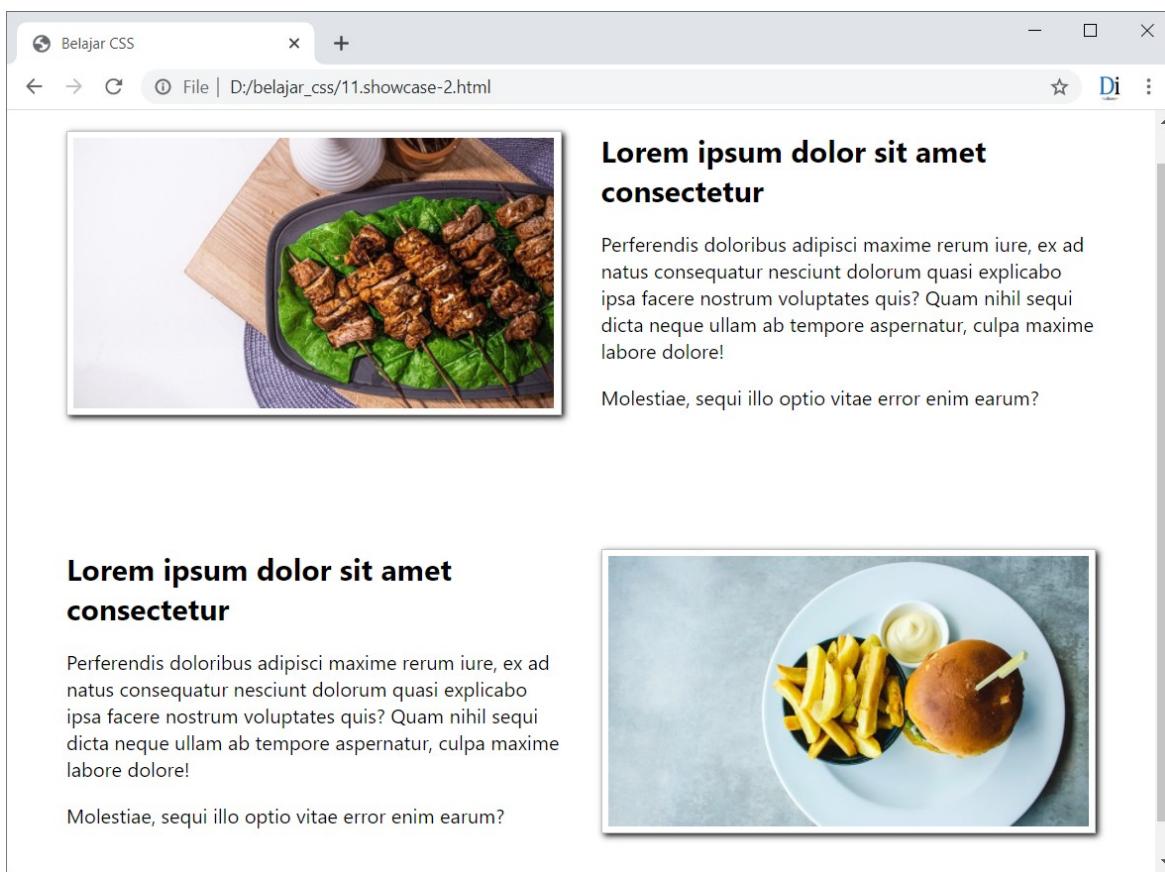
25     

26 </div>  

27 </section>

```

Di baris 15 – 27, terdapat komponen showcase kedua. Nama class yang digunakan tetap sama, hanya saja sekarang posisi gambar dibalik dan berada setelah teks. Tanpa mengubah satu pun kode CSS, struktur showcase ini sudah langsung ter-style:



Gambar: Tampilan 2 buah showcase

Atau agar lebih menarik, saya bisa memberi warna background berbeda untuk setiap showcase dengan tambahan kode CSS berikut;

```

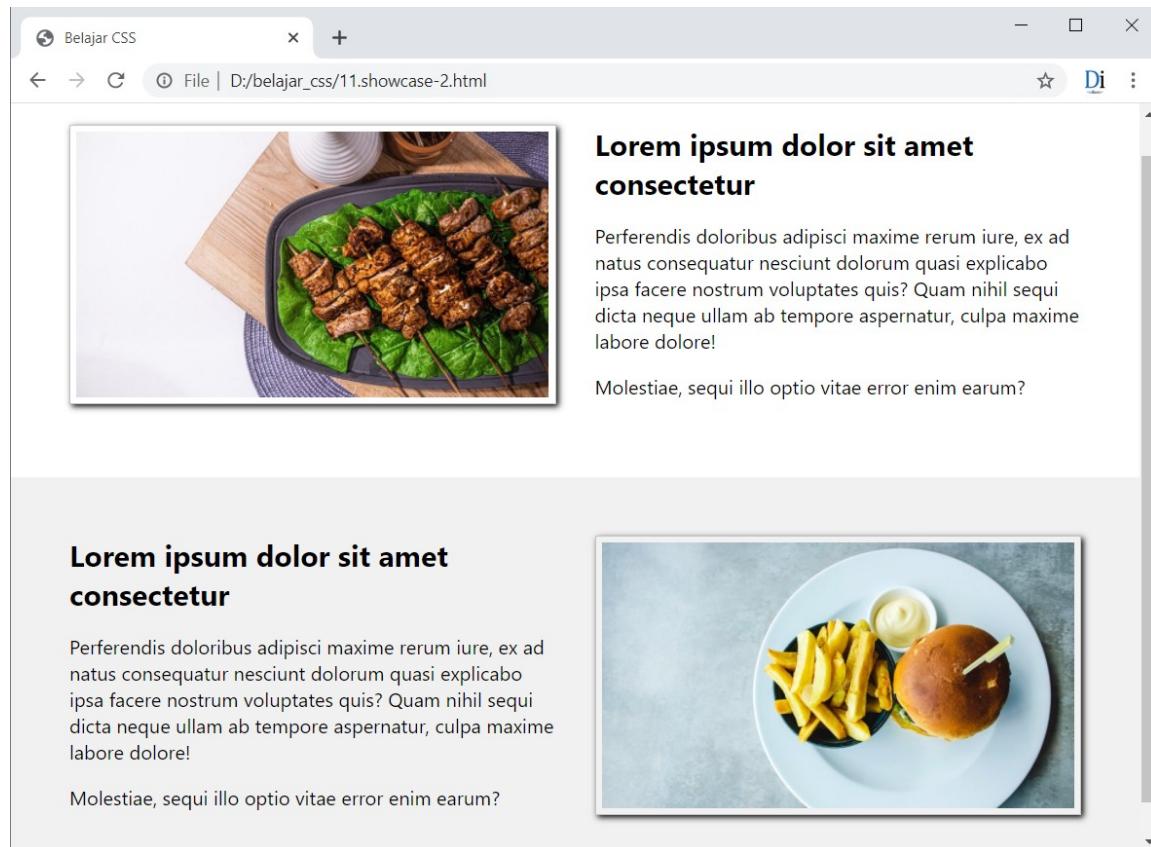
1 .showcase:nth-child(even) {  

2   background-color: #f0f0f0;  

3 }

```

Hasilnya, showcase yang ada di urutan genap akan memiliki background berwarna abu-abu:



Gambar: Tampilan 2 buah showcase dengan warna background

22.8. Membuat Product and Services

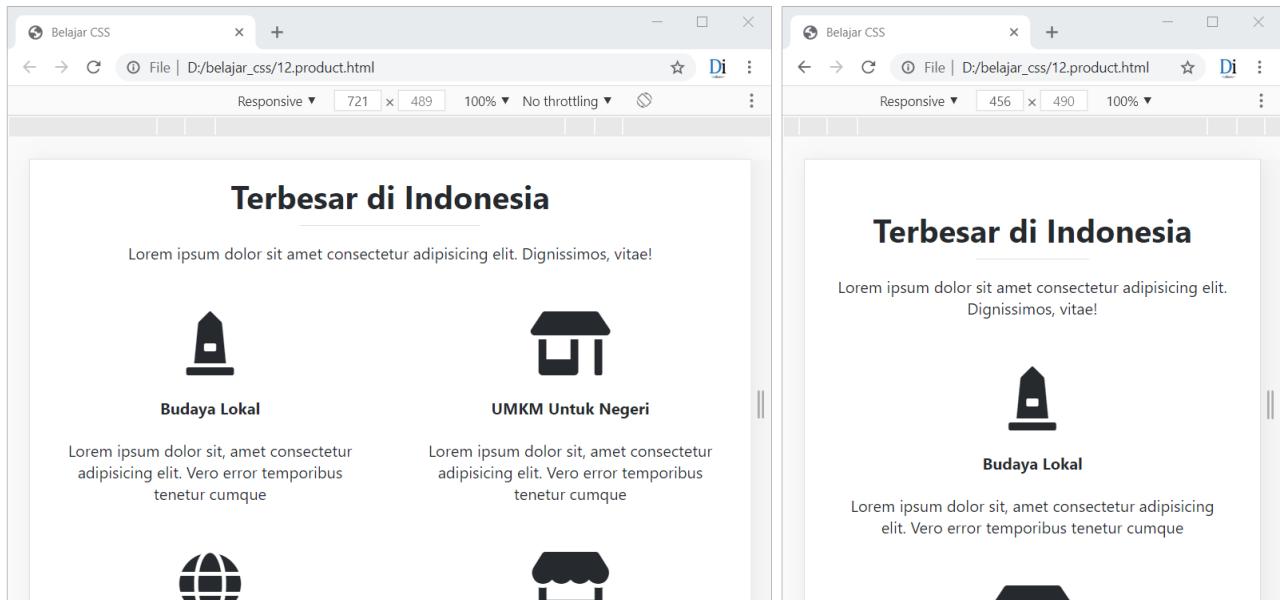
Bagian yang akan kita rancang ini juga tidak memiliki nama khusus, saya sebut sebagai *product and services* karena bisa dipakai untuk menampilkan jenis-jenis produk atau jasa:

A screenshot of a web browser window titled "Belajar CSS". The address bar shows "File | D:/belajar_css/12.product.html". The page title is "Terbesar di Indonesia". Below it is a short paragraph of placeholder text: "Lorem ipsum dolor sit amet consectetur adipisicing elit. Dignissimos, vitae!". There are four cards, each with an icon and text: 1. "Budaya Lokal" with a traditional lamp icon. 2. "UMKM Untuk Negeri" with a small shop icon. 3. "Go Internasional" with a globe icon. 4. "1000 Peserta" with a storefront icon. Each card also has a short paragraph of placeholder text below it.

Gambar: Tampilan komponen product and services

Komponen ini terdiri dari judul dibagian atas, lalu 4 gambar serta teks penjelasan yang berjejer dibagian bawah. Gambar *product and services* di buat menggunakan icon dari font-awesome.

Sebagai efek responsive, gambar icon akan tampil dalam 4, 2 atau 1 gambar dalam setiap baris. CSS flexbox sangat membantu kita dalam membuat tampilan seperti ini:



Gambar: Tampilan untuk versi tablet (kiri) dan smartphone (kanan)

Baik, berikut struktur kode HTML yang diperlukan:

12.product.html

```
1 ...  
2 <link href="fontawesome/css/all.min.css" rel="stylesheet">  
3 ...  
4 <div class="product-container">  
5   <header>  
6     <h1>Terbesar di Indonesia</h1>  
7     <hr>  
8     <p>Lorem ipsum dolor sit amet consectetur  
9       adipisicing elit. Dignissimos, vitae!</p>  
10    </header>  
11  
12    <section>  
13      <div>  
14        <i class="fas fa-monument fa-4x"></i>  
15        <h4>Budaya Lokal</h4>  
16        <p>Lorem ipsum dolor sit, amet consectetur  
17          adipisicing elit. Vero error temporibus tenetur cumque</p>  
18      </div>  
19  
20      <div>  
21        <i class="fas fa-store-alt fa-4x"></i>  
22        <h4>UMKM Untuk Negeri</h4>  
23        <p>Lorem ipsum dolor sit, amet consectetur
```

```

24      adipisicing elit. Vero error temporibus tenetur cumque</p>
25  </div>
26
27  <div>
28      <i class="fas fa-globe fa-4x"></i>
29      <h4>Go Internasional</h4>
30      <p>Lorem ipsum dolor sit, amet consectetur
31          adipisicing elit. Vero error temporibus tenetur cumque</p>
32  </div>
33
34  <div>
35      <i class="fas fa-store fa-4x"></i>
36      <h4>1000 Peserta</h4>
37      <p>Lorem ipsum dolor sit, amet consectetur
38          adipisicing elit. Vero error temporibus tenetur cumque</p>
39  </div>
40 </section>
41 </div>

```

Karena kita akan memakai icon font awesome, maka di bagian `<head>` perlu ditambah tag `<link href="fontawesome/css/all.min.css">` untuk mengakses file CSS font awesome.

Seluruh kode berada di dalam container `.product-container` yang memiliki 2 child element, yakni tag `<header>` untuk judul, serta tag `<section>` untuk gambar icon.

Pemilihan gambar icon diakses dari class milik font awesome di dalam tag `<i>`. Sebagai contoh, tag `<i class="fas fa-monument fa-4x">` akan menampilkan icon monument yang diperbesar sebanyak 4x. Lebih jauh tentang cara penggunaan font awesome sudah kita bahas dalam bab sebelumnya.

Silahkan pelajari sejenak struktur HTML di atas karena akan di style dengan kode CSS berikut:

12.product.html

```

1  * {
2      box-sizing: border-box;
3  }
4  html, body {
5      padding: 0;
6      margin: 0;
7      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
8      color: #212529;
9      text-align: center;
10 }
11 .product-container {
12     max-width: 1100px;
13     margin: 3rem auto;
14     text-align: center;
15 }
16 header hr {
17     width: 25%;
18     border: 0;
19     border-top: 1px solid #ddd;

```

```

20 }
21 section {
22   margin-top: 2rem;
23   display: flex;
24   flex-flow: row wrap;
25 }
26 section div {
27   width: 25%;
28   padding: 1em 2em;
29 }
30 header h1 {
31   font-size: 2rem;
32   margin-bottom: 0;
33 }
34
35 @media screen and (max-width: 768px){
36   section div {
37     width: 50%;
38   }
39 }
40
41 @media screen and (max-width: 461px){
42   section div {
43     width: 100%;
44   }
45 }
```

Di baris 11, class .product-container di set dengan max-width: 1100px dan margin: 3rem auto. Ini akan membuat tampilan komponen *product and services* pas di tengah-tengah halaman. Dengan property max-width: 1100px, maka lebar maksimal element dibatasi 1100px, namun tetap bisa mengecil sesuai lebar layar.

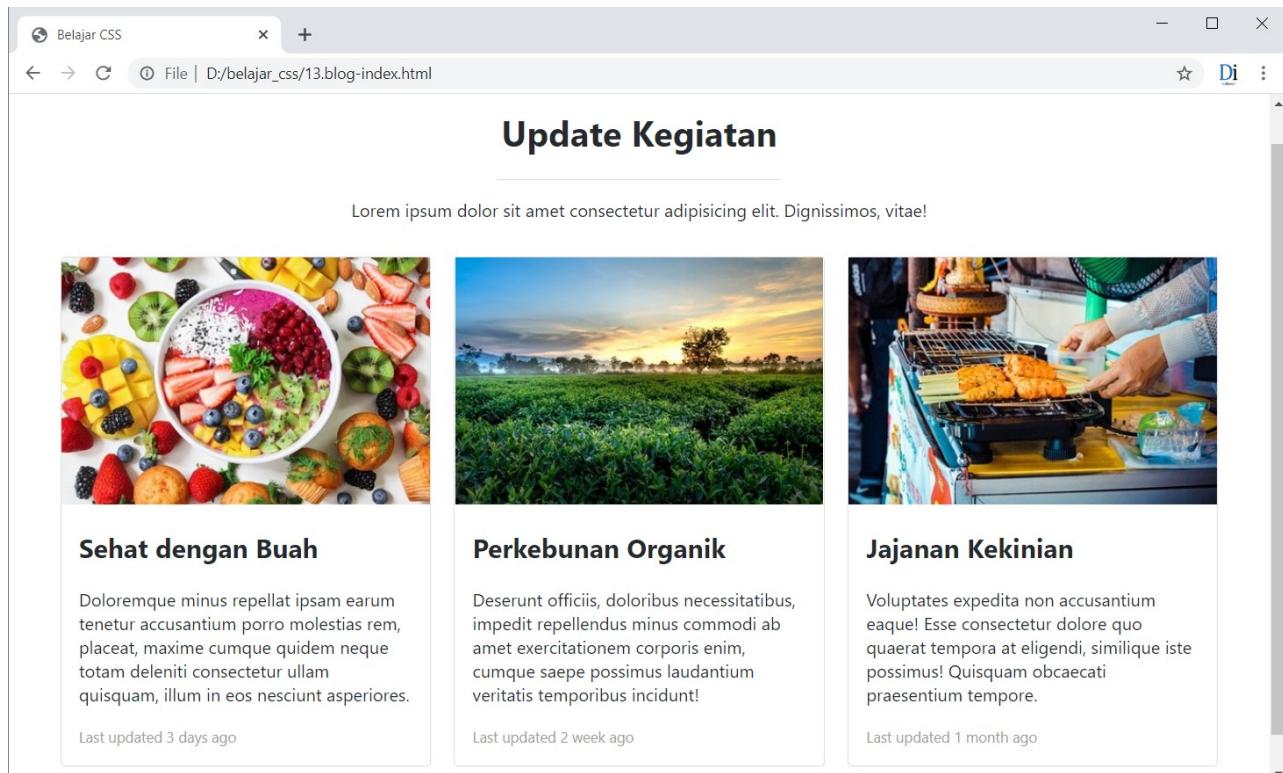
Selector header hr di baris 16 dipakai untuk mempercantik tampilan tag <hr> bawaan HTML. Trik ini terinspirasi dari kode yang digunakan framework Bootstrap, dimana border bawaan tag <hr> di 0-kan dan diganti dengan border-top: 1px solid #ddd.

Bagian yang perlu property flexbox ada di selector section, karena inilah yang berperan sebagai container dari 4 gambar icon. Efek flexbox diaktifkan dengan property display: flex dan flex-flow: row wrap.

Setiap gambar icon berada dalam selector section div. Untuk tampilan desktop, lebar div di set menjadi 25% (baris 27). Kemudian ketika masuk ke breakpoint tablet, lebarnya menjadi 50% (baris 37), dan berubah ke 100% untuk breakpoint smartphone (baris 43). Inilah cara membuat efek responsive untuk tampilan *product and services*.

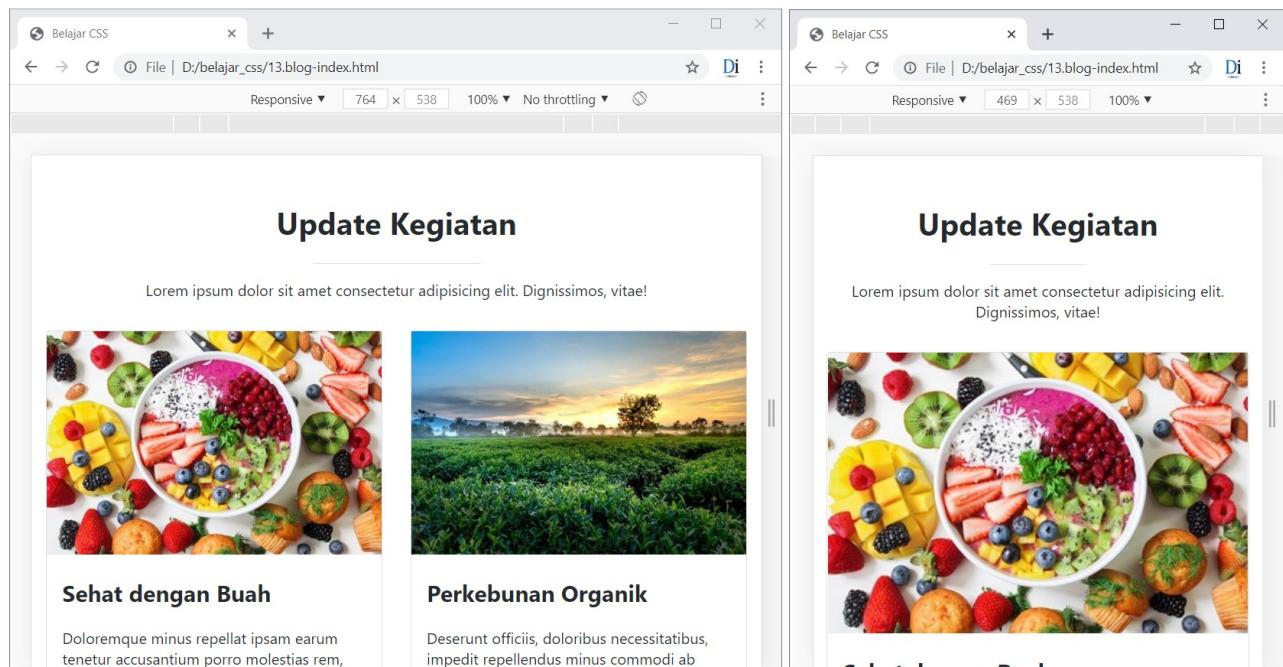
22.9. Membuat Blog Index

Komponen *blog index* berisi cuplikan artikel atau blog yang ada di sebuah website. Berikut tampilan yang dimaksud:



Gambar: Tampilan komponen blog index

Sebenarnya ini modifikasi dari komponen *product and services* yang sudah kita buat sebelumnya. Dibagian atas terdapat sedikit teks pengantar, diikuti 3 cuplikan blog di bagian bawah. Agar lebih menarik, setiap cuplikan blog memiliki gambar dan teks di dalam box. Untuk efek responsive, jumlah blog menjadi 2 atau 1 dalam satu baris tergantung lebar layar:



Gambar: Tampilan blog index untuk versi tablet (kiri) dan smartphone (kanan)

Berikut kode HTML yang diperlukan:

13.blog-index.html

```
1 <div class="blog-container">
2   <header>
3     <h1>Update Kegiatan</h1>
4     <hr>
5     <p>Lorem ipsum dolor sit amet consectetur
6       adipisicing elit. Dignissimos, vitae!</p>
7   </header>
8
9   <div class="blog-index">
10
11    <div class="snippet">
12      
13      <div class="snippet-txt">
14        <h2>Sehat dengan Buah</h2>
15        <p>Doloremque minus repellat ipsam earum tenetur accusantium porro
16          molestias rem, placeat, maxime cumque quidem neque totam deleniti
17          consectetur ullam quisquam, illum in eos nesciunt asperiores.</p>
18        <small>Last updated 3 days ago</small>
19      </p>
20    </div>
21  </div>
22
23  <div class="snippet">
24    
25    <div class="snippet-txt">
26      <h2>Perkebunan Organik</h2>
27      <p>Deserunt officiis, doloribus necessitatibus, impedit repellendus
28        minus commodi ab amet exercitationem corporis enim, cumque saepe
29        possimus laudantium veritatis temporibus incident! </p>
30      <p>
31        <small>Last updated 2 week ago</small>
32      </p>
33    </div>
34  </div>
35
36  <div class="snippet">
37    
38    <div class="snippet-txt">
39      <h2>Jajanan Kekinian</h2>
40      <p>Voluptates expedita non accusantium eaque! Esse consectetur
41        dolore quo quaerat tempora at eligendi, similique iste possimus!
42        Quisquam obcaecati praesentium tempore.</p>
43      <small>Last updated 1 month ago</small>
44      </p>
45    </div>
46  </div>
47
48 </div>
49
50 </div>
```

Struktur yang saya pakai mirip seperti *product and services*, dimana tag <div class="blog-container"> berperan sebagai container dan memiliki 2 child element berupa tag <header> dan <div class="blog-index">.

Di dalam tag <div class="blog-index"> inilah terdapat 3 buah cuplikan blog dari tag <div class="snippet">. Setiap snippet memiliki gambar tag serta teks penjelasan dari tag <p>.

Langsung saja kita masuk ke kode CSS:

13.blog-index.html

```

1  * {
2      box-sizing: border-box;
3  }
4  html, body {
5      padding: 0;
6      margin: 0;
7      font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
8      color: #212529;
9  }
10 .blog-container {
11     max-width: 1100px;
12     margin: 3rem auto;
13 }
14 .blog-container header {
15     padding: 0 2em;
16     text-align: center;
17 }
18 .blog-container header hr {
19     width: 25%;
20     border: 0;
21     border-top: 1px solid #ddd;
22 }
23 .blog-index {
24     margin-top: 1rem;
25     display: flex;
26     flex-flow: row wrap;
27     justify-content:space-between;
28     padding: 0 2%;
29 }
30 .blog-index .snippet {
31     width: 32%;
32     margin-top: 1rem;
33     border: 1px solid #ddd;
34     border-radius: 0.25rem;
35 }
36 .blog-index .snippet-img {
37     width: 100%;
38 }
39 .blog-index .snippet-txt {
40     padding: 0 1em;
41 }
```

```

42 .blog-index .snippet-txt small {
43   color: #999;
44 }
45
46 @media screen and (max-width: 768px){
47   .blog-container .snippet {
48     width: 48%;
49   }
50 }
51
52 @media screen and (max-width: 481px){
53   .blog-container .snippet {
54     width: 98%;
55   }
56   .blog-index {
57     justify-content:center;
58   }
59 }

```

Kode yang dipakai juga mirip seperti *product and services*. Lebar `.blog-container` di set dengan `max-width: 1100px`, serta property `margin: 3rem auto` dipakai agar `.blog-container` tampil di tengah-tengah halaman.

Flexbox di set dari selector `.blog-index` dan setiap `.snippet` memiliki lebar 32% untuk breakpoint desktop, 48% untuk breakpoint tablet, serta 98% untuk breakpoint smartphone. Garis border `.snippet` dibuat dari property `border: 1px solid #ddd` serta `border-radius: 0.25rem` untuk membuat efek melengkung.

Agar lebih menarik lagi (sekaligus sebagai studi kasus tambahan), saya ingin membuat efek hover memutar dan zoom-out untuk gambar blog. Idenya adalah, tampilkan gambar dengan sedikit diputar dan diperbesar. Begitu terjadi hover, putar kembali gambar ke posisi normal dan kecilkan (efek zoom-out).

Untuk membuatnya, perlu mengubah sedikit struktur HTML. Kali ini setiap gambar blog akan berada di dalam sebuah container tag `<figure>`:

```

14.blog-index-style.html
1 <div class="snippet">
2   <figure>
3     
4   </figure>
5   <div class="snippet-txt">
6     ...
7   </div>
8 </div>

```

Setelah itu tambah kode CSS berikut:

14.blog-index-style.html

```
1 /* Efek untuk hover gambar */
2 .snippet figure {
3   padding: 0;
4   margin: 0;
5   width: 100%;
6   overflow: hidden;
7 }
8 .snippet-img {
9   transform: rotate(10deg) scale(1.4);
10  transition: .3s ease-in-out;
11 }
12 .snippet figure:hover img {
13   transform: rotate(0) scale(1.1);
14 }
```

Triknya adalah, saat halaman tampil pertama kali, gambar yang berada di dalam tag `` akan di ubah dengan property di baris 9, yakni `transform: rotate(10deg) scale(1.4)`. Ini akan memutar gambar 10 derajat ke kanan, dan diperbesar 1,4 kali.

Begitu gambar di hover, giliran property di baris 13 yang akan berjalan, yakni `transform: rotate(0) scale(1.1)`. Efek ini akan memutar gambar kembali ke posisi semula serta diperkecil menjadi 1.1.

Di dalam struktur HTML, tambahan tag `<figure>` berfungsi sebagai container gambar dan memiliki property `overflow: hidden`. Ini diperlukan karena gambar akan "melimpah" keluar karena kita perbesar sebanyak 1,4 kali.

Silahkan jalankan langsung file `14.blog-index-style.html` untuk melihat hasilnya:



Sehat dengan Buah

Doloremque minus repellat ipsam earum
tenetur accusantium porro molestias rem,

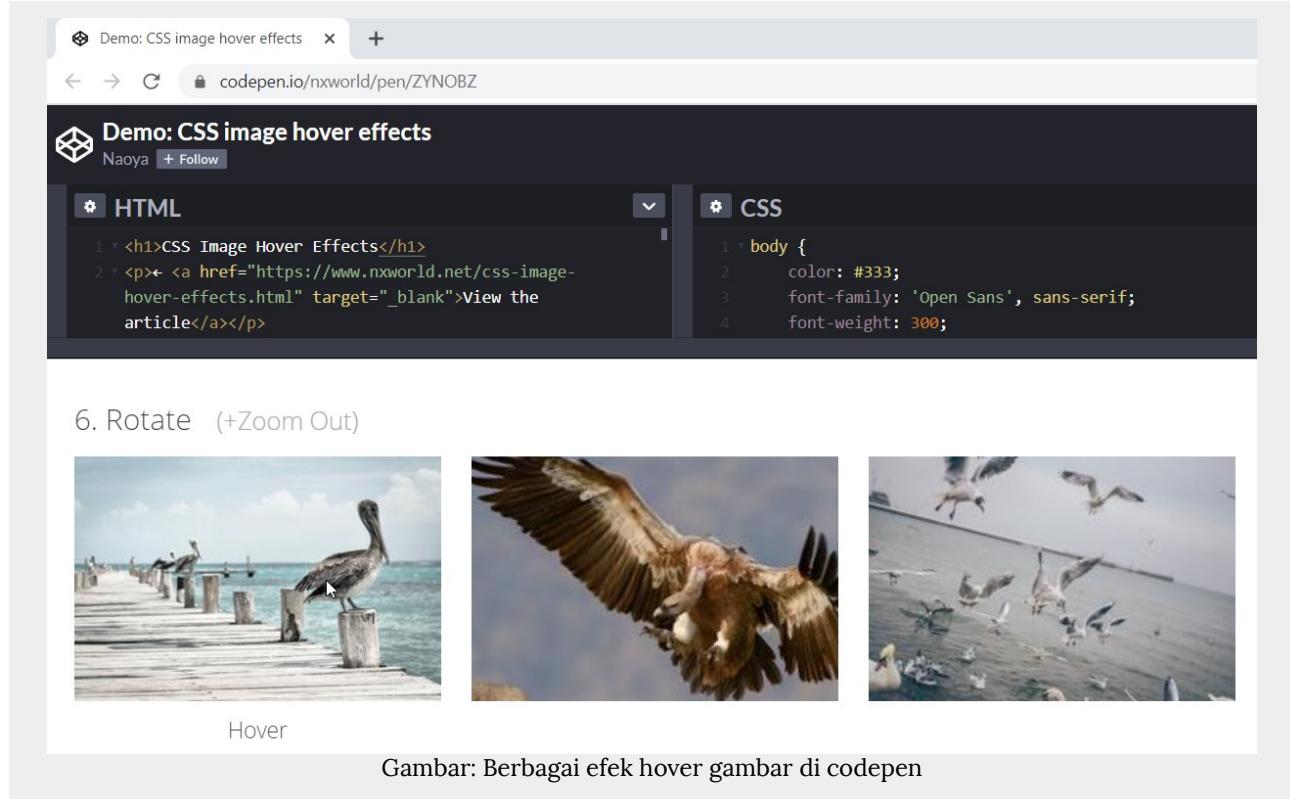
Gambar: Tampilan gambar blog sebelum di hover (kiri) dan saat di hover (kanan)



Sehat dengan Buah

Doloremque minus repellat ipsam earum
tenetur accusantium porro molestias rem,

Seperti trik CSS pada umumnya, efek di atas juga saya temukan dari internet, tepatnya di <https://codepen.io/nxworld/pen/ZYNObZ>. Di halaman tersebut tersedia kode untuk membuat efek-efek menarik lain.



The screenshot shows a CodePen demo titled "Demo: CSS image hover effects" by Naoya. The demo displays three images of birds (a pelican on a pier, a large bird in flight, and a flock of seagulls) with CSS hover effects applied. The CodePen interface shows the HTML code for the first image:

```
1 <h1>CSS Image Hover Effects</h1>
2 <p><a href="https://www.nxworld.net/css-image-hover-effects.html" target="_blank">View the article</a></p>
```

The CSS code for the body is:

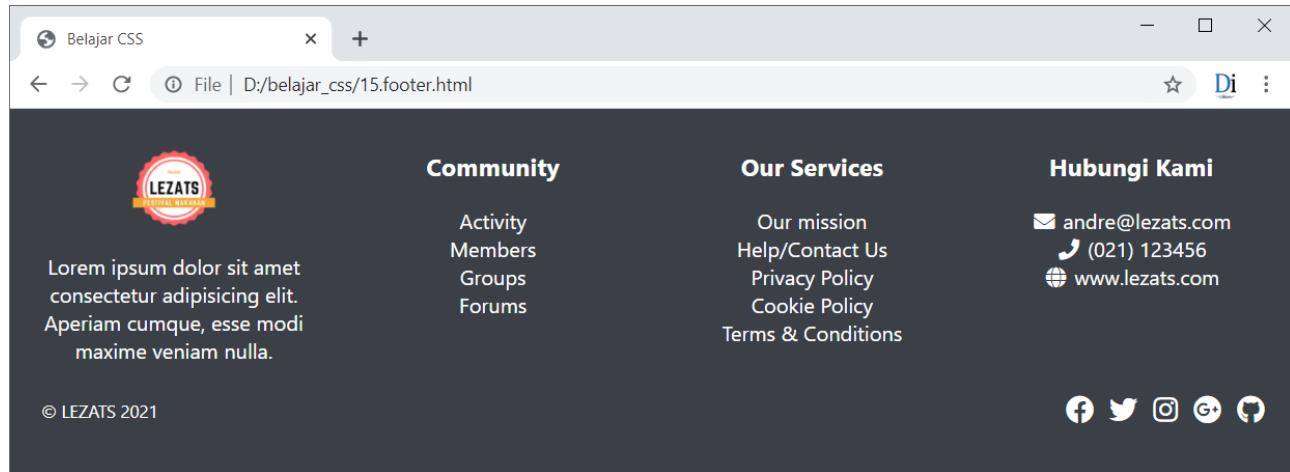
```
body {
  color: #333;
  font-family: 'Open Sans', sans-serif;
  font-weight: 300;
```

Below the images, a caption reads "6. Rotate (+Zoom Out)". The first image has a "Hover" label below it.

Gambar: Berbagai efek hover gambar di codepen

22.10. Membuat Footer

Bagian footer sebuah website biasanya menjadi tempat untuk konten tambahan yang tidak terlalu urgent seperti menu navigasi tambahan, alamat kantor, link ke akun social media serta teks copyright. Berikut tampilan footer yang akan kita rancang:



The screenshot shows a desktop browser window with a dark-themed footer. The footer is divided into four main sections: a logo on the left, and three columns on the right. The columns are labeled "Community", "Our Services", and "Hubungi Kami". Each column contains links to various site pages. Below the columns are social media icons for Facebook, Twitter, Instagram, Google+, and a messaging app.

LEZATS

Community

- Activity
- Members
- Groups
- Forums

Our Services

- Our mission
- Help/Contact Us
- Privacy Policy
- Cookie Policy
- Terms & Conditions

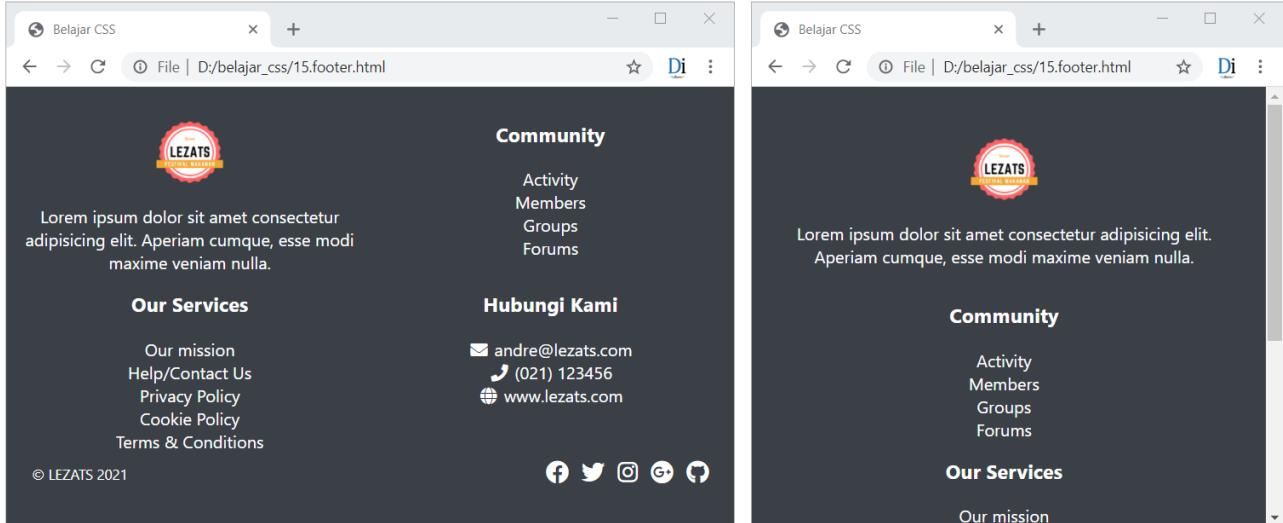
Hubungi Kami

- andre@lezats.com
- (021) 123456
- www.lezats.com

© LEZATS 2021

Gambar: Tampilan footer untuk breakpoint desktop

CSS Case Study



Gambar: Tampilan footer untuk breakpoint tablet (kiri) dan smartphone (kanan)

Dengan menganalisis tampilan yang ada, bisa disimpulkan kalau footer ini terdiri dari 4 kolom, kemudian menjadi 2 untuk versi tablet serta memanjang 1 kolom ke bawah untuk versi smartphone.

Berikut struktur HTML yang diperlukan:

15.footer.html

```
1 <footer class="footer-container">
2   <div class="footer-menu">
3     <section>
4       <div class="logo-container">
5         <a href="#">
6           
7         </a>
8       </div>
9       <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
10          Aperiam cumque, esse modi maxime veniam nulla.
11        </p>
12      </section>
13
14      <section>
15        <h3>Community</h3>
16        <ul>
17          <li><a href="#">Activity</a></li>
18          <li><a href="#">Members</a></li>
19          <li><a href="#">Groups</a></li>
20          <li><a href="#">Forums</a></li>
21        </ul>
22      </section>
23
24      <section>
25        <h3>Our Services</h3>
26        <ul>
27          <li><a href="#">Our mission</a></li>
```

```

28      <li><a href="#">Help/Contact Us</a></li>
29      <li><a href="#">Privacy Policy</a></li>
30      <li><a href="#">Cookie Policy</a></li>
31      <li><a href="#">Terms & Conditions</a></li>
32    </ul>
33  </section>
34
35  <section>
36    <h3>Hubungi Kami</h3>
37    <div><i class="fas fa-envelope fa-fw"></i> andre@lezats.com</div>
38    <div><i class="fas fa-phone fa-fw"></i> (021) 123456</div>
39    <div><i class="fas fa-globe fa-fw"></i> www.lezats.com</div>
40  </section>
41 </div>
42
43 <div class="footer-copyright">
44   <div>
45     <small>&copy; LEZATS 2021</small>
46   </div>
47   <div>
48     <a href="#"><i class="fab fa-facebook fa-lg"></i></a>
49     <a href="#"><i class="fab fa-twitter fa-lg"></i></a>
50     <a href="#"><i class="fab fa-instagram fa-lg"></i></a>
51     <a href="#"><i class="fab fa-google-plus fa-lg"></i></a>
52     <a href="#"><i class="fab fa-github fa-lg"></i></a>
53   </div>
54 </div>
55
56 </footer>

```

Seluruh footer berada di dalam tag `<footer class="footer-container">`. Tag ini memiliki 2 buah child element, yaitu `<div class="footer-menu">` dari baris 2 – 41, serta tag `<div class="footer-copyright">` dari baris 43 – 54.

Tag `<div class="footer-menu">` pada gilirannya berisi 4 buah tag `<section>` dengan konten yang berlainan. Section pertama (baris 3 – 12) dipakai untuk menampilkan gambar logo serta sedikit teks penjelasan. Section kedua dan ketiga (baris 14 – 33) berisi beberapa menu. Dan section keempat (baris 35 – 40) berisi menu yang disertai icon font awesome.

Setelah itu tag `<div class="footer-copyright">` berisi teks copyright beserta beberapa icon social media yang dibuat menggunakan font awesome.

Kita masuk ke kode CSS:

15.footer.html

```

1  * {
2    box-sizing: border-box;
3  }
4  html, body {
5    padding: 0;
6    margin: 0;
7    font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

```

CSS Case Study

```
8  }
9  .footer-container {
10   color: white;
11   background-color: #343a40;
12   padding: 2rem 1rem;
13 }
14 .footer-container a{
15   text-decoration: none;
16   color: white;
17 }
18 .footer-container a:hover{
19   text-decoration: underline;
20 }
21 .footer-menu{
22   max-width: 1100px;
23   margin: 0 auto;
24   display: flex;
25   flex-flow: row wrap;
26   justify-content: space-between;
27 }
28 .footer-menu section {
29   width: 23%;
30   text-align: center;
31 }
32 .footer-menu ul {
33   list-style-type: none;
34   margin: 0;
35   padding: 0;
36 }
37 .footer-menu h3{
38   margin-top: 0;
39 }
40 .logo-container img {
41   width: 65px;
42 }
43 .footer-copyright {
44   max-width: 1100px;
45   margin: 0 auto;
46   display: flex;
47   flex-flow: row wrap;
48   justify-content: space-between;
49   padding: 0.5rem;
50 }
51 .footer-copyright a {
52   margin-left: 0.5rem;
53 }
54 .footer-copyright a:hover {
55   text-decoration: none;
56 }
57
58 @media screen and (max-width: 900px){
59   .footer-menu section {
60     width: 48%;
61   }
62 }
```

```

63
64 @media screen and (max-width: 500px){
65   .footer-menu section {
66     width: 98%;
67     margin-top: 1rem;
68   }
69   .footer-copyright {
70     margin-top: 1rem;
71     justify-content: center;
72   }
73 }
```

Sebagian besar kode CSS ini berisi perintah untuk mengatur box model (padding, margin, dan width), serta property teks dasar seperti mengatur warna font, ukuran font, dan background-color.

Perintah flexbox ada di dua tempat, pertama di dalam selector `.footer-menu` pada baris 21–27. Property yang dipakai adalah `flex-flow: row wrap` serta `justify-content: space-between`. Ini akan mengurutkan ke-4 tag `<section>` secara horizontal dan berbagi spasi diantaranya.

Untuk breakpoint point desktop (tampilan default), setiap `.footer-menu section` akan mengambil tempat 23% seperti perintah di baris 29. Dengan demikian total ruang dari semua tag `<section>` adalah $23 \times 4 = 92\%$. Sisa 8% lain akan dipakai sebagai jarak spasi antar section.

Ketika masuk ke breakpoint tablet, lebar section di set menjadi 48% (baris 60). Akibatnya hanya ada 2 tag `<section>` yang tampil berdampingan. Terakhir untuk breakpoint smartphone, lebar section dibuat menjadi 98% agar mengambil tempat satu baris penuh (baris 66).

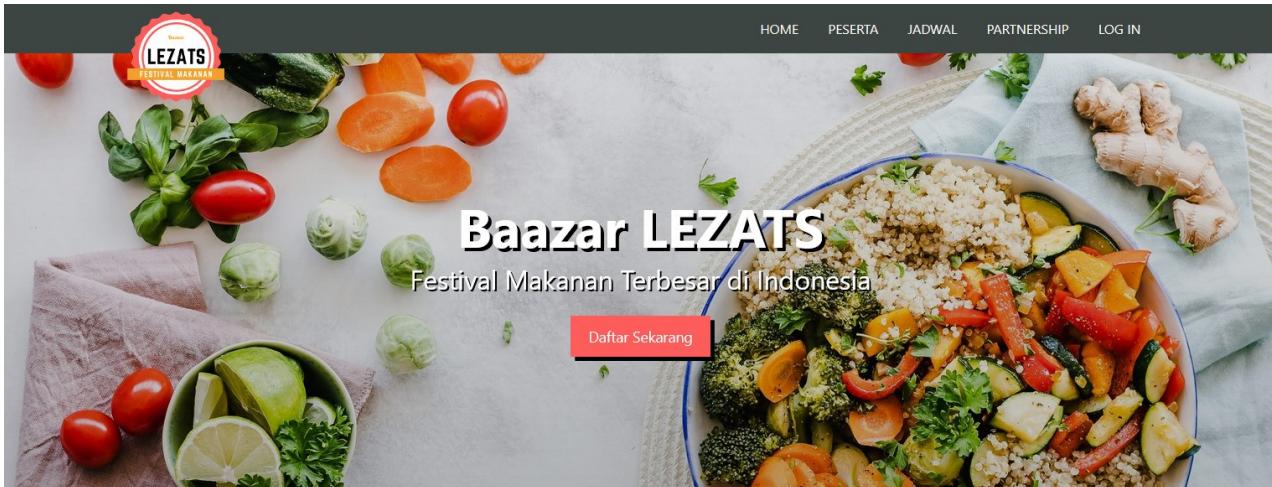
Perintah flexbox berikutnya ada di dalam selector `.footer-copyright`, yang dipakai untuk menampilkan teks copyright serta icon media social (baris 43–50). Dengan memakai `flex-flow: row wrap` dan `justify-content: space-between`, teks copyright akan mengambil tempat di sisi kiri, serta icon social media akan ter dorong ke sisi kanan.

Ketika masuk ke breakpoint smartphone, nilai property `justify-content` untuk selector `.footer-copyright` diubah menjadi `center` agar tampil di tengah-tengah halaman (baris 71).

22.11. Final Project

Sepanjang bab ini kita telah membahas cara pembuatan berbagai komponen web, mulai dari menu navigasi (hamburger menu), header, hero image, showcase, product and services, blog index hingga footer. Sekarang saatnya menggabungkan semua komponen ini menjadi sebuah tampilan web utuh.

Mayoritas kode HTML dan CSS yang digunakan langsung saya ambil dari komponen yang sudah kita buat. Namun tetap perlu beberapa penyesuaian agar kode CSS lebih terstruktur. Berikut tampilan akhir dari website "Lezats" hasil penggabungan semua komponen:



Lore ipsum dolor sit amet consectetur

Perferendis doloribus adipisci maxime rerum iure, ex ad natus
consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum
voluptates quis? Quam nihil sequi dicta neque ullam ab tempore
aspernatur, culpa maxime labore dolore!

Molestiae, sequi illo optio vitae error enim earum?

Lore ipsum dolor sit amet consectetur

Perferendis doloribus adipisci maxime rerum iure, ex ad natus
consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum
voluptates quis? Quam nihil sequi dicta neque ullam ab tempore
aspernatur, culpa maxime labore dolore!

Molestiae, sequi illo optio vitae error enim earum?



Terbesar di Indonesia

Lore ipsum dolor sit amet consectetur adipisicing elit. Dignissimos, vitae!



Budaya Lokal

Lore ipsum dolor sit, amet
consequetur adipisicing elit.
Vero error temporibus
tenetur cumque



UMKM Untuk Negeri

Lore ipsum dolor sit, amet
consequetur adipisicing elit.
Vero error temporibus
tenetur cumque



Go Internasional

Lore ipsum dolor sit, amet
consequetur adipisicing elit.
Vero error temporibus
tenetur cumque



1000 Peserta

Lore ipsum dolor sit, amet
consequetur adipisicing elit.
Vero error temporibus
tenetur cumque

Gambar: Bagian atas website "Lezat"



Update Kegiatan

Lore ipsum dolor sit amet consectetur adipisicing elit. Dignissimos, vitae!



Sehat dengan Buah

Doloremque minus repellat ipsam earum tenetur accusantium porro molestias rem, placeat, maxime cumque quidem neque totam deleniti consectetur ullam quisquam, illum in eos nesciunt asperiores.

Dolore provident expedita hic soluta maxime distinctio, error delectus.

Last updated 3 days ago



Perkebunan Organik

Deserunt officiis, doloribus necessitatibus, impedit repellendus minus commodi ab amet exercitationem corporis enim, dolorum excepturi numquam architecto accusamus cumque saepe possimus laudantium veritatis temporibus incident!

Nihil tempora porro neque animi debitis nemo accusamus placeat cupiditate dolor totam delectus quo eum minima?

Last updated 2 week ago

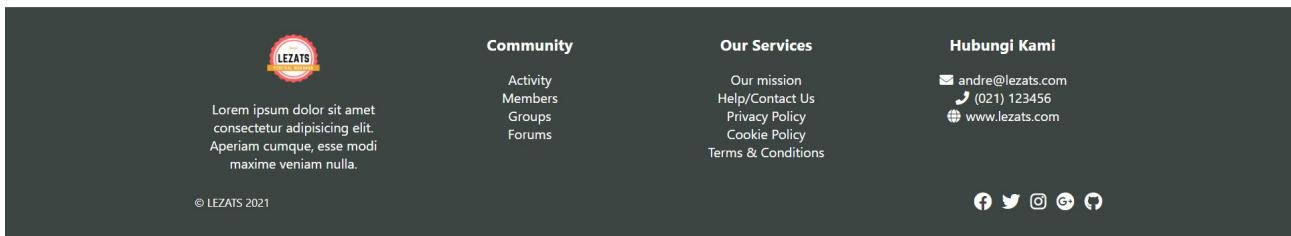


Jajanan Kekinian

Voluptates expedita non accusantium eaquel! Esse consectetur dolore quo querat tempora at eligendi, similiqe iste possimus! Quisquam obcaecati praesentium tempore aut nihil odio distinctio, alias itaque, nostrum eum et vitae asperiores iure vel nam.

Officiis esse explicabo iure pariatur illum obcaecati suscipit Facilis fuga quaerat, placeat vel quis cumque aut.

Last updated 1 month ago



Gambar: Bagian bawah website "Lezat"

Komponen baru yang belum kita bahas adalah Explore yang ada di antara product and services dan blog index. Ini sebenarnya modifikasi dari hero image dengan memperkecil tinggi element.

Berikut kode HTML untuk membuat tampilan di atas:

16.lezats-web-design.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <link href="fontawesome/css/all.min.css" rel="stylesheet">
7      <link href="style.css" rel="stylesheet">
8  <title>Lezats Food Festival</title>
```

```
9  </head>
10 <body>
11
12   <div class="nav-container">
13     <!-- HEADER DAN NAVIGASI -->
14     <header>
15       <label for="hamburger">#9776;</label>
16       <input type="checkbox" id="hamburger"/>
17       <div class="logo">
18         
19       </div>
20       <nav>
21         <ul>
22           <li><a href="#" class="active">Home</a></li>
23           <li><a href="#">Peserta</a></li>
24           <li><a href="#">Jadwal</a></li>
25           <li><a href="#">Partnership</a></li>
26           <li><a href="#">Log In</a></li>
27         </ul>
28       </nav>
29     </header>
30   </div>
31
32   <!-- HERO -->
33   <div class="hero-container">
34     <div class="hero-text">
35       <h1>Baazar LEZATS</h1>
36       <p>Festival Makanan Terbesar di Indonesia</p>
37       <a href="#" class="button">Daftar Sekarang</a>
38     </div>
39   </div>
40
41   <!-- SHOWCASE -->
42
43   <div class="showcase-container">
44     <section class="showcase">
45       <div class="showcase-image">
46         
47       </div>
48       <div class="showcase-text">
49         <h1>Lorem ipsum dolor sit amet consectetur</h1>
50         <p>Perferendis doloribus adipisci maxime rerum iure, ex ad natus
51             consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum
52             voluptates quis? Quam nihil sequi dicta neque ullam ab tempore
53             aspernatur, culpa maxime labore dolore!</p>
54         <p>Molestiae, sequi illo optio vitae error enim earum?</p>
55       </div>
56     </section>
57   </div>
58
59   <div class="showcase-container">
60     <section class="showcase">
61       <div class="showcase-text">
62         <h1>Lorem ipsum dolor sit amet consectetur</h1>
63         <p>Perferendis doloribus adipisci maxime rerum iure, ex ad natus
```

```
64      consequatur nesciunt dolorum quasi explicabo ipsa facere nostrum  
65      voluptates quis? Quam nihil sequi dicta neque ullam ab tempore  
66      aspernatur, culpa maxime labore dolore!</p>  
67      <p>Molestiae, sequi illo optio vitae error enim earum?</p>  
68  </div>  
69  <div class="showcase-image">  
70        
71  </div>  
72  </section>  
73</div>  
74  
75  <!-- PRODUCT AND SERVICE -->  
76  <div class="product-container">  
77      <header>  
78          <h1>Terbesar di Indonesia</h1>  
79          <hr>  
80          <p>Lorem ipsum dolor sit amet consectetur  
81              adipisicing elit. Dignissimos, vitae!</p>  
82      </header>  
83  
84      <section>  
85          <div>  
86              <i class="fas fa-monument fa-4x"></i>  
87              <h4>Budaya Lokal</h4>  
88              <p>Lorem ipsum dolor sit, amet consectetur  
89                  adipisicing elit. Vero error temporibus tenetur cumque</p>  
90          </div>  
91  
92          <div>  
93              <i class="fas fa-store-alt fa-4x"></i>  
94              <h4>UMKM Untuk Negeri</h4>  
95              <p>Lorem ipsum dolor sit, amet consectetur  
96                  adipisicing elit. Vero error temporibus tenetur cumque</p>  
97          </div>  
98  
99          <div>  
100             <i class="fas fa-globe fa-4x"></i>  
101             <h4>Go Internasional</h4>  
102             <p>Lorem ipsum dolor sit, amet consectetur  
103                 adipisicing elit. Vero error temporibus tenetur cumque</p>  
104         </div>  
105  
106         <div>  
107             <i class="fas fa-store fa-4x"></i>  
108             <h4>1000 Peserta</h4>  
109             <p>Lorem ipsum dolor sit, amet consectetur  
110                 adipisicing elit. Vero error temporibus tenetur cumque</p>  
111         </div>  
112     </section>  
113</div>  
114  
115  <!-- EXPLORE -->  
116  <section class="explore-container">  
117      <div class="explore-text">  
118          <h1>Explore</h1>
```

```

119   <p>
120     Lorem ipsum dolor sit, amet consectetur adipisicing elit.
121     Minus voluptate quam, rem possimus ut corrupti velit eaque
122     odit assumenda ipsam fugiat soluta, in vero tenetur omnis
123     at magni voluptatibus adipisci.
124   </p>
125   <a href="#" class="button">Find Out More</a>
126   </div>
127 </section>
128
129 <!-- BLOG UPDATE -->
130 <div class="blog-container">
131   <header>
132     <h1>Update Kegiatan</h1>
133     <hr>
134     <p>Lorem ipsum dolor sit amet consectetur
135       adipisicing elit. Dignissimos, vitae!</p>
136   </header>
137
138   <div class="blog-index">
139
140     <div class="snippet">
141       <figure>
142         
143       </figure>
144       <div class="snippet-txt">
145         <h2>Sehat dengan Buah</h5>
146         <p>Doloremque minus repellat ipsam earum tenetur accusantium porro
147           molestias rem, placeat, maxime cumque quidem neque totam deleniti
148           consectetur ullam quisquam, illum in eos nesciunt asperiores.</p>
149         <p>Dolore provident expedita hic soluta maxime distinctio,
150           error delectus.</p>
151         <p>
152           <small>Last updated 3 days ago</small>
153         </p>
154       </div>
155     </div>
156
157     <div class="snippet">
158       <figure>
159         
160       </figure>
161       <div class="snippet-txt">
162         <h2>Perkebunan Organik</h5>
163         <p>Deserunt officiis, doloribus necessitatibus, impedit repellendus
164           minus commodi ab amet exercitationem corporis enim, dolorum
165           excepturi numquam architecto accusamus cumque saepe possimus
166           laudantium veritatis temporibus incident! </p>
167         <p>Nihil tempora porro neque animi debitis nemo accusamus placeat
168           cupiditate dolor totam delectus quo eum minima?</p>
169         <p>
170           <small>Last updated 2 week ago</small>
171         </p>
172       </div>
173     </div>

```

```
174
175      <div class="snippet">
176          <figure>
177              
178          </figure>
179          <div class="snippet-txt">
180              <h2>Jajanan Kekinian</h2>
181              <p>Voluptates expedita non accusantium eaque! Esse consectetur
182                  dolore quo quaerat tempora at eligendi, similique iste possimus!
183                  Quisquam obcaecati praesentium tempore aut nihil odio distinctio,
184                  alias itaque, nostrum eum et vitae asperiores iure vel nam.</p>
185              <p>
186                  Officiis esse explicabo iure pariatur illum obcaecati suscipit
187                  Facilis fuga quaerat, placeat vel quis cumque aut.</p>
188              <p>
189                  <small>Last updated 1 month ago</small>
190              </p>
191          </div>
192      </div>
193
194  </div>
195
196 </div>
197
198 <!-- FOOTER -->
199 <footer class="footer-container">
200     <div class="footer-menu">
201         <section>
202             <div class="logo-container">
203                 <a href="#">
204                     
205                 </a>
206             </div>
207             <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
208                 Aperiam cumque, esse modi maxime veniam nulla.
209             </p>
210         </section>
211
212         <section>
213             <h3>Community</h3>
214             <ul>
215                 <li><a href="#">Activity</a></li>
216                 <li><a href="#">Members</a></li>
217                 <li><a href="#">Groups</a></li>
218                 <li><a href="#">Forums</a></li>
219             </ul>
220         </section>
221
222         <section>
223             <h3>Our Services</h3>
224             <ul>
225                 <li><a href="#">Our mission</a></li>
226                 <li><a href="#">Help/Contact Us</a></li>
227                 <li><a href="#">Privacy Policy</a></li>
228                 <li><a href="#">Cookie Policy</a></li>
```

```

229         <li><a href="#">Terms & Conditions</a></li>
230     </ul>
231 </section>
232
233 <section>
234     <h3>Hubungi Kami</h3>
235     <div><i class="fas fa-envelope fa-fw"></i> andre@lezats.com</div>
236     <div><i class="fas fa-phone fa-fw"></i> (021) 123456</div>
237     <div><i class="fas fa-globe fa-fw"></i> www.lezats.com</div>
238 </section>
239 </div>
240
241 <div class="footer-copyright">
242     <div>
243         <small>&copy; LEZATS 2021</small>
244     </div>
245     <div>
246         <a href="#"><i class="fab fa-facebook fa-lg"></i></a>
247         <a href="#"><i class="fab fa-twitter fa-lg"></i></a>
248         <a href="#"><i class="fab fa-instagram fa-lg"></i></a>
249         <a href="#"><i class="fab fa-google-plus fa-lg"></i></a>
250         <a href="#"><i class="fab fa-github fa-lg"></i></a>
251     </div>
252 </div>
253
254 </footer>
255
256 </body>
257 </html>

```

Kode HTML yang dibutuhkan memang sangat panjang, namun seperti inilah kode yang dibutuhkan untuk project "real world" sebenarnya. Untuk perintah CSS, saya pindahkan ke sebuah file external `style.css` dengan kode sebagai berikut:

```

style.css

1  /* Global Variabel */
2  :root{
3      --main-color : #FF5C5C;
4      --main-color-hover : #f83232;
5      --dark-color : #3c4542;
6      --container-width : 1100px;
7  }
8
9  /* Global Style */
10 * {
11     box-sizing: border-box;
12 }
13 html, body {
14     padding: 0;
15     margin: 0;
16     font-family:'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
17 }
18 h1 {

```

```
19  padding: 0;
20 margin: 0;
21 font-size: 2rem;
22 text-shadow: 2px 2px 3px silver;
23 }
24 .button {
25   text-decoration: none;
26   display: inline-block;
27   margin-top: 1.3em;
28   padding: 0.7em 1.2em;
29   background-color: var(--main-color);
30   color: white;
31   box-shadow: 5px 5px black;
32   font-size: 1.1em;
33 }
34 .button:hover {
35   background-color: var(--main-color-hover);
36 }
37
38 /* ===== */
39 /* HEADER DAN NAVIGASI */
40 /* ===== */
41
42 .nav-container {
43   background-color: var(--dark-color);
44   position: sticky;
45   top: 0px;
46   z-index: 100;
47 }
48 .nav-container header {
49   display: flex;
50   flex-direction: row;
51   justify-content: space-between;
52   width: 80vw;
53   height: 60px;
54   margin: 0 auto;
55 }
56 .nav-container .logo img {
57   width: 120px;
58   padding-top: 10px;
59 }
60 .nav-container ul {
61   list-style-type: none;
62   margin: 0;
63   padding: 0;
64   display: flex;
65 }
66 .nav-container li a {
67   display: block;
68   color: white;
69   padding: 0 1.1em;
70   text-decoration: none;
71   text-transform: uppercase;
72   height: 60px;
73   line-height: 60px;
```

```
74 }
75 .nav-container li a:hover {
76   color: var(--main-color);
77   transition: all 0.3s;
78   border-bottom: 5px solid var(--main-color);
79 }
80 /* Sembunyikan hamburger icon */
81 .nav-container header label, #hamburger {
82   display: none;
83 }
84
85 /* === */
86 /* HERO */
87 /* === */
88
89 .hero-container {
90   background-image: linear-gradient(rgba(0, 0, 0, 0.1),
91                                     rgba(0, 0, 0, 0.3)), url("img/hero.jpg");
92   height: 70vh;
93   background-position: center;
94   background-repeat: no-repeat;
95   background-attachment: fixed;
96   background-size: cover;
97   display: flex;
98   align-items: center;
99   justify-content: center;
100 }
101 .hero-text {
102   text-align: center;
103   color: white;
104 }
105 .hero-text h1{
106   font-size: 4em;
107   text-shadow: 4px 4px black;
108 }
109 .hero-text p{
110   font-size: 2em;
111   margin: 0;
112   text-shadow: 2px 2px black;
113 }
114
115 /* ===== */
116 /* SHOWCASE */
117 /* ===== */
118
119 .showcase-container:nth-child(even) {
120   background-color: #f0f0f0;
121 }
122 .showcase {
123   display: flex;
124   justify-content: space-between;
125   max-width: 1100px;
126   margin: 0 auto;
127   padding: 2em ;
128 }
```

CSS Case Study

```
129 .showcase-image, .showcase-text{  
130   width: 50%;  
131   margin: 1em;  
132 }  
133 .showcase-image img{  
134   width: 100%;  
135   border: 1px solid silver;  
136   padding: 0.3rem;  
137   box-shadow: 2px 2px 5px black;  
138 }  
139 /* ===== */  
140 /* PRODUCT AND SERVICE */  
141 /* ===== */  
143  
144 .product-container {  
145   max-width: var(--container-width);  
146   margin: 3rem auto;  
147   text-align: center;  
148 }  
149 .product-container header hr {  
150   width: 25%;  
151   border: 0;  
152   border-top: 1px solid #ddd;  
153 }  
154 .product-container section {  
155   margin-top: 2rem;  
156   display: flex;  
157   flex-flow: row wrap;  
158 }  
159 .product-container section div {  
160   width: 25%;  
161   padding: 1em 2em;  
162 }  
163 /* ===== */  
164 /* EXPLORE */  
165 /* ===== */  
167  
168 .explore-container {  
169   padding: 0 2em;  
170   background-repeat: no-repeat;  
171   background-size: cover;  
172   background-attachment: fixed;  
173   background-image: linear-gradient(rgba(0, 0, 0, 0.4),  
174                               rgba(0, 0, 0, 0.6)), url("img/hero.jpg");  
175   text-align: center;  
176   color: white;  
177 }  
178 .explore-text {  
179   max-width: var(--container-width);  
180   margin: 0 auto;  
181   padding: 3rem 0;  
182 }  
183 .explore-text p {
```

```
184   font-size: 1.3rem;
185   text-shadow: 1px 1px 2px black;
186 }
187 .explore-text h1 {
188   text-shadow: 1px 1px 2px black;
189 }
190
191 /* ===== */
192 /* BLOG INDEX */
193 /* ===== */
194
195 .blog-container {
196   max-width: var(--container-width);
197   margin: 3rem auto;
198 }
199 .blog-container header {
200   padding: 0 2em;
201   text-align: center;
202 }
203 .blog-container header hr {
204   width: 25%;
205   border: 0;
206   border-top: 1px solid #ddd;
207 }
208 .blog-index {
209   margin-top: 1rem;
210   display: flex;
211   flex-flow: row wrap;
212   justify-content:space-between;
213   padding: 0 2%;
214 }
215 .blog-index .snippet {
216   width: 32%;
217   margin-top: 1rem;
218   border: 1px solid #ddd;
219   border-radius: 0.25rem;
220 }
221 .blog-index .snippet-img {
222   width: 100%;
223 }
224 .blog-index .snippet-txt {
225   padding: 0 1em;
226 }
227 .blog-index .snippet-txt small {
228   color: #999;
229 }
230
231 /* Efek untuk hover gambar */
232 .snippet figure {
233   padding: 0;
234   margin: 0;
235   width: 100%;
236   overflow: hidden;
237 }
238 .snippet-img {
```

```
239   transform: rotate(10deg) scale(1.4);
240   transition: .3s ease-in-out;
241 }
242 .snippet figure:hover img {
243   transform: rotate(0) scale(1.1);
244 }
245 /* ===== */
246 /*     FOOTER      */
247 /* ===== */
248
249
250 .footer-container {
251   margin-top: 3rem;
252   color: white;
253   background-color: var(--dark-color);
254   padding: 2rem 1rem;
255 }
256 .footer-container a{
257   text-decoration: none;
258   color: white;
259 }
260 .footer-container a:hover{
261   text-decoration: underline;
262 }
263 .footer-menu{
264   max-width: var(--container-width);
265   margin: 0 auto;
266   display: flex;
267   flex-flow: row wrap;
268   justify-content: space-between;
269 }
270 .footer-menu section {
271   width: 23%;
272   text-align: center;
273 }
274 .footer-menu ul {
275   list-style-type: none;
276   margin: 0;
277   padding: 0;
278 }
279 .footer-menu h3{
280   margin-top: 0;
281 }
282 .logo-container img {
283   width: 65px;
284 }
285 .footer-copyright {
286   max-width: var(--container-width);
287   margin: 0 auto;
288   display: flex;
289   flex-flow: row wrap;
290   justify-content: space-between;
291   padding: 0.5rem;
292 }
293 .footer-copyright a {
```

```
294   margin-left: 0.5rem;
295 }
296 .footer-copyright a:hover {
297   text-decoration: none;
298 }
299
300 /* ===== */
301 /* BREAKPOINT TABLET */
302 /* ===== */
303
304 @media screen and (max-width: 768px){
305
306   /*===== HEADER DAN NAVIGASI =====*/
307
308   /* Tampilkan hamburger icon */
309   .nav-container header label {
310     display: inline-block;
311     color: white;
312     font-style: normal;
313     font-size: 2rem;
314     padding: 0.5rem;
315     padding-left: 1rem;
316   }
317
318   /* Susun ulang menu menjadi vertikal */
319   .nav-container header {
320     width: 100%;
321   }
322   .nav-container nav ul {
323     flex-direction: column;
324     width: 100%;
325     position: absolute;
326     top: 60px;
327     left: 0;
328   }
329   .nav-container li a {
330     display: block;
331     color: white;
332     background-color: var(--dark-color);
333     padding: 0 2em;
334     text-decoration: none;
335     text-transform: uppercase;
336     height: auto;
337   }
338   .nav-container li a:hover {
339     border-bottom: none;
340   }
341
342   /* Pindahkan posisi logo ke kanan, dan perkecil */
343   .nav-container .logo {
344     position: absolute;
345     right: 20px;
346     z-index: 1000;
347   }
348   .nav-container .logo img {
```

```
349     width: 80px;
350 }
351
352 /* Untuk menampilkan / menyembunyikan menu */
353 .nav-container nav { display: none; }
354 .nav-container header input:checked ~ nav {
355     display: flex;
356 }
357
358 /*===== SHOWCASE =====*/
359 .showcase {
360     flex-direction: column;
361     padding: 1em 2em ;
362 }
363 .showcase-image, .showcase-text{
364     width: auto;
365 }
366
367 /*===== PRODUCT =====*/
368 .product-container section div {
369     width: 50%;
370 }
371
372 /*===== BLOG UPDATE =====*/
373 .blog-container .snippet {
374     width: 48%;
375 }
376
377 /*===== FOOTER =====*/
378 .footer-menu section {
379     width: 48%;
380 }
381 }
382
383 /* ===== */
384 /* BREAKPOINT SMARTPHONE */
385 /* ===== */
386 @media screen and (max-width: 481px){
387
388     /*===== SERVICES =====*/
389     .product-container section div {
390         width: 100%;
391     }
392
393     /*===== BLOG UPDATE =====*/
394     .blog-container .snippet {
395         width: 98%;
396     }
397     .blog-index {
398         justify-content:center;
399     }
400
401     /*===== FOOTER =====*/
402     .footer-menu section {
403         width: 98%;
```

```

404     margin-top: 1rem;
405   }
406   .footer-copyright {
407     margin-top: 1rem;
408     justify-content:center;
409   }
410 }
```

Total halaman ini butuh lebih dari 400 baris kode CSS. Bagi pemula mungkin merasa terintimidasi dengan banyaknya perintah yang ada. Tapi sekali lagi, begitulah kode untuk web yang sebenarnya. Dan ini juga baru satu halaman, jika ada halaman kedua, ketiga, dst, tentu akan lebih panjang lagi.

Mayoritas kode ini sudah kita bahas. Salah satu perubahan yang saya tambah adalah mendefinisikan 4 variabel di awal kode program, yakni:

```

1  :root{
2    --main-color : #FF5C5C;
3    --main-color-hover : #f83232;
4    --dark-color : #3c4542;
5    --container-width : 1100px;
6 }
```

Dua variabel awal berfungsi sebagai warna utama berupa warna oranye / jingga. Warna ini dipakai pada bagian menu dan tombol. Selanjutnya variabel --dark-color dipakai untuk bagian yang butuh warna latar belakang gelap, yakni di bagian menu navigasi serta footer. Lalu variabel --container-width berfungsi untuk menentukan lebar maksimum container.

Perubahan lain adalah pembuatan "global style" di antara baris 10-36. Di sini saya mendefinisikan beberapa style yang akan dipakai berulang seperti jenis font, style untuk tag `<h1>`, dan style untuk tombol.

Sisa kode lain dipakai dikelompokkan sesuai komponen yang di-style. Terdapat baris komentar sebagai pembatas dari setiap komponen. Di akhir file terdapat perintah CSS media query agar tampilan web menjadi responsive.

Meskipun hanya terdiri dari satu halaman, semoga studi kasus ini bisa menjadi gambaran cara implementasi kode-kode CSS ke sebuah website yang sebenarnya. Saya juga sangat sarankan untuk coba rancang komponen web lain. Silahkan lihat inspirasi dari web yang sudah ada, lalu coba buat kode CSSnya sendiri.

Penutup: CSS Uncover

Tidak terasa hampir 700 halaman buku **CSS Uncover** menemani anda dalam mempelajari CSS. Buku ini memang ditujukan untuk level pemula, namun saya rasa sudah cukup memberikan fondasi dasar CSS yang kuat.

Dengan bekal pengetahuan ini, silahkan cari berbagai trik dan tips penggunaan CSS, termasuk studi kasus lain. Salah satu sumber yang saya rekomendasikan adalah css-tricks.com yang juga jadi referensi dalam riset materi buku ini.

Prinsip ATM (amati, tiru, modifikasi) menjadi teknik belajar paling cepat untuk menguasai sesuatu, tidak terkecuali web design. Silahkan cari website yang tampilannya dirasa menarik lalu coba tiru.

Salah satu keunggulan (dan juga bisa disebut sebagai kekurangan) dari cara kerja website adalah kita bisa melihat struktur HTML dan CSS hampir semua web. Situs marketplace web template seperti themeforest.net bisa jadi referensi untuk mencari inspirasi design.

Pelajari Seni Desain, UI dan UX

CSS adalah bahasa kode untuk mendesain tampilan web, akan tetapi tidak semua ahli CSS bisa membuat desain yang menarik.

Di perusahaan software besar, memang ada pemisahan antara tim yang merancang tampilan desain dengan tim yang menerjemahkan desain tersebut ke dalam kode HTML dan CSS. Akan tetapi untuk perusahaan kecil atau *freelance*, kadang kita dituntut untuk bisa membuat design dan coding sendiri.

Karena itu tidak ada salahnya untuk meningkatkan skill seni design seperti mengenal aturan golden ratio, psikologi warna, penggunaan ruang kosong (whitespace), serta teori pembuatan UI (User Interface) dan UX (User eXperience) yang baik.

Pelajari CSS Preprocessor

Jika memutuskan fokus di CSS, terdapat teknologi lain yang disebut CSS Preprocessor. **CSS Preprocessor** adalah semacam 'bahasa pemrograman CSS' yang memungkinkan kita menulis perulangan serta kondisi if di dalam CSS.

Saat ini CSS Preprocessor yang paling populer adalah **Sass** (sass-lang.com). Pilihan lain bisa ke **Less** atau **Stylus**, meskipun tidak sepopuler Sass.

Pelajari CSS Framework

Karena sudah memiliki bekal CSS, anda bisa "naik level" dengan mempelajari **CSS Framework**. CSS Framework berisi kumpulan kode CSS siap pakai yang memudahkan kita dalam membuat tampilan dan layout.

Merancang website dengan framework akan jauh lebih cepat. Misalnya jika menggunakan framework CSS Bootstrap, hamburger menu bisa dibuat tanpa menulis satu pun kode CSS! Yang perlu kita lakukan hanya menyusun struktur HTML sesuai aturan Bootstrap serta menambah nama class yang di syaratkan.

Selain Bootstrap, Framework CSS lain yang cukup populer saat ini adalah Tailwind CSS. Pilihan lain bisa ke Materialize, Semantic UI, Bulma, PureCSS, Kube CSS Framework, YALM 4, Gumby Framework, dan masih banyak lagi.

Jika tertarik, di DuniaIlkom juga sudah tersedia eBook / buku [Bootstrap Uncover](#).

Jadi, sebaiknya lanjut kemana?

Jika ingin fokus ke sisi web design, lanjut ke **Bootstrap** atau **Sass** bisa jadi pilihan menarik. Meskipun referensi Sass dalam bahasa Indonesia memang masih agak jarang.

Namun jika ingin belajar web programming secara lengkap (jalur *full-stack*), bisa lanjut ke teknologi dasar web berikutnya, yakni bisa pilih salah satu antara JavaScript atau PHP.

Alasan untuk ke JavaScript terlebih dahulu agar selesai semua materi front-end (HTML, CSS, JS), setelah itu baru ke back-end (PHP-MySQL).

Alasan untuk ke PHP terlebih dahulu karena programming di PHP sedikit lebih mudah dibandingkan JS. Selain itu untuk aplikasi sederhana seperti pembuatan CRUD, kadang tidak butuh JS (cukup HTML, CSS, PHP dan MySQL). Ini lebih saya sarankan terutama bagi yang belum pernah belajar bahasa pemrograman apapun / bukan dari jurusan IT.

Akhir kata, semoga materi yang ada di buku CSS Uncover ini bisa bermanfaat. Mohon maaf jika ada kata-kata atau penjelasan yang salah.

Terima kasih juga atas dukungannya dengan membeli versi asli buku CSS Uncover (bukan dari sumber bajakan / copy-an). Donasi pembelian buku ini menjadi penyemangat saya untuk bisa terus berkarya.

Sampai jumpa di buku DuniaIlkom selanjutnya, semoga ilmu yang di dapat bisa berkah dan bermanfaat :)

Daftar Pustaka

Sepanjang penulisan buku CSS Uncover, saya mengumpulkan bahan dari berbagai sumber. Anda bisa mengunjungi daftar pustaka ini untuk menambah pengetahuan seputar CSS:

- **MDN (Mozilla Developer Network):**
<https://developer.mozilla.org/en-US/docs/Web/CSS>
- **CSS-Tricks:** <https://css-tricks.com>
- **W3C:** <https://www.w3.org>
- **W3schools CSS Tutorial:** <https://www.w3schools.com/css/default.asp>
- **Kevin Powell YouTube Channel:** <https://www.youtube.com/kepowob>