

BAB II CSS

Pengenalan CSS

Tentang CSS (Cascading Style Sheet)

CSS (Cascading Style Sheet) secara sederhana adalah sebuah metode yang digunakan untuk mempersingkat penulisan tag HTML, seperti font, color, text dan tabel menjadi lebih ringkas sehingga tidak terjadi pengulangan penulisan.

CSS digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS, memungkinkan Kita untuk menampilkan halaman yang sama dengan format berbeda.

CSS merupakan teknologi internet yang direkomendasikan oleh World Wide Web Consortium atau W3C pada tahun 1996.

Saat ini terdapat 3 versi CSS (CSS1, CSS2, CSS3) mayoritas property - property yang digunakan adalah CSS2 karena property CSS3 masih dapat berubah - ubah selama pengembangan.

Kegunaan CSS

Beberapa kegunaan CSS diantaranya :

- Mempersingkat penulisan tag HTML
- Penulisan tag dengan property dan nilai yang sama tidak perlu dituliskan pada setiap tag HTML.
- Mempercepat proses rendering atau pembacaan HTML karena tidak terdapat pengulangan penulisan.
- Mudah untuk memelihara (maintain) skrip : karena CSS dapat dibuat terpisah maka tidak perlu merombak semua elemen atau property dalam HTML.
- CSS dapat melakukan apa yang tidak bisa dilakukan oleh HTML seperti memberikan warna pada input box atau scrollbar.

Aturan Penulisan CSS

CSS memiliki aturan dalam penulisannya, yang terdiri atas ***Selector*** dan ***Declaration***.

- **Selector** : mengidentifikasi elemen atau elemen - elemen yang akan di deklarasikan.
- **Declaration** : untuk mendeskripsikan property dan nilai yang akan digunakan pada *selector*.



CSS *Declaration* yang berada didalam kurung kurawal terdiri dari 2 (dua) bagian **property** dan **Value** yang dipisahkan oleh tanda titik dua “:”. Kita dapat menentukan beberapa property dalam satu deklarasi, masing - masing dipisahkan dengan tanda titik koma “;”.



Cara Penulisan CSS

Cara penulisan skrip CSS dibagi menjadi 3 (tiga) bagian, yaitu :

1. *Inline Style Sheet*

Penulisan skrip CSS didalam elemen HTML. Untuk metode ini fungsinya hamper sama dengan menuliskan property pada tag HTML, penggunaannya untuk hal - hal tertentu saja. Misalkan memberi warna hijau pada tag “<p>” maka cara penulisannya sebagai berikut :

```
<p style="color:green">
```

Penulisan skrip CSS dilakukan pada tag pembuka suatu element html. Property **style** akan men-support semua property yang di-support oleh CSS.

2. *Embedded Style Sheet*

Penulisan skrip CSS didalam dokumen HTML. Metode ini menggunakan tag **<style>** dan **</style>**

Contoh :

```
<style type="text/css">
```

```
p {color : red}
```

```
</style>
```

Biasanya penulisan skrip CSS ini dilakukan pada bagian elemen “head”.

3. *Linked Style Sheet / External Style Sheet*

Penulisan skrip CSS di halaman yang berbeda atau terpisah dari HTML. Jadi, Kita tinggal melakukan link ke *file* CSS yang telah dibuat. Untuk metode ini, menggunakan tag `<link rel>` yang ditempatkan pada bagian tag `<head>`. Metode ini paling banyak digunakan dalam penulisan CSS. *Linked style sheet* merupakan metode penulisan yang paling banyak dipakai pada pengembangan website.

Beberapa keuntungan menggunakan aturan *style* ini diantaranya:

- Menghemat Kita mengulangi aturan *style* yang sama di setiap halaman.
- Dapat mengubah tampilan beberapa halaman dengan mengubah hanya pada *style sheet* - nya saja daripada masing-masing halaman. Ini berarti lebih mudah untuk memperbarui website Anda jika Anda ingin, misalnya, mengubah gaya font yang digunakan dalam semua judul atau mengubah warna semua tautan.
- Setelah pengunjung ke situs Anda mengunduh CSS dengan halaman pertama situs Anda yang menggunakannya, halaman berikutnya akan lebih cepat untuk memuat (karena *browser* menyimpan salinan *style sheet* dan aturan tidak perlu diunduh untuk setiap halaman). Ini juga mengurangi ketegangan pada server (komputer yang mengirim halaman web ke orang yang melihat situs) karena halaman yang dikirim lebih kecil.
- *Style sheet* dapat bertindak sebagai *template style* untuk membantu penulis yang berbeda mencapai *style sheet* dokumen yang sama tanpa mempelajari semua pengaturan *style* individu.
- Karena halaman web tidak mengandung aturan gaya (*style rules*), berbagai *style sheet* dapat dilampirkan ke dokumen yang sama. Jadi, Anda dapat menggunakan dokumen HTML yang sama dengan satu *style sheet* saat *viewer* berada di komputer desktop, *style sheet* lain saat pengguna memiliki perangkat genggam (*mobile*), *style sheet* lain saat halaman sedang dicetak (*print*), *style sheet* lain saat halaman sedang dilihat di TV, dan sebagainya. Anda dapat menggunakan kembali dokumen yang sama dengan *style sheet* berbeda untuk kebutuhan pengunjung yang berbeda.
- *Style sheet* dapat mengimpor dan menggunakan *style* dari *style sheet* lain, memungkinkan untuk pengembangan modular dan penggunaan kembali yang baik.
- Jika *style sheet* dihapus, Anda dapat membuat situs lebih mudah diakses bagi mereka yang memiliki keterbatasan penglihatan karena Anda tidak lagi mengendalikan font dan skema warna.

Oleh karena itu, cukup adil untuk mengatakan bahwa setiap kali Kita menulis seluruh situs, Anda harus menggunakan style sheet eksternal untuk mengontrol penyajiannya (daripada meletakkan aturan CSS di halaman web individual).

Atribut Rel

Atribut rel diperlukan dan menentukan hubungan antara dokumen (relationship) yang berisi tautan dan dokumen yang ditautkan. Nilai yang digunakan untuk bekerja dengan style sheet adalah stylesheet:

rel="stylesheet"

Atribut Href

Atribut href menentukan URL untuk dokumen yang ditautkan :

href = "../ stylesheets / interface.css"

Nilai dari atribut ini dapat berupa URL absolut atau relatif (dibahas pada buku panduan definitif html5), tetapi biasanya merupakan URL relatif karena style sheet adalah bagian dari situs.

Atribut Media

Atribut media menentukan perangkat output yang dimaksudkan untuk digunakan dengan dokumen:

media = "screen"

Meskipun atribut ini tidak selalu digunakan, ini penting karena orang mengakses Internet dengan cara yang berbeda menggunakan perangkat yang berbeda. Tabel 7-1 menunjukkan nilai yang mungkin.

2.1.4.1	Nilai (value)	2.1.4.2	Penggunaan
2.1.4.3	screen	2.1.4.4	Layar komputer (seperti komputer desktop dan laptop)
2.1.4.5	tty	2.1.4.6	Media dengan kisi karakter pitch tetap, seperti teletype, terminal, atau perangkat portabel dengan kemampuan tampilan terbatas
2.1.4.7	Tv	2.1.4.8	Perangkat TV dengan resolusi rendah, layar warna, dan kemampuan terbatas untuk menggulir ke bawah halaman
2.1.4.9	print	2.1.4.10	Dokumen tercetak, yang kadang-kadang disebut media paged (dan dokumen ditampilkan di layar dalam mode pratinjau cetak (<i>preview mode</i>))
2.1.4.11	projection	2.1.4.12	Projector
2.1.4.13	handheld	2.1.4.14	Perangkat genggam, yang sering memiliki layar kecil, bergantung pada grafik yang dipetakan, dan memiliki bandwidth terbatas

2.1.4.15	braille	2.1.4.16	Perangkat umpan balik sentuhan Braille
2.1.4.17	embossed	2.1.4.18	Printer beranda braille
2.1.4.19	speech	2.1.4.20	Penyintesis ucapan
2.1.4.21	all	2.1.4.22	Cocok untuk semua perangkat

Komentar pada CSS

Sama seperti Bahasa pemrograman lainnya komentar akan diberikan paa skrip untuk menjelaskan sesuatu. Komentar pada CSS hampir sama dengan komentar pada Bahasa C atau C++, yaitu menggunakan :

```
/* isi komentar */
```

Contoh :

```
p {
    color : red; /* memberikan warna merah pada paragraf*/
}
```

Macam-Macam Selector

Sebelum melanjutkan untuk melihat lebih banyak properti, Anda perlu melihat beberapa masalah yang lebih mendasar. Mulailah dengan melihat bagaimana Anda dapat menggunakan jenis *selector* yang berbeda untuk menentukan elemen mana yang menjadi aturan aturan *style sheet*. Sebenarnya ada beberapa cara untuk melakukan ini, tidak hanya dengan menggunakan nama elemen seperti yang telah Kita lihat sejauh ini dalam bab ini (yang, kebetulan, dikenal sebagai *selector* sederhana) atau menggunakan nilai atribut kelas atau atribut id. Pelajari caranya di beberapa bagian berikutnya.

Universal Selector

Selector universal adalah tanda bintang; itu seperti wildcard dan cocok dengan semua jenis elemen dalam dokumen.

```
* { }
```

Jika Anda ingin aturan diterapkan ke semua elemen, Anda dapat menggunakan selector ini. Kadang-kadang digunakan untuk nilai-nilai default yang berlaku untuk seluruh dokumen (seperti **font-family** dan ukuran **font**) kecuali selector lain yang lebih spesifik menunjukkan bahwa elemen harus menggunakan nilai yang berbeda untuk properti yang sama.

Ini sedikit berbeda dari menerapkan gaya default ke elemen **<body>** karena pemilih universal berlaku untuk setiap elemen dan tidak bergantung pada properti yang diwarisi dari aturan yang berlaku untuk elemen **<body>**.

Tag / Element HTML

Penggunaan tag yang terdapat pada HTML. Setiap tag yang ada dalam HTML dapat dijadikan sebagai selector.

Contoh :

h1 {color : red; }

Class Selector

Class selector memungkinkan Anda untuk mencocokkan aturan dengan yang membawa attribute **class** yang nilainya cocok dengan yang Anda tentukan di *class selector*.

Sebagai contoh, pada tag “<p>” akan Kita tambahkan atribut class dengan nama *isiteks* seperti berikut :

```
<p class="isiteks">ini adalah isiteks class paragraph</p>
```

Penggunaan *class selector* akan diawali dengan tanda titik “.” Diawal penulisannya. Kita dapat menggunakan *class selector* dengan 2 (dua) cara.

1. Kita dapat memberikan aturan penulisan CSS pada semua element yang memiliki atribut class “isiteks”.

```
.isiteks{  
    color: #99ccff;  
    font-size: 36px;  
}
```

2. Kita dapat memberikan aturan penulisan CSS hanya pada spesifik elemen yang sudah diberikan class tertentu saja.

```
p.isiteks{  
    color: #FF0000;  
    font-size: 36px;  
}
```

Kemudian untuk mengimplementasikan style pada class yang sudah dibuat, pada dokumen html diantara tag **body** dapat dibuat sebagai berikut :

```
<h1 class="isiteks" >  
ini adalah style pada class  
</h1>
```

```
<p class="isiteks" >  
ini adalah style pada class spesifik  
</p>
```

Maka pada hasil di *browser* dapat terlihat sebagai berikut :



ini adalah style pada class

ini adalah style pada class spesifik

ID Selector

Hampir sama dengan class selector, *ID selector* bersifat unik dan dapat diterapkan untuk hampir semua tag HTML, tetapi penggunaanya hanya sekali dalam satu halaman untuk satu tag HTML tertentu. Penggunaan ID selector menggunakan awalan tanda pagar “#” untuk inisialisasi sebuah nama id yang sudah dibuat pada dokumen HTML.

```
#footer {  
    color: blue;  
    border: 1px solid black;  
}
```

pada CSS ditulis :

```
<div id="footer">
    copy-Right Cyber Bussiner School - 2011
</div>
```

Sekali lagi di tekankan selector ID digunakan hanya untuk 1 elemen pada satu halaman web. Misalnya saja ID **#footer** diatas hanya digunakan sekali karena dalam satu halaman web hanya ada 1 id **footer**.

Hasil pada *browser* :



Child Selector

Child selector digunakan untuk mencocokkan semua elemen yang merupakan anak dari elemen yang ditentukan. Ini memberi hubungan antara dua elemen. Selector **elemen > elemen** memilih elemen yang merupakan anak-anak dari orang tua tertentu. Operan di sisi kiri ">" adalah induk dan operan di sebelah kanan adalah elemen anak-anak.

Style CSS :

```
div > p {
    color:white;
    background: green;
    padding:2px;
}
```

Dokumen Html:


```

<div>
    <h2 style = "color:green;">
        CSS Child Selector
    </h2>
    <p>
        A computer science portal for geeks.
    </p>
</div>

```



Descendant Selector

Selector Descendant digunakan untuk memilih semua elemen yang merupakan anak dari elemen (bukan elemen tertentu). Ini memilih elemen di dalam elemen yaitu menggabungkan dua penyeleksi sehingga elemen yang cocok dengan *selector* kedua dipilih jika mereka memiliki elemen leluhur yang cocok dengan *selector* pertama.

Style CSS :

```

div h2 {
    font-size:26px;
    text-align: center;
}

```

Dokumen HTML:

```

<div>
  <h2 style = "color:green;" >
    GeeksForGeeks
  </h2>
  <div>
    <h2>GeeksForGeeks</h2>
  </div>
</div>
<p>
  GeeksforGeeks in green color is example of child
  Selector
  <br>other GeekforGeeks is example of descendant Selector
</p>

```



Adjacent Sibling Selector

Sebuah *adjacent sibling selector* digunakan untuk memilih diantara 2 pasang selector dan dipilih yang kedua atau selanjutnya.

Jika, x, y dan z adalah tiga elemen HTML dan y dan z berada bersebelahan di dalam x, maka y dan z disebut sebagai *adjacent sibling selector*.

Saat menulis *adjacent sibling selector*, *selector* harus dipisahkan dengan kombinator "+".

Misalnya jika Anda ingin membuat paragraf pertama setelah setiap level 1 dengan gaya yang berbeda dari elemen <p> lainnya, Anda dapat menggunakan *adjacent sibling selector* seperti itu untuk menentukan aturan untuk hanya elemen <p> pertama yang muncul setelah setiap elemen <h1>.

Contoh penggunaan *adjacent sibling selector* :

CSS Style :

```
h1 + h2 {  
    color: red;  
}
```

Dokumen HTML:

```
<h1>w3resource CSS examples</h1>  
<h2>w3resource CSS CSS adjacent selectors examples</h2>
```

w3resource CSS examples

w3resource CSS CSS adjacent selectors examples

Jika Kita menemukan h2 kembali tanpa h1 pada paragraf lainnya maka mereka bukan adjacent sibling selector karena selector sebelumnya bukan h1.

General Sibling Selector

General sibling selector memilih semua elemen yang merupakan saudara kandung dari elemen yang ditentukan.

Contoh berikut memilih semua elemen <p> yang merupakan saudara dari elemen <div>:

Saat menulis general sibling selector selector harus dipisahkan dengan kombinator “~”

Style CSS :

```
div ~ p {  
    background-color: yellow;  
}
```

Dokument HTML :

```

<p>Paragraph 1.</p>
<div>
  <p>Paragraph 2.</p>
</div>
<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

```

Paragraph 1.

Paragraph 2.

Paragraph 3.

Some code.

Paragraph 4.

Font

Selanjutnya Kita akan mempelajari mengenai kontrol teks saya akan membagi menjadi 2 (dua) bab yang membahas mengenai kontrol teks ini. bab ini adalah bagian yang pertama yaitu kontrol teks yang langsung mempengaruhi font dan penampilannya. Misalnya huruf tebal, cetak miring dan ukuran teks.

Berikut adalah property font yang dapat mempengaruhi secara langsung (*directly*) pada sebuah font teks.

2.1.4.23	Properti	2.1.4.24	Kegunaan
2.1.4.25	font	2.1.4.26	Memungkinkan Anda untuk menggabungkan beberapa properti berikut ini menjadi satu.
2.1.4.27	font-family	2.1.4.28	Menentukan jenis huruf atau keluarga font yang harus digunakan.
2.1.4.29	font-size	2.1.4.30	Menentukan ukuran font.
2.1.4.31	font-weight	2.1.4.32	Menentukan apakah font harus normal atau tebal.
2.1.4.33	font-style	2.1.4.34	Menentukan apakah font harus normal, miring, atau miring.
2.1.4.35	font-variant	2.1.4.36	Menentukan apakah font harus berukuran normal atau kecil.

Sebelum melihat properti ini secara detail, ada baiknya Anda memahami beberapa istilah kunci yang digunakan dalam tipografi.

Mungkin yang paling penting, **font** tidak sama dengan jenis huruf (*typeface*):

- jenis huruf (*typeface*) adalah keluarga font, seperti keluarga Arial.
- Font adalah anggota spesifik keluarga itu, seperti huruf tebal 12 poin Arial.

Anda sering melihat istilah yang digunakan secara bergantian, tetapi perlu diperhatikan perbedaannya.

Typeface cenderung menjadi salah satu dari dua grup: font **serif** dan **sans-serif**. Font serif memiliki ikal tambahan pada huruf. Sebagai contoh, pada gambar dibawah ini, gambar pertama berisi serif di bagian atas huruf dan di bagian bawah huruf, sedangkan font sans-serif memiliki ujung lurus ke huruf, seperti pada contoh kedua.

Gaya umum ketiga dari jenis huruf adalah font serif **monospace**. Setiap huruf dalam font monospasi memiliki lebar yang sama, sedangkan font nonmonospasi memiliki lebar berbeda untuk huruf yang berbeda. (Misalnya, dalam font serif dan sans-serif, huruf l cenderung lebih sempit daripada huruf m.)



Font serif umumnya dianggap lebih mudah dibaca untuk sejumlah besar teks yang dicetak. Tetapi di Internet banyak orang menemukan font serif lebih sulit dibaca untuk waktu yang lama, terutama karena resolusi layar komputer tidak sebagus dokumen cetak, yang membuat font sans-serif yang kurang rinci lebih mudah dibaca.

Agar Anda dapat mempelajari properti yang memengaruhi font, sebagian besar contoh di bagian berikut menggunakan struktur yang serupa. Paragraf teks akan diulangi, dan setiap elemen `<p>` membawa atribut kelas dengan nilai yang berbeda, misalnya:

```

<html>
<head>
<style>
  p.sans-serif {
    font-family : arial, verdana, sans-serif;
  }
  p.serif {
    font-family : times, "times new roman", serif;
  }
  p.monospace {
    font-family : courier, "courier new", monospace;
  }
</style>
</head>
<body>
  <p class="sans-serif">Here is some text in a sans-serif font.</p>
  <p class="serif">Here is some text in a serif font.</p>
  <p class="monospace">Here is some text in a monospaced font.</p>
</body>
</html>

```

Anda kemudian dapat melihat bagaimana properti berbeda memengaruhi setiap elemen <p> dengan menulis aturan terpisah untuk setiap paragraf. Anda dapat menggunakan nilai atribut kelas di pemilih CSS untuk membuat aturan yang berlaku hanya untuk satu elemen <p> pada suatu waktu.

Here is some text in a sans-serif font.

Here is some text in a serif font.

Here is some text in a monospaced font.

Properti Font-Family

Properti font-family memungkinkan Anda untuk menentukan jenis huruf yang harus digunakan untuk teks apa pun di dalam elemen yang diterapkan aturan CSS.

Saat memilih tipografi, Anda harus tahu bahwa tanpa menggunakan arahan @ font-face, *browser* hanya dapat menampilkan teks HTML dalam font yang Anda tentukan jika jenis huruf itu diinstal pada komputer itu. Jadi, jika Anda menentukan font seperti ***Futura*** atau ***Garamond***, dan saya tidak memilikinya di komputer saya, saya akan melihat teks dalam font yang berbeda bukan yang Anda tentukan.

Inilah sebabnya, jika Anda melihat pilihan situs web, sebagian besar sangat bergantung pada pilihan kecil tipografi yang dipasang pada sebagian besar komputer yang mengakses web, khususnya Arial, Kurir / Kurir Baru, Georgia, Times / Times New Roman, dan Verdana .

(Dari daftar ini, Arial dan Verdana sangat populer karena dianggap mudah dibaca secara online.)

Untuk membantu masalah, Anda dapat menentukan daftar tipografi sehingga jika pengguna tidak memiliki pilihan jenis huruf yang diinstal pada komputer mereka, *browser* dapat mencoba menampilkan teks dalam pilihan kedua atau ketiga. Setiap jenis huruf dalam daftar dipisahkan oleh koma, dan jika nama berisi **spasi** (seperti *times new roman* atau *courier new*), Anda harus menempatkan nama jenis huruf dalam tanda kutip ganda (gambar 23)

Anda mungkin memperhatikan bahwa setiap daftar tipografi pada contoh sebelumnya diakhiri dengan apa yang disebut nama **font generik** (sans-serif, serif, dan monospace).

Gagasan di balik ini adalah bahwa setiap komputer akan memiliki font yang sesuai dengan salah satu dari lima grup font generik (**sans-serif, serif, monospace, cursive, dan fantasy**), dan jika tidak dapat menemukan tipografi yang telah Anda tentukan, dapat menggunakan pilihan font yang sesuai dengan grup font umum.

Berikut adalah beberapa nama font yang umum.

2.1.4.37	Nama Font Umum	2.1.4.38	Tipe font	2.1.4.39	Contoh
2.1.4.40	Serif	2.1.4.41	Font dengan serif	2.1.4.42	Times
2.1.4.43	Sans-serif	2.1.4.44	Font tanpa serif	2.1.4.45	Arial
2.1.4.46	Monospace	2.1.4.47	Font lebar fix	2.1.4.48	Courier
2.1.4.49	Cursive	2.1.4.50	Font yang meniru tulisan tangan	2.1.4.51	Comis sans
2.1.4.52	fantasy	2.1.4.53	Font dekoratif untuk judul dan sebagainya	2.1.4.54	Impact

Satu hal yang perlu dipertimbangkan ketika memilih daftar font adalah bahwa setiap font dapat memiliki ketinggian atau lebar yang berbeda, jadi Anda mungkin ingin memilih daftar font yang memiliki ukuran yang sama. (Kalau tidak, tata letak bisa terlihat berbeda dengan yang Anda harapkan.) Misalnya, Courier New cukup pendek dan lebar, jadi jika ini adalah pilihan pertama Anda, itu tidak baik untuk memiliki Dampak sebagai pilihan kedua karena Dampak cukup tinggi dan sempit.

Jika Anda ingin menggunakan jenis huruf tertentu, Anda harus banyak mencoba - coba dan mempelajari mengenai web font lebih mendalam lagi.

Properti Font-Size

Properti font-size digunakan untuk menentukan ukuran font. Biasanya nilai untuk property ini ditentukan dalam ukuran pixel (px).

Namun Kita dapat memberikan nilai dengan banyak cara :

- **Length** : selain pixel ada beberapa satuan panjang lain yang dapat digunakan pada ukuran font.

px, em, ex, pt, in, cm, pc, mm, rem, vw, vh

- **Ukuran absolut** : masing-masing nilai ini sesuai dengan ukuran tetap :

xx-small, x-small, small, medium, large, x-large, xx-large

- **Ukuran relative** : nilai ini relatif terhadap teks disekitarnya :

smaller, larger

- **Persentase** : dihitung sebagai proporsi dari elemen induk :

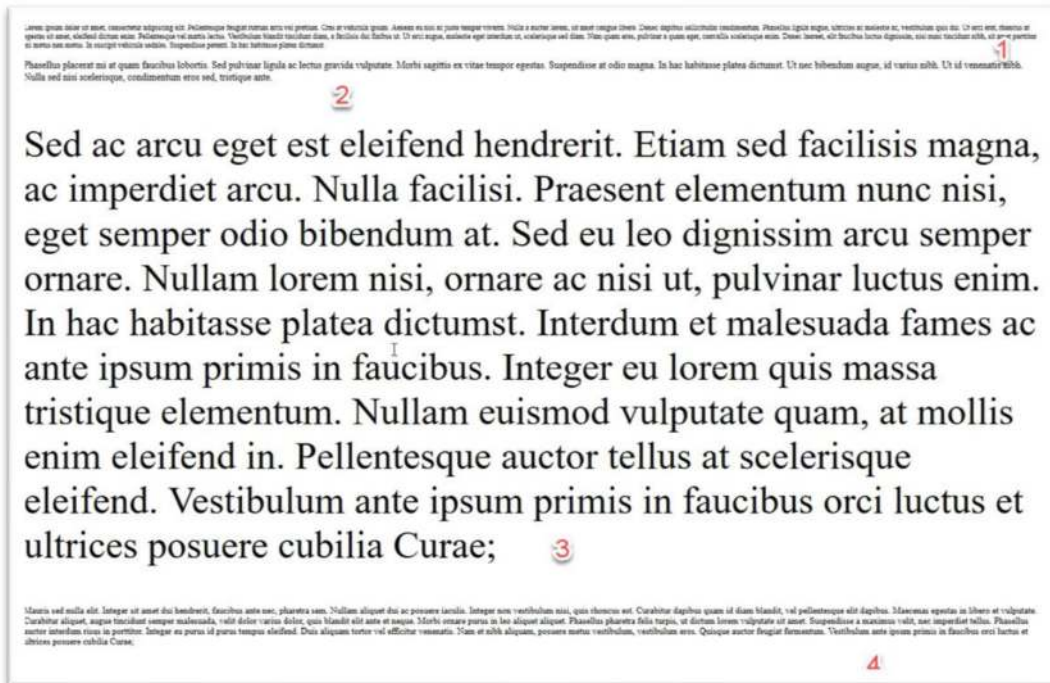
2%, 10%, 25% 100%

Berikut adalah contoh beberapa cara untuk menentukan ukuran font pada css :

Pada contoh ini saya membuat sebuah document html dengan 4 paragraf yang memiliki class yang berbeda. Pada class tersebut ditentukan style css sebagai berikut :

```
6
7 ▼ <style>
8 ▼ p.satu {
9   font-size : xx-small;
10  }
11 ▼ p.dua {
12   font-size : 12px;
13  }
14 ▼ p.tiga {
15   font-size : 3pc;
16  }
17 ▼ p.empat {
18   font-size : 70%;
19  }
20 </style>
```

Yang menghasilkan format dokumen html sebagai berikut :



Properti Font-Weight

Property ini digunakan untuk membuat variasi font yang berbeda seperti cetak tebal dan cetak miring. Pada sebuah *browser* cenderung menggunakan sebuah algoritma untuk membuat versi font terlihat lebih tebal atau lebih tipis. Inilah tujuan dari property font-weight.

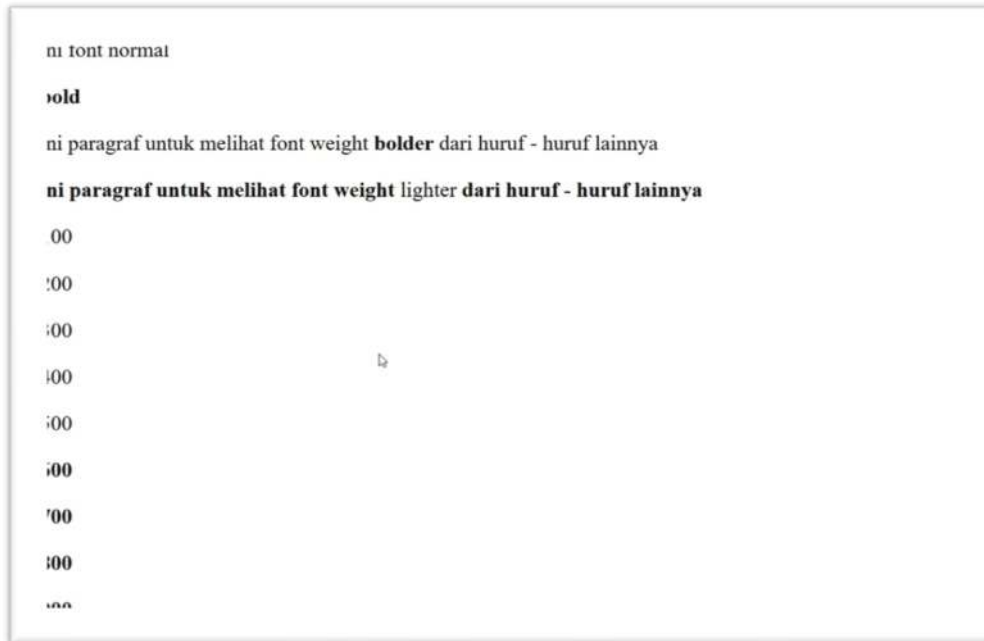
Nilai yang mungkin untuk font-weight adalah :

normal bold bolder lighter 100 200 300 400 500 600 700 800 900

berikut adalah contoh penggunaan property font-weight pada css :

```
p.normal{ font-weight: normal; }
p.bold{ font-weight: bold; }
p.bold span{font-weight: lighter;}
p.normal span{font-weight: bolder;}
p.satu{font-weight: 100;}
p.dua{font-weight: 200;}
p.tiga{font-weight: 300;}
p.empat{font-weight: 400;}
p.lima{font-weight: 500;}
p.enam{font-weight: 600;}
p.tujuh{font-weight: 700;}
p.delapan{font-weight: 800;}
p.sembilan{font-weight: 900;}
```

Hasil yang dapat dilihat pada *browser* sesuai dengan dokumen html yang sudah dibuat sebelumnya:



Properti Font-Style

Property ini digunakan untuk mengatur bentuk style pada font. Apakah secara normal, italic, atau oblique. Namun style bold tidak dikategorikan ke property font-style karena style bold dikategorikan ke property weight, hal ini berbeda dengan HTML.

Dalam tipografi, versi italic dari sebuah font biasanya akan menjadi versi font khusus berdasarkan kaligrafi, sedangkan versi oblique dari font akan mengambil versi normal font yang sudutnya dimiringkan.

Berikut adalah contoh penggunaan property font-style:

```
<style>
  h1 {
    font-style: normal;
  }
  h2 {
    font-style: italic;
  }
  h3 {
    font-style: oblique;
  }
</style>
```

Hasil yang terlihat pada *browser*:

ini font style normal

ini font style italic

ini font style oblique

Properti Font-Variant

Property ini untuk mengubah bentuk huruf yang tadinya berukuran kecil menjadi semuanya berbentuk huruf capital atau normal. Terdapat dua nilai pada property ini : **normal** dan **small-caps**.

Berikut contoh penggunaan font-variant pada dokumen html.

```

5 </style><font var font= document.</style>
6 <style>
7   p{
8     font-variant: normal;
9   }
10  span.smallcaps{
11    font-variant: small-caps;
12  }
13 </style>
14 </head>
15 <body>
16   <p>ini adalah font normal, tetapi terdapat <span
17     class="smallcaps">variant smallcaps</span> dibagian
18     tengah</p>
19 </body>
20 </html>

```

ini adalah font normal, tetapi terdapat VARIANT SMALLCAPS dibagian tengah

Teks

Bab ini melanjutkan kontrol teks pada css di bab sebelumnya. Kontrol teks yang kedua adalah kontrol yang memiliki efek pada teks terlepas dari font yang digunakan. Misalnya, warna teks, jarak antara kata atau huruf dll.

Berikut adalah tabel yang menunjukkan property format teks :

2.1.4.55	Properti	2.1.4.56	Tujuan
2.1.4.57	color	2.1.4.58	Menentukan warna teks.
2.1.4.59	text-align	2.1.4.60	Menentukan perataan horizontal teks dalam elemen yang mengandungnya.
2.1.4.61	vertical-align	2.1.4.62	Menentukan perataan vertical teks dalam elemen yang mengandungnya.
2.1.4.63	text-decoration	2.1.4.65	Menentukan apakah teks harus diberi underline, overline atau dicoret.
2.1.4.64			
2.1.4.66	text-indent	2.1.4.67	Menentukan indentasi dari batas kiri untuk teks.
2.1.4.68	text-transform	2.1.4.70	Menentukan bahwa konten elemen semuanya harus huruf besar, huruf kecil atau kapitalisasi.
2.1.4.69			
2.1.4.71	text-shadow	2.1.4.72	Menentukan bahwa teks harus memiliki bayangan.
2.1.4.73	letter-spacing	2.1.4.74	Mengontrol lebar antara huruf.
2.1.4.75	word-spacing	2.1.4.76	Mengontrol jumlah ruang antara setiap kata.

2.1.4.77 white-space 2.1.4.78	2.1.4.79 Menentukan apakah whitespace harus dikecilkan, dipertahankan atau dicegah dari area yang ada.
2.1.4.80 direction	2.1.4.81 Menentukan arah teks.

Properti Color

Property ini digunakan untuk mengatur warna pada teks yang dibuat. Nilai warna bisa menggunakan nama secara langsung (*red, blue, yellow*), nilai RGB (*Red, Green, Blue*), nilai hexadecimal, atau HSL (*Hue Saturation Lightness*).

- Untuk penulisan menggunakan format RGB intensitas warna memiliki nilai diantara 0 dan 255 pada setiap parameter (red, green, blue) dan format penulisan adalah **rgb(x,x,x)** dimana x adalah nilai parameter.
- Untuk penulisan menggunakan nilai hexadecimal bagian depan diberi awalan symbol “#” tanpa tanda kutip dengan format #rrggbb dimana rr (red), gg (green), bb (blue) adalah nilai hexadecimal dimulai dari 00 sampai ff.
- Untuk penulisan menggunakan format hsl memiliki nilai sebagai berikut :
 - Hue adalah derajat pada roda warna dari 0 hingga 360. 0 berwarna merah, 120 berwarna hijau, dan 240 berwarna biru.
 - Saturation adalah nilai persentase penuh tidaknya sebuah warna, 0% berarti warna abu-abu, dan 100% adalah warna penuh.
 - Lightness juga adalah persentase gelap terangnya sebuah warna. 0% adalah hitam, 50% tidak terang atau gelap, 100% putih.

Dengan format penulisan **hsl(h,s,l)**. Berikut adalah contoh penggunaan property color pada dokumen html.

```

7 ▼ <style>
8 ▼   h1.satu{
9     color: red;
10  }
11 ▼   h1.dua{
12     color: #0000ff;
13  }
14 ▼   h1.tiga{
15     color: rgb(255,255,0);
16  }
17 ▼   h1.empat{
18     color: hsl(248,53%,58%);
19  }
20 </style>
21

```

body h1 .empat

```

24 ▼ <body>
25     <h1 class="satu">Ini contoh penggunaan kontrol teks css
      color</h1>
26     <h1 class="dua">Ini contoh penggunaan kontrol teks css
      color</h1>
27     <h1 class="tiga">Ini contoh penggunaan kontrol teks css
      color</h1>
28     <h1 class="empat">Ini contoh penggunaan kontrol teks css
      color</h1>
29 </body>

```

Ini contoh penggunaan kontrol teks css color

Ini contoh penggunaan kontrol teks css color

Ini contoh penggunaan kontrol teks css color

Ini contoh penggunaan kontrol teks css color

Properti Text-Align

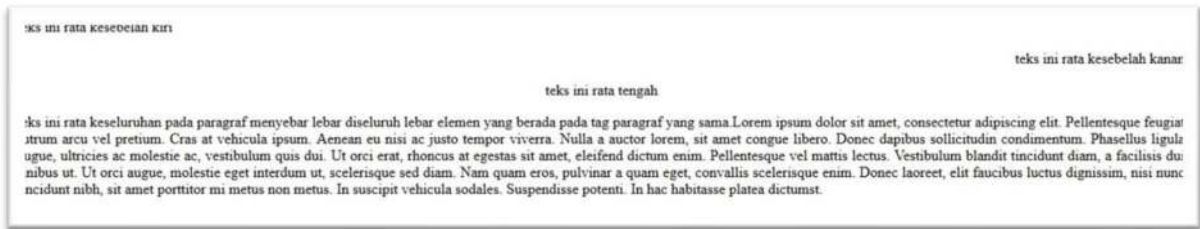
Property text-align bekerja seperti atribut pada format HTML. *Align* atau perata yang bisa diatur diantaranya rata kiri (*left*), kanan (*right*), tengah (*center*) dan rata keseluruhan paragraf (*justify*).

Berikut adalah contoh penggunaan property **text-align** pada sebuah elemen paragraf dengan **class** yang sudah ditentukan pada *script* css.

```

6 ▼ <style>
7 ▼ .leftAlign {
8     text-align : left;
9 }
10 ▼ .rightAlign {
11     text-align : right;
12 }
13 ▼ .center {
14     text-align : center;
15 }
16 ▼ .justify {
17     text-align : justify;
18 }
19 </style>

```



Properti Vertical-Align

Properti ini berguna saat bekerja dengan elemen sebaris (*inline*), pada gambar dan bagian teks tertentu. Ini memungkinkan Kita untuk mengontrol posisi vertikal mereka di dalam elemen yang berada didalamnya. Beberapa nilai yang dapat digunakan pada property ini diantaranya :

2.1.4.82	Nilai	2.1.4.83	Kegunaan
2.1.4.84	baseline	2.1.4.85	Menyelaraskan dengan garis dasar elemen induk. (Ini adalah pengaturan standar.)
2.1.4.86	sub	2.1.4.87	Membuat subscript elemen. Dengan gambar, bagian atas gambar harus pada garis dasar. Dengan teks, bagian atas badan font harus pada garis dasar.
2.1.4.88	super	2.1.4.89	Membuat superscript elemen. Dengan gambar, bagian bawah gambar harus sejajar dengan bagian atas font. Dengan teks, bagian bawah descender (bagian huruf seperti g dan p yang berada di bawah garis teks) harus sejajar dengan bagian atas badan font.
2.1.4.90	top	2.1.4.91	Sejajarkan bagian atas teks dan bagian atas gambar dengan bagian atas elemen tertinggi di baris.
2.1.4.92	text-top	2.1.4.93	Sejajarkan bagian atas teks dan bagian atas gambar dengan bagian atas teks tertinggi di baris.
2.1.4.94	middle	2.1.4.95	Sejajarkan titik tengah vertikal elemen dengan titik tengah vertikal induk.
2.1.4.96	bottom	2.1.4.97	Sejajarkan bagian bawah teks dan bagian bawah gambar dengan bagian bawah elemen terendah pada baris.
2.1.4.98	text-bottom	2.1.4.99	Sejajarkan bagian bawah teks dan bagian bawah gambar dengan bagian bawah teks terendah pada baris.

Properti ini juga dapat menurunkan (inherit) dari nilai dengan persentase. Berikut adalah contoh penggunaan property vertical-align. Script css : dengan class berbeda pada setiap nilai properti


```

<style>
img.a { vertical-align: none;}
img.b { vertical-align: baseline; }
img.c { vertical-align: sub; }
img.d { vertical-align: super; }
img.e { vertical-align: top; }
img.f { vertical-align: text-top; }
img.g { vertical-align: middle; }
img.h { vertical-align: bottom;}
img.i { vertical-align: text-bottom; }
img.j { vertical-align: 100%; }
img.k { vertical-align: 50%;}
</style>

```

Script HTML : dengan membuat sebuah teks dan gambar sehingga teks menyesuaikan penyerataan pada sebuah gambar.

```

<body>
<p>teks standard tanpa vertical align </p>
<p>teks standard baseline vertical align </p>
<p>teks standard sub vertical align </p>
<p>teks standard super vertical align </p>
<p>teks standard top vertical align </p>
<p>teks standard text-top vertical align </p>
<p>teks standard middle vertical align </p>
<p>teks standard bottom vertical align </p>
<p>teks standard text-bottom vertical align </p>
<p>teks standard 100% vertical align </p>
<p>teks standard 50% vertical align </p>
</body>

```

Hasil pada *Browser* :



Properti Text-Decoration

Properti ini digunakan untuk memberikan variasi pada suatu teks atau penanda. Beberapa nilai yang dapat digunakan pada properti ini adalah:

2.1.4.100 Nilai	2.1.4.101 Kegunaan
2.1.4.102 underline	2.1.4.103 Menambahkan baris di bawah konten.
2.1.4.104 overline	2.1.4.105 Menambahkan baris di atas konten.
2.1.4.106 line-through	2.1.4.107 Menambahkan garis melalui bagian tengah konten, seperti teks yang dicoret. Secara umum, ini harus digunakan hanya untuk menunjukkan teks yang ditandai untuk dihapus.
2.1.4.108 none	2.1.4.109 Menghapus semua variasi teks pada suatu elemen.

Berikut adalah contoh penggunaan properti text-decoration. Script css : ditulis pada bagian head

```
<style>
h1.underline {
    text-decoration : underline;
}
h1.overline {
    text-decoration : overline;
}
h1.line-through {
    text-decoration : line-through;
}
h1.none {
    text-decoration : none;
}
</style>
```

Script HTML: ditulis pada bagian body

```
<body>
  <h1 class="underline">teks ini seharusnya ada garis bawah</h1>
  <h1 class="overline">teks ini seharusnya ada garis atas</h1>
  <h1 class="line-through">teks ini seharusnya ada garis coret</h1>
  <h1 class="none">teks ini seharusnya tidak memiliki variasi</h1>
</body>
```

Hasil pada *browser*:



Properti Text-Indent

Properti ini memungkinkan Kita untuk mengatur indentasi baris teks pertama dalam suatu elemen, sehingga baris pertama pada paragraf tersebut menjorok kedalam sesuai dengan nilai yang sudah Kita tentukan. Berikut adalah contoh penggunaan properti text-indent:

Script css:

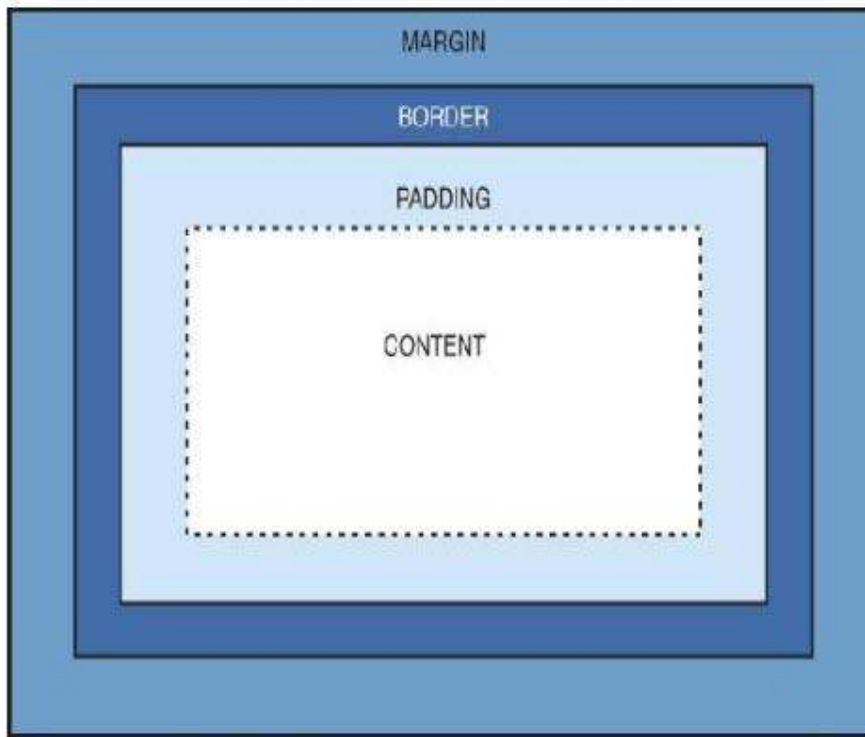
```
<style>
.indent {
    text-indent : 3em;
}
</style>
```

Box Model

Box model adalah konsep penting dalam CSS karena menentukan bagaimana elemen diposisikan dalam jendela *browser*. Karena CSS memperlakukan setiap elemen seolah-olah berada didalam sebuah kotak (*box*). Properti box model yang penting dapat dilihat pada tabel dibawah ini :

2.1.4.110 Properti	2.1.4.111 Deskripsi
2.1.4.112 border	2.1.4.113 Bahkan jika Anda tidak dapat melihatnya, setiap kotak memiliki batas (border). Ini memisahkan tepi satu kotak dari kotak lain di sekitarnya.

2.1.4.114 margin	2.1.4.115 Margin adalah jarak antara batas kotak dan kotak di sebelahnya.
2.1.4.116 padding	2.1.4.117 Padding adalah ruang antara isi kotak dan perbatasannya



Kita dapat menggunakan CSS untuk secara individual mengontrol *border*, *margin*, dan *padding* di setiap sisi kotak. Kita dapat menentukan lebar, gaya garis, dan warna yang berbeda untuk setiap sisi batas kotak.

Sifat *padding* dan *margin* sangat penting dalam menciptakan apa yang oleh perancang disebut sebagai ruang putih (*white space*), ini adalah ruang di antara berbagai bagian halaman.

Misalnya, jika Kita memiliki kotak dengan batas hitam dan kotak berisi teks hitam, Kita tidak ingin teks menyentuh perbatasan karena akan membuat teks lebih sulit dibaca. Memberikan *padding* kotak membantu memisahkan teks dari garis di seKitar tepi.

Sementara itu, anggaplah Kita memiliki dua kotak di samping satu sama lain, keduanya berbatasan. Jika tidak ada margin di antara mereka, kotak akan bertemu satu sama lain, dan garis tempat kotak bertemu bisa terlihat lebih tebal daripada garis lainnya.

Ilustrasi Box Model

Untuk mengilustrasikan *box model*, Kita bisa menambahkan *border* pada setiap elemen dalam halaman web.

Script css: setiap elemen memiliki border box masing-masing dan pada element `<h1>` dan `` diberi warna abu-abu untuk membantu membedakan dari elemen lain.

```

<style>
body, h1, p, img, b {
  border-style : solid;
  border-width : 2px;
  border-color : #000000;
  padding:2px;
}
h1, b {
  background-color : #cccccc;
}
</style>

```

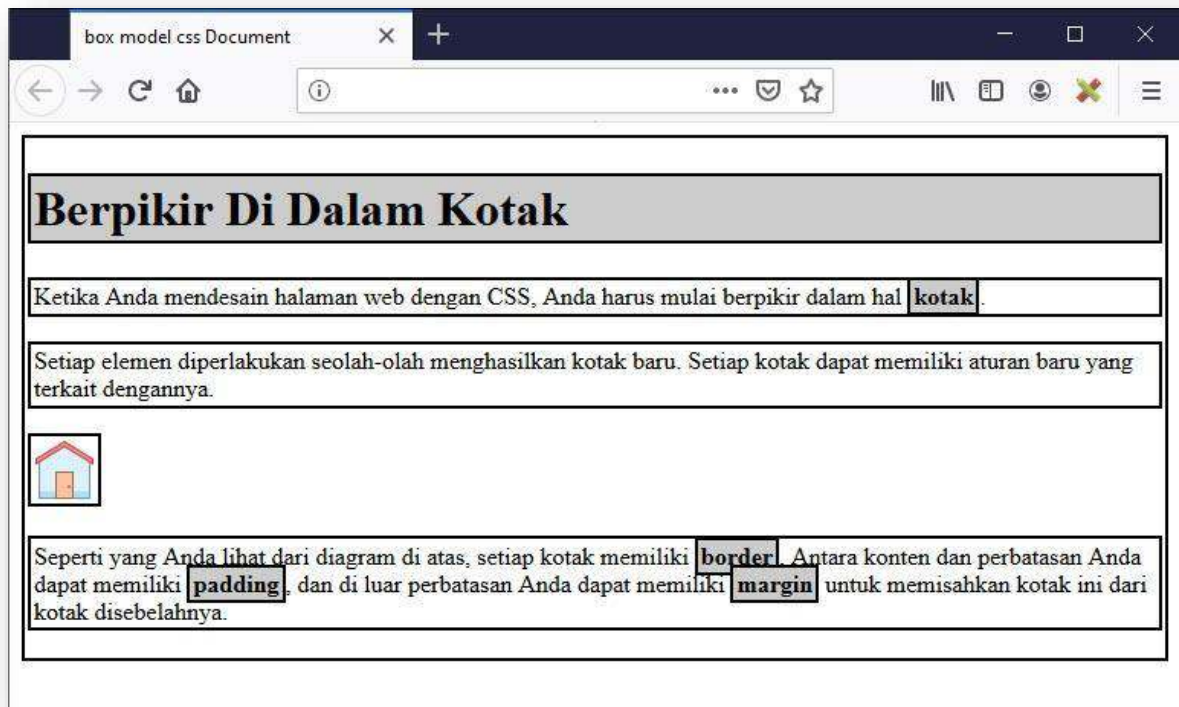
Script HTML :

```

<body>
  <h1>Berpikir Di Dalam Kotak</h1>
  <p class="description">Ketika Anda mendesain halaman web dengan CSS, Anda harus mulai berpikir dalam hal <b>kotak</b>.</p>
  <p>Setiap elemen diperlakukan seolah-olah menghasilkan kotak baru. Setiap kotak dapat memiliki aturan baru yang terkait dengannya.</p>
  
  <p>Seperti yang Anda lihat dari diagram di atas, setiap kotak memiliki <b>border</b>. Antara konten dan perbatasan Anda dapat memiliki <b>padding</b>, dan di luar perbatasan Anda dapat memiliki <b>margin</b> untuk memisahkan kotak ini dari kotak disebelahnya.</p>
</body>

```

Hasil pada *browser* :



Pada buku lain yang membahas dasar HTML (Panduan Definitif untuk HTML5) dibahas mengenai perbedaan antara elemen blok dan elemen *inline*, perbedaan ini menjadi sangat penting ketika bekerja dengan CSS karena ia menentukan bagaimana setiap kotak ditampilkan.

Border

Properti Box model yang pertama adalah border. Berbeda dengan border pada HTML, border dalam CSS berdiri sendiri tanpa bantuan tabel. Salah satu kelebihan border dalam CSS adalah Kita bisa memilih *style* yang beragam, selain itu Kita bisa mengatur bagian atas, bawah, kanan, dan kiri border dengan pengaturan yang berbeda.

Properti Border

Properti border memungkinkan Kita menentukan bagaimana batas kotak yang mewakili elemen akan terlihat. Ada 3 (tiga) properti border yang dapat diubah :

1. ***border-color*** digunakan untuk mengatur warna pada border.
2. ***border-style*** digunakan untuk mengatur hiasan pada border dapat berupa *solid*, *dashed*, *double line* atau salah satu dari nilai yang mungkin lainnya.
3. ***border-width*** digunakan untuk mengatur lebar sebuah border.

Properti Border-Width

Properti ini memungkinkan Kita untuk menentukan lebar dari sebuah border, biasanya ditentukan dalam satuan *pixel* (px). Nilai properti *border-width* tidak dapat menggunakan

persentase, meskipun Kita dapat menggunakan absolut unit atau relative unit, atau salah satu dari nilai berikut :

- *thin*
- *medium*
- *thick*

Nilai diatas tidak ditentukan dalam rekomendasi css, sehingga lebar sebenarnya yang sesuai dengan nilai diatas ini bergantung pada *browser*.

Kita dapat mengubah lebar batas bagian bawah, kiri, atas, dan kanan kotak secara individual menggunakan properti berikut:

- border-bottom-width
- border-right-width
- border-top-width
- border-left-width

Properti Border-Style

Properti ini memungkinkan Kita untuk menentukan garis apa yang akan digunakan sebagai pembatas (*border*). Nilai *default* dari properti ini adalah “**none**” jadi tidak ada border yang akan ditampilkan secara otomatis. Berikut adalah nilai yang dapat digunakan pada *border-style*.

2.1.4.118	Nilai	2.1.4.119	Deskripsi
2.1.4.120	none	2.1.4.121	Tidak ada border (setara dengan border-width:0;)
2.1.4.122	solid	2.1.4.123	Border adalah garis solid tunggal.
2.1.4.124	dotted	2.1.4.125	Border adalah serangkaian titik.
2.1.4.126	dashed	2.1.4.127	Border adalah serangkaian garis pendek.
2.1.4.128	double	2.1.4.129	Border adalah dua garis solid, nilai properti border-width menciptakan jumlah dari dua garis dan ruang diantara mereka.
2.1.4.130	groove	2.1.4.131	Border tampak seolah-olah diukir kehalaman.
2.1.4.132	ridge	2.1.4.133	Border terlihat kebalikan dari groove.
2.1.4.134	inset	2.1.4.135	Border membuat kotak terlihat seperti tertanam di page.
2.1.4.136	outset	2.1.4.137	Border membuat kotak tampak seperti keluar kanvas.
2.1.4.138	hidden	2.1.4.139	Sama seperti none, kecuali dalam hal resolusi konflik untuk elemen tabel.

Kita dapat mengubah *style* batas bagian bawah, kiri, atas, dan kanan kotak secara individual menggunakan properti berikut:

- border-bottom-style

- border-right- style
- border-top- style
- border-left- style

Properti Border-Color

Properti ini memungkinkan Kita untuk mengubah warna border disekitar kotak.

Sintaks:

Elemen { border-color : nilai valid warna; }

Nilai dapat digunakan menggunakan nilai valid warna pada css (hexadecimal, rgb, rgba, hsl, hsla, nama warna).

Kita dapat mengubah warna batas bagian bawah, kiri, atas, dan kanan kotak secara individual menggunakan properti berikut:

- border-bottom-color
- border-right-color
- border-top-color
- border-left-color

Properti Border (singkat)

Properti border memungkinkan Kita secara singkat menentukan warna, style dan lebar dalam satu property.

Jika Kita menggunakan singkatan ini, nilainya tidak boleh memiliki apa pun (selain spasi) di antara mereka.

Contoh:

```
p { border : 4px solid red; }
```

Kita juga dapat menentukan warna, style, dan lebar garis secara terpisah untuk setiap sisi kotak dengan cara yang sama menggunakan properti ini:

- border-bottom
- border-top
- border-left
- border-right

berikut adalah contoh penggunaan properti border:

script html:

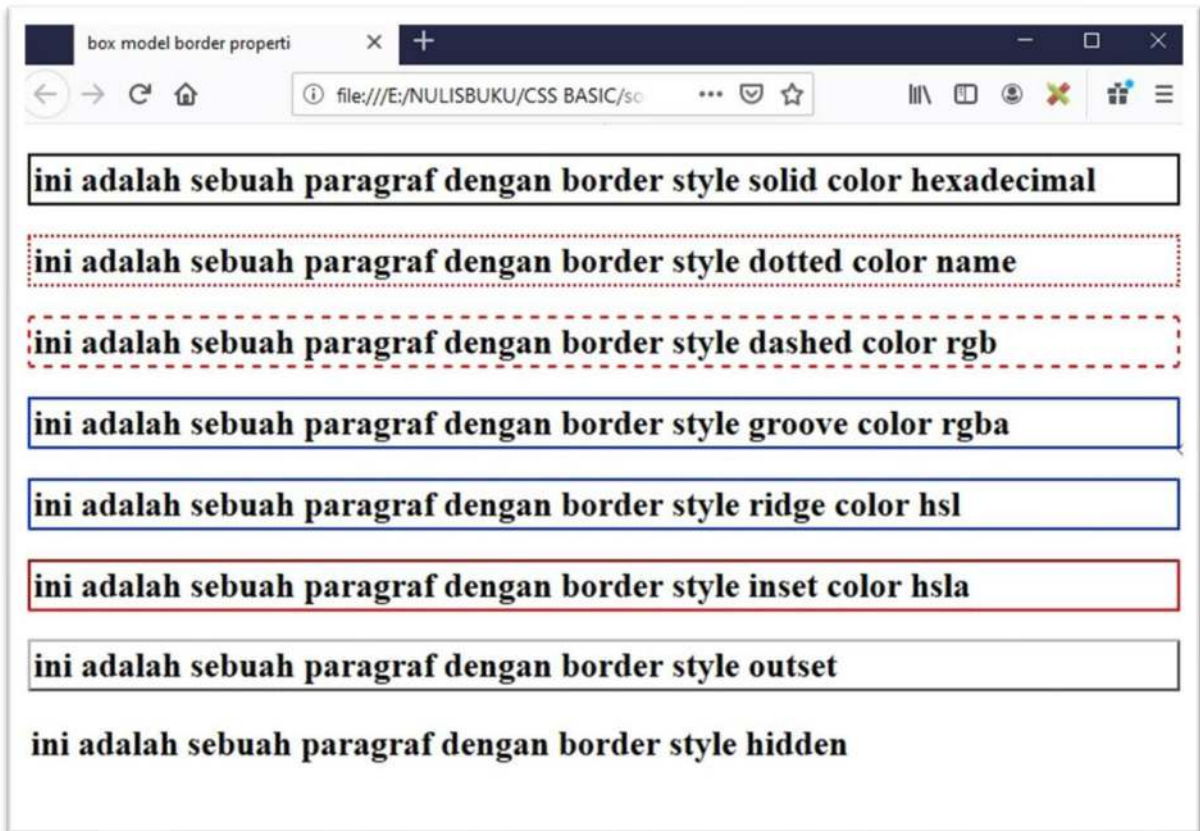
```
<h2 class="border-solid">ini adalah sebuah paragraf dengan border style solid color hexadecimal</h2>
<h2 class="border-dotted">ini adalah sebuah paragraf dengan border style dotted color name</h2>
<h2 class="border-dashed">ini adalah sebuah paragraf dengan border style dashed color rgb</h2>
<h2 class="border-groove">ini adalah sebuah paragraf dengan border style groove color rgba</h2>
<h2 class="border-ridge">ini adalah sebuah paragraf dengan border style ridge color hsl</h2>
<h2 class="border-inset">ini adalah sebuah paragraf dengan border style inset color hsla</h2>
<h2 class="border-outset">ini adalah sebuah paragraf dengan border style outset</h2>
<h2 class="border-hidden">ini adalah sebuah paragraf dengan border style hidden</h2>
</body>
```

Script css dibagi menjadi 2 (dua) gambar yang dituliskan pada bagian **head html**:

```
<style>
  .border-solid{
    border-style : solid;
    border-width : 2px;
    border-color : #000000;
    padding:2px;
  }
  .border-dotted{
    border-style : dotted;
    border-width : 2px;
    border-color : red;
    padding:2px;
  }
  .border-dashed{
    border-style : dashed;
    border-width : 2px;
    border-color : rgb(255,0,0);
    padding:2px;
  }
  .border-groove{
    border-style : groove;
    border-width : 2px;
    border-color : rgba(0,51,255,1.00);
    padding:2px;
  }
}
```

```
.border-ridge{
  border-style : ridge;
  border-width : 2px;
  border-color : hsl(228,100%,50%);
  padding:2px;
}
.border-inset{
  border-style : inset;
  border-width : 2px;
  border-color : hsla(359,100%,50%,1.00);
  padding:2px;
}
.border-outset{
  border-style : outset;
  border-width : 2px;
  border-color : #000000;
  padding:2px;
}
.border-hidden{
  border-style : hidden;
  border-width : 2px;
  padding:2px;
}
</style>
```

Hasil pada *browser*:



Margin dan Padding

Pengaturan *margin* dan *padding* yang baik dapat membuat dokumen HTML terlihat lebih rapih dan profesional. Dengan menggunakan CSS Kita bisa mengaturnya sesuai dengan keinginan.

Properti Padding

Properti padding memungkinkan Kita menentukan berapa banyak ruang yang harus muncul antara konten elemen dan border.

Nilai properti ini paling sering ditentukan dalam piksel. Meskipun, itu dapat menggunakan salah satu unit panjang yang Kita temui sebelumnya, persentase, atau kata yang diwarisi (*inherit*).

Padding suatu elemen tidak mewarisi secara *default*, jadi jika elemen **<body>** memiliki properti *padding* dengan nilai 50 piksel, ini tidak secara otomatis berlaku untuk semua elemen lain di dalamnya. Jika nilai *inherit* diterapkan ke elemen apa pun, hanya dengan demikian mereka dapat memiliki *padding* yang sama dengan elemen induknya.

Jika persentase digunakan, persentase kotak berisi, dan jika nilai 10 persen ditentukan, akan ada 5 persen dari masing-masing sisi kotak sebagai *padding*.

Kita dapat menentukan jumlah *padding* yang berbeda di dalam setiap sisi kotak menggunakan properti berikut:

- *padding-bottom*
- *padding-top*
- *padding-left*
- *padding-right*

Atribut *padding* sangat membantu untuk membuat ruang putih (*white space*) antara konten elemen dan setiap perbatasan yang dimilikinya.

Berikut adalah contoh penggunaan properti padding.

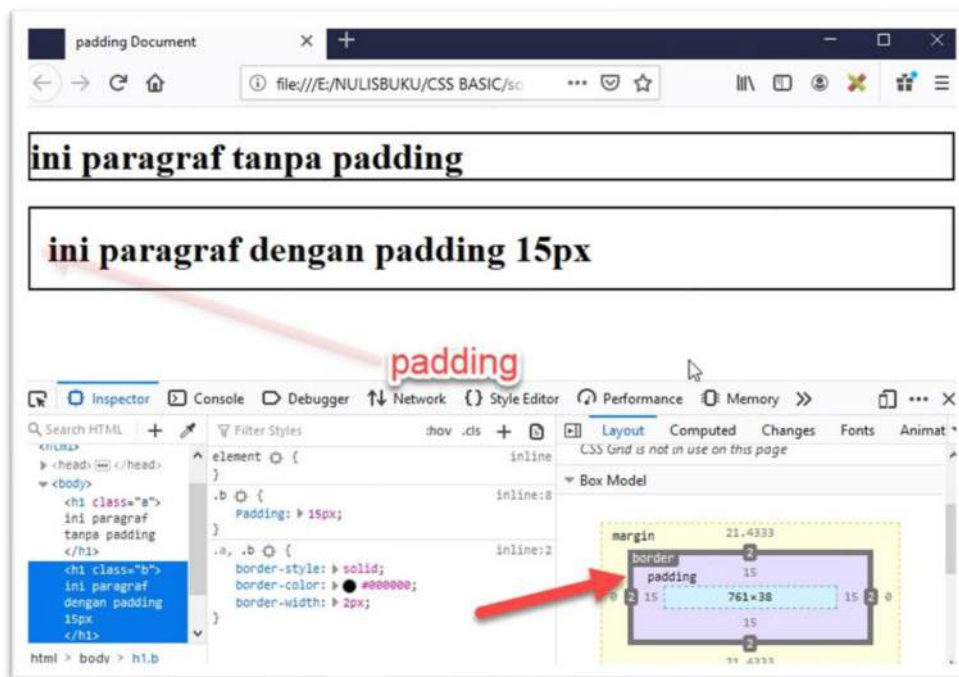
Script html dan css:

```

1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>padding Document</title>
6
7 <style>
8 .a, .b {
9     border-style : solid;
10    border-color : #000000;
11    border-width : 2px;
12
13 }
14 .b {
15     Padding : 15px;
16 }
17 </style>
18
19 </head>
20
21 <body>
22 <h1 class="a">ini paragraf tanpa padding</h1>
23 <h1 class="b"> ini paragraf dengan padding 15px</h1>
24 </body>
25 </html>
26

```

Hasil pada *browser* dengan *inspect element* :



Terkadang, suatu elemen mungkin tidak memiliki *border* yang terlihat, tetapi memiliki warna atau pola latar belakang. Dalam kasus seperti itu, berikan *padding* pada beberapa kotak dapat membantu membuat desain Kita lebih menarik.

Properti Margin

Properti margin mengontrol celah antara kotak, dan nilainya panjang menggunakan angka, persentase, atau warisan (*inherit*), yang masing-masing memiliki arti yang persis sama seperti yang terjadi pada properti *padding* yang baru saja Kita lihat.

Seperti halnya properti *padding*, nilai properti *margin* tidak diwarisi oleh elemen turunan kecuali Anda menggunakan nilai *inherit*.

Juga, ingatlah bahwa ketika satu kotak berada di atas kotak yang lain, hanya yang lebih besar dari dua margin yang ditampilkan (atau jika keduanya sama, ukuran satu margin).

Kita juga dapat mengatur nilai yang berbeda untuk margin di setiap sisi kotak menggunakan properti berikut:

- *margin-bottom*
- *margin-top*
- *margin-left*
- *margin-right*

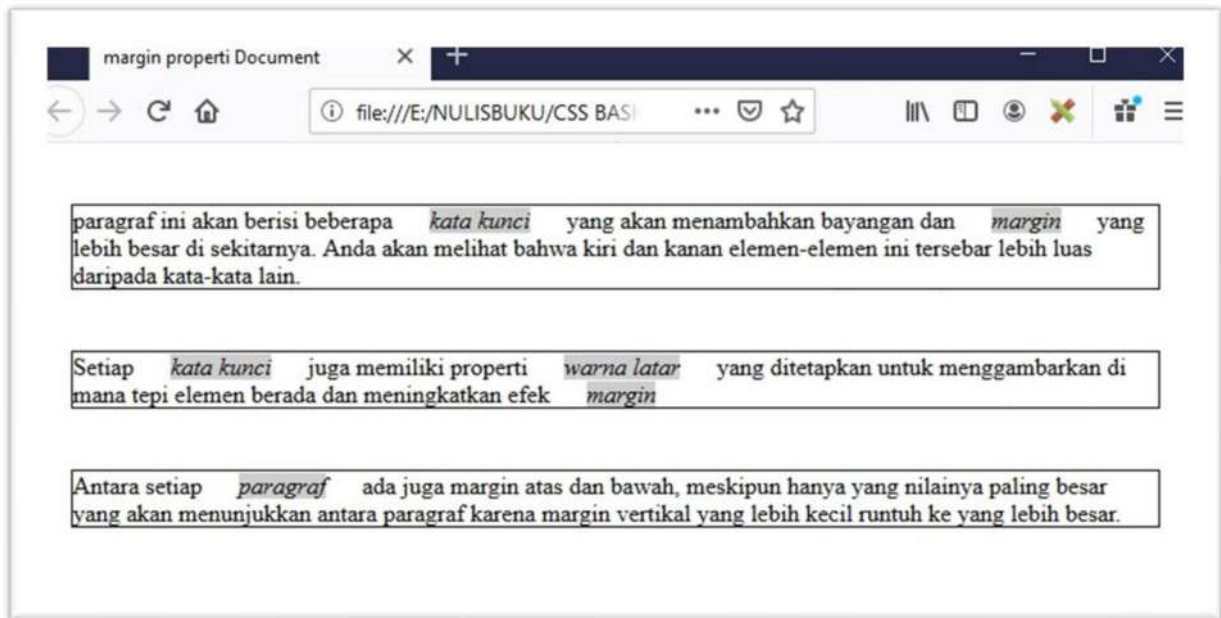
Berikut adalah contoh penggunaan properti margin pada html dan css:

```
<style>
p {
  margin-top : 40px;
  margin-bottom : 30px;
  margin-left : 20px;
  margin-right : 20px;
  border-style : solid;
  border-width : 1px;
  border-color : #000000;}
em {
  background-color : #cccccc;
  margin-left : 20px;
  margin-right : 20px;
}
</style>

</head>

<body>
<p>paragraf ini akan berisi beberapa <em>kata kunci</em> yang akan menambahkan bayangan dan
<em>margin</em> yang lebih besar di sekitarnya. Anda akan melihat bahwa kiri dan kanan elemen-
elemen ini tersebar lebih luas daripada kata-kata lain.</p>
<p>Setiap <em>kata kunci</em> juga memiliki properti <em>warna latar</em> yang ditetapkan untuk
menggambarkan di mana tepi elemen berada dan meningkatkan efek <em>margin</em></p>
<p>Antara setiap <em>paragraf</em> ada juga margin atas dan bawah, meskipun hanya yang nilainya
paling besar yang akan menunjukkan antara paragraf karena margin vertikal yang lebih kecil
runtuh ke yang lebih besar.</p>
</body>
</html>
```

Hasil pada *browser*:



Kita dapat melihat tiga paragraf, yang terlihat seolah-olah diberi spasi sama. Namun, mereka memiliki *margin* yang lebih tinggi di bagian atas daripada bagian bawah, dan oleh karena itu di mana dua kotak bertemu, margin bawah diabaikan — marginnya diciutkan. (Ini hanya terjadi pada **margin vertikal**, bukan margin kiri dan kanan.)

Script HTML:

```
<body>
  <p>Paragraf ini harus disejajarkan dengan sisi kiri browser</p>
  <p class="indent">Hanya baris pertama paragraf ini yang harus di indentasi oleh
    3 em, ini seharusnya tidak berlaku untuk baris berikutnya dalam paragraf yang
    sama.</p>
</body>
```

Hasil pada *browser*:



Properti Text-Shadow

Property ini memberikan efek bayangan pada sebuah teks atau paragraf. Sering digunakan di media cetak. Nilai untuk properti ini cukup rumit karena dapat mengambil warna diikuti 3 (tiga) nilai yang lainnya.

Dua nilai yang pertama menentukan seberapa jauh bayangan dari teks asli menggunakan koordinat X dan Y, nilai yang ketiga sebelum nilai warna adalah menentukan seberapa blur bayangan seharusnya, nilai terakhir adalah nilai warna pada umumnya sebuah css.

Contoh penggunaan properti text-shadow adalah sebagai berikut:

Script css: pada bagian head dibuat tag style dengan class dropShadow. Nilai koordinat x dan y adalah 2px dan nilai blur 3px dengan warna merah (#ff0000).

```
<style>
.dropShadow {
    text-shadow : 2px 2px 3px #ff0000;
}
</style>
```

Script HTML:

```
<body>
    <h1 class="dropShadow">ini text dengan properti text-shadow</h1>
</body>
```

Hasil pada *browser*:



Properti Text-Transform

Properti ini digunakan untuk mengubah teks pada suatu konten elemen. Baik menjadi *capitalize*, *uppercase* atau *lowercase*. Beberapa nilai yang dapat digunakan pada property ini adalah sebagai berikut :

2.1.4.140 Nilai	2.1.4.141 Kegunaan
2.1.4.142 none	2.1.4.143 tidak ada perubahan yang terjadi
2.1.4.144 capitalize	2.1.4.145 mengubah karakter huruf pertama menjadi kapital / huruf besar.

2.1.4.146 uppercase	2.1.4.147 membuat seluruh konten elemen huruf besar.
2.1.4.148 lowercase	2.1.4.149 membuat seluruh konten elemen huruf kecil.

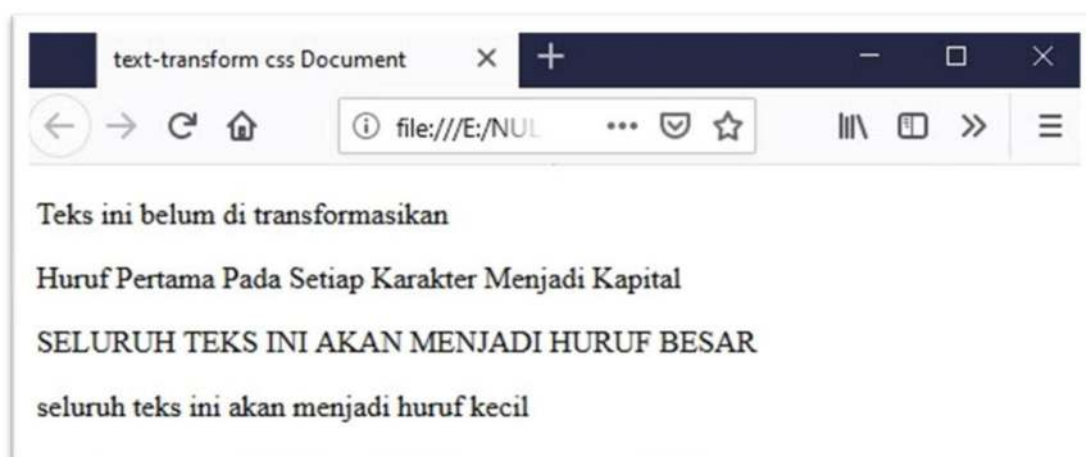
Berikut adalah contoh penggunaan *text-decoration* properti. Script css: pada script ini kita akan membuat class dengan fungsi yang sama dengan nilai properti text-decoration.

```
<style>
p.none {
  text-transform : none;
}
p.capitalize {
  text-transform : capitalize;
}
p.uppercase {
  text-transform : uppercase;
}
p.lowercase {
  text-transform : lowercase;
}
</style>
```

Script HTML: kita akan membuat beberapa paragraf elemen dengan class yang sudah dibuat pada script css.

```
<body>
<p class="none">Teks ini belum di transformasikan</p>
<p class="capitalize">huruf pertama pada setiap karakter menjadi kapital</p>
<p class="uppercase">seluruh teks ini akan menjadi huruf besar</p>
<p class="lowercase">SELURUH TEKS INI AKAN MENJADI HURUF KECIL</p>
</body>
```

Hasil: hasil pada *browser* dapat dilihat pada gambar dibawah ini.



Properti Letter-Spacing

Properti ini memungkinkan Anda untuk mengatur jarak antar karakter. Hal ini biasanya digunakan oleh seorang desainer cetak untuk mengatur jarak antar karakter tersebut. Jika Kita ingin menambah atau mengurangi jarak antar karakter, kemungkinan besar Kita menentukannya dalam **pixel** atau sesuatu yang dikenal dengan **em**. Meskipun satuan panjang lain yang didukung oleh CSS bisa digunakan).

Berikut adalah contoh penggunaan properti letter-spacing. Script css: Saya akan menggunakan 3 buah class untuk memberi nilai pada properti letter-spacing.

```
<style>
.satu {
  letter-spacing : 3px;
}
.dua {
  letter-spacing : 0.5em;
}
.tiga {
  letter-spacing : -1px;
}
</style>
```

Script html: memiliki 3 paragraf dengan class yang sudah dibuat pada script css.

```
<body>
<p>ini adalah normal letter spacing</p>
<p class="satu">pada kalimat ini menggunakan letter spacing bernilai 3px</p>
<p class="dua"> pada kalimat ini menggunakan letter spacing bernilai 0.5 em</p>
<p class="tiga">pada kalimat ini menggunakan letter spacing -1px</p>
</body>
```

Hasil pada *browser*:



Properti Word-Spacing

Pada properti ini Kita dapat mengatur jarak antar kata pada sebuah dokumen HTML. Penggunaannya sama seperti letter-spacing properti. Berikut adalah contoh penggunaan properti word-spacing.

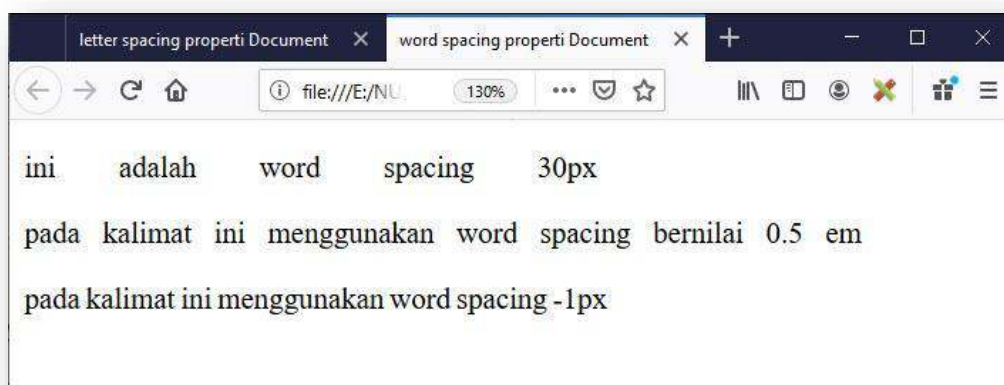
Script css:

```
<style>
.satu {
  word-spacing: 30px;
}
.dua {
  word-spacing : 0.5em;
}
.tiga {
  word-spacing : -1px;
}
</style>
```

Script HTML:

```
<body>
  <p class="satu">ini adalah word spacing 30px </p>
  <p class="dua"> pada kalimat ini menggunakan word spacing bernilai 0.5 em</p>
  <p class="tiga">pada kalimat ini menggunakan word spacing -1px</p>
</body>
```

Hasil pada *browser*:



Properti White-Space

Properti ini mengontrol apakah white space (ruang putih pada editor) dipertahankan atau diabaikan. Ada beberapa nilai yang dapat digunakan pada properti white-space terkait penggunaan white space pada sebuah dokumen html.

2.1.4.150 Nilai	2.1.4.151 Kegunaan
2.1.4.152 normal	2.1.4.153 normal sesuai halaman web.
2.1.4.154 pre	2.1.4.155 mempertahankan white space seperti pada editor HTML.
2.1.4.156 pre-line	2.1.4.157 mempertahankan white space seperti pada editor HTML tetapi pada value pre-line jarak setiap per-kata hanya akan menampilkan satu spasi seperti pada umumnya walaupun anda mengetikkan lebih dari satu spasi.dapat membuat baris baru jika diperlukan.
2.1.4.158 pre-wrap	2.1.4.159 mempertahankan white space seperti pada editor apabila baris teks akan mengalir keluar melebihi batas elemen ia akan membuat baris baru sehingga baris teks tetap berada di dalam area pembungkusnya.
2.1.4.160 nowrap	2.1.4.161 teks akan terus memanjang (tidak membuat baris baru untuk menyesuaikan tampilan di <i>browser</i>) jika tidak secara eksplisit menggunakan elemen <code>
</code> .

Berikut adalah contoh penggunaan properti white-space. Script css: akan dibuat sebuah class dengan nilai properti white-space yang berbeda-beda.

```
<style>

```

Script HTML :

```

<h2>normal white space</h2>
<h3 class="normal">
Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-
space dengan mengklik salah satu properti di sebelah kiri.
</h3>
<h2>nowrap white space</h2>
<h3 class="nowrap">
Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-
space dengan mengklik salah satu properti di sebelah kiri.
</h3>
<h2>pre white space</h2>
<h3 class="pre">
Contoh ini menunjukkan properti
white space. Anda dapat melihat hasil
dari berbagai properti
white-space dengan mengklik
salah satu properti di sebelah kiri.
</h3>
<h2>pre-line white space</h2>
<h3 class="preline">
Contoh ini menunjukkan properti
white space. Anda dapat melihat hasil
dari berbagai properti
white-space dengan mengklik
salah satu properti di sebelah kiri.
</h3>
<h2>pre-wrap white space</h2>
<h3 class="prewrap">
Contoh ini menunjukkan properti
white space. Anda dapat melihat hasil
dari berbagai properti
white-space dengan mengklik
salah satu properti di sebelah kiri.
</h3>

```

Hasil pada *browser* :

normal white space

Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-space dengan mengklik salah satu properti di sebelah kiri.

nowrap white space

Contoh ini menunjukkan properti white space. Anda dapat melihat

pre white space

Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-space dengan mengklik salah satu properti di sebelah kiri.

pre-line white space

Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-space dengan mengklik salah satu properti di sebelah kiri.

pre-wrap white space

Contoh ini menunjukkan properti white space. Anda dapat melihat hasil dari berbagai properti white-space dengan mengklik salah satu properti di sebelah kiri.

Properti Direction

Properti ini seperti elemen <dir> pada HTML yaitu menentukan arah dimana teks harus mengalir. Berikut adalah properti dari *direction* :

2.1.4.162 Nilai	2.1.4.163 Kegunaan
2.1.4.164 ltr	2.1.4.165 teks mengalir dari kiri ke kanan.

2.1.4.166 rtl	2.1.4.167 teks mengalir dari kanan ke kiri.
2.1.4.168 inherit	2.1.4.169 teks mengalir sesuai dengan arah dari elemen induk.

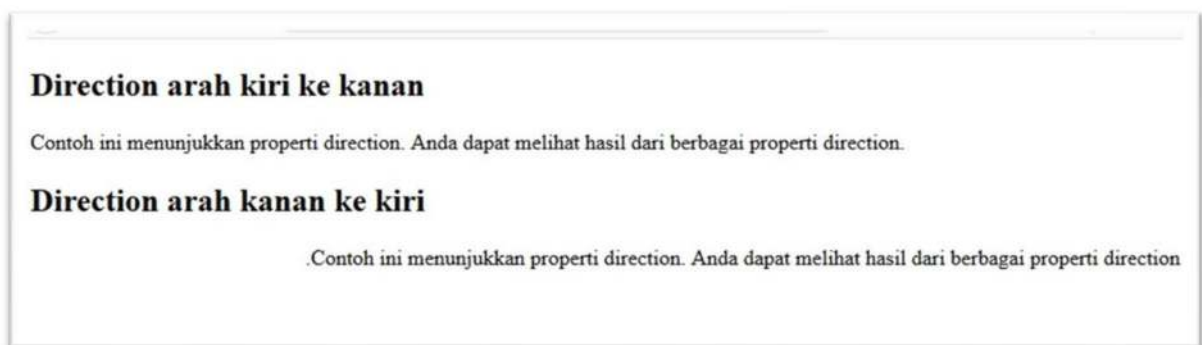
Berikut adalah contoh penggunaan properti direction: Script CSS:

```
<style>
p.ltr { direction : ltr; }
p.rtl { direction : rtl; }
</style>
```

Script HTML :

```
<body>
  <h2>Direction arah kiri ke kanan</h2>
  <p class="ltr">
    Contoh ini menunjukkan properti direction. Anda dapat melihat hasil dari berbagai properti
    direction.
  </p>
  <h2>Direction arah kanan ke kiri</h2>
  <p class="rtl">
    Contoh ini menunjukkan properti direction. Anda dapat melihat hasil dari berbagai properti
    direction.
  </p>
</body>
```

Hasil pada *browser* :



CSS Layout

Properti **display** adalah properti CSS paling penting untuk mengontrol tata letak. Properti **display** menentukan jika / bagaimana sebuah elemen ditampilkan. Setiap elemen HTML

memiliki nilai tampilan default tergantung pada jenis elemennya. Nilai tampilan default untuk sebagian besar elemen adalah **blok** atau **inline**.

Block Level Elements

Block Level Elements selalu dimulai pada baris baru dan menempati lebar penuh yang tersedia (membentang ke kiri dan kanan sejauh mungkin).

Contoh:

- `<div>`
- `<h1>` - `<h6>`
- `<p>`
- `<form>`
- `<header>`
- `<footer>`
- `<section>`

Inline Elements

Inline Elements tidak dimulai pada baris baru dan hanya membutuhkan lebar yang diperlukan.

Contoh :

- ``
- `<a>`
- ``

Properti Display: none

Biasanya digunakan dengan JavaScript untuk menyembunyikan dan menampilkan elemen tanpa menghapus dan membuatnya kembali.

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  h1.hide {
6  display: none;
7  }
8  </style>
9  </head>
10 <body>
11
12 <h1>Ini Adalah Heading yang terlihat</h1>
13 <h1 class="hide">Ini adalah Heading yang tidak terlihat</h1>
14 <p>Perhatikan bahwa elemen h1 dengan display: none; tidak memakan tempat.</p>
15 </body>
16 </html>
17

```



Override the Default Display Value

Seperti yang disebutkan, setiap elemen memiliki nilai tampilan default. Namun, Anda dapat menyimpannya.

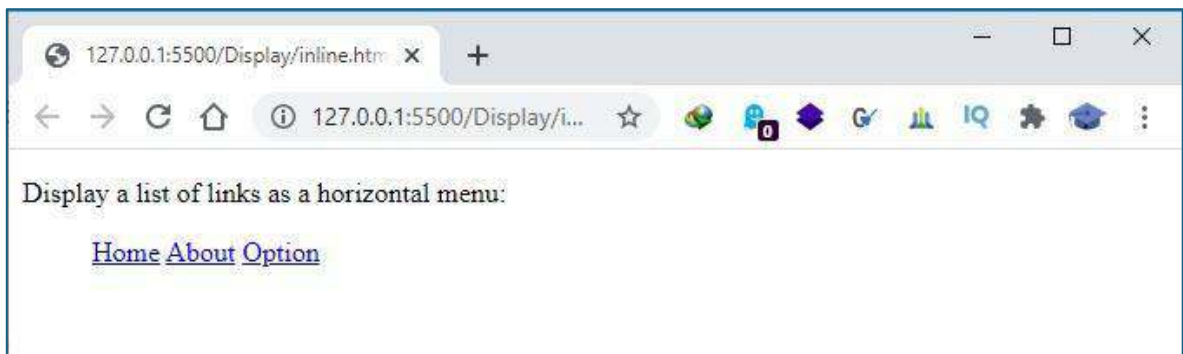
Mengubah elemen inline menjadi elemen blok, atau sebaliknya, dapat berguna untuk membuat laman terlihat dengan cara tertentu, dan tetap mengikuti standar web.

Contoh umum adalah membuat elemen inline `` untuk menu horizontal:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  li {
6  display: inline;
7  }
8  </style>
9  </head>
10 <body>
11
12 <p>Display a list of links as a horizontal menu:</p>
13
14 <ul>
15     <li class="current"><a href="#">Home</a></li>
16     <li><a href="#">About</a></li>
17     <li><a href="#">Option</a></li>
18 </ul>
19
20 </body>
21 </html>

```



Contoh : Inline menjadi blok

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 span {
6   display: block;
7 }
8 </style>
9 </head>
10 <body>
11
12 <span>Properti Display dengan nilai "block" menghasilkan</span>
13 <span>pemisah baris (baris baru) antara dua elemen.</span>
14
15 </body>
16 </html>
17
```



Properti Visibility: hidden

Selain dengan menggunakan `display : none;` ada properti css yang berfungsi untuk menyembunyikan elemen yaitu `visibility : hidden`. Namun, elemen tersebut akan tetap menempati ruang yang sama seperti sebelumnya. Elemen tersebut akan disembunyikan, tetapi masih memengaruhi tata letak:

```

1  <!DOCTYPE html>
2  <html>
3  <head>
4  <style>
5  h1.hidden{
6  visibility: hidden;
7  }
8  </style>
9  </head>
10 <body>
11
12 <h1>Ini adalah Heading yang Terlihat</h1>
13 <h1 class="hidden">Ini adalah Heading yang di sembunyikan</h1>
14 <p>Perhatikan bahwa judul tersembunyi masih membutuhkan tempat.</p>
15
16 </body>
17 </html>

```



CSS *Layout* Float dan Clear

Jika Kita sering melakukan penempatan gambar pada sebuah teks dalam ms word maka terkadang sebuah gambar ditempatkan seolah - oleh berada pada lapisan atas sebuah paragraf tertentu.

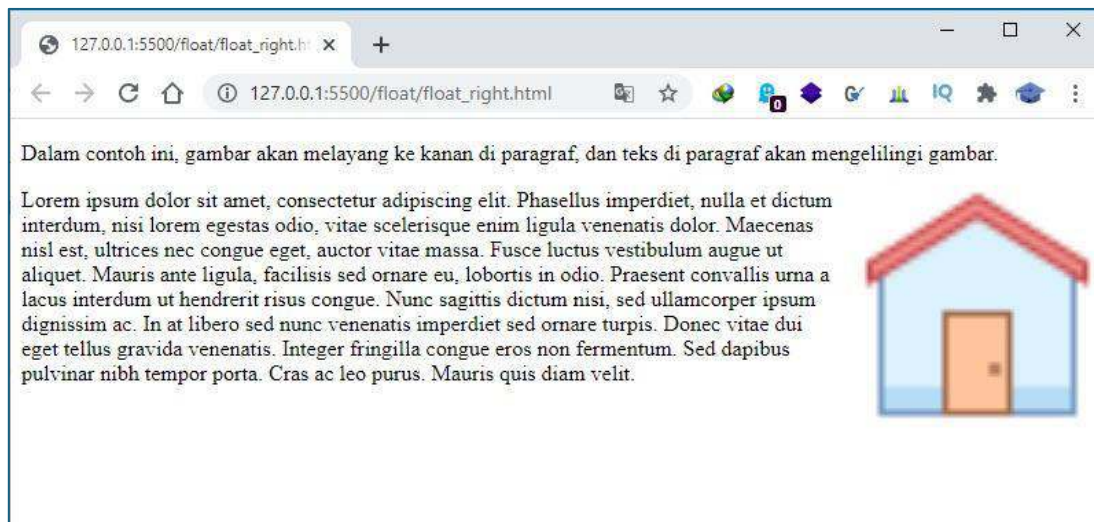
Properti CSS float menentukan bagaimana sebuah elemen harus mengapung (float pada *layout* tertentu) sehingga tampilan seolah - olah berada pada lapisan atasnya.

Properti CSS clear menentukan elemen apa yang dapat mengapung (float) di samping elemen yang dibersihkan dan di sisi mana.

Properti Float

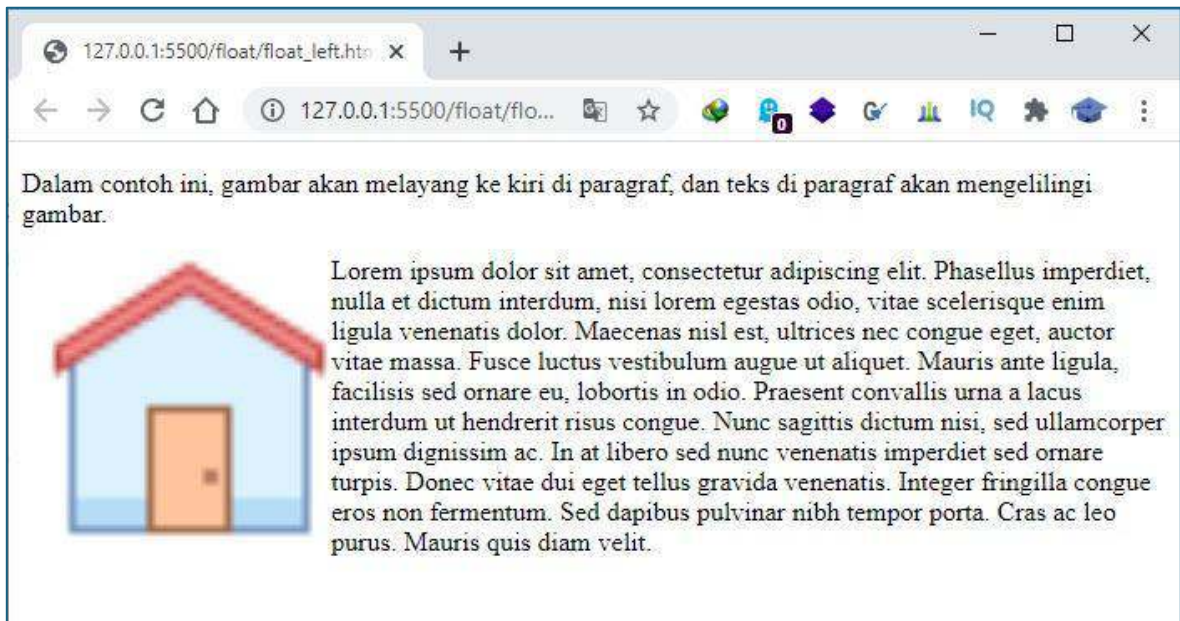
Contoh Float

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 img {
6   float: right;
7 }
8 </style>
9 </head>
10 <body>
11
12 <p>Dalam contoh ini, gambar akan melayang ke kanan di paragraf, dan teks di paragraf akan
mengelilingi gambar.</p>
13 <p>
14 Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum
interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl
est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris
ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut
hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero
sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis.
Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo
purus. Mauris quis diam velit.</p>
15
16 </body>
17 </html>
```



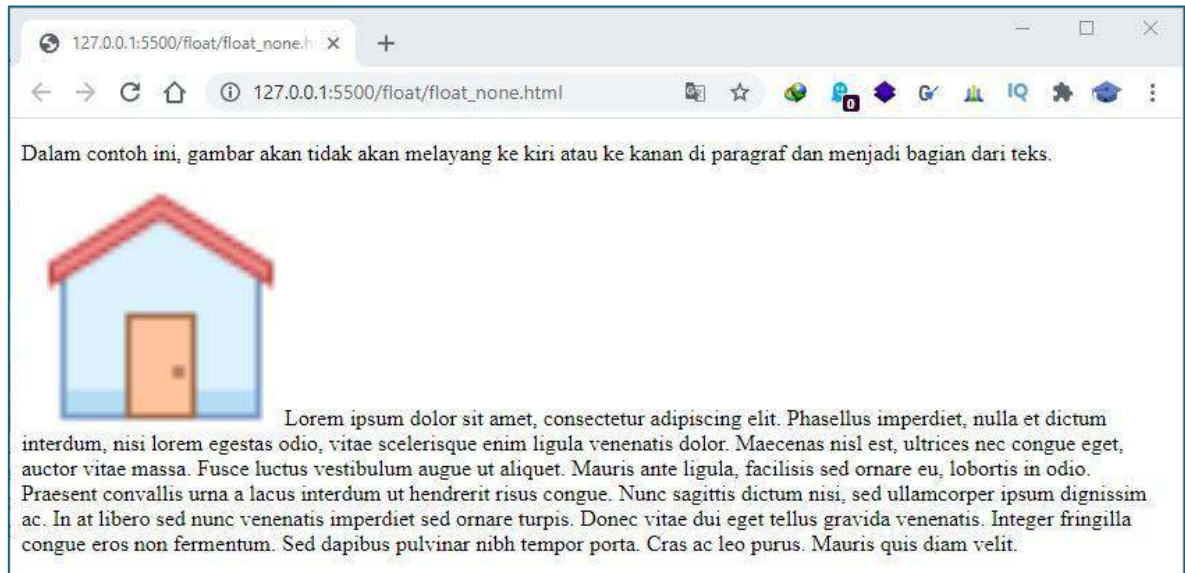
Float left


```
<style>
img {
  float: left;
}
</style>
```



Float : none

```
<style>
img {
  float: none;
}
</style>
```

Properti Clear

Properti menentukan elemen apa yang bisa mengapung di samping elemen yang clear kan dan di sisi mana. Properti dapat memiliki salah satu nilai berikut:

- none - Memungkinkan elemen mengambang di kedua sisi. Ini default
- kiri - Elemen mengambang tidak diperbolehkan di sisi kiri
- kanan- Tidak ada elemen mengambang yang diizinkan di sisi kanan
- keduanya - Tidak ada elemen mengambang yang diizinkan di sisi kiri atau kanan
- inherit - Elemen mewarisi nilai yang jelas dari induknya

Cara paling umum untuk menggunakan properti clear adalah setelah Anda menggunakan properti float pada sebuah elemen.

Saat membersihkan float, Anda harus mencocokkan yang jelas dengan floating-nya: Jika elemen di float kan ke kiri, maka Anda harus jelas ke kiri. Elemen float Anda akan terus mengambang, tetapi elemen yang telah clear-kan akan muncul di bawahnya pada halaman web.

Contoh berikut membersihkan float ke kiri. Berarti tidak ada elemen mengambang yang diizinkan di sisi kiri (div):

```

<style>
.div1 {
  float: left;
  width: 100px;
  height: 50px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.div2 {
  border: 1px solid red;
}

.div3 {
  float: left;
  width: 100px;
  height: 50px;
  margin: 10px;
  border: 3px solid #73AD21;
}

.div4 {
  border: 1px solid red;
  clear: left;
}
</style>

```

Html body :

```

<body>

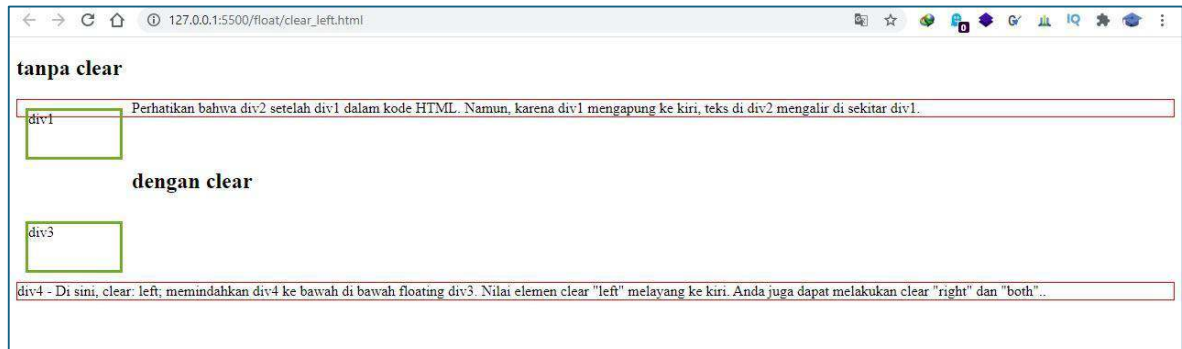
<h2>tanpa clear</h2>
<div class="div1">div1</div>
<div class="div2">Perhatikan bahwa div2 setelah div1 dalam kode HTML. Namun, karena div1 mengapung ke kiri, teks di div2 mengalir di sekitar div1.</div>
<br><br>

<h2>dengan clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Di sini, clear: left; memindahkan div4 ke bawah di bawah floating div3. Nilai elemen clear "left" melayang ke kiri. Anda juga dapat melakukan clear "right" dan "both".</div>

</body>

```

Hasil:



Jika sebuah elemen lebih tinggi dari elemen yang memuatnya, dan itu floatingkan, itu akan "overflow" di luar penampungnya:

Kemudian kita bisa menambahkan **overflow: auto;** ke elemen yang mengandung untuk memperbaiki masalah ini:

Style css:

```
<style>
div {
  border: 3px solid #4CAF50;
  padding: 5px;
}

.img1 {
  float: right;
}

.clearfix {
  overflow: auto;
}

.img2 {
  float: right;
}
</style>
```

Body html

```
<body>

<p>Dalam contoh ini, gambar lebih tinggi dari elemen yang memuatnya, dan itu di-float-kan, sehingga overflow dari penampungnya:</p>

<div>
  
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...
</div>

<p style="clear:right">Tambahkan kelas clearfix dengan overflow: auto; ke elemen yang mengandung, untuk memperbaiki masalah ini:</p>

<div class="clearfix">
  
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...
</div>


</body>
```

127.0.0.1:5500/float/overflow_au

127.0.0.1:5500/float/ove...

Dalam contoh ini, gambar lebih tinggi dari elemen yang memuatnya, dan itu di-float-kan, sehingga overflow dari penampungnya:

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...



Tambahkan kelas clearfix dengan overflow: auto; ke elemen yang mengandung, untuk memperbaiki masalah ini:

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum...

