



BINAR
ACADEMY

HTML & CSS

Gambaran Umum

Materi tersebut menjelaskan tentang apa itu **HTML** (Hypertext Markup Language), dan apa keterkaitannya dengan **Browser**, pengertian tentang apa itu **bahasa markup**, dan juga seperti apa **struktur** dan **entitas HTML**.



Tujuan Pembelajaran

- a. Siswa dapat memahami apa itu **HTML** dan kaitannya dengan **browser**.
- b. Siswa dapat menjelaskan apa yang dimaksud dengan bahasa **markup**.
- c. Siswa dapat membuat sebuah dokumen dalam **HTML**.
- d. Siswa dapat menampilkan sebuah gambar di dalam suatu halaman.
- e. Siswa dapat memakai **hyperlink** di dalam sebuah dokumen HTML untuk menampilkan halaman lain.
- f. Siswa dapat melakukan *debugging* untuk dokumen HTML mereka.



PART I - HTML

Hypertext Markup Language

Apa itu HTML?

Hypertext Markup Language

HTML

adalah

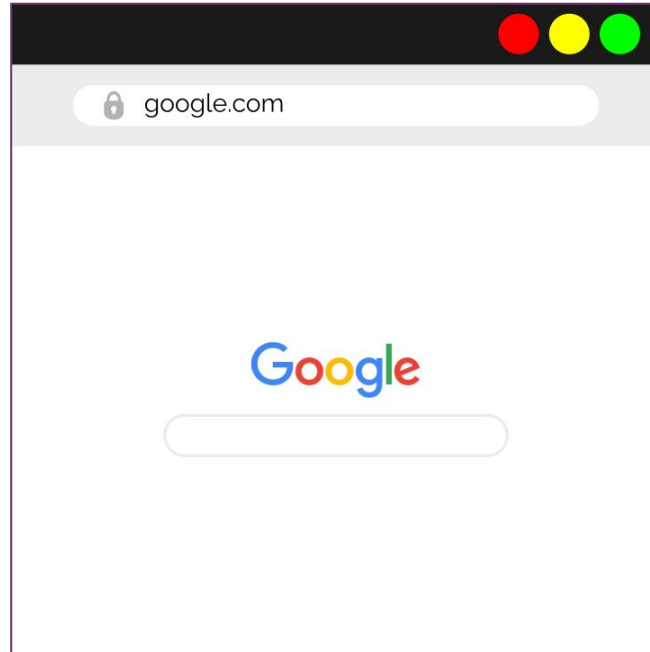
sebuah **bahasa**

markup

untuk membuat

Web Pages

Apa yang kamu lihat disini



dibuat
memakai
HTML

Bahasa Markup

adalah
sebuah format
dalam
menulis

dimana **tulisan** itu
akan **diproses**
menjadi **tampilan**

Hello, world! Bagaimana kabarmu?

10:30 AM

Hello, ketikin teks bold dong?

10:31 AM

Oke, **Bold**

10:31 AM



Oke, *Bold*

Secara Garis Besar

Karena **HTML** itu adalah bahasa *markup*, maka dia digunakan untuk memberi tahu browser untuk menyusun sebuah halaman web dan memberi tahu bagaimana browser akan menampilkannya, selayaknya kita memberi ***pada teks*** di **Whatsapp** tadi.

Kita bisa memberi tahu **browser** untuk menampilkan hal-hal lain, seperti gambar, teks yang dicetak miring, atau bahkan sebuah daftar.

Kalau di Whatsapp kita pakai * (bintang) atau simbol lain untuk menampilkan sesuatu, kalau di HTML, kita menggunakan yang namanya **Tag**.



Tapi,
Tag di **HTML** itu apa sih?



Jadi tag itu

Semacam kode yang dibungkus dengan karakter *Enclosing Tag* (<>) yang biasanya digunakan untuk membungkus suatu teks lain atau bahkan suatu konten.

Untuk menampilkan sesuatu di dalam browser yang seperti :

Hello world

Kita perlu menggunakan tag yang akan membungkus teks **Hello world** tersebut.

Jadi di dalam dokumen HTML akan terlihat seperti berikut :

<h1>Hello world</h1>

Terus gimana sih caranya
Bikin dokumen HTML?



Mari kita mulai dengan membuat

Halaman dengan teks Hello World

Halaman dengan teks Hello World

1. Buat file baru dengan ekstensi **.html**
2. Buka file tersebut dengan menggunakan **text editor** kalian.
3. Lalu **isi** file tersebut dengan **teks berikut** :

```
<h1>Hello World</h1>
```
4. Lalu **simpan** file tersebut, dan **buka** file tersebut dengan menggunakan **browser**





/home/sabrina/html-introduction/index.html

Hello World



Teks ini berada di dalam `<h1>`

Mari kita

Pecah satu-persatu

`<h1>` Ini tag

Hello World Ini konten

`<h1>Hello World</h1>` Dan ini elemen

h1

menampilkan *Header*. Biasanya untuk menyatakan judul.

h2

menampilkan *Header*, tapi tidak lebih penting dari **h1**.

p

menampilkan **paragraf**.

strong

menampilkan **teks** yang **dicetak tebal**.

i

menampilkan **teks** yang **dicetak miring**.

img

menampilkan sebuah **gambar**.

ul

menampilkan sebuah **daftar** yang **tidak teratur**.

li

menampilkan **elemen** dalam **sebuah daftar**.

Setiap

Tag

Mempunyai
Representasi
Yang
Berbeda-beda


```

```



/home/sabrina/html-introduction/index.html

scelerisque neque. Nunc ex turpis, dapibus a feugiat in, gravida ac tellus. Nunc imperdiet sed lacus id fermentum. Interdum et malesuada fames ac ante ipsum primis in faucibus.





Kita bisa menambahkan atribut di dalam suatu elemen

Seperti contoh elemen **img** tadi. Elemen tersebut membutuhkan atribut bernama **src** dimana itu akan merujuk ke suatu **URL** atau suatu **file** bertipe **gambar**.

Kutip satu atau kutip dua?

Ketika ketika melihat suatu elemen di dalam HTML, terkadang terdapat atribut yang nilainya dibungkus dengan **kutip satu**, ada pula yang dibungkus **kutip dua**.

Kedua elemen ini akan menghasilkan tampilan yang sama.

Namun, jika anda menggabungkan kedua kutip itu untuk membungkus nilai dari suatu atribut, browser tidak akan bisa memahaminya.

**Simpan</button>
```

Seperti yang kita lihat, ada atribut **disabled**, nah atribut disabled itu tidak memiliki nilai, bukan berarti dia tidak bernilai. Namun, **disabled** sebenarnya memiliki nilai

**Boolean (true atau false)**. Artinya bila **disabled** ditulis di dalam elemen itu, maka dia akan bernilai **true**, jika tidak maka akan dianggap **false**.

```
<button type="submit" disabled="true">Simpan</button>
```

Atau bisa juga ditulis seperti itu.



# Hyperlink

`<a href="https://binar.co.id">Cek website kita</a>`



/home/sabrina/html-introduction/index.html

- [Home](#)
- [About](#)
- [Contact](#)

# Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque lectus nisi, pellentesque volutpat vestibulum feugiat, blandit vel risus. Phasellus eleifend sollicitudin turpis at posuere. Phasellus ut mauris est. Curabitur tortor est, commodo at ante sit amet, pharetra hendrerit sem. In commodo porttitor lacus non cursus. Praesent bibendum porta pretium. Sed dignissim bibendum odio ac accumsan. Sed venenatis tincidunt enim, id fringilla neque faucibus in. Suspendisse cursus suscipit ante quis dapibus. Ut convallis lorem aliquet, accumsan mauris in, scelerisque neque. Nunc ex turpis, dapibus a feugiat in, gravida ac tellus. Nunc imperdiet sed lacus id fermentum. Interdum et malesuada fames ac ante ipsum primis in faucibus.

## Here's my achievement:

- The fastest writer in the world
- The most beloved person in the world
- The cutest person in the world

[Google](#)

<!DOCTYPE html>

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>Hello World</title>

</head>

<body>

<ul>

<li>

<a href="index.html">Home</a>

</li>

<li>

<a href="about.html">About</a>

</li>

<li>

<a href="contact.html">Contact</a>

</li>

</ul>

<h1>Hello World</h1>

<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam at purus massa. Duis dignissim ligula vitae lacus sagittis, non sollicitudin urna mattis. Morbi malesuada a elit non porta. Suspendisse posuere massa eget cursus blandit. Phasellus eu est scelerisque, condimentum leo eu, suscipit nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla iaculis est eget ipsum finibus scelerisque. Ut dapibus hendrerit ipsum, id facilisis odio dapibus quis. Ut eget lacinia diam. Integer libero odio, malesuada in felis vel, eleifend lacinia eros. Vivamus at eros faucibus, placerat est id, vulputate libero. Fusce vitae leo ac mauris rhoncus dictum auctor id augue. Mauris ante arcu, ultrices pharetra erat quis, tempus laoreet tortor.</p>

<h2>Here's my achievement</h2>

<ul>

<li>The fastest writer in the world</li>

<li>The most beloved person in the world</li>

<li>The cutest person in the world</li>

</ul>

<a href="www.google.com">Google</a>

</body>

</html>

Ini yang kita sebut elemen bersarang

# Struktur Dokumen HTML

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <meta charset="utf-8">
 <title></title>
 </head>
 <body>
 </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <meta charset="utf-8">
 <title></title>
 </head>
 <body>
 </body>
</html>
```

# Anatomi HTML

## DOCTYPE

Dalam masa lalu, ketika HTML masih muda (sekitar 1991/2), DOCTYPE dimaksudkan untuk bertindak sebagai link ke serangkaian aturan yang harus diikuti oleh halaman HTML untuk dianggap sebagai HTML yang baik, yang dapat berarti pengecekan kesalahan otomatis dan hal lainnya yang bermanfaat. Mereka dulu terlihat seperti ini:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Namun, akhir-akhir ini tidak ada yang benar-benar peduli tentang hal tersebut, dan mereka benar-benar hanya artefak sejarah yang perlu dimasukkan agar semuanya berjalan dengan baik. **<!DOCTYPE html>** adalah string karakter terpendek yang dianggap sebagai doctype yang valid. Hanya itu yang perlu kita ketahui.



```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <meta charset="utf-8">
 <title></title>
 </head>
 <body>
 </body>
</html>
```

# Anatomi HTML

## HTML

Elemen **<html>**. Elemen ini membungkus semua konten di seluruh halaman, dan kadang-kadang dikenal sebagai elemen root.

## HEAD

Elemen **<head>**. Element ini bertindak sebagai wadah untuk semua hal yang ingin kita sertakan pada halaman HTML yang bukan konten yang akan kita tampilkan kepada user halaman kita. Ini termasuk hal-hal seperti keyword dan deskripsi halaman yang ingin kita tampilkan di hasil pencarian, CSS akan menata konten kita, deklarasi set karakter, dan banyak lagi. Kita akan belajar lebih banyak tentang ini di materi selanjutnya dalam seri ini.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <meta charset="utf-8">
 <title></title>
 </head>
 <body>
 </body>
</html>
```

# Anatomi HTML

## METADATA

Elemen **<meta charset="utf-8">**. Elemen ini mengatur set karakter yang harus digunakan dokumen kita ke UTF-8, yang mencakup sebagian besar karakter dari sebagian besar bahasa tulisan manusia. Pada dasarnya itu sekarang dapat menangani konten teks yang mungkin kita masukkan ke dalamnya. Tidak ada alasan untuk tidak mengatur ini, dan ini dapat membantu menghindari beberapa masalah nanti.

## TITLE

Elemen **<title>**. Ini mengatur judul halaman kita, yang merupakan judul yang muncul di tab browser tempat halaman dimuat, dan digunakan untuk menjelaskan halaman saat kita membookmark/memfavoritkannya.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
 <head>
 <meta charset="utf-8">
 <title></title>
 </head>
 <body>
 </body>
</html>
```

# Anatomi HTML

## METADATA

Elemen **<meta charset="utf-8">**. Elemen ini mengatur set karakter yang harus digunakan dokumen kita ke UTF-8, yang mencakup sebagian besar karakter dari sebagian besar bahasa tulisan manusia. Pada dasarnya itu sekarang dapat menangani konten teks yang mungkin kita masukkan ke dalamnya. Tidak ada alasan untuk tidak mengatur ini, dan ini dapat membantu menghindari beberapa masalah nanti.

## BODY

Elemen **<body>**. Ini berisi semua konten yang ingin kita tampilkan kepada user ketika mereka mengunjungi halaman kita, apakah itu teks, gambar, video, game, trek audio yang dapat diputar, atau apa pun lainnya.

# Referensi Entitas: Termasuk Karakter Khusus dalam HTML

Dalam HTML, karakter `<`, `>`, `"`, `'`, dan `&` adalah karakter khusus. Mereka adalah bagian dari syntax **HTML** itu sendiri, jadi bagaimana kita memasukkan salah satu karakter ini ke dalam teks kita, misalnya jika kita benar-benar ingin menggunakan **ampersand** atau tanda kurang dari, dan tidak menafsirkannya sebagai kode seperti yang dilakukan beberapa browser?

Kita harus menggunakan karakter referensi - kode khusus yang mewakili karakter, dan dapat digunakan dalam keadaan yang tepat. Setiap referensi karakter dimulai dengan **ampersand** `&`, dan diakhiri dengan **tanda titik koma** `;`.

Literal Character	Character Reference Equivalent
<code>&lt;</code>	<code>&amp;lt;</code>
<code>&gt;</code>	<code>&amp;gt;</code>
<code>"</code>	<code>&amp;quot;</code>
<code>'</code>	<code>&amp;apos;</code>
<code>&amp;</code>	<code>&amp;amp;</code>

# Referensi Entitas: Termasuk Karakter Khusus dalam HTML

Dalam contoh di bawah ini, kita dapat melihat dua paragraf, yang membicarakan tentang teknologi web. Kita menginginkan hasil seperti berikut:

**" Dalam HTML, kita mendefinisikan paragraf menggunakan element <p>. "**

Berikut penulisan yang benar dan salah:

**<p>Dalam HTML, kita mendefinisikan paragraf menggunakan element <p>.</p>**

**<p>Dalam HTML, kita mendefinisikan paragraf menggunakan element &lt;p&gt;.</p>**

Maka hasilnya akan seperti dibawah ini:

**Dalam HTML, kita mendefinisikan paragraf menggunakan element.**

**Dalam HTML, kita mendefinisikan paragraf menggunakan element <p>.**

Jika kita melihat hasilnya, kita akan melihat bahwa paragraf pertama salah dan tidak sesuai yang kita inginkan, karena browser berpikir bahwa instance kedua dari <p> memulai paragraf baru. Paragraf kedua terlihat bagus, karena kita telah mengganti kurung sudut dengan karakter referensi.

# Komentar dalam HTML

Apa sih yang dimaksud dengan komentar?

## Jadi komentar itu

Suatu **teks**, atau **kode** yang ditulis dalam suatu file, katakanlah **HTML**, dimana teks tersebut akan **dihiraukan** ketika dijalankan.

```
<h1>Hello World</h1>
```

```
<!-- Hiraukan saja diriku -->
```

```
<p>Kalau salah jangan cuma pake satu huruf</p>
```

```
<!-- <p>Paragraf ini ga bakal ditampilkan kok!</p> -->
```



/home/sabrina/html-introduction/index.html

- [Home](#)
- [About](#)
- [Contact](#)

# Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque lectus nisi, pellentesque volutpat vestibulum feugiat, blandit vel risus. Phasellus eleifend sollicitudin turpis at posuere. Phasellus ut mauris est. Curabitur tortor est, commodo at ante sit amet, pharetra hendrerit sem. In commodo porttitor lacus non cursus. Praesent bibendum porta pretium. Sed dignissim bibendum odio ac accumsan. Sed venenatis tincidunt enim, id fringilla neque faucibus in. feugiat in,

Here's m

- The
- The
- The cutest person in the world

Google



Elements

<html>

Di dalam tab ini, akan terlihat  
HTML yang tertampil di dalam  
tampilan browser




# Debugging HTML

- [Home](#)
- [About](#)
- [Contact](#)



## Hello World

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam at purus massa. Duis dignissim ligula vitae lacus sagittis, non sollicitudin urna mattis. Morbi malesuada a elit non porta. Suspendisse posuere massa eget cursus blandit. Phasellus eu est scelerisque, condimentum leo eu, suscipit nulla. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla iaculis est eget ipsum finibus scelerisque. Ut dapibus hendrerit ipsum, id facilisis odio dapibus quis. Ut eget lacinia diam. Integer libero odio, malesuada in felis vel, eleifend lacinia eros. Vivamus at eros faucibus, placerat est id, vulputate libero. Fusce vitae leo ac mauris rhoncus dictum auctor id augue. Mauris ante arcu, ultrices pharetra erat quis, tempus laoreet tortor.



Elements

Console

Sources

Network

Performance

Memory

Application

Security

Audits

Redux

```
<!doctype html>
<html>
 <head>...</head>
 <body> == $0
 ...

 <h1>Hello World</h1>
 <p>...</p>
 Google
 </body>
</html>
```

Styles

Computed

Event Listeners

»

Filter

:hov .cls +

element.style {

}

body {

display: block;

margin: 8px;

}

Inherited from html

html {

color: -internal-root-color;

}

Di dalam tab ini, akan terlihat  
HTML yang tertampil di dalam  
tampilan browser

# PART II - CSS

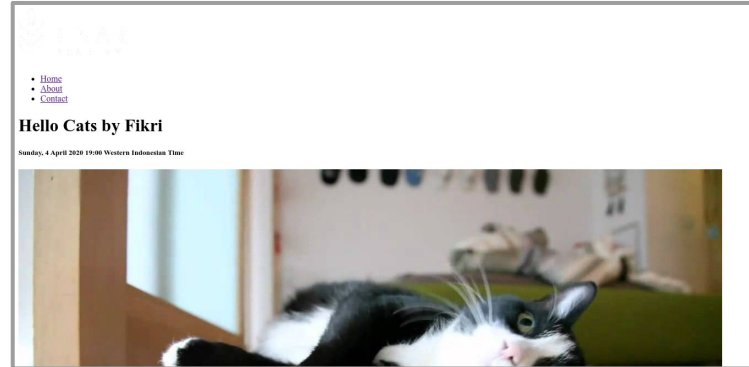
---

## Cascading Stylesheet

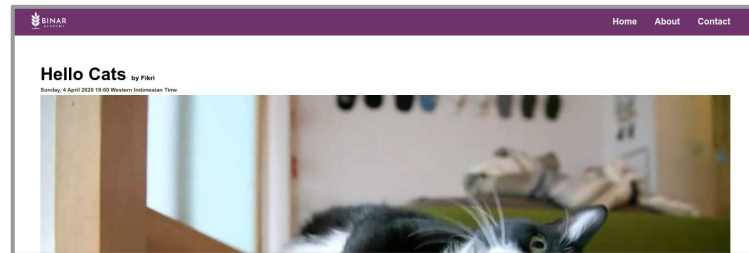
# CSS

adalah  
bahasa  
untuk  
menentukan  
bagaimana  
Dokumen  
itu  
disajikan

Dari ini



Menjadi ini



## Apa itu CSS?

Seperti yang telah kita sebutkan sebelumnya, CSS adalah bahasa untuk menentukan bagaimana dokumen disajikan kepada user. Bagaimana mereka disajikan, ditata, dll.

Sebuah **dokumen** biasanya merupakan file teks yang disusun menggunakan *bahasa markup*. HTML adalah bahasa markup yang paling umum, tetapi kita juga akan menemukan bahasa markup lainnya seperti SVG atau XML.

**Menyajikan** dokumen kepada user berarti mengubahnya menjadi bentuk yang dapat digunakan untuk pembaca kita. Browser, seperti Firefox, Chrome atau Internet Explorer, dirancang untuk menyajikan dokumen secara visual, misalnya, pada layar komputer, proyektor, atau printer.

# Bagaimana CSS mempengaruhi HTML?

Web browser menerapkan **aturan CSS** ke dokumen untuk memengaruhi cara mereka ditampilkan. Aturan CSS dibentuk dari:

- Seperangkat properti, yang memiliki nilai yang diatur untuk memperbarui cara konten HTML ditampilkan. Misalnya, kita ingin lebar element menjadi 50% dari element induknya, dan latar belakangnya menjadi merah.
- Selector, yang memilih element yang ingin kita terapkan nilai properti yang baru saja diperbarui. Misalnya, kita ingin menerapkan aturan CSS kita ke semua paragraf di dokumen HTML kita.

Serangkaian aturan CSS yang terkandung dalam **stylesheet** menentukan bagaimana tampilan halaman web nantinya. Kita akan belajar lebih banyak tentang seperti apa Syntax CSS di materi modul berikutnya.



## Contoh CSS Sederhana

Deskripsi di atas mungkin atau mungkin tidak masuk akal, jadi mari kita pastikan semuanya jelas dengan menghadirkan contoh sederhana. Pertama-tama, mari kita ambil dokumen **HTML** sederhana, yang berisi `<h1>` dan `<p>` (perhatikan bahwa stylesheet diterapkan ke **HTML** menggunakan element `<link>`)

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>My CSS experiment</title>
6 <link rel="stylesheet" href="style.css">
7 </head>
8 <body>
9 <h1>Hello World!</h1>
10 <p>This is my first CSS example</p>
11 </body>
12 </html>
```

index.html

```
1 h1 {
2 color: blue;
3 background-color: yellow;
4 border: 1px solid black;
5 }
6
7 p {
8 color: red;
9 }
```

style.css



# Kita bahas satu per satu

Di dalam file **CSS** tadi, kita menerapkan **dua aturan** untuk **elemen-elemen** yang ada di file HTML kita.

Aturan **pertama** dimulai dengan selector **h1**, yang berarti akan menerapkan nilai propertinya ke element **<h1>**. Ini berisi tiga properti dan nilainya (setiap properti/nilai disebut dengan **deklarasi**)

1. Yang pertama **berfungsi** untuk mengatur **warna teks** menjadi **biru**.
2. Yang kedua **berfungsi** untuk mengatur **warna latar belakang** menjadi **kuning**.
3. Dan yang ketiga **berfungsi** untuk **menempatkan border** di **sekitar header** yang **memiliki lebar 1 pixel, solid** (tidak putus-putus, atau putus-putus, dll), dan berwarna hitam.

```
1 h1 {
2 color: blue;
3 background-color: yellow;
4 border: 1px solid black;
5 }
6
7 p {
8 color: red;
9 }
```

style.css



## Kita bahas satu per satu

Aturan **kedua** dimulai dengan **selector p**, yang artinya akan menerapkan **nilai propertinya** ke element **<p>**. Ini berisi satu deklarasi, yang mengatur **warna teks** menjadi **merah**.

Ini tidak terlalu cantik, tapi setidaknya itu mulai memberi kita gambaran tentang cara kerja CSS

```
1 h1 {
2 color: blue;
3 background-color: yellow;
4 border: 1px solid black;
5 }
6
7 p {
8 color: red;
9 }
```

style.css

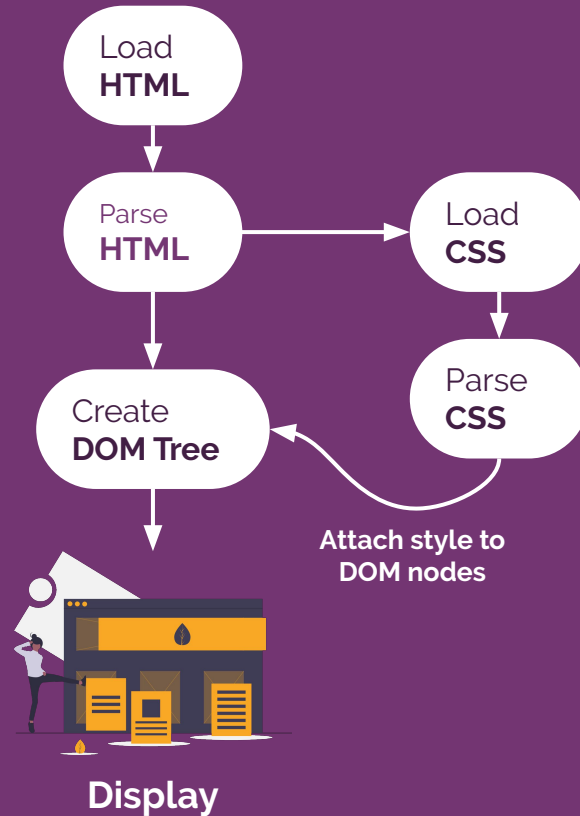
Terus gimana sih cara  
**CSS bekerja?**



## Jadi seperti ini cara CSS bekerja

Ketika **browser menampilkan dokumen**, itu harus menggabungkan konten **dokumen** dengan **informasi stylenya**. Ini memproses dokumen dalam dua tahap:

1. **Browser mengubah HTML dan CSS menjadi DOM** (*Document Object Model*). DOM mewakili **dokumen dalam memori komputer**. Ini menggabungkan konten dokumen dengan stylenya.
2. Browser menampilkan konten DOM.



## Tentang DOM

**DOM** memiliki **struktur** seperti **pohon**. Setiap **element**, **attribute**, dan **potongan teks** dalam bahasa markup **menjadi DOM node** dalam **struktur pohon**. **Node** ditentukan oleh **hubungannya** dengan **DOM node lainnya**. Beberapa **elemen** merupakan **parent** dari **node anak**, dan **node anak** memiliki **saudara kandung**. Memahami **DOM** akan **membantu kita mendesain, men-debug, dan memelihara CSS** kita karena DOM adalah **tempat CSS dan konten dokumen bertemu**.



## Representasi DOM

Daripada **penjelasan yang panjang** dan **membosankan**, mari kita ambil contoh untuk melihat **bagaimana DOM dan CSS bekerja bersama**..

Mari kita asumsikan kode HTML berikut

```
<p>
 Let's use:
 Cascading
 Style
 Sheets
</p>
```

Di DOM, node yang terkait dengan element **<p>** kita adalah parent/induk. Anak-anaknya adalah node teks dan node yang sesuai dengan element **<span>** kita. Node SPAN juga termasuk parent, dengan node teks sebagai anak-anak mereka

```
P
├─ "Let's use:"
├─ SPAN
│ └─ "Cascading"
├─ SPAN
│ └─ "Style"
└─ SPAN
 └─ "Sheets"
```

## Menerapkan CSS ke DOM

Katakanlah kita **menambahkan** beberapa **CSS** ke **dokumen** kita, untuk mendesain-nya. Sekali lagi, HTML yang kita gunakan adalah sebagai berikut

```
<p>
 Let's use:
 Cascading
 Style
 Sheets
</p>
```

Jika kita menerapkan CSS berikut untuk itu

```
span {
 border: 1px solid black;
 background-color: red;
}
```

Browser akan **mem-parsing** HTML dan **membuat DOM** **darinya**, lalu **mem-parsing CSS**. Karena **satu-satunya aturan** yang tersedia di **CSS** memiliki **span selector**, itu akan menerapkan aturan itu untuk masing-masing dari tiga span.

## Cara Menerapkan CSS ke HTML

Ada tiga cara berbeda untuk menerapkan CSS ke dokumen HTML yang biasa kita temui, beberapa lebih bermanfaat daripada yang lain. Di sini kita akan meninjau secara singkat masing-masingnya.



# Stylesheet Eksternal

Kita telah melihat **stylesheet eksternal** di materi ini, tetapi tidak dengan nama itu. **Stylesheet eksternal** adalah ketika kita membuat **CSS** lalu **ditulis dalam file terpisah** dengan ekstensi **.css**, dan kita mereferensikannya dari element HTML **<link>**

File HTML dan CSSnya akan **terlihat seperti dibawah**

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8">
 <title>My CSS experiment</title>
 <link rel="stylesheet" href="style.css">
 </head>
 <body>
 <h1>Hello World!</h1>
 <p>This is my first CSS example</p>
 </body>
</html>
```

index.html

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}

p {
 color: red;
}
```

style.css

Metode ini bisa dibilang yang terbaik, karena kita dapat menggunakan satu stylesheet untuk menata beberapa dokumen, dan hanya perlu memperbarui CSS di satu tempat jika diperlukan perubahan.



## Stylesheet Internal

**Stylesheet internal** adalah ketika kita tidak memiliki file CSS eksternal, tetapi CSS ditempatkan di dalam element `<style>`, yang terdapat di dalam kepala HTML.

Jadi HTML akan terlihat seperti kode disamping →

Ini dapat berguna dalam beberapa keadaan (mungkin kita bekerja dengan sistem manajemen konten di mana kita tidak dapat memodifikasi file CSS secara langsung), tetapi itu tidak seefisien stylesheet eksternal. Di situs web, CSS akan dibutuhkan untuk digunakan di beberapa halaman, dan diperbarui di banyak tempat jika diperlukan perubahan.

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8">
 <title>My CSS experiment</title>
 <style>
 h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
 }

 p {
 color: red;
 }
 </style>
 </head>
 <body>
 <h1>Hello World!</h1>
 <p>This is my first CSS example</p>
 </body>
</html>
```

## Inline Style

**Inline styles** adalah deklarasi CSS yang hanya memengaruhi satu element, yang terkandung dalam attribute style

Jadi HTML akan terlihat seperti kode disamping →

**Sangat dianjurkan**, untuk tidak menggunakan cara ini, karena tidak praktis. Dan apabila kita ingin melakukan perubahan maka akan sangat merepotkan.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8">
 <title>My CSS experiment</title>
</head>

<body>
 <h1 style="color: blue;background-color: yellow;border: 1px solid black;">
 Hello World!
 </h1>
 <p style="color:red;">This is my first CSS example</p>
</body>

</html>
```

## Syntax CSS

Selanjutnya, kita akan menyelam ke dalam syntax CSS untuk mempelajari lebih banyak lagi, melihat bagaimana *properties* dan *values* mereka membentuk ke dalam *deklarasi*, beberapa deklarasi membentuk *blok deklarasi*, dan blok deklarasi dan selector melengkapi *aturan CSS*. Pada materi kali ini kita akan membahas fitur syntax CSS lainnya seperti komentar dan whitespace.



## Sentuhan Kosa Kata

Pada tingkat paling dasar, CSS terdiri dari dua blok bangunan:

- **Properties:** Pengidentifikasi yang dapat dibaca manusia yang menunjukkan fitur stylistic (misalnya. Font, lebar, warna background) yang ingin kita ubah.
- **Values:** Setiap properties yang ditentukan diberikan value, yang menunjukkan bagaimana kita ingin mengubah fitur-fitur stylistic (misalnya. Apa yang kita ingin ubah pada font, lebar atau warna background)

Properties yang dipasangkan dengan value disebut deklarasi CSS. Deklarasi CSS dimasukkan ke dalam *Blok Deklarasi CSS*. Dan akhirnya, blok deklarasi CSS dipasangkan dengan *selector* untuk menghasilkan *CSS Rulesets* (atau *Aturan CSS*)



Sebelum terlalu mendalam dalam teori dan penjelasan tertulis, mari kita lihat contoh konkret (kita melihat sesuatu yang sangat mirip dalam materi kita sebelumnya)

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8">
 <title>My CSS experiment</title>
 <link rel="stylesheet" href="style.css">
 </head>
 <body>
 <h1>Hello World!</h1>
 <p>This is my first CSS example</p>

 This is
 a list

 </body>
</html>
```

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}

p {
 color: red;
}

p, li {
 text-decoration: underline;
}
```

Ada **lebih dari 300 properti** berbeda di **CSS** dan hampir tak terhingga nilainya. Tidak semua pasangan properti dan nilai diizinkan. Setiap properti memiliki daftar spesifik dari value valid yang ditentukan untuknya.

**Penting:** Jika properties tidak diketahui atau jika value tidak valid untuk properties yang diberikan, deklarasi akan dianggap *tidak valid* dan sepenuhnya diabaikan oleh browser engine CSS. Di CSS (dan standar web lainnya), ejaan negara US telah disetujui sebagai standar untuk tetap mengikutinya. Misalnya, color harus selalu dieja color. Jika kita menggunakan colour, maka itu tidak akan berfungsi.

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}

p {
 color: red;
}

p, li {
 text-decoration: underline;
}
```

# Blok Deklarasi CSS

Deklarasi dikelompokkan dalam **blok**, dengan setiap set deklarasi dibungkus oleh kurung kurawal pembuka, ({) dan penutup (}). Setiap deklarasi yang terkandung di dalam **blok deklarasi** harus dipisahkan oleh tanda titik koma (;). Jika tidak, kode tidak akan berfungsi (atau setidaknya akan memberikan hasil yang tidak diharapkan.) Deklarasi terakhir dari sebuah blok tidak perlu diakhiri dengan titik koma, meskipun sering dianggap *style yang baik* untuk melakukannya karena mencegah ketika kita lupa untuk menambahkannya, ketika memperluas blok dengan deklarasi lain.

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black
}
```

Titik koma berfungsi untuk  
memisahkan dua deklarasi

Penambahan titik koma disini  
bersifat opsional, tetap lebih  
baik kita tambahkan

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}

p {
 color: red;
}

p, li {
 text-decoration: underline;
}
```

## Catatan:

Blok terkadang bisa disarangkan, dalam kasus seperti itu buka dan tutup kurung harus bersarang secara logis, dengan cara yang sama seperti tag element HTML bersarang. Contoh paling umum yang akan kita temui adalah *@-rules*, yang merupakan blok yang dimulai dengan @identifikasi seperti @media, @font-face, dll. Blok deklarasi mungkin memiliki value kosong, hal tersebut benar-benar valid.

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}
```

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

```
p, li {
 text-decoration: underline;
}
```

Ini adalah  
**Selector**

```
h1 {
 color: blue;
 background-color: yellow;
 border: 1px solid black;
}
```

Selector bisa berupa banyak kata.

Sebagai contoh:

```
div ul li {
 color: red;
}
```

Selector bisa berupa sebuah **grup** yang ditandai dengan **koma**.

Sebagai contoh:

```
div ul li, p span {
 color: red;
}
```

Selector bisa berupa **penujuk suatu kejadian** ditandai dengan **titik dua**.

Sebagai contoh:

```
button:hover {
 background: blue;
 border-color: white;
 color: white;
}
```

**Selector**  
Dan  
**Aturan**  
**CSS**



## CSS Statements

Ada beberapa jenis *CSS Statement*, yaitu:

- **At-Rules** digunakan dalam CSS untuk menyampaikan metadata, informasi kondisional, atau informasi deskriptif lainnya. Mereka dimulai dengan tanda at (@), diikuti oleh pengenalan untuk mengatakan seperti apa aturannya, lalu blok syntax semacam itu, berakhir dengan semi-colon atau titik koma (;). Setiap jenis at-rule, yang ditentukan oleh identifier, akan memiliki syntax dan semantik internalnya sendiri. Contohnya termasuk:
  - @charset dan @import (metadata)
  - @media atau @document (informasi bersyarat, juga disebut statement bersarang, lihat di bawah.)
  - @font-face (informasi deskriptif)
- **Nested Statement** adalah subset spesifik *at-rule*, syntax yang merupakan blok aturan CSS yang bersarang yang hanya akan diterapkan pada dokumen jika kondisi tertentu cocok:
  - Konten *at-rule* @**media** hanya diterapkan jika perangkat yang menjalankan browser cocok dengan kondisi yang diungkapkan.
  - Konten *at-rule* @**support** hanya diterapkan jika browser benar-benar mendukung fitur yang diuji.
  - Konten *at-rule* @**document** hanya diterapkan jika halaman saat ini cocok dengan beberapa ketentuan.

```
@import 'custom.css';
```

Ini contoh untuk ***at-rules syntax***

Berguna untuk mengimpor file CSS lain ke dalam CSS saat ini

```
@media (min-width: 801px) {
 body {
 margin: 0 auto;
 width: 800px;
 }
}
```

Ini contoh untuk  
***Nested Statement***

Statements bersarang di atas hanya menerapkan aturan bersarang saat lebar halaman melebihi 800 pixel.

Gimana caranya buat syntax CSS  
**Yang mudah dibaca?**



**Sangat penting kita menulis syntax CSS dalam beberapa aturan yang kita sepakati. Agar kodenya terpelihara dan mudah dibaca.**

**Slide berikutnya, akan menjelaskan hal-hal apa saja yang bisa kita jadikan acuan dalam menulis syntax CSS.**

# Whitespace

Whitespace meliputi **spasi**, **tab** dan **enter**. Kita dapat menambahkan *whitespace* untuk membuat *stylesheet* kita agar lebih mudah dibaca.

```
@import 'custom.css';
```

Menambahkan Tab di dalam Deklarasi Blok.

```
h1 {
 color: red;
 text-decoration: underline;
}
```

Menambahkan Enter (Newline) di dalam Deklarasi Blok.

Menambahkan Tab di dalam Nested Statement.

```
@media (min-width: 801px) {
 body {
 margin: 0 auto;
 width: 800px;
 }
}
```

Menambahkan Enter (Newline) di dalam Nested Statement.

*Coba bandingkan jika kamu menghapus semua Whitespace!*

*Lebih enak dilihat yang mana, meskipun sama-sama jalan.*



## Tapi Ingat

*Whitespace* hanya berlaku pada separator saja..

```
h1 {
 margin: 0 auto;
 padding-left: 10px;
}
```

Kode diatas akan berjalan dengan baik.

```
h1 {
 margin: 0auto;
 padding- left: 10px;
}
```

Namun tidak dengan ini.

**Pro Tips:** Ada yang namanya *syntax highlighting* di dalam *text editor*, ini sangat berfungsi untuk mendeteksi apakah kode kita akan jalan atau tidak.

## Komentar

Nah, komentar di CSS sama persis dengan komentar di HTML.

```
/* Mengatur tajuk h1 */
h1 {
 margin: 0 auto;
 padding-left: 10px;
}
```

```
/* Mengatur semua konten untuk berada di tengah */
div {
 margin: 0 auto;
}
```



## Shorthand

Katakanlah, kita punya **padding-left**, **padding-right**, **padding-top**, **padding-bottom**. Betapa merepotkannya kita harus memberi nilai kepada **properti-properti** itu **satu per satu**. Dan ternyata ada **shortcutnya** lho untuk menulis itu.

```
/* Daripada nulis gini */
.content {
 padding-top: 4px;
 padding-right: 6px;
 padding-bottom: 6px;
 padding-left: 4px;
}
```

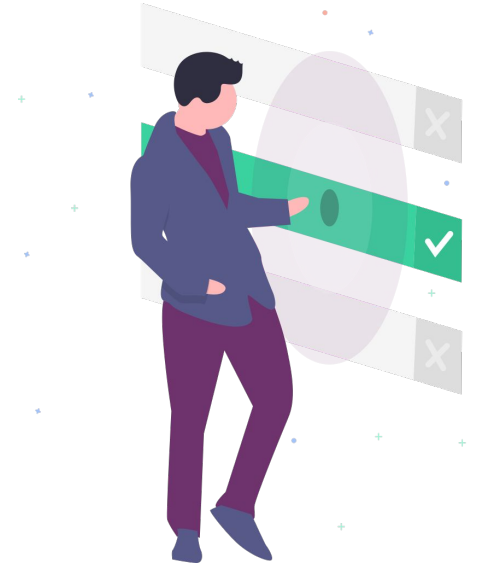
```
/* Mending kayak gini */
.content {
 padding: 4px 6px 6px 4px;
 /* Top: 4px Right: 6px Bottom: 6px Left: 4px */
}
```





## CSS Selector

Dalam CSS, selector digunakan untuk menargetkan element HTML pada halaman website yang ingin kita berikan sentuhan style. Ada berbagai macam selector CSS yang tersedia, memungkinkan untuk melakukan presisi yang halus ketika memilih element sesuai style. Dalam beberapa materi berikutnya kita akan membahas berbagai jenis selector dengan sangat terperinci, dan melihat cara kerjanya.



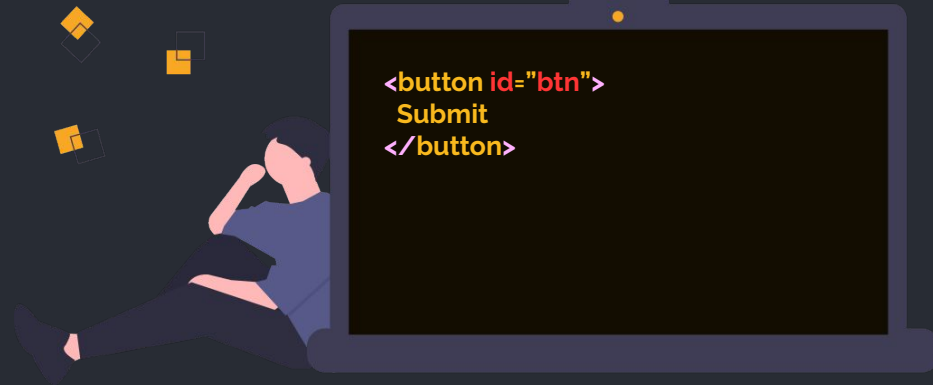
# Simple Selector

Memilih dengan cara memanggil **elemen**, **id** atau **class**..

```
/* Kita milih body
 * untuk dikasih suatu
 * aturan */
body { /* Biasa kita sebut dengan simple selector */
 margin: 0;
 padding: 0;
 width: 100%;
}
```

```
/* Kita element yang
 * memiliki ID btn
 * untuk dikasih suatu
 * aturan */
#btn {

}
```



## Attribute Selector

Memilih dengan cara memanggil **attribut** dari **suatu element**..

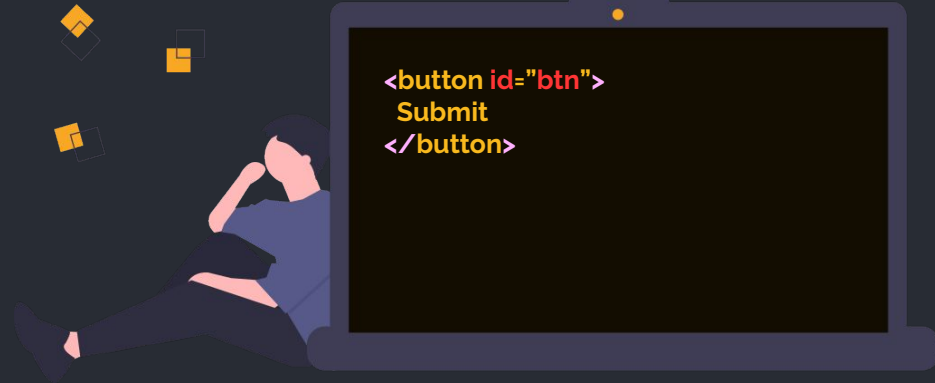
```
/* Kita memilih
 * element input
 * yang memiliki
 * type text */
input[type="text"] {
 color: white;
 outline: none;
}
```



# Pseudo Class

Mencocokkan satu atau lebih element yang ada **dalam keadaan tertentu**.

```
/*
 * Aturan ini akan
 * berjalan jika pointer
 * mouse berada di
 * dalam box ini
 * */
#btn:hover {
 color: black;
 background: red;
}
```



## Pseudo Element

Mencocokkan satu atau lebih bagian konten yang berada dalam posisi tertentu sehubungan dengan suatu element, misalnya kata pertama dari setiap paragraf, atau konten yang dihasilkan muncul sebelum element

```
/* Dengan aturan
* seperti ini
* setiap elemen yang
* termasuk kedalam kelas
* paragraph, akan mendapat
* tambahan konten Hello,
* di awal konten elemen tersebut
* */
.paragraph::before {
 content: "Hello"
}

/* Begitu pula sebaliknya */
.paragraph::after {
 content: "Bye"
}
```

## Dan banyak lagi jenis-jenis selector

Seperti atribut pada CSS, ada banyak jenis-jenis selector, dan sangat tidak mungkin kita bahas satu per satu disini. Maka, kunci adalah eksplorasi. Kalian bisa lihat link ini sebagai referensi tentang CSS Selector.

[https://www.w3schools.com/cssref/css\\_selectors.asp](https://www.w3schools.com/cssref/css_selectors.asp)

[https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_Selectors](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors)

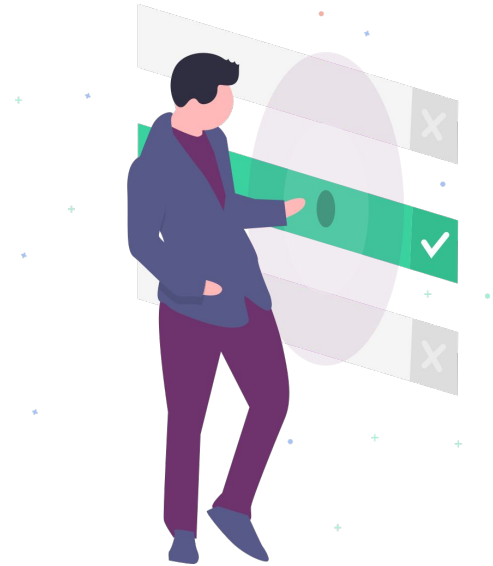
## Yang perlu kamu ketahui.

- **CSS Values and Unit**

[https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction\\_to\\_CSS/Values\\_and\\_units](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Values_and_units)

- **Box model**

[https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction\\_to\\_CSS/Box\\_model](https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Box_model)



## Debugging CSS

Pada materi terakhir ini, kita akan melihat prinsip-prinsip debugging CSS, termasuk menjelajahi CSS yang diterapkan ke halaman, dan tools lain yang dapat membantu kita menemukan kesalahan dalam kode CSS.





## CSS dan debugging

Sama seperti HTML, CSS adalah *permisif*. Dalam kasus CSS, jika deklarasi tidak valid (mengandung kesalahan syntax, atau browser tidak mendukung fitur itu), browser akan mengabaikan sepenuhnya dan beralih ke yang berikutnya. Jika selector tidak valid, maka selector tidak akan memilih apa pun, dan seluruh aturan tidak akan melakukan apa pun. Sekali lagi, browser hanya beralih ke aturan berikutnya. Ini bagus dalam banyak hal. Dalam banyak kasus, konten kita akan ditampilkan kepada user kita, bahkan jika itu tidak ditata dengan benar. Tetapi ini tidak terlalu membantu ketika kita mencoba men-debug masalah dan kita bahkan tidak mendapatkan pesan kesalahan apa pun untuk membantu kita menemukannya. Ini bahkan lebih menyakitkan ketika konten tidak dapat dilihat oleh user kita. Mungkin styling yang kritis tidak diterapkan, mengakibatkan tata letak yang salah?

Untungnya ada beberapa tools yang tersedia untuk membantu kita. Mari kita lihat ini sekarang.



## Menginspeksi DOM dan CSS

Saat ini, semua web browser menyediakan tools developer yang dibuat untuk membantu kita memeriksa dan memahami halaman web. Di antara berbagai tools yang mereka sediakan, ada dua yang tersedia di semua browser yaitu: DOM Inspector dan CSS Editor, yang tersedia pada Firefox di halaman inspector tool. Kita telah melihat Inspektur DOM di Debugging HTML dan bagaimana hal itu dapat digunakan untuk memeriksa HTML. Di sini kita akan melihat ini dan CSS Editor, dan bagaimana menggunakannya bersama-sama untuk men-debug masalah yang terdapat pada CSS.



## Bikin file HTML seperti dibawah ini

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8">
 <title>Debugging CSS</title>
 <link rel="stylesheet" href="style.css">
 </head>
 <body>
 <header>
 <h1>My imperfect page</h1>
 </header>

 <main>

 <h2>My article</h2>

 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero. Sed cursus ante dapibus diam. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum. Praesent mauris. Fusce nec tellus sed augue semper porta. Mauris massa. Vestibulum lacinia arcu eget nulla.</p>

 Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Integer nec odio. Praesent libero.
 Sed cursus ante dapibus diam.

 <p>Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Curabitur sodales ligula in libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aenean quam. In scelerisque sem at dolor.</p>

 </main>

 <footer>
 <p>©1978 Chris' brain</p>
 </footer>
 </body>
</html>
```

```
/* General styles */

html {
 font-family: 'Helvetica neue', Arial, 'sans serif';
 font-size: 10px;
}

body {
 width: 80em;
 margin: 0 auto;
}

/* Typography */

h1 {
 font-size: 4em;
}

h2 {
 font-size: 3.2em;
}

p li {
 font-size: 1.7em;
}

/*+++ bagian styles yang lebih spesifik +++*/

/* header dan footer */

header, footer {
 background-color: teal;
 height: 10em;
 padding: 2em;
}

h1, footer p {
 margin: -70px;
}

h1 {
 text-align: center;
 padding: 0.5em 0;
}

/* main content */

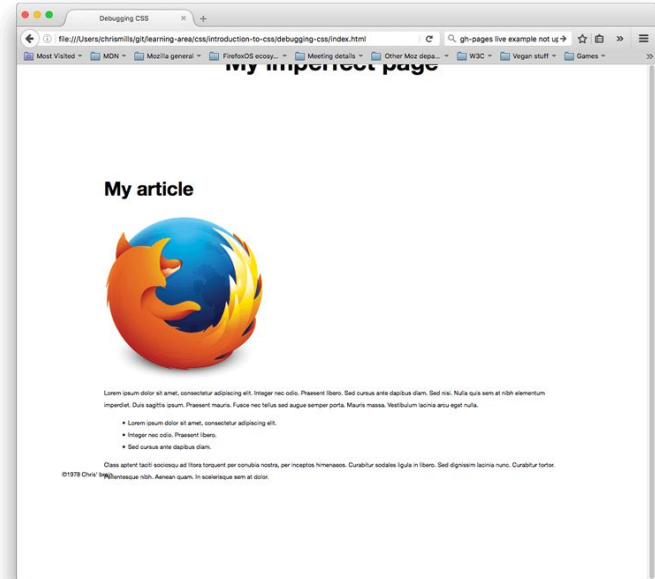
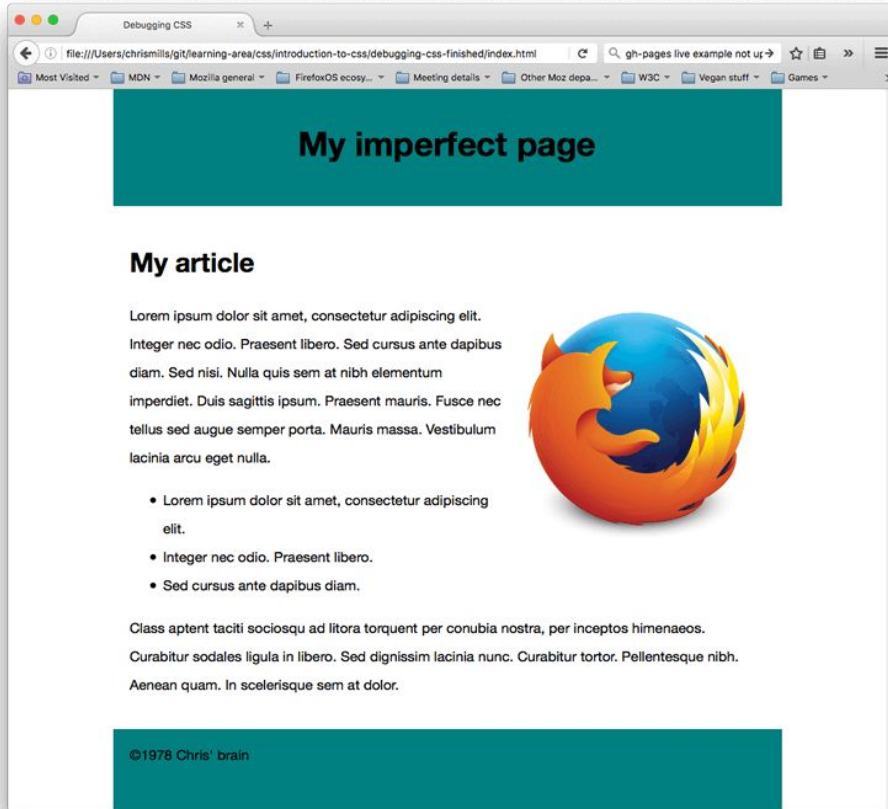
main {
 padding: 2em;
}

main p, main li {
 line-height: 2;
}

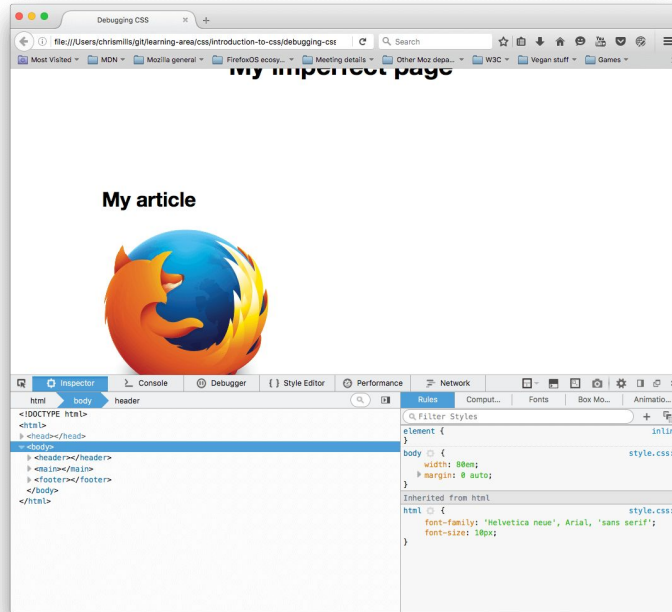
img {
 float: right;
 margin: 1.2em 2em;
}
```

## File CSS Seperti Ini

# Hasilnya seperti ini



Coba tekan F12 di browsermu



Atau inspect element

# Rangkuman

## HTML

Sebuah bahasa **markup** yang digunakan untuk memberi tahu browser apa yang harus ditampilkan.

## Bahasa Markup

Sebuah teks yang akan diproses menjadi tampilan..

## Tag

Suatu karakter yang dibungkus oleh **Enclosing Tag** di dalam suatu dokumen **HTML**, mempunyai representasi yang unik di dalam browser.

## Elemen Bersarang

Elemen yang terdapat di dalam elemen.

## Atribut

Suatu informasi yang terdapat di dalam suatu elemen.

## Hyperlink

Suatu elemen yang mempunyai referensi ke halaman lain, dimana kalau kita klik elemen tersebut akan menuju ke halaman lain.

# Rangkuman

## CSS

Suatu bahasa yang digunakan untuk mengatur tampilan di browser..

## CSS Selector

**.sesuatu**

menerapkan aturan berdasarkan kelas

**#sesuatu**

menerapkan aturan berdasarkan id

**p**

menerapkan aturan berdasarkan nama elemen

**input[ type="text" ]**

menerapkan aturan berdasarkan atribut dari sesuatu yang dipilih.

## Komentar di CSS

```
/* Tulisan ini bakal dicuekin browser :(*/
```

# Buatlah

Satu **halaman**

Tentang **dirimu**

- Halaman tersebut harus memiliki **Biodata** tentang dirimu (Nama lengkap, TTL, dll)
- Tampilkan **foto** dirimu di dalam **halaman** tersebut
- Sertakan **link** menuju **media sosialmu** dengan menggunakan **Hyperlink**
- **Desain bebas** (kreativitas tanpa batas, tugas ini bisa jadi portofolio kamu nantinya)
- Kirim halaman tersebut ke **ikarina@binar.co.id** dengan subjek:  
**HTML - CSS**



## Tugas



# Referensi

[https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction\\_to\\_HTML/Getting\\_started](https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started)

# Terima Kasih