

# CSS LAYOUT

# Sebelum Belajar

- HTML
- CSS

# PENGENALAN CSS LAYOUT

# PENGENALAN CSS LAYOUT

- Di materi CSS Dasar, kita sudah belajar bagaimana menambahkan gaya pada element-element di HTML
- Selain menambahkan gaya di element, CSS juga biasanya digunakan untuk membuat layout (tata letak) halaman web
- Hampir semua web di dunia, akan menggunakan CSS untuk melakukan layouting (tata letak) komponen-komponen HTML nya
- Salah satu tujuan melakukan tata letak adalah agar tampilan halaman web lebih menarik

# JANGAN GUNAKAN TABEL UNTUK LAYOUT

- Kadang mungkin ada beberapa pengembang web menggunakan Tabel untuk membuat tata letak, karena dianggap lebih mudah
- Namun, sangat tidak disarankan untuk menggunakan tabel, terutama pada kondisi layout yang sangat kompleks, maka akan sangat menyulitkan

MEMBUAT PROJECT

# MEMBUAT PROJECT

- Silahkan buat folder baru untuk menyimpan kode program belajar kita
- Misal nama foldernya belajar-css-layout
- Silahkan buka menggunakan Text Editor yang digunakan, misal Visual Studio Code

NORMAL FLOW



# NORMAL FLOW

- Normal Flow adalah bagaimana Web Browser menampilkan tata letak halaman HTML secara default, ketika kita tidak mengubah apapun pada tata letak nya
- Secara default, jika kita tidak menambahkan layout CSS sama sekali, maka Web Browser akan menampilkan halaman Web dalam Normal Flow
- Biasanya setiap Web Browser hampir memiliki Normal Flow yang sama

## KODE : CONTOH

```
7   </head>
8   ✓ <body>
9     <h1>Normal Flow</h1>
10
11    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Doloremque fugiat asperiores
    culpa quae sequi optio voluptates, dolor laboriosam eius, magnam expedita dolore officii
12
13    <h2>Normal Flow</h2>
14
15    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Doloremque fugiat asperiores
    culpa quae sequi optio voluptates, dolor laboriosam eius, magnam expedita dolore officii
16  </body>
17  </html>|
```

# URUTAN KOMPONEN HTML

- Secara default, Web Browser akan menampilkan urutan sesuai dengan posisi kode HTML yang dibuat
- Walaupun kita tambahkan Style menggunakan CSS pada komponen HTML nya, tetap saja, urutan nya secara default akan mengikuti urutan sesuai kode HTML yang kita buat

## KODE : NORMAL FLOW CSS

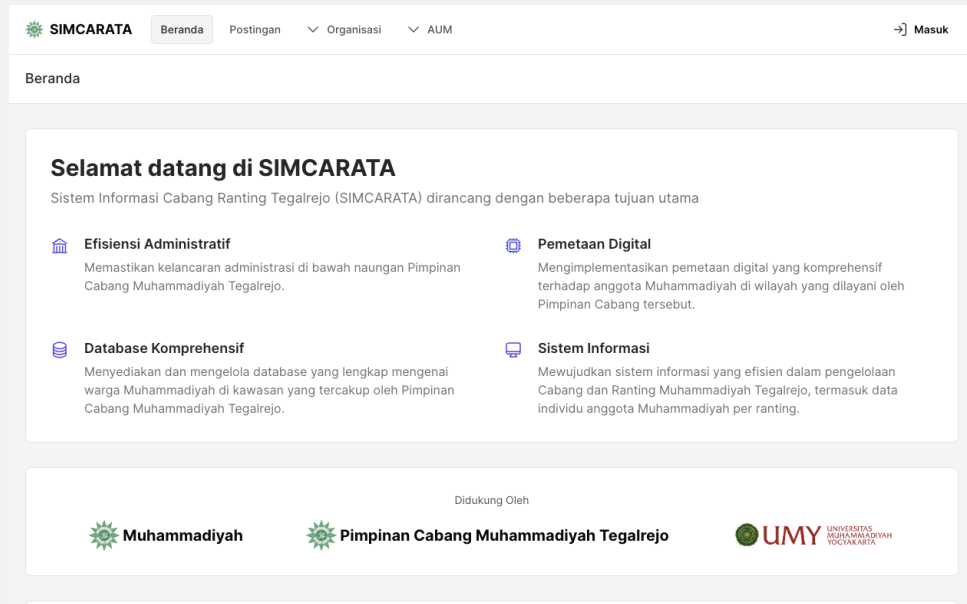
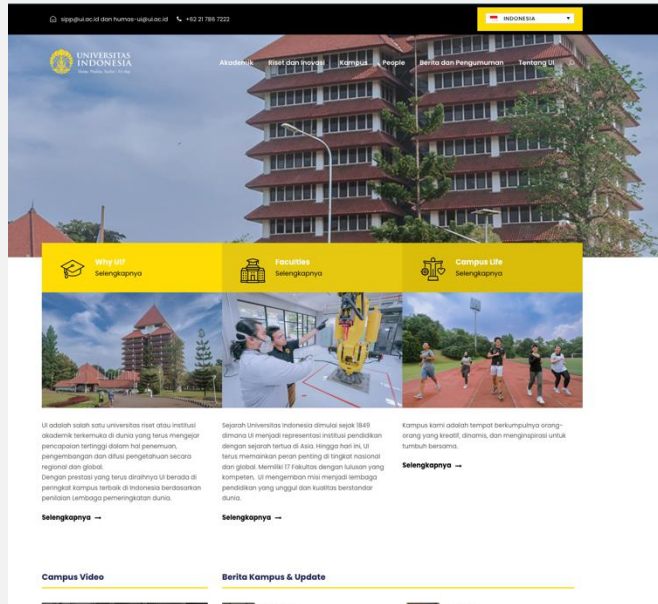
```
<title>Normal Flow</title>
<style>
  body {
    width: 500px;
    margin: 0 auto;
  }

  p {
    background-color: aqua;
    border: 2px solid blue;
    padding: 10px;
    margin: 10px;
  }
</style>
</head>
```

## KENAPA TIDAK CUKUP DENGAN NORMAL FLOW?

- Untuk membuat halaman yang menarik, kadang butuh tata letak yang tidak bisa sesuai dengan Normal Flow
- Kadang kita butuh meletakkan beberapa posisi komponen ditempat yang kita inginkan, sehingga tidak bisa jika mengikuti aturan Normal Flow
- Maka kita bisa lihat bahwa tata letak komponen tidak ditampilkan secara Normal Flow

# WEB



DISPLAY

# DISPLAY

- Sebelum kita belajar melakukan tata letak, kita perlu bahas ulang tentang Display
- Kita tahu bahwa komponen di HTML memiliki default display, ada yang block dan ada yang inline
- Sebenarnya, kita bisa mengubah nilai display untuk komponen HTML menggunakan atribut display di CSS
- <https://developer.mozilla.org/en-US/docs/Web/CSS/display>



## NILAI CSS DISPLAY

- **inline**, artinya komponen ditampilkan secara inline (hanya mengambil tempat secukupnya)
- **block**, artinya komponen ditampilkan secara block (mengambil tempat kiri ke kanan komponen di atasnya)
- **inline-block**, artinya komponen ditampilkan secara inline, tapi kita bisa mengubah tinggi dan lebar komponennya seperti layaknya block
- **none**, artinya komponen akan dihapus dan tidak ditampilkan

## KODE : DISPLAY HTML

```
8  </head>
9  <body>
0      <div class="content">
1          <h1>Ini Konten 1</h1>
2          <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Magni :
3      </div>
4
5      <div class="content">
6          <h1>Ini Konten 2</h1>
7          <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Explica
8      </div>
9  </body>
0  </html>
```

## KODE : DISPLAY CSS

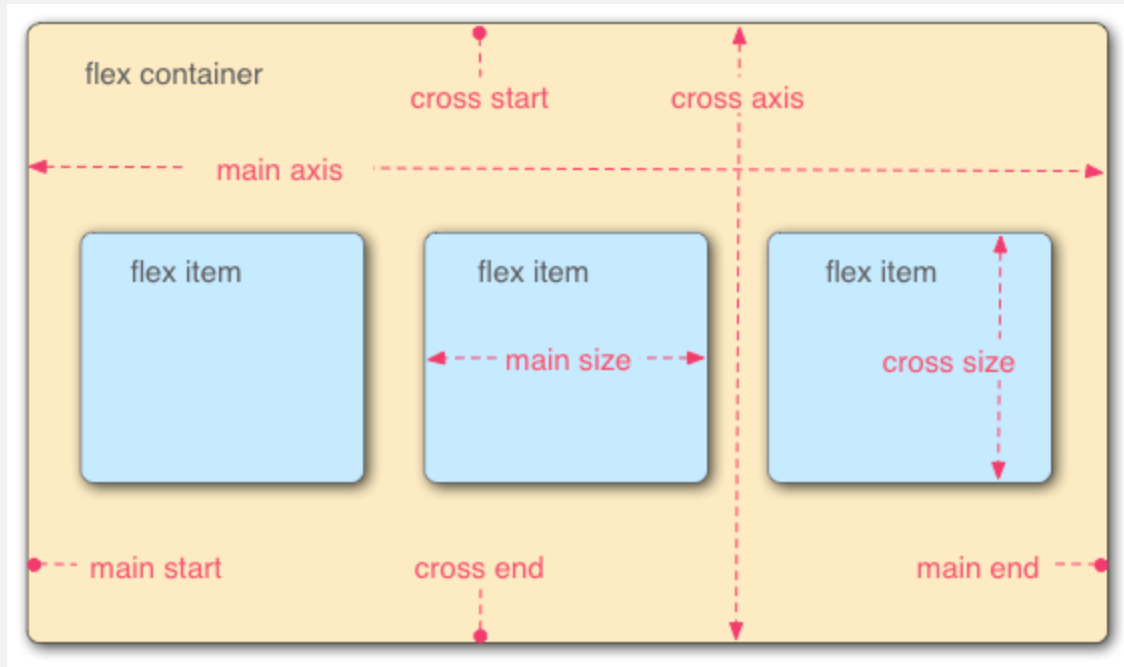
```
<title>Display</title>
<style>
  .content {
    vertical-align: top;
    display: inline-block;
    width: 200px;
    background-color: aqua;
    border: 2px solid blue;
    padding: 10px;
    margin: 10px;
  }
</style>
```

# FLEXBOX

# FLEXBOX

- Flexbox Layout bertujuan untuk menyediakan cara yang lebih efisien untuk menata letak, menyelaraskan, dan mendistribusikan ruang antar item dalam wadah (container), bahkan ketika ukurannya tidak diketahui dan/atau dinamis (sehingga disebut “fleksibel”).
- Flexbox Layout paling sesuai untuk komponen aplikasi, dan tata letak skala kecil, sedangkan Grid Layout ditujukan untuk tata letak skala besar.
- Grid Layout akan dibahas di materi tersendiri

# DIAGRAM FLEXBOX



# FLEX CONTAINER

- Untuk membuat Flex Container, kita bisa menggunakan display value flex
- Semua child element yang terdapat di dalam Flex Container, maka kita sebut sebagai Flex Item

## KODE : FLEXBOX

```
<style>
  .container {
    display: flex;
  }
  .content {
    background-color: aqua;
    border: 2px solid blue;
    margin: 10px;
    padding: 10px;
  }
</style>
```

```
<div class="container">
  <div class="content">
    <h1>Ini Konten 1</h1>
    <p>Lorem ipsum dolor sit, amet consectetur adipis
  </div>
  <div class="content">
    <h1>Ini Konten 2</h1>
    <p>Lorem ipsum dolor sit, amet consectetur adipis
  </div>
  <div class="content">
    <h1>Ini Konten 2</h1>
    <p>Lorem ipsum dolor sit, amet consectetur adipis
  </div>
</div>
```



# FLEX DIRECTION

- Secara default, Flex Item akan ditampilkan dengan arah dari kiri ke kanan, kita bisa mengubah dengan atribut flex-direction :
- <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-direction>

## KODE : FLEX DIRECTION

```
7 <style>
8   .container {
9     display: flex;
10    flex-direction: row-reverse;
11  }
12  .content {
13    background-color: aqua;
14    border: 2px solid blue;
15    margin: 10px;
16    padding: 10px;
17  }
18 </style>
```

# FLEX WRAP

- Secara default, Flex Item akan ditampilkan dalam satu garis, baik itu vertical (row) atau horizontal (column)
- Namun jika kita ingin Flex Item di wrap pada garis berbeda ketika dibutuhkan, maka kita bisa gunakan atribut flex-wrap
- <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-wrap>

## KODE : FLEX WRAP

```

<style>
  .container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
  }
  .content {
    width: 200px;
    background-color: aqua;
    border: 2px solid blue;
    margin: 10px;
    padding: 10px;
  }
</style>
```

FLEX ITEMS

# FLEX ITEMS

- Komponen yang terdapat di dalam Flex Container, kita sebut dengan nama Flex Item
- Selain melakukan pengaturan ke Flex Container, kita juga bisa melakukan pengaturan ke Flex Item

# ORDER

- Secara default urutan Flex Item akan ditampilkan sesuai dengan urutan kode HTML yang dibuat
- Namun, jika kita ingin mengubah urutan tampilannya, kita bisa menggunakan atribut order
- <https://developer.mozilla.org/en-US/docs/Web/CSS/order>

## KODE : FLEX ITEM ORDER

```
✓ <body>
✓ <div class="container">
✓   <div class="content" style="order: 2;">
      <h1>Ini Konten 1</h1>
      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Magni
    </div>
✓   <div class="content" style="order: 3;">
      <h1>Ini Konten 2</h1>
      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Magni
    </div>
✓   <div class="content" style="order: 1;">
      <h1>Ini Konten 3</h1>
      <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Magni
    </div>
  </div>
```



# FLEX GROW

- Atribut flex-grow mendefinisikan kemampuan item fleksibel untuk berkembang jika diperlukan.
- Atribut flex-grow menerima nilai yang berfungsi sebagai proporsi dari total.
- Ini menentukan jumlah ruang yang tersedia di dalam wadah fleksibel yang harus digunakan oleh item tersebut.
- Misal kita punya 5 Flex Item, dimana total dari 5 Flex Item tersebut memiliki jumlah flex-grow 20, artinya per 1 grow adalah  $100\% / 20 = 5\%$
- Sehingga Flex Item yang memiliki nilai flex-grow 2 artinya memiliki ruang sebesar 10%
- <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-grow>

# KODE : FLEX GROW

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Flex Grow</title>
<style>
    .container {
        display: flex;
        flex-direction: row;
    }
    .content {
        background-color: aqua;
        border: 2px solid blue;
        padding: 5px;
        margin: 5px;
    }
    .small {
        flex-grow: 1;
    }
    .large {
        flex-grow: 2;
    }
</style>
```

```
<body>
  <div class="container">
    <div class="content small">A</div>
    <div class="content large">B</div>
    <div class="content small">C</div>
    <div class="content large">D</div>
    <div class="content small">E</div>
  </div>
</body>
</html>
```

# FLEX SHRINK

- Jika flex-grow merupakan kemampuan untuk Flex Item untuk berkembang, maka atribut flex-shrink untuk menyusut ketika dibutuhkan
- Biasanya kemampuan untuk menyusut dibutuhkan ketika memang ukuran halaman tidak cukup, dan kita bisa mengatur nilai factor dari penyusutannya antar Flex Item, sama seperti flex-grow
- <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-shrink>

# KODE : FLEX SHRINK

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Flex Shrink</title>
<style>
    .container {
        display: flex;
    }
    .content {
        background-color: aqua;
        border: 2px solid blue;
        padding: 10px;
        margin: 10px;
    }
    .content1 {
        flex-shrink: 1;
    }
    .content2 {
        flex-shrink: 2;
    }
    .content3 {
        flex-shrink: 1;
    }
</style>
```

```
</style>
</head>
<body>
    <div class="container">
        <div class="content content1">Lorem ipsum dolor sit amet
        <div class="content content2">Lorem ipsum dolor sit, ame
        <div class="content content3">Lorem ipsum dolor sit, ame
    </div>
</body>
</html>
```

# FLEX BASIS

- Atribut flex-basis digunakan untuk membuat Flex Item mengambil ruang sebesar yang ditentukan, atau jika ruangan tidak tersedia, ambil semampunya
- Jika kita menggunakan flex-shrink : 0, maka akan dipastikan bahwa Flex Item akan mengambil sejumlah flex-basis, walaupun ukuran layar tidak mencukupi
- <https://developer.mozilla.org/en-US/docs/Web/CSS/flex-basis>

# KODE : FLEX BASIS

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Flex Basis</title>
<style>
  .container {
    display: flex;
  }
  .content {
    background-color: aqua;
    border: 2px solid blue;
    padding: 10px;
    margin: 10px;
  }
  .content1 {
    flex-basis: 300px;
    flex-shrink: 0;
  }
</style>
</head>
```

```
</head>
<body>
  <div class="container">
    <div class="content content1">Lorem ipsum dolor si
    <div class="content content2">Lorem ipsum dolor si
    <div class="content content3">Lorem ipsum dolor si
  </div>
</body>
</html>
```

# FLEX ALIGNMENT

# FLEX ALIGNMENT

- Flex Item yang terdapat di dalam Flex Container, bisa kita rapikan atau selaraskan dengan beberapa atribut



# JUSTIFY CONTENT

- Saat kita menggunakan Flexbox, kita bisa menggunakan atribut justify-content menentukan bagaimana Web Browser menentukan jarak antar Flex Item
- Ada banyak nilai yang bisa kita gunakan dalam justify-content, seperti yang terlihat di gambar
- <https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content>

flex-start



flex-end



center



space-between



space-around



space-evenly



# KODE : JUSTIFY CONTENT

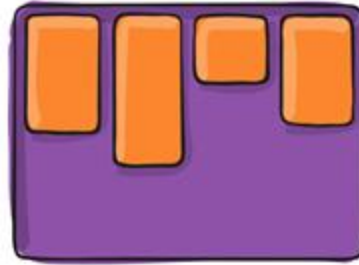
```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Flex Align</title>
<style>
  .container {
    background-color: yellow;
    height: 500px;
    display: flex;
    justify-content: center;
  }
  .content {
    background-color: aqua;
    border: 1px solid blue;
    padding: 10px;
    margin: 10px;
    width: 200px;
    height: auto;
  }
</style>
</head>
```

```
</head>
<body>
  <div class="container">
    <div class="content">
      <h1>Content 1</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    </div>
    <div class="content">
      <h1>Content 2</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    </div>
    <div class="content">
      <h1>Content 3</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.
    </div>
  </div>
</body>
```

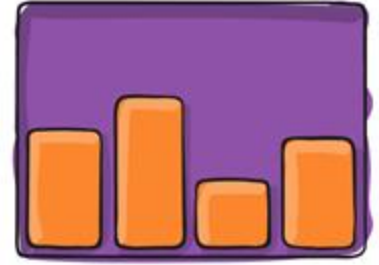
# ALIGN ITEMS

- Atribut align-items bisa digunakan untuk perataan Flex Item secara cross-axis
- <https://developer.mozilla.org/en-US/docs/Web/CSS/align-items>

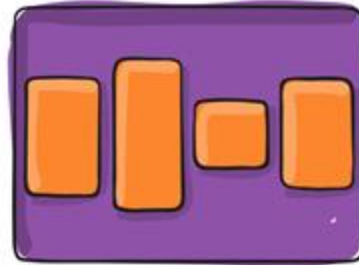
flex-start



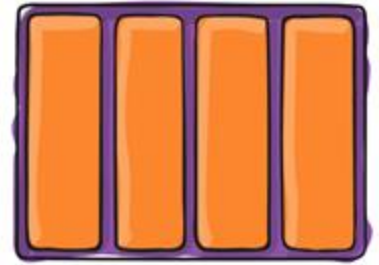
flex-end



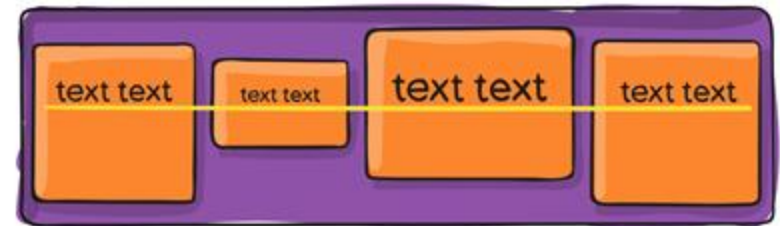
center



stretch



baseline



## KODE : FLEX ALIGN ITEMS

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Flex Align Items</title>
<style>
  .container {
    background-color: yellow;
    height: 500px;
    display: flex;
    align-items: flex-end;
  }
  .content {
    background-color: aqua;
    border: 1px solid blue;
    padding: 10px;
    margin: 10px;
    width: 200px;
    height: auto;
  }
</style>
```

```
</head>
<body>
  <div class="container">
    <div class="content">
      <h1>Content 1</h1>
      <p>Lorem ipsum dolor sit amet</p>
    </div>
    <div class="content">
      <h1>Content 2</h1>
      <p>Lorem ipsum dolor sit amet</p>
      <p>Lorem ipsum dolor sit amet</p>
    </div>
    <div class="content">
      <h1>Content 3</h1>
      <p>Lorem ipsum dolor sit amet</p>
    </div>
  </div>
</body>
</html>
```

# ALIGN CONTENT

- Atribut align-content digunakan agar menyelaraskan Flex Item seperti justify-content
- Namun pada align-content, ini hanya bisa dilakukan jika menggunakan flex-wrap dengan nilai wrap atau wrap-reverse
- <https://developer.mozilla.org/en-US/docs/Web/CSS/align-content>

flex-start



flex-end



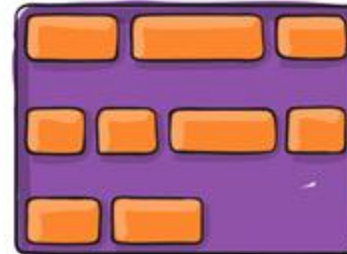
center



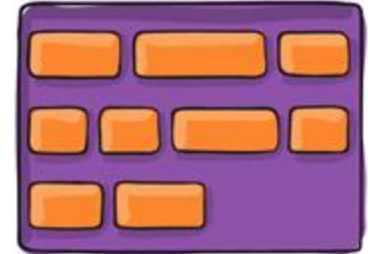
stretch



space-between



space-around



# KODE : ALIGN CONTENT

```
<title>Flex Align Content</title>
<style>
  .container {
    background-color: yellow;
    height: 1000px;
    display: flex;
    flex-wrap: wrap;
    align-content: flex-start;
  }
  .content {
    background-color: aqua;
    border: 1px solid blue;
    padding: 10px;
    margin: 10px;
    width: 200px;
    height: auto;
  }
</style>
</head>
```

```
<body>
  <div class="container">
    <div class="content">
      <h1>Content 1</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 2</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 3</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 4</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 5</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
  </div>
</body>
```

# GAP

- Sebelumnya, untuk menambahkan gap (jarak) antar Flex Item, kita biasanya menggunakan margin
- Saat menggunakan Flexbox, lebih baik gunakan Gap untuk menentukan jarak dalam row (baris) atau column (kolom)
- <https://developer.mozilla.org/en-US/docs/Web/CSS/gap>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/row-gap>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-gap>



# KODE : FLEX GAP

```
<title>Flex Gap</title>
<style>
  .container {
    background-color: yellow;
    display: flex;
    flex-wrap: wrap;
    /* gap: 10px 20px; */
    row-gap: 10px;
    column-gap: 20px;
    padding: 10px;
  }
  .content {
    background-color: aqua;
    border: 1px solid blue;
    padding: 10px;
    width: 200px;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="content">
      <h1>Content 1</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 2</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 3</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 4</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
    <div class="content">
      <h1>Content 5</h1>
      <p>Lorem ipsum dolor sit amet consectetur</p>
    </div>
  </div>
</body>
```



# GRIDS

# GRID

- Grid Layout adalah sistem tata letak berbasis grid dua dimensi
- Flexbox adalah Layout yang bagus, tapi hanya satu arah, dan cocok pada kasus tertentu
- Grid Layout adalah fitur CSS yang dibuat untuk mengatasi permasalahan tata letak yang kompleks

# GRID CONTAINER

- Untuk membuat Grid Container, kita bisa menggunakan atribut display dengan nilai grid
- Tidak seperti Flexbox, saat menggunakan Grid Container, secara default akan terdapat satu kolom, sehingga tampilan awal mungkin tidak akan berbeda dengan Normal Flow

## KODE : GRID

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Grid</title>
<style>
    .container {
        display: grid;
    }
    .content {
        background-color: aqua;
        border: 1px solid blue;
        padding: 10px;
        margin: 10px;
    }
</style>
</head>
```

```
</head>
<body>
    <div class="container">
        <div class="content">
            <h1>Content 1</h1>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit</p>
        </div>
        <div class="content">
            <h1>Content 2</h1>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit</p>
        </div>
        <div class="content">
            <h1>Content 3</h1>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit</p>
        </div>
        <div class="content">
            <h1>Content 4</h1>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit</p>
        </div>
        <div class="content">
            <h1>Content 5</h1>
            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit</p>
        </div>
    </div>
</body>
```

# GRID TERMINOLOGY

# GRID TERMINOLOGY

- Sebelum kita lanjut tentang Grid Container, kita akan bahas dulu tentang Grid Terminology, agar tidak bingung di materi selanjutnya
- Seperti yang sudah kita tahu, bentuk dari Grid itu mirip seperti tabel

# GRID CONTAINER

- Komponent yang menggunakan display: grid, kita sebut dengan Grid Container

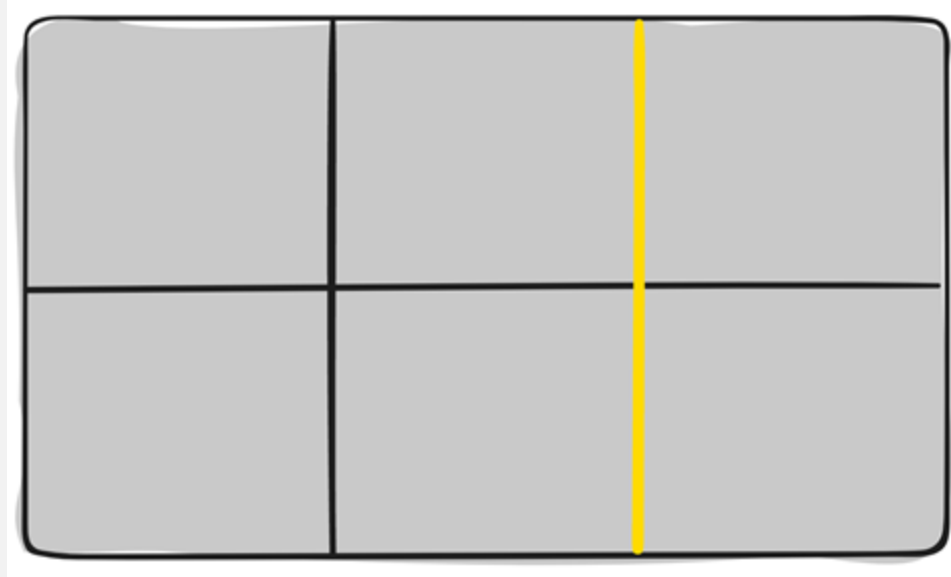
## GRID ITEM

- Komponen Children dari Grid Container kita sebut dengan nama Grid Item



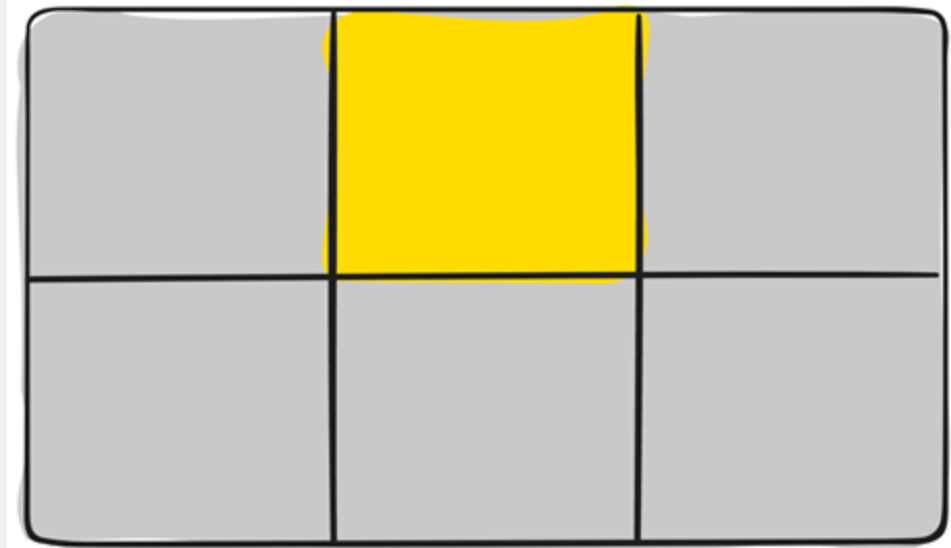
# GRID LINE

- Garis pemisah dalam grid baik itu yang vertical atau horizontal, kita sebut dengan Grid Line



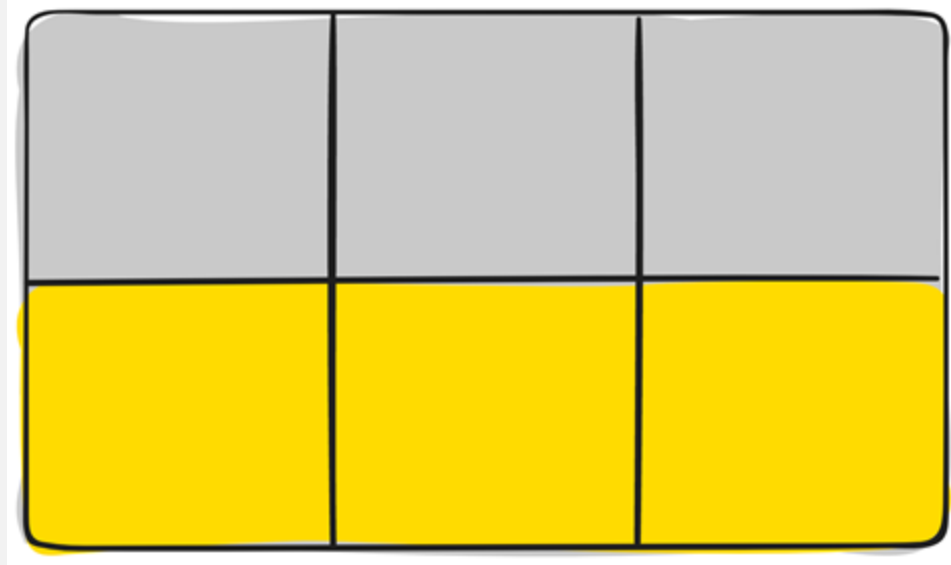
# GRID CELL

- Area yang terdapat didalam kolom dan baris yang dipisah oleh Grid Line, kita sebut dengan Grid Cell



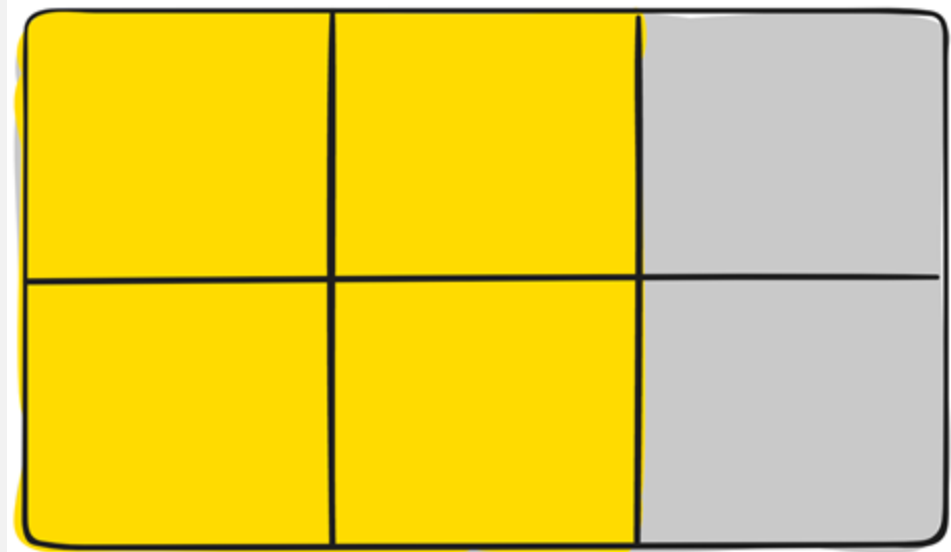
# GRID TRACK

- Bagian antara dua Grid Line, atau bisa dibilang baris, kita sebut dengan Grid Track



# GRID AREA

- Total area dari beberapa Grid Cell, kita sebut dengan Grid Area



# GRID TEMPLATES

# GRID TEMPLATE

- Untuk menentukan kolom dan baris dalam Grid, kita bisa menggunakan Grid Template
- Atribut grid-template-columns digunakan untuk menentukan kolom
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-columns>
- Atribut grid-template-rows digunakan untuk menentukan baris
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-rows>

# GRID TEMPLATE

```
<meta name="viewport" content="width=device-width">
<title>Grid</title>
<style>
  .container {
    display: grid;
    grid-template-columns: 200px auto 200px;
    grid-template-rows: 500px auto;
  }
  .content {
    background-color: aqua;
    border: 1px solid blue;
    padding: 10px;
    margin: 10px;
  }
</style>
</head>
```

```
<body>
  <div class="container">
    <div class="content">
      <h1>Content 1</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
    <div class="content">
      <h1>Content 2</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
    <div class="content">
      <h1>Content 3</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
    <div class="content">
      <h1>Content 4</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
    <div class="content">
      <h1>Content 5</h1>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
  </div>
```

# GRID ITEMS



# GRID ITEMS

- Komponen Children langsung dari Grid Container kita sebut dengan Grid Item
- Ada banyak hal yang bisa kita lakukan di Grid Item, seperti memilih kolom, baris atau area

# GRID ITEM START DAN END

- Grid Item bisa ditentukan mau mulai dari kolom atau baris berapa
- Bahkan, untuk mempermudah, ketika membuat row template dan column template, kita bisa memberi nama sehingga mudah digunakan
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-column-start>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-column-end>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-row-start>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-row-end>

## GRID LINE NAME

- Sebelum kita praktek tentang Grid Start dan End, kadang ada baiknya kita membuat nama untuk Grid Line saat membuat Grid Template
- Kita bisa buat Grid Line dengan menggunakan [namaline]

## KODE : GRID LINE NAME

```
<html lang= en >
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Grid Start dan End</title>
  <style>
    .container {
      display: grid;
      grid-template-columns: [cline1] 100px [cline2] auto [cline3] 100px [cline4];
      grid-template-rows: [rline1] 50px [rline2] auto [rline3] 100px [rline4];
    }
    .header {
```

## KODE : GRID ITEM START DAN END

```
}  
.header {  
  background-color: aqua;  
  grid-column-start: cline1;  
  grid-column-end: cline4;  
  grid-row-start: rline1;  
  grid-row-end: rline1;  
}  
.content {  
  background-color: bisque;  
  grid-area: content;  
  grid-column-start: cline2;  
  grid-column-end: cline3;  
  grid-row-start: rline2;  
  grid-row-end: rline3;  
}  
.footer {  
  background-color: aqua;  
  grid-column-start: cline1;  
  grid-column-end: cline4;  
  grid-row-start: rline3;  
  grid-row-end: rline4;  
}  
</style>
```

```
</head>  
<body>  
  <div class="container">  
    <div class="header">  
      Selamat Data di web Programmer Zaman Now  
    </div>  
    <div class="content">  
      <h1>Programmer Zaman Now</h1>  
      <p>Lorem ipsum dolor, sit amet consectetur adipisi<br>Lorem ipsum dolor sit amet consectetur adipisic</p>  
    </div>  
    <div class="footer">  
      Develop with Love by Programmer Zaman Now  
    </div>  
  </div>  
</body>
```

# GRID AREA

- Saat kita membuat Grid, kita bisa membuat Grid Area dengan menentukan nama dari Grid Area tersebut
- Untuk membuat Grid Area, kita bisa menggunakan atribut grid-template-areas
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-template-areas>
- Dan untuk menentukan Grid Item muncul di area mana, kita bisa gunakan atribut grid-area
- <https://developer.mozilla.org/en-US/docs/Web/CSS/grid-area>
- Penggunaan Grid Area lebih mudah dibanding Grid Item Start dan End

## KODE : GRID TEMPLATE AREA

```
<meta name="viewport" content="width=device-width,
<title>Grid Area</title>
<style>
.container {
  display: grid;
  grid-template-columns: 100px auto 100px;
  grid-template-rows: 50px auto 100px;;
  grid-template-areas:
    "header header header"
    ". content ."
    "footer footer footer";
}
.header {
  background-color: #aqua;
  grid-area: header;
}
.content {
  background-color: #bisque;
  grid-area: content;
}
.footer {
  background-color: #aqua;
  grid-area: footer;
}
</style>
```

```
</head>
<body>
  <div class="container">
    <div class="header">
      Selamat Data di web Programmer Zaman Now
    </div>
    <div class="content">
      <h1>Programmer Zaman Now</h1>
      <p>Lorem ipsum dolor, sit amet consectetur adipisicing
      <p>Lorem ipsum dolor sit amet consectetur adipisicing
    </div>
    <div class="footer">
      Develop with Love by Programmer Zaman Now
    </div>
  </div>
</body>
</html>
```

# GRID ALIGNMENT



# GRID ALIGNMENT

- Grid juga sama seperti Flexbox, kita bisa lakukan penyelarasan seperti layaknya di Flexbox
- Salah satu perbedaanya, dalam Grid, kita bisa melakukan penyelarasan Seluruh Grid Cell, atau hanya untuk satu Grid Cell saja

## JUSTIFY CONTENT

- Saat kita menggunakan Grid, kita bisa menggunakan atribut justify-content menentukan bagaimana Web Browser menentukan jarak antar Grid Column
- Ada banyak nilai yang bisa kita gunakan dalam justify-content, seperti yang terlihat di gambar
- <https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content>

## KODE : GRID JUSTIFY CONTENT

```
<meta name= viewport content= width=device-width, 1
<title>Grid Justify Content</title>
<style>
  .container {
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
    justify-content: space-evenly;
  }
  .content {
    background-color: ■bisque;
    border: 1px solid □black;
    padding: 10px;
    text-align: center;
  }
</style>
</head>
```

```
</head>
<body>
  <div class="container">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
  </div>
</body>
```

# ALIGN CONTENT

- Atribut align-content digunakan agar menyelaraskan Grid Row seperti justify-content
- <https://developer.mozilla.org/en-US/docs/Web/CSS/align-content>

## KODE : GRID ALIGN CONTENT

```
<meta name="viewport" content="width=device-width,
<title>Grid Align Content</title>
<style>
    .container {
        background-color: aqua;
        height: 500px;
        display: grid;
        grid-template-columns: 100px 100px 100px;
        grid-template-rows: 100px 100px 100px;
        align-content: space-evenly;
    }
    .content {
        background-color: bisque;
        border: 1px solid black;
        padding: 10px;
        text-align: center;
    }
</style>
```

```
</head>
<body>
    <div class="container">
        <div class="content">1</div>
        <div class="content">2</div>
        <div class="content">3</div>
        <div class="content">4</div>
        <div class="content">5</div>
        <div class="content">6</div>
        <div class="content">7</div>
        <div class="content">8</div>
        <div class="content">9</div>
    </div>
</body>
```

# JUSTIFY ITEMS

- Justify Content akan melakukan penyalarsan seluruh Grid Item beserta Grid Line, namun pada Justify Items, penyalarsan hanya dilakukan di level Grid Item di dalam Grid Cell, tanpa mengubah posisi Grid Line
- <https://developer.mozilla.org/en-US/docs/Web/CSS/justify-items>

## KODE : GRID JUSTIFY ITEMS

```
<meta name= viewport content= width=device-width,
<title>Grid Justify Items</title>
<style>
  .container {
    background-color: aqua;
    height: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
    justify-items: center;
  }
  .content {
    background-color: bisque;
    border: 1px solid black;
    padding: 10px;
    text-align: center;
  }
</style>
</head>
```

```
<body>
  <div class="container">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
  </div>
</body>
```

# ALIGN ITEMS

- Align Items akan melakukan penyelarasan di level Grid Cell
- <https://developer.mozilla.org/en-US/docs/Web/CSS/align-items>



## KODE : GRID ALIGN ITEMS

```
<meta name="viewport" content="width=device-width;" />
<title>Grid Align Items</title>
<style>
  .container {
    background-color: aqua;
    height: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
    align-items: center;
  }
  .content {
    background-color: bisque;
    border: 1px solid black;
    padding: 10px;
    text-align: center;
  }
</style>
</head>
```

```
</head>
<body>
  <div class="container">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
  </div>
</body>
```

# JUSTIFY DAN ALIGN SELF

- Sebelumnya untuk Justify Content / Items dan Align Content / Items, akan berdampak ke semua Grid Item
- Namun jika kita ingin membuat Justify dan Align hanya untuk salah satu Grid Item, kita bisa menggunakan Justify dan Align Self
- <https://developer.mozilla.org/en-US/docs/Web/CSS/justify-self>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/align-self>

## KODE : JUSTIFY DAN ALIGN SELF

```
<meta name= viewport content= width=device-width,
<title>Grid Self</title>
<style>
  .container {
    background-color: aqua;
    height: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
  }
  .content {
    background-color: bisque;
    border: 1px solid black;
    padding: 10px;
    text-align: center;
  }
  .content1 {
    justify-self: center;
    align-self: center;
  }
</style>
```

```
<body>
  <div class="container">
    <div class="content content1">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
  </div>
</body>
```

# GAP

- Gap, selain digunakan di Flexbox, juga bisa digunakan di Grid
- Tujuannya juga sama, untuk menambah jarak antar Grid Cell
- <https://developer.mozilla.org/en-US/docs/Web/CSS/gap>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/row-gap>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-gap>

## KODE : GRID GAP

```
<meta name="viewport" content="width=device-width,
<title>Grid Gap</title>|
<style>
  .container {
    background-color: aqua;
    height: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
    row-gap: 10px;
    column-gap: 10px;
  }
  .content {
    background-color: bisque;
    border: 1px solid black;
    padding: 10px;
    text-align: center;
  }
</style>
</head>
```

```
</head>
<body>
  <div class="container">
    <div class="content">1</div>
    <div class="content">2</div>
    <div class="content">3</div>
    <div class="content">4</div>
    <div class="content">5</div>
    <div class="content">6</div>
    <div class="content">7</div>
    <div class="content">8</div>
    <div class="content">9</div>
  </div>
</body>
```

SUBGRID

## SUBGRID

- Saat kita membuat tata letak yang sangat kompleks, kadang kita sering menjadikan Grid Item menjadi Grid Container lagi
- Pada kasus tertentu, walaupun Grid Item tersebut merupakan Grid Container, namun kadang kita ingin aturan row dan column nya ingin mengikuti aturan Grid Container yang ada di atasnya
- Pada kasus seperti ini, kita bisa menggunakan Subgrid

# KODE : GRID DI DALAM GRID

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Grid In Grid</title>
<style>
  .container {
    background-color: aqua;
    height: 500px;
    width: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px 100px 100px;
    grid-template-rows: 100px 100px 100px 100px 100px;
  }
  .content1 {
    display: grid;
    grid-column: 2 / 5;
    grid-row: 2 / 5;
    background-color: bisque;
    border: 1px solid black;
    grid-template-columns: 100px 100px 100px;
    grid-template-rows: 100px 100px 100px;
  }
  .inner {
```

```
    background-color: brown;
    border: 1px solid black;
    grid-column: 2/4;
    grid-row: 2/4;
  }
</style>
</head>
<body>
  <div class="container">
    <div class="content1">
      <div class="inner"></div>
    </div>
  </div>
</body>
```



## KODE : SUBGRID

```
<meta name= viewport content= width=device-width, initial-scale=1 />
<title>Subgrid</title>
<style>
  .container {
    background-color: aqua;
    height: 500px;
    width: 500px;
    display: grid;
    grid-template-columns: 100px 100px 100px 100px 100px;
    grid-template-rows: 100px 100px 100px 100px 100px;
  }
  .content1 {
    display: grid;
    grid-column: 2 / 5;
    grid-row: 2 / 5;
    background-color: bisque;
    border: 1px solid black;
    grid-template-columns: subgrid;
    grid-template-rows: subgrid;
  }
</style>
<div class="container">
  <div class="content1">
    <div class="inner"></div>
  </div>
</div>
```

```
  .inner {
    background-color: brown;
    border: 1px solid black;
    grid-column: 2/5;
    grid-row: 2/5;
  }
</style>
<div class="container">
  <div class="content1">
    <div class="inner"></div>
  </div>
</div>
```

MULTIPLE COLUMN

# MULTIPLE COLUMN

- Multiple column layout adalah cara untuk membuat tata letak seperti kolom dalam koran
- Cara untuk membuat column adalah dengan menggunakan attribute column-count
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-count>
- Atau jika kita ingin menentukan lebar kolom, tanpa peduli jumlah kolom yang akan dibuat, kita bisa gunakan atribut column-width
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-width>

## KODE : MULTIPLE COLUMN

```
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Column</title>
<style>
  .container {
    column-width: 200px;
  }
</style>
</head>
<body>
  <div class="container">
    <h1>Judul Artikel</h1>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit.
    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit.
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
  </div>
</body>
</html>
```

# COLUMN STYLE

- Kita juga bisa mengubah Style untuk Column
- Atribut column-gap untuk mengatur jarak Column
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-gap>
- Atribut column-rule untuk mengatur border dari Column
- <https://developer.mozilla.org/en-US/docs/Web/CSS/column-rule>

## KODE : COLUMN STYLE

```
<title>Column</title>
<style>
  .container {
    column-width: 200px;
    column-gap: 20px;
    column-rule: 1px solid black;
  }
</style>
</head>
<body>
  <div class="container">
    <h1>Judul Artikel</h1>
    <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. I
    <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. U
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. E
  </div>
```

FLOAT

# FLOAT

- CSS memiliki atribut float, yang sebelum ada Flexbox dan Grid, dulu Float biasanya digunakan untuk membuat Layout
- Atribut float digunakan untuk memposisikan elemen pada web, sekarang biasanya digunakan untuk gambar
- <https://developer.mozilla.org/en-US/docs/Web/CSS/float>



## KODE : FLOAT

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Float</title>
<style>
    .container {
        background-color: #00FFFF;
    }
    .image {
        float: right;
        width: 200px;
        height: 200px;
        background-color: #FFB6C1;
        text-align: center;
    }
</style>
```

```
<body>
    <div class="container">
        <h1>Judul Artikel</h1>
        <div class="image">A</div>
        <p>Lorem ipsum dolor sit amet consectetur, adipisci velit, sed quia non  
<p>Lorem ipsum dolor, sit amet consectetur adipisci velit, sed quia non  
<p>Lorem ipsum dolor sit, amet consectetur adipisci velit, sed quia non  
</div>
</body>
</html>
```

# POSITIONING

# POSITIONING

- Positioning memungkinkan kita meletakkan posisi elemen di tempat yang tidak sesuai dengan Normal Flow
- Misal meletakkan elemen di atas elemen lain, atau meletakkan elemen di posisi yang selalu sama di Viewport browser
- Untuk mengubah posisi elemen, kita bisa menggunakan atribut position
- <https://developer.mozilla.org/en-US/docs/Web/CSS/position>
- Secara default, position bernilai static, artinya dia akan ditempatkan sesuai dengan Normal Flow

# TOP, BOTTOM, LEFT DAN RIGHT

- Jika kita ubah atribut position dari default value-nya, kita bisa mengatur elemen menggunakan atribut top, bottom, left dan right
- <https://developer.mozilla.org/en-US/docs/Web/CSS/top>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/bottom>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/left>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/right>

# RELATIVE POSITIONING

- Relative positioning adalah posisi mirip seperti static positioning, dimana element akan ditempatkan sesuai Normal Flow.
- Namun setelah ditempatkan, kita bisa mengubah posisi elemen.
- Pada static positioning, mengubah posisi tidak akan berdampak apapun

## KODE : RELATIVE POSITIONING

```
<meta name="viewport" content="width=device-width, height=device-height">
<title>Relative Positioning</title>
<style>
    .content {
        background-color: aqua;
        width: 200px;
        height: 200px;
        display: inline-block;
    }
    .content2 {
        background-color: pink;
        position: relative;
        top: 20px;
        left: 20px;
    }
</style>
</head>
```

```
</head>
<body>
    <div class="content content1">
        <h1>Content 1</h1>
    </div>
    <div class="content content2">
        <h1>Content 2</h1>
    </div>
    <div class="content content3">
        <h1>Content 3</h1>
    </div>
</body>
</html>
```

# ABSOLUTE POSITIONING

- Absolute positioning adalah menghapus elemen dari Normal Flow, bahkan tidak ada space yang digunakan sama sekali
- Posisi awal untuk elemen absolute adalah relative ke posisi terdekat dengan elemen sebelumnya, atau jika tidak ada, berarti di awal block element parent nya
- Jika elemen absolute tidak memiliki parent, maka parent nya adalah html





# Z-INDEX

- Saat kita menggunakan Relative dan Absolute Positioning, yang mulai mengganggu adalah elemen akan saling bertumpuk
- Secara default saat menggunakan relative dan absolute, maka posisi akan diatas element yang static, tapi bagaimana jika ternyata kita ingin mengubah posisi tumpukan?
- Untuk mengubah posisi tumpukan elemen, kita bisa menggunakan atribut z-index, yang secara default bernilai auto atau 0
- Semakin tinggi nilai z-index, artinya posisi akan semakin diatas
- <https://developer.mozilla.org/en-US/docs/Web/CSS/z-index>

KODE : ABSOLUTE POSITIONING DENGAN Z-INDEX

```
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
<title>Absolute Positioning</title>
<style>
    .content {
        background-color: #00FFFF;
        width: 200px;
        height: 200px;
        display: inline-block;
    }
    .content2 {
        background-color: #FFB6C1;
        position: absolute;
        left: 20px;
        top: 20px;
        z-index: -1;
    }
</style>
```

[illegible]

# FIXED POSITIONING

- Fixed positioning adalah menghapus elemen dari Normal Flow, bahkan tidak ada space yang digunakan sama sekali, jadi sama seperti Absolute positioning
- Posisi awal untuk elemen absolute adalah relative ke posisi terdekat dengan elemen sebelumnya, atau jika tidak ada, berarti di awal block element parent nya
- Namun yang membedakan dari Absolute adalah, perubahan posisi pada element fixed, dia akan relative ke viewport (halaman web yang terlihat)
- Oleh karena itu, jika halaman web kita scroll, maka elemen fixed akan diam ditempat, dan tidak akan mengikuti scroll

KODE : FIXED POSITIONING

```
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
<title>Fixed Positioning</title>
<style>
    .content {
        background-color: aqua;
        width: 200px;
        height: 200px;
        display: inline-block;
    }
    .content2 {
        background-color: pink;
        position: fixed;
        left: 20px;
        top: 20px;
        z-index: -1;
    }
</style>
</head>
```

[illegible]

# STICKY POSITIONING

- Sticky positioning adalah gabungan antara relative dan fixed position
- Sticky positioning akan menampilkan elemen seperti relative positioning, yang artinya dalam Normal Flow, namun ketika elemen di scroll dalam ambang batas yang sudah disesuaikan, maka otomatis akan menjadi fixed positioning

## KODE : STICKY POSITIONING

```
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Sticky Positioning</title>
7 <style>
8   .sticky {
9     background-color: pink;
10    position: sticky;
11    top: 20px;
12    z-index: 1;
13  }
14 </style>
15 </head>
16 <body>
17   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
18   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
19   <p class="sticky">Lorem ipsum dolor sit amet consectetur adipisicing elit. I
20   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
21   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
22   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
23   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
24   <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. In repellat arch
```

# FRACTION UNIT

# FRACTION UNIT

- Saat kita membuat layout, kadang kita akan sering menggunakan satuan unit bernama fr (fraction)
- Fraction adalah sisa ruang dalam Grid, sisa ruang biasanya adalah sisa ruang yang bisa diisi setelah di kurangan Grid Item yang tidak flexible (ukurannya sudah fix)
- Cara perhitungan fr mirip dengan flex-grow, dimana ukurannya akan dihitung dari total fr
- Misal jika sisa ruang adalah 1000px, dan total fr adalah 20fr, maka 1 pr bernilai 50px



## KODE : FRACTION UNIT

```
<title>Fraction</title>
<style>
  .container {
    display: grid;
    grid-template-columns: 200px repeat(4, 1fr);
    column-gap: 10px;
  }
  .sidebar {
    background-color: pink;
  }
  .content {
    background-color: aqua;
  }
</style>
```

```
</head>
<body>
  <div class="container">
    <div class="sidebar">
      <h1>Sidebar</h1>
    </div>
    <div class="content">Content</div>
    <div class="content">Content</div>
    <div class="content">Content</div>
    <div class="content">Content</div>
  </div>
</body>
</html>
```

# MEDIA QUERIES

# MEDIA QUERIES

- Media Queries adalah fitur di CSS yang bisa digunakan untuk memodifikasi tampilan web sesuai dengan kondisi Device, Web Browser atau System Setting milik pengguna
- Seperti yang kita tahu, saat membuat web, pastinya kita tahu bahwa perangkat pengguna pasti berbeda-beda, ada yang membuat web kita dari Smartphone, Laptop, Komputer, bahkan ukuran layarnya bisa berbeda-beda
- Oleh karena itu kadang mungkin kita ingin mengubah tampilan sesuai dengan kondisi perangkat pengguna

# MENGGUNAKAN MEDIA QUERIES

- Untuk menggunakan Media Queries, kita bisa lakukan di HTML ketika menggunakan file CSS yang berbeda
- Atau bisa langsung menggunakan @media di file CSS yang kita buat
- Kita akan coba praktekan dua-dua nya
- [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_media\\_queries/Using\\_media\\_queries](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries)

## KODE : HTML MEDIA QUERY


```

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Media Query</title>
  <link rel="stylesheet" href="small.css" media="(min-width: 10px)">
  <link rel="stylesheet" href="large.css" media="(min-width: 500px)">
</head>
<body>
  <div class="container">
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
    <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
  </div>
</body>
</html>



```

## KODE : LARGE CSS

```
# large.css >  .content
```

```
1  .container {  
2      display: grid;  
3      grid-template-columns: repeat(4, 1fr);  
4      column-gap: 10px;  
5      row-gap: 10px;  
6  }  
7  
8  .content {  
9      background-color:  aqua;  
10 }
```

## KODE : SMALL CSS

```
# small.css >  .content  
1  .container {  
2      display: grid;  
3      grid-template-columns: repeat(1, 1fr);  
4      column-gap: 10px;  
5      row-gap: 10px;  
6  }  
7  
8  .content {  
9      background-color:  aqua;  
10 }
```

## @MEDIA

- Selain menggunakan Media Query di HTML, kita juga bisa langsung menggunakan Media Query di file CSS menggunakan at-rule @media
- At-rule @media memiliki aturan penulisan seperti berikut :

@media MediaType Operator MediaFeature Operator MediaFeature



## @MEDIA RULE

- Untuk Media Type, kita bisa lihat disini untuk daftar yang tersedia :  
<https://www.w3.org/TR/CSS21/media.html>
- Dan untuk Media Feature, kita bisa melihat disini untuk daftar yang tersedia :  
<https://web.dev/learn/design/media-features>
- Dan untuk Operator terdapat tiga pilihan :
  - and untuk kombinasi beberapa media feature, dan semua media feature wajib terpenuhi
  - or, menggunakan , (koma), untuk kombinasi beberapa media feature yang hanya wajib salah satunya
  - not untuk menyatakan tidak boleh memenuhi aturan media feature yang ditentukan

# KODE : CSS MEDIA QUERY

```
# media-query.css > {} @media screen and (min-width: 500px)
1  @media all {
2      .container {
3          display: grid;
4          column-gap: 10px;
5          row-gap: 10px;
6      }
7      .content {
8          background-color: aqua;
9      }
10 }
11
12 @media screen and (min-width: 10px) and (max-width: 500px) {
13     .container {
14         grid-template-columns: repeat(1, 1fr);
15     }
16 }
17
18 @media screen and (min-width: 500px) {
19     .container {
20         grid-template-columns: repeat(4, 1fr);
21     }
22 }
```

## KODE : HTML MEDIA QUERY

```
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Media Query CSS</title>
7 <link rel="stylesheet" href="media-query.css">
8 </head>
9 <body>
10   <div class="container">
11     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
12     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
13     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
14     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
15     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
16     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
17     <div class="content">Lorem, ipsum dolor sit amet consectetur adipisicing
18   </div>
19 </body>
20 </html>
```

## REFERENSI

- Berikut contoh Media Feature untuk ukuran-ukuran layar dan perangkat yang banyak digunakan :  
[https://www.w3schools.com/css/css\\_rwd\\_mediaqueries.asp](https://www.w3schools.com/css/css_rwd_mediaqueries.asp)

DISPLAY NONE

# DISPLAY NONE

- Sebelumnya, di materi Display kita bahas sedikit bahwa atribut display bisa memiliki value none (dihapus / dihilangkan)
- Pertanyaannya, untuk apa kita membuat elemen, tapi harus dihilangkan?
- Sebenarnya banyak kegunaannya, contohnya di materi ini, kita akan coba membuat Menu Bar memanfaatkan Display None

## KODE : HTML MENU BAR

```
</head>
<body>
  <div class="menubar">
    <button class="menu">Social Media</button>
    <div class="menulist">
      <a href="#">Facebook</a>
      <a href="#">Instagram</a>
      <a href="#">Tiktok</a>
      <a href="#">Youtube</a>
    </div>
  </div>
  <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsum na
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Quis ad
</body>
</html>
```

## KODE : MENU BAR CSS

```
<title>Menu Bar</title>
<style>
  .menubar {
    display: inline-block;
  }
  .menu {
    background-color: aqua;
    color: black;
    padding: 10px;
    border: none;
    cursor: pointer;
  }
  .menulist {
    display: none;
    position: absolute;
    background-color: aqua;
    z-index: 1;
  }
}
```

```

  .menulist a {
    color: black;
    text-decoration: none;
    display: block;
    padding: 10px;
  }
  .menulist a:hover {
    background-color: bisque;
  }
  .menubar:hover .menu {
    background-color: bisque;
  }
  .menubar:hover .menulist {
    display: block;
  }
</style>
```



PENUTUP

# BELAJAR APA LAGI?

- Belajar JavaScript
- Latihan membuat halaman web dari yang sederhana sampai yang kompleks menggunakan HTML dan CSS
- Belajar Library CSS
  - Bootstrap : <https://getbootstrap.com/>
  - Tailwind CSS : <https://tailwindcss.com/>