

**LAPORAN RESMI**  
**PRAKTIKUM DESAN WEB**

**DOKUMENTASI**  
**TAILWIND CSS**



**NIM** : 21104410049  
**NAMA** : Viery Nugroho  
**JURUSAN** : Teknik Informatika  
**KELAS** : TI 3B  
**TGL. PRAKTEK** : Senin, 16 Januari 2023

**JURUSAN TEKNIK INFORMATIKA**  
**FAKULTAS TEKNOLOGI INFORMASI**  
**UNIVERSITAS ISLAM BALITAR**  
**2023**

LEMBAR PERSETUJUAN

**DOKUMENTASI**  
**TAILWIND CSS**

**NIM** : 21104410049  
**NAMA** : Viery Nugroho  
**JURUSAN** : Teknik Informatika  
**KELAS** : TI 3B  
**TGL. PRAKTEK** : Senin, 16 Januari 2023

Disetujui,  
Blitar, 16 Januari 2023  
Dosen

Mohammad Faried Rahmat, S.ST., M.Tr.T  
NIDN.

## Tailwind CSS



**Tailwind CSS** adalah framework CSS open source. Tailwind CSS bekerja dengan memindai semua file HTML, komponen JavaScript, dan template lain untuk nama kelas, menghasilkan gaya yang sesuai, lalu menuliskannya ke file CSS statis. Tailwind CSS bersifat cepat, fleksibel dan andal tanpa runtime. Sama seperti framework CSS lain seperti bootstrap, tailwind juga menggunakan nama class sebagai penggunaan setiap utilitas classnya untuk membangun antar muka kustom dengan cepat.

Informasi selengkapnya bisa dilihat pada halaman resmi TailwindCSS [disini](#).

## Instalasi Tailwind CSS

**Instalasi Tailwind CSS** bisa dilakukan dengan beberapa cara, yaitu:

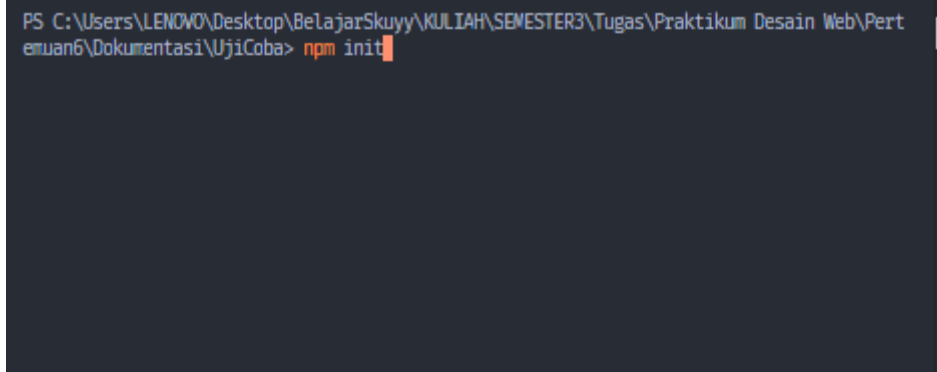
1. Tailwind CLI
2. Menggunakan PostCSS
3. Framework Guides
4. Menggunakan CDN

Contoh instalasi TailwindCSS bisa dilihat [disini](#).

## Install dan Setup Tailwind

### 1. Membuat package.json

Lakukan perintah **npm init** untuk membuat file **package.json**. Kemudian isikan pada bagian yang sekiranya diperlukan.



```
PS C:\Users\LENDOW\Desktop\BelajarSkuyy\KULIAH\SEMESTER3\Tugas\Praktikum Desain Web\Pertemuan6\Dokumentasi\UjiCoba> npm init
```

Gambar 1 npm init

Jika sudah maka akan terdapat file baru bernama **package.json**.



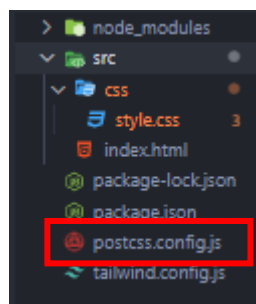
Gambar 2 package.json

### 2. Install TailwindCSS using PostCSS

Masukkan perintah **npm install -D tailwindcss postcss autoprefixer** untuk menginstall tailwind dan postcss serta autoprefixer.

### 3. Tambahkan TailwindCSS pada konfigurasi file PostCSS

Buat file baru bernama **postcss.config.js** untuk membuat konfigurasi pada postcss. Jika sudah masukkan kode berikut pada file **psotcss.config.js**



Gambar 3 postcss.config.js

```
Module.exports = {
  Plugin: {
    tailwindcss: {},
    autoprefixer: {}
  }
}
```

4. Konfigurasi template path pada file `tailwind.config.js`  
Masukkan perintah **`npx tailwindcss init`** untuk membuat file **`tailwind.config.js`**.  
Jia sudah masukkan kode berikut:

```
Module.exports = {
  content: ["/src/**/*.html,js"],
  theme:{
    extend:{},
  },
  Plugin:[],
}
```

5. Tambahkan variable tailwind pada file CSS `style.css`  
Masukkan kode berikut pada file **`style.css`**

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

6. Tambahkan script untuk mem-build tailwind  
Pada file **`package.json`** tambahkan script untuk menjalankan tailwind css.

```
"scripts": {
  "dev": "tailwindcss -i ./src/css/style.css -o ./public/css/tailwind.css --watch"
}
```

- **Build-tailwind**: nama perintah untuk menjalankan tailwind
- **`./src/css/styles.css`**: nama file css yang telah dibuat sebelumnya
- **`./public/css/tailwind.css`**: nama file css tailwind

7. Menjalankan tailwind  
Untuk menjalankan tailwind lakukan perintah berikut untuk mem-build tailwind pada proyek kita. Dan jangan lupa menambahkan link ke `tailwind.css` pada file html kita.

**`npm run dev`**

```
> ujicoba@1.0.0 dev
> tailwindcss -i ./src/css/style.css -o ./public/css/tailwind.css --watch

Rebuilding...
Done in 369ms.
```

Gambar 4 build tailwind

# HOMEPAGE SEKOLAH KODING

## 1. Bagian 1

Menggunakan framework TailwindCSS sangat memudahkan developer dalam pembuatan sebuah website. Dengan memanfaatkan utilities class yang disediakan developer mampu membuat web dengan lebih praktis tanpa perlu menuliskan sintaks CSS secara manual.

Di tailwind sering dijumpai istilah sumbu X dan Y seperti pada contoh class my-2 atau px-2. Dimana X merujuk ke sumbu horizontal (kanan-kiri) dan Y merujuk ke sumbu vertical (atas-bawah).

Dan berikut ini merupakan beberapa class utilities yang digunakan dalam membangun halaman homepage.

Class Utilities	Fungsi
p	Mengatur Padding (p, pt, pb, pl, pr, px, py)
M	Mengatur Margin (m, mt, mb, ml, mr, mx, my)
Flex	Display flex
Justify-center/between/around/evenly	Mengatur letak elemen ke arah horizontal
Items-center/start/end	Mengatur letak elemen ke arah vertikal
Rounded	Membuat elemen menjadi bulat
bg	Mengatur background
Bg-gray-500	500 di sini merupakan ketebalan background
Text-xs	Mengatur ukuran text (text-xs, text-sm, text-md, text-xl)
Text-white	Mengatur warna text (text-(color))

w	Mengatur width
h	Mengatur height
Lebih lengkap kunjungi dokumentasi resmi tailwind <a href="#">disini</a>	

## 2. Bagian 2

Konsep **responsive** pada tailwind menggunakan Breakpoint prefix. Konsep ini akan sering digunakan ketika ingin membuat tampilan website yang berbeda pada saat ukuran tertentu.

Untuk mengatur lebar pada point tertentu biasanya digunakan sintaks **breakpoint:size** atau sebagai contoh **md:w-4/6**.

Breakpoint Prefix	Minimum width
sm	640px
md	768px
lg	1024px
xl	1280px
2xl	1536px
Penjelasan lebih lanjut kunjungi dokumentasi tailwind <a href="#">disini</a>	

Konsep **hover** adalah aksi dimana cursor terletak di atas elemen. Dengan tailwind penggunaan metode hover sangat mudah, yaitu dengan menggunakan sintaks **hover:aksi** atau sebagai contoh **hover:bg-red-500**.

Konsep **fokus** adalah aksi dimana elemen tertentu sedang digunakan. Dengan tailwind penggunaan metode fokus hampir sama dengan hover, yaitu dengan menggunakan sintaks **focus:aksi** atau sebagai contoh **focus:outline-none**.

### 3. Bagian 3

Tailwind mendukung developer untuk melakukan kustomisasi pada class ataupun membuat class sendiri. Untuk membuat kustomisasi pada tailwind dapat dilakukan pada file **tailwind.config.js**. Sebagai contoh untuk kustomisasi warna tambahan pada tailwind dapat dilakukan pada baris **theme**.

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ['./src/**/*.{html,js}'],
  theme: {
    extend: {
      // Kustomisasi Tailwind
      colors: {
        mycolor: {
          DEFAULT: '#7b99b9',
          dark: '#517dad',
        },
      },
    },
  },
  plugins: [],
};
```

Pada baris kode diatas memasukkan warna kustom pada baris kode extend theme. Di letakkan pada baris extend karena disini kita hanya ingin menambahkan keyword warna baru pada proyek website kita. Jika sudah rebuild ulang tailwind untuk melakukan perbaruan pada tailwind.

Untuk penggunaannya sama dengan warna pada tailwind biasa. Sebagai contoh perhatikan baris kode dengan keyword mycolor seperti yang telah dibuat di atas

```
<button class="mt-2 w-full text-center bg-mycolor hover:bg-mycolor-dark text-white p-2 text-large rounded-md">Daftar</button>
```



## 4. Bagian 4

Dalam menggunakan tailwind mungkin saja developer menemui kesulitan dimana harus menuliskan banyak kode untuk setiap elemen. Sebagai contoh untuk membuat **button** harus menambahkan beberapa utilities class tailwind. Dengan menggunakan konsep fungsi **apply** pada tailwind akan memudahkan developer dalam menulis kode agar lebih optimal.

Penggunaan konsep **apply** pada tailwind saya kira hampir mirip seperti pada pembuatan class dan css pada umumnya. Penulisan **apply** tailwind terjadi pada file main css pada project atau satu file pada import variable pada tailwindnya. Sebagai contoh perhatikan penjelasan berikut.

Jika ingin membuat button tanpa menggunakan konsep apply maka akan menggunakan beberapa class pada satu buah element.

```
<a href="" class="text-sm text-white py-1 px-6 rounded-md bg-red-400">Google</a>
      <a href="" class="text-sm text-white py-1 px-6 rounded-md bg-gray-400">Github</a>
      <a href="" class="text-sm text-white py-1 px-6 rounded-md bg-blue-400">Twitter</a>
```

Dengan menggunakan **apply** baris kode tersebut bisa terlihat lebih rapi. Cara penggunaannya sebagai berikut.

Pada file utama css buat sebuah selector yang akan digunakan sebagai nama keyword class lalu masukkan keyword **@apply** dan diikuti nama class tailwind yang digunakan sebelumnya pada bagian deklarator. Dalam penggunaan **apply** sebisa mungkin hindari penggunaan nama class yang terlalu spesifik. Sebagai contoh perhatikan baris kode berikut.

```
@tailwind base;
@tailwind components;
@tailwind utilities;

.button {
  @apply text-sm text-white py-1 px-6 rounded-md;
}
```

Jika sudah cara penggunaannya sama seperti halnya penggunaan class tailwind pada umumnya. Sebagai contoh pada baris class sebelumnya baris kode terlihat sangat Panjang, dan setelah penggunaan metode **apply** baris kode akan terlihat lebih rapi.

```
<a href="" class="button bg-red-400 hover:bg-red-500">Google</a>
      <a href="" class="button bg-gray-700 hover:bg-
gray-900">Github</a>
      <a href="" class="button bg-blue-600 hover:bg-
blue-800">Twitter</a>
```

## 5. Bagian 5

Penggunaan **group hover ke child** sama seperti konsep css position relative dan absolute. Dimana kedua metode tersebut saling berhubungan. Sama halnya konsep **group hover to child** pada parent element harus menerapkan class **group** untuk menandai bahwa class tersebut berada pada parent dari elemen HTML. Dan pada child class nya untuk menggunakan hover yaitu dengan menggunakan class **group-hover:aksi**. Sebagai contoh **group-hover:text-gray-900**. Karena property pada tailwind masih terbatas, maka jika ingin menambahkan sebuah property baru bisa ditambahkan pada file **tailwind.config.js** pada bagian variants lalu extend.

Sebagai contoh:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ['./src/**/*.html', './src/**/*.js'],
  theme: {
    extend: {
      // Kustomisasi Tailwind
      colors: {
        mycolor: {
          DEFAULT: '#7b99b9',
          dark: '#517dad',
        },
      },
    },
  },
}
```

```
    },  
    variants: {  
      extends: {  
        divideColor: ['group-hover']  
      }  
    }  
  },  
  plugins: [],  
};
```

Setiap melakukan penambahan atau kustomisasi pada tailwind lakukan re build untuk melihat perubahan yang terjadi.

## 6. Bagian 6

Responsive menu mempunyai konsep yang sama pada responsive pada tailwind yang sudah dijelaskan. Namun pada responsive bagian menu terdapat beberapa trik untuk mengakali tampilan website yang berbeda pada setiap device. Dengan penggunaan sintaks yang sama dan diikuti dengan pengaturan display pada elemen HTML hal itu dapat digunakan untuk mengatur tampilan website antara device satu dengan lainnya.

# Output:

## Desktop



## Mobile



**Keseluruhan Source Code dari penjelasan di atas dapat diunduh pada link github disini.**