Curso de Tailwind CSS

# ¿Qué es Tailwind CSS?

**¿**[**Qué es TailWind CSS**](https://tailwindcss.com/)**?** 🧐  
Es un *[framework](https://en.wikipedia.org/wiki/Software_framework" \t "_blank)* que permite la contrucción de diseños altamente personalizados y de bajo nivel.

**¿**[**Por qué usar TailWind CSS**](https://tailwindcss.com/#what-is-tailwind)**?** 🤨  
A diferencia de otros frameworks css, como lo es [bootstrap](https://getbootstrap.com/" \t "_blank), que otorgan componentes prediseñados, Tailwind provee clases a bajo nivel de css que nos permiten construir diseños completamente personalizados por nostros y sin tener que pelear con el framework en sí.

Nos externa una [responsividad](https://tailwindcss.com/docs/responsive-design/" \t "_blank) desde la misma sintaxis.

Trabaja de manera amigable en el crecimiento de nuestro proyecto al proveernos herramientas para [extraer clases](https://tailwindcss.com/docs/extracting-components/).

Pero sobre todo, nos abre su código para [personalizarlo](https://tailwindcss.com/docs/configuration/) ya que esta escrito en [PostCSS](https://postcss.org/" \t "_blank) y configurado en JavaScript.

Tailwind is more than a CSS framework, *it’s an engine for creating design systems*.

# ¿Qué es Tailwind CSS?

Dejo por aquí unos apuntes que cree para iniciar la configuración desde 0, por si le sirve a la comunidad.

## Dependencias

- `npm init -y`

- `npm install tailwindcss autoprefixer postcss-cli`

### Inicializamos las herramientas instaladas:

<!-- Genera archivo configuracion vacio de nombre tailwind.config.js -->

- `npx tailwindcss init`

<!-- Genera archivo configuracion completo -->

- `npx tailwindcss init tailwind.config.full.js --full`

<!-- Plugin recomendado para VSCode: Tailwind CSS IntelliSense -->

### Creamos archivo de configuracion postcss.config.js

<!-- Instrucciones archivo postcss.config.js: -->

- `touch postcss.config.js`

- `module.exports = { plugins: [require('tailwindcss'), require('autoprefixer')], };`

### Creacion archivo html y origen CSS

- `mkdir css`

- `touch css/tailwind.css`

### Configuracion archivo css/tailwind.css

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

### Completamos script en package.json

- `"scripts": { "build": "postcss css/tailwind.css -o public/css/styles.css",`

<!-- Para autoregenerar el tailwind.css cuando creamos paquetes -->

- `"dev": "postcss css/tailwind.css -o public/css/styles.css --watch"}`

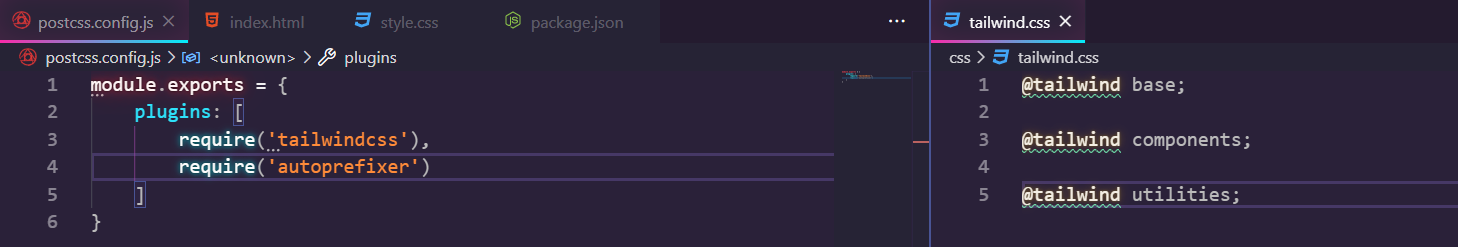
<!-- Ejecutar para compilar -->

- `npm run dev`

### Inicializamos script para crear el css

<!-- genera una directorio css con su styels.css en la carpeta public -->

- `npm run build`



# Directivas de Tailwind

Directiva es una instrucción que utiliza tailwind para insertar código en el archivo final de css que genera.

@tailwind base  
Esto inyecta los estilos base de Tailwind y cualquier estilo base registrado por plugins.

@tailwind components  
Esto inyecta las clases de componentes de Tailwind y cualquier clase de componente registrado por los plugins.

@tailwind utilities  
Esto inyecta las clases de utilidad de Tailwind y cualquier clase de utilidad registrada por los plugins.

@tailwind screens  
Esta directiva sirve para controlar donde Tailwind inyecta las variaciones responsivas de cada utilidad. Si se omite, Tailwind añadirá estas clases al final de tu css, por defecto.

# Personalización y configuración

## Configuración

Ya que Tailwind es un framework para construir UI a la medida, por default, se tiene un archivo opcional llamado tailwind.config.js en la raíz de la carpeta, donde está el package.json.

## Creando un archivo de configuración

Para generar un archivo de configuración para Tailwind, podemos usar el Tailwind CLI:

npx tailwind init

**Nota**. Podemos utilizar esta herramienta cuando instalamos la dependencia via npm.  
Donde como resultado tendremos tailwind.config.js:

module.exports = {

theme: {},

variants: {},

plugins: [],

}

Cabe mencionar que cada sección, del archivo de configuración, es opcional.

## La sección Theme

Esta sección es donde definimos los aspectos relacionados con el diseño visual de nuestro sitio.

...

theme: {

screens: {

sm: '640px',

md: '768px',

lg: '1024px',

xl: '1280px',

},

fontFamily: {

display: ['Gilroy', 'sans-serif'],

body: ['Graphik', 'sans-serif'],

},

borderWidth: {

**default**: '1px',

'0': '0',

'2': '2px',

'4': '4px',

},

extend: {

colors: {

cyan: '#9cdbff',

},

spacing: {

'96': '24rem',

'128': '32rem',

}

}

}

...

## La sección Variants

Esta sección nos permite controlar el comportamiento de las utilidades core, como responsive variants y pseudo-class variants.

...

variants: {

appearance: ['responsive'],

// ...

borderColor: ['responsive', 'hover', 'focus'],

// ...

outline: ['responsive', 'focus'],

// ...

zIndex: ['responsive'],

},

...

## La sección Plugins

Esta sección nos permite registrar plugins de terceros con el objetivo de extender utilidades, componentes, estilos, etc.

...

plugins: [

require('tailwindcss-transforms'),

require('tailwindcss-transitions'),

require('tailwindcss-border-gradients'),

],

...

# Responsive Design, Mobile First y Utility First

**Responsive Web Design**

La web como la conocemos, hoy en día, no es una tecnología pensando en un **UX** o *User Experience* generando, desafortunadamente, que muchos de los sitios web no estén optimizados para los **dispotivos móviles**.

Definido por [Ethan Marcotte](https://alistapart.com/article/responsive-web-design/" \t "_blank), es una filosofía que responde a las necedades de los usuarios y a los dispositivos que estamos usando.

**Mobile First**

Como su nombre sugiere, significa que iniciaremos con el diseño de móviles y expandiendo éstas características para crear una verión en tableta o escritorio/web tradicional.

Cabe mencionar que esta filosofía no es sinónimo de limitación, por lo que tenemos que tener el mismo contenido tanto en escritorio como en móvil. Google describe las *[best practices](https://developers.google.com/search/mobile-sites/mobile-first-indexing" \t "_blank)* en su sitio.

**Utility first**

CSS posee diferentes tipos de paradigmas para abstraer un diseño, como BEM descrito por [Tailwind](https://tailwindcss.com/docs/utility-first/" \t "_blank) en ésta sección:

# Colores

Los elementos que pueden ser afectados por los colores son:

* Fondo.
* Texto.
* Bordes.
* Placeholder.

Taildwind por default tiene valores ya predefinidos, para conocerlos tenemos que generar un archivo de configuración con todos los valores completos:  
npx tailwindcss init tailwind.config.full.js --full

Se puede ver la paleta de colores por defecto aquí:  
<https://tailwindcss.com/docs/customizing-colors/#default-color-palette>

# Dimensiones y Espacios

Tailwind utiliza un sistema de espacios que se puede configurar. Se maneja internamente utilizando rem. Se puede visualizar desde el archivo tailwind.config.full.js.

Se puede utilizar en:

* Height.
* Width.
* Margin.
* Padding.

En las clases de los elementos se añaden cosas como:  
h-32 w-1/2 pt-2 mx-auto

Para manajear el height:  
<https://tailwindcss.com/docs/height/#app>

Para manejar width en porcentajes se puede ver esta documentación:  
<https://tailwindcss.com/docs/width/#app>

