TAILWIND CSS - Un framework CSS de utilidades básicas

Es un "framework CSS que prioriza las utilidades" que proporciona varias de estas clases de un solo propósito que puedes utilizar directamente dentro de tu marcado para diseñar un elemento.

Desde el cmd, tenemos que tener el Node.js instalado, nos vamos a la carpeta donde tenemos todos los proyectos web creados. Para instalarlo:

```
npm install -D tailwindcss
npx tailwindcss init
```

Con Ctrl+J puedo abrir el terminal y lanzar también los comandos, si da error teclear:

```
set-executionpolicy -scope currentuser unrestricted
```

Se crea un fichero tailwind.config.js y añadimos la ruta donde tenemos los proyectos:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./webs/**/*.{html,js}"],  /* webs es una carpeta en el nivel de tailwind.config.js */
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Se añade los @tailwind a un fichero principal css dentro de la carpeta indicada en el fichero anterior, por ejemplo, webs/input.css:

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

Se vuelve a la carpeta del proyecto en el cmd, y se importa el archivo

```
npx tailwindcss -i ./webs/input.css -o ./webs/output.css --watch (no se cierra aunque se puede ocultar, para parar de escuchar Ctrl + C)
```

Y el fichero se crea en tal carpeta, webs/index.html, añadiendo el fichero output.css recién creado:

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="./output.css" rel="stylesheet">
</head>
<body>
    <h1 class="text-3xl font-bold underline"> /* Tamaño texto, negrita, subrayado */
        Hello world!
    </h1>
</body>
</html>
```

Colores

1) Agregamos Tailwind CSS Intellise para que nos ayude a completar los valores, por ejemplo, color de fondo, bg- (nos muestra todos los colores posibles). En la página de tailwind podemos ver todos los colores (customizing-colors).

2) Podemos poner otros colores con el uso RGB, text-[#d2d255].
3) Podemos definir nuevos colores, para ello lo introducimos en tailwind.config.js:

```
extend: {
    colors: {
        celeste:'#243cff',
    }
},
```

Y se usaría con → text-celeste /* al poner el ratón indica el código CSS */, para darle opacidad podemos usar → text-celeste/50

Gradientes

Se define indicando el color de inicio y el final, también se puede usar colores intermedios. La clase a usar .bg-gradient-to-r:

"bg-gradient-to-r from-celeste to-green border-red-500 border-2"

Si queremos que el texto coja color de un gradiente, hay que definir el gradiente, ponerlo al final de la caja que contiene el texto con bg-clip-text y poner el texto transparente:

```
<div class="text-4xl font-extrabold">
    <span class="bg-clip-txt text-transparent bg-gradient-to-r from-blue to-green">
        Texto
    </span>
</div>
```

Otros elementos

Padding → p- (número, en píxeles se multiplica por 4)
Border radius → rounded- (número)
Centrado → mx-auto
Para convertir elementos en bloque → block
Margin vertical → my- (número)

Medidas

1. El ancho lo podemos definir con valores ya predefinidos con la ayuda del Tailwind CSS Intellise → w- (valores definidos)
2. O sin los valores definidos → w-[170px]
3. O ponerlo en el archivo de configuración de tailwind.config.js:

```
extend: {
    colors: {
        'azul-claro':'#243cff',
    },
    spacing: {          → tanto ancho como alto, si solo se quiere ancho se usa width
        '42':'170px',
    }
},
```

Y para usarlo, w-42

4. Con la altura se hace de la misma forma, pero la clase ahora es h- (de height)
5. También se puede usar fracciones para indicar una porción de la pantalla
6. Con h-screen se refiere a toda la pantalla
7. Y con h-full al 100% del contenedor

```
<div class="h-[100px]">
        <div class="bg-red-400 h-full">Texto</div>
</div>
```

Estados

1) hover → al poner el ratón encima, hover:bg-blue-400
2) disabled → deshabilitado, se pone la propiedad y después en la clase se puede cambiar el color, disabled:bg-red-200
3) focus → focus:outline-none (sin borde) focus:ring-1 (grosor) focus:ring-purple-600
4) invalid → para valores invalidos, invalid:focus:ring-red-400
   Si queremos que aparezca mensaje cuando lo introducido es incorrecto, ponemos un texto oculto, que aparecerá cuando el valor sea incorrecto:
   `<p class="text-red-400 hidden peer-invalid:block">Correo incorrecto</p>`
   Cambiamos márgenes y padding para que quede bonito y añadimos en la clase del input la clase peer.

Pseudoclases

1. after:content-['Hola'] → texto que aparece después del elemento donde se está colocando la pseudoclase
   after:absolute → posición, si se pone relative coge la referencia del elemento donde se está definiendo
   after:top-0 → coge de referencia el body
   after:left-32 → se pone a la izquierda del elemento, puede ser otro número
   after:border-8 → si se quita el texto, aparece un borde grueso de tono gris
   after:border-l-orange-600 (sale un triángulo)
   after:border-transparent (tono transparente)
2. placeholder → relleno de una caja de texto en un formulario. Podemos cambiar color o formato de texto
   placeholder:text-orange-600 placeholder:italic
3. file → para definir ficheros, las clases serían del tipo file:rounded-lg …
4. marker → relacionado con las listas tanto ordenadas como desordenadas
   `<ul class="list-disc ml-6 marker:text-red-500">`
   `<ol class="list-decimal marker:font-bold">`
5. selection → puede cambiar color a la hora de seleccionar una parte de un texto
   `<p class="selection:bg-green-500">`
6. first-line first-letter → first-line:uppercase first-line:tracking-widest (espaciado mayor) first-letter:float-left (se ubica a la izquierda)

Diseño responsivo

Al igual que en Bootstrap se usa en Tailwind CSS las distintas medidas de las pantallas,

```
<body class="sm:bg-amber-200 md:bg-blue-200">
```

Si se quiere saber el tamaño de las pantallas, se puede poner en la parte superior, seleccionando en la parte inferior inspeccionar. Se puede aplicar más elementos en otras etiquetas

```
<h2 class="text-xl font-bold sm:text-2xl">
```

Podemos personalizar puntos de quiebre en el fichero de configuración, taildwind.config.js,

```
extend: {
    screens: {
        'tablet':'900px',
    },
```

Flexbox

En input.css se puede definir componentes para después usar en html, por ejemplo,

```
@Layer components {
    .card {
        @apply bg-red-500 w-40 h-40 grid place-content-center rounded-lg text-white
        font-bold border-2 border-red-600 text-4xl;
    }
}
```

Ya en HTML

```
<div class="flex place-content-around">
    <div class="card">1</div>
</div>
```

Más clases que se pueden usar → flex-col[-reverse], place-content-between. Las cajas se pueden alinear dentro de su contenedor → items-end, items-center, pero se puede alinear sólo un elemento → self-center, self-end

Con flex-wrap las cajas no cambian de tamaño para ajustarse, sino que cambian de fila. A los elementos hijos le podemos indicar que crezcan si hay espacio para ello con .grow, y si no tiene espacio que mantenga su forma y se hagan más pequeños con .shrink-0

A cada contenedor le puedo dar un tamaño respecto a la página con .basis-1/4, .basis-1/2, …

Grid

```
<div class="grid grid-cols-4 gap-4"> /* indico número de columnas y espaciado */
    <div class="card">1</div>
    ...
</div>
```

Podemos indicar a un hijo que se expanda dos columnas desde donde se encuentra → col-span-2. También se le puede indicar que empiece en un sitio determinado y que termine en otro → col-start-2 col-end-4 = col-[2/4]

Al igual con filas.

Dark

El usuario puede indicar al navegador que quiere visualizarlo de modo oscuro, parte superior derecha – Configuración – Apariencia. En el HTML se puede usar

```
<body class="dark:bg-slate-900">
```

Se le puede dar la posibilidad al usuario que lo elija. En el fichero de configuración tailwind.config.js

```
module.exports = {
        darkMode:'class',
        ...
}
```

En el HTML se activa en la parte superior

```
        <html lang=”es” class=”dark”>
```

dándole al usuario la posibilidad de elegirlo

```
    <label for=”darkmode” class=”dark:text-white”>Dark Mode</label>
    <input type=”checkbox” id=”darkmode”>
    <script src=”./main.js”></script> /* Lanza un ejecutable */
```

Y en el main.js

```
    const darkModeInput = document.querySelector('#darkmode');
    darkModeInput.addEventListener('click',()=>{
       document.documentElement.classList.toggle('dark'); /* Añade o quita la clase dark */
    })
```