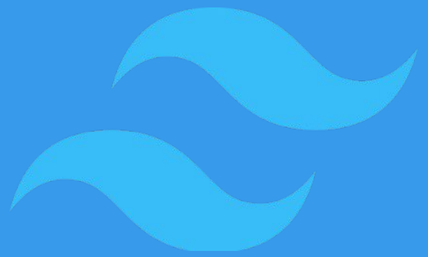
DGCAL

2022



tailwindcss

Crea interfacce moderne per il Web con il solo codice CSS necessario

In collaborazione con Edgemony

DGCAL SRL

Marco Ingraiti

Software Developer con 10+ anni di esperienza.
Mi occupo di progettazione e realizzazione di soluzioni per il Web, integrazione tra sistemi ed infrastrutture cloud.
La mia esperienza mi permette di poter scegliere lo stack tecnologico più efficiente per la realizzazione del progetto ed il suo mantenimento.

A cosa serve Tailwind

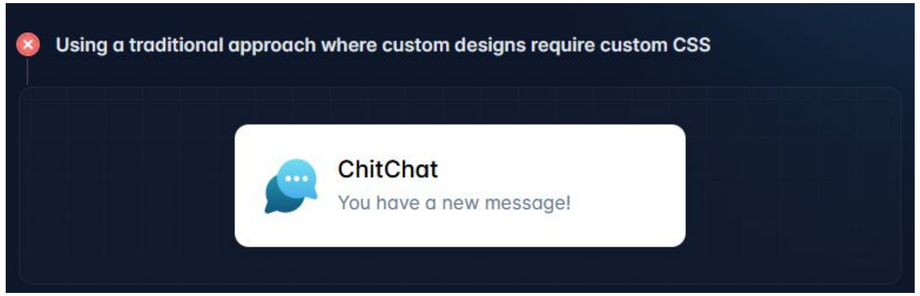
Tailwind è un Framework CSS che utilizza un approccio moderno, riusabile e versatile per costruire interfacce per il Web.

L’approccio di Tailwind consiste nel dichiarare direttamente nelle classi del codice HTML che tipo di proprietà si vogliono applicare al codice, grazie a delle classi predefinite e dei “modificatori condizionali” si possono per esempio gestire:

- Breakpoint diversi
- Stati e pseudoclassi
- Dark mode

Old vs new

Approccio tradizionale



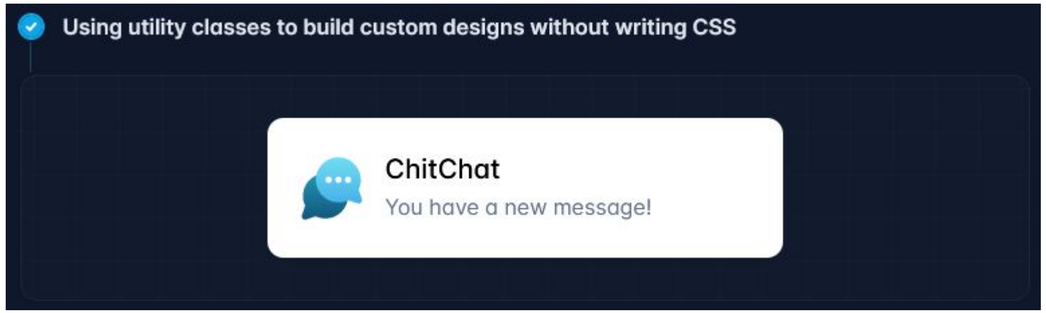
- ➔ Codice ripetitivo e ridondante
- ➔ Senza preprocessori c’è il rischio di inserire valori non omogenei
- ➔ File .css infiniti
- ➔ Perdita di tempo per pensare i nomi giusti per le classi

```
<div class="chat-notification">
  <div class="chat-notification-logo-wrapper">
    <img class="chat-notification-logo" src="/img/logo.svg" alt="ChitChat Logo">
  </div>
  <div class="chat-notification-content">
    <h4 class="chat-notification-title">ChitChat</h4>
    <p class="chat-notification-message">You have a new message!</p>
  </div>
</div>

<style>
  .chat-notification {
    display: flex;
    max-width: 24rem;
    margin: 0 auto;
    padding: 1.5rem;
    border-radius: 0.5rem;
    background-color: #fff;
    box-shadow: 0 20px 25px -5px rgba(0, 0, 0, 0.1), 0 10px 10px -5px rgba(0, 0, 0, 0
  }
  .chat-notification-logo-wrapper {
    flex-shrink: 0;
  }
  .chat-notification-logo {
    height: 3rem;
    width: 3rem;
  }
  .chat-notification-content {
    margin-left: 1.5rem;
    padding-top: 0.25rem;
  }
  .chat-notification-title {
    color: #1a202c;
    font-size: 1.25rem;
    line-height: 1.25;
  }
  .chat-notification-message {
    color: #718096;
    font-size: 1rem;
    line-height: 1.5;
  }
</style>
```

Old vs new

Con Tailwind

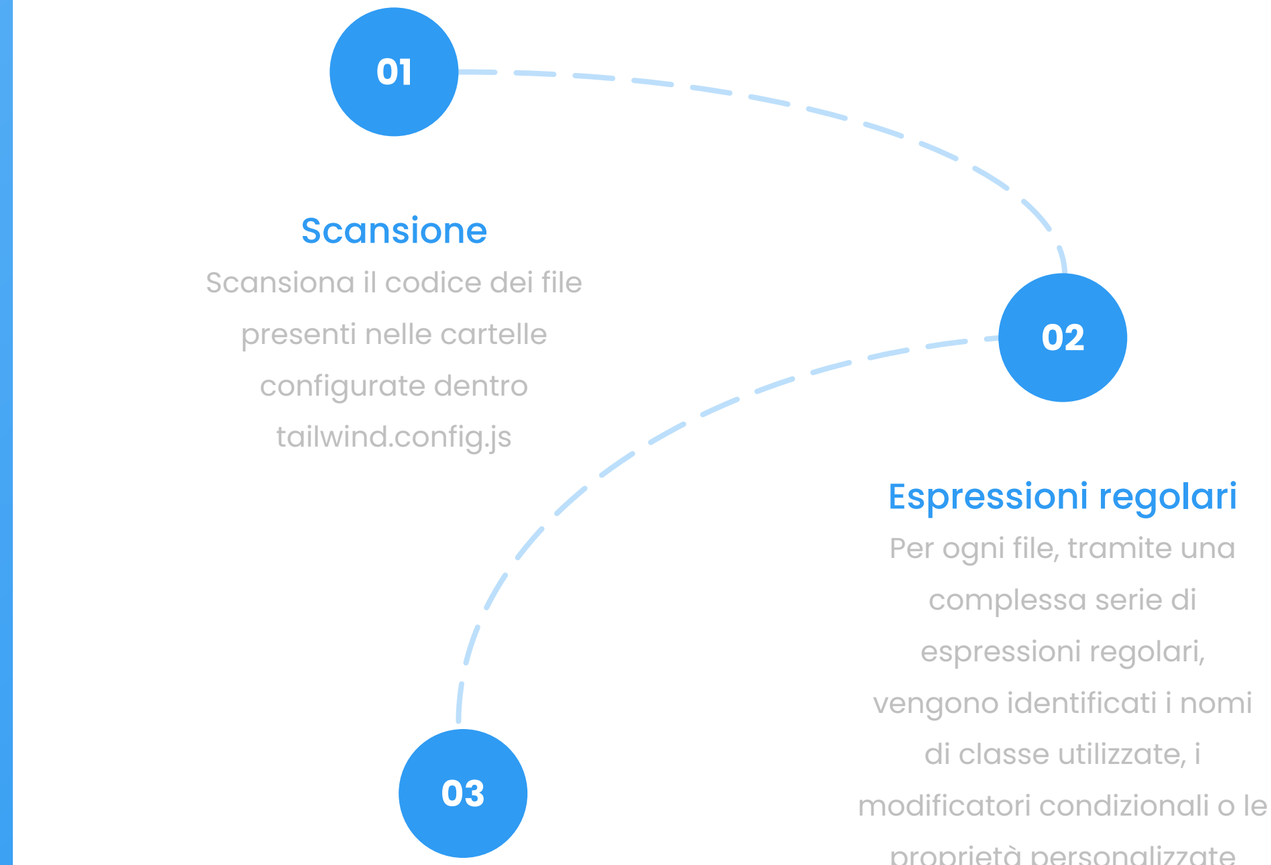



```
<div class="p-6 max-w-sm mx-auto bg-white rounded-xl shadow-lg flex items-center space-x-4">
  <div class="shrink-0">
    <img class="h-12 w-12" src="/img/logo.svg" alt="ChitChat Logo">
  </div>
  <div>
    <div class="text-xl font-medium text-black">ChitChat</div>
    <p class="text-slate-500">You have a new message!</p>
  </div>
</div>
```

- ➔ Meno codice da scrivere
- ➔ Sei vincolato ad un sistema solido di misure, palette colori, spaziature, ombre, ecc
- ➔ Scrivi Webapp più performanti grazie alla generazione di un file .css con le sole istruzioni che hai utilizzato (anzichè includere tutto il framework)
- ➔ Riutilizzo del codice e possibilità di accesso ad una libreria di componenti già pronti

Funzionamento

Come fa Tailwind a capire quale codice CSS deve generare?



Setup

Avviamo un progetto

Tailwind lo possiamo incorporare in diversi modi a seconda del nostro stack.

Come pacchetto NPM indipendente attraverso la Tailwind CLI, , con i più famosi Framework e librerie o includendolo su qualsiasi pagina HTML tramite CDN.

Vediamo come utilizzarlo con Next.js.

01.

Creazione del progetto

```
npx create-next-app my-project
cd my-project
```

02.

Installiamo Tailwind

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

03.

Configuriamo l'ambiente

```
/** @type {import('tailwindcss').Config} */
module.exports = {
 content: [
          "./pages/**/*.{js,ts,jsx,tsx}",
          "./components/**/*.{js,ts,jsx,tsx}",
 ],
 theme: {
          extend: {},
 },
 plugins: [],
}
```

04.

Aggiungiamo le direttive

```
#./styles/globals.css
@tailwind base;
@tailwind components;
@tailwind utilities;
```

05.

Avviamo il server

```
npm run dev
```

Layout, colori, spaziatura, ombre...

Panoramica delle classi

Layout

container

block, flex

justify-center

Colori

text-blue-400

text-orange-700

bg-white-100

Spaziatura

m-4, mt-12

px-6, p-2

space-x-4

Dimensioni

w-full, w-10

min-w-full

max-h-screen

Carattere

text-2xl

font-bold

text-center

Effetti

rounded-lg

shadow-md

outline outline-offset-2 outline-pink-500

Breakpoint, stati e pseudoclassi, dark mode

Modificatori condizionali

Permettono di combinare una proprietà ad una condizione specifica, aggiungendo flessibilità al codice

Breakpoints

Condizione specifica in merito alla dimensione della viewport.

Non determina una condizione specifica che “si verifica fino a” ma “si verifica a partire da…in poi” .

Questa logica permette di avere un approccio mobile-first in quanto ci costringe ad inserire le classi mobile senza prefisso per poi salire di breakpoints fino ai Desktop.

Breakpoint prefix	Minimum width	CSS
`sm`	640px	`@media (min-width: 640px) { ... }`
`md`	768px	`@media (min-width: 768px) { ... }`
`lg`	1024px	`@media (min-width: 1024px) { ... }`
`xl`	1280px	`@media (min-width: 1280px) { ... }`
`2xl`	1536px	`@media (min-width: 1536px) { ... }`

Breakpoint, stati e pseudoclassi, dark mode

Modificatori condizionali

Permettono di combinare una proprietà ad una condizione specifica, aggiungendo flessibilità al codice

Pseudoclassi

Condizione specifica in merito ad una pseudoclasse: hover, focus, active, first, last, odd, even, ecc.

Può essere utilizzato anche con le pseudoclassi after e before per aggiungere contenuto tramite CSS.

Modifier	CSS
hover	`&:hover`
focus	`&:focus`
focus-within	`&:focus-within`
focus-visible	`&:focus-visible`
active	`&:active`
visited	`&:visited`
target	`&:target`
first	`&:first-child`
last	`&:last-child`
only	`&:only-child`
odd	`&:nth-child(odd)`

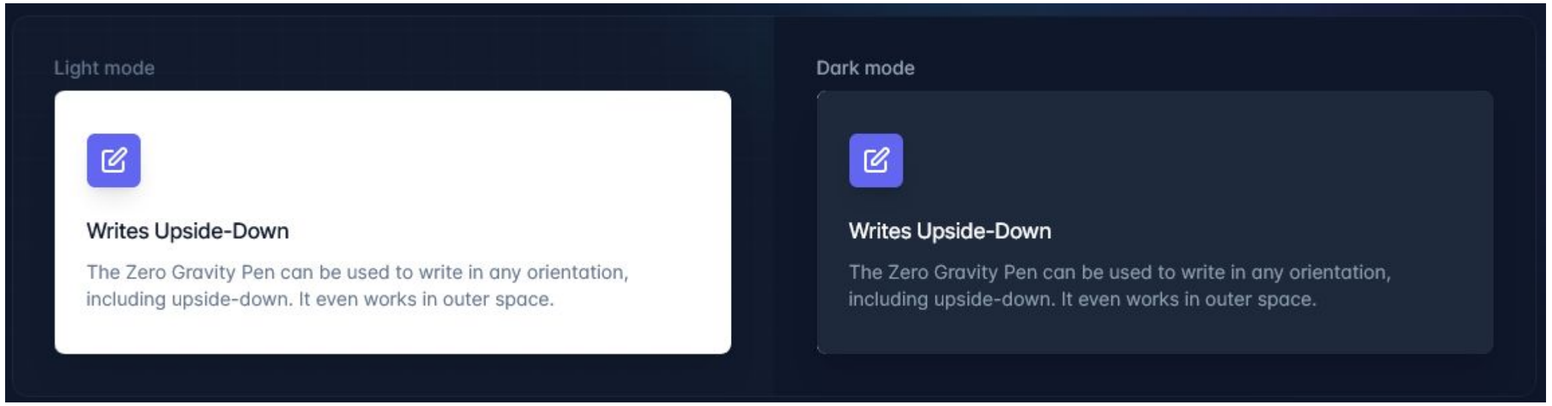
Breakpoint, stati e pseudoclassi, dark mode

Modificatori condizionali

Permettono di combinare una proprietà ad una condizione specifica, aggiungendo flessibilità al codice

Dark Mode

Permette di gestire classi specifiche in base alla funzione Dark mode del sistema operativo e del browser o per la preferenza specifica del sito web.



Personalizza il tuo Tailwind

Estensioni, valori personalizzati e direttive

Permette di creare classi personalizzate senza aggiungere overhead, personalizzare la palette colori o i breakpoint e creare una configurazione di Tailwind adatta al tuo progetto

Estensioni

Estensione e modifica della configurazione di base del file tailwind.config.js

```
module.exports = {
  theme: {
    screens: {
      sm: '480px',
      md: '768px',
      lg: '976px',
      xl: '1440px',
    },
    colors: {
      'blue': '#1fb6ff',
      'pink': '#ff49db',
      'orange': '#ff7849',
      'green': '#13ce66',
      'gray-dark': '#273444',
      'gray': '#8492a6'
```

Valori personalizzati

Puoi aggiungere valori arbitrari all'interno dei nomi di classe.

This is basically like inline styles, with the major benefit that you can combine it with interactive modifiers like `hover` and responsive modifiers like `lg`:

```
<div class="top-[117px] lg:top-[344px]">
  <!-- ... -->
</div>
```

This works for everything in the framework, including things like background colors, font sizes, pseudo-element content, and more:

```
<div class="bg-[#bada55] text-[22px] before:content-['Festivus']">
  <!-- ... -->
</div>
```

Classi di utility

Possiamo combinare classi personalizzate come insieme di classi di Tailwind per creare delle nostre utility.

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

@layer components {
  .select2-dropdown {
    @apply rounded-b-lg shadow-md;
  }
  .select2-search {
    @apply border border-gray-300 rounded;
  }
  .select2-results__group {
    @apply text-lg font-bold text-gray-900;
  }
  /* ... */
}
```

Stili di base

Possiamo applicare degli stili di base ad alcuni tag HTML aggiungendo queste direttive alla configurazione CSS.

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

@layer base {
  h1 {
    @apply text-2xl;
  }
  h2 {
    @apply text-xl;
  }
  /* ... */
}
```

Funzioni

Grazie alle funzioni possiamo accedere a dei valori di configurazione direttamente nel file CSS

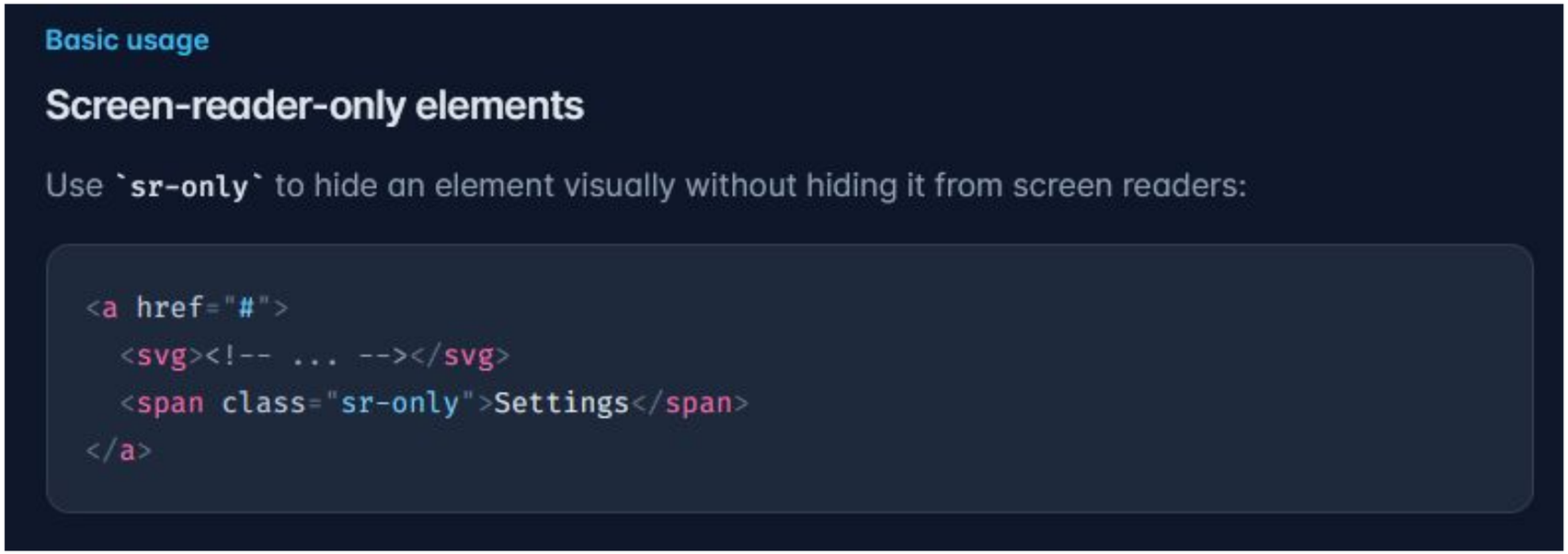
theme()

Use the `theme()` function to access your Tailwind config values using dot notation.

```
.content-area {
  height: calc(100vh - theme(spacing.12));
}
```

Accessibilità

Possiamo scegliere quale tag è read-only e renderlo leggibile solo per le impostazioni di accessibilità in lettura.



Question & Answers

Ponetemi pure qualsiasi domanda

Grazie

Scrivetemi su Linkedin per qualsiasi ulteriore approfondimento