2023

RÉCAPITULATIF DE LA FORMATION : TAILWIND CSS

WORKFLOW ENGINE PROJECT

Réalisé par :KHOUDRAJI OUIAM

SOMMAIRE

1-Qu'est-ce que Tailwind CSS ?

Tailwind CSS est un framework CSS populaire intégralement utility-first. Il fournit aux utilisateurs des classes CSS de bas niveau en PostCSS qui peuvent être utilisées pour **définir des composants et des designs de manière indépendante.**

2-Avantages de Tailwind CSS :

- Pas besoin d'énormes fichiers CSS avec CSS personnalisé
- Modifications de conception faciles à apporter directement à partir des fichiers de vue
- Meilleure expérience développeur
- Des conceptions axées sur le mobile dès le départ
- Conçu avec des valeurs par défaut pour un look et une sensation raffinés et conçus
- Facilement personnalisable avec le fichier de configuration

3-Instalation de Tailwaind CSS :

- installer Tailwind CSS et créer le fichier config 'tailwind.config.js' :
  npm install -D tailwindcss
  npx tailwindcss init
- ajouter le chemin de tout les fichier de ta template dans le fichier 'tailwind.config.js' :

```
/** @type {import('tailwindcss').Config}
*/
module.exports = {
  content: ["./src/**/*.{html,js}"],
  theme: {
    extend: {},
  },
  plugins: [],}
```

- Ajoutez les directives @tailwind pour chacune des couches de Tailwind à votre fichier CSS principal.(dans le cas de react : le fichier App.css) :

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

- Démarrer le processus de création de la CLI Tailwind :

```
npx tailwindcss -i ./src/input.css -o
./dist/output.css –watch
```

4-Les classes de Background:

cet ensemble de classes change la couleur d'arrière-plan d'un élément en utilisant une échelle de 100 à 900 pour les nuances et une palette de plus de 90 nuances.

bg-*-{100-900} {}

black, white, gray, red, orange, yellow, green, teal, indigo, blue, purple, pink

5-Le dimensionnement des éléments est réalisé à l'aide des classes w- et h- :

Tous les nombres dans Tailwind sont basés autour de l'unité de mesure rem. 1 rem est égal à la taille de la police de base du document. Par exemple, si la taille de la police de base est de 16 pixels, 1 rem est égal à 16px et nous pouvons en déduire que 1,25 rem est égal à 20px. Pour vous aider avec ces nombres fractionnaires, Les classes numérotées de Tailwind sont multipliées par 4 pour éviter d'avoir des nombres avec des décimales.

6-Système de numérotation de Tailwind :

En supposant que la taille de police de base du document est de 16px

Pixels	rem	Tailwind
0px	0 rem	0
4px	0.25 rem	1
8px	0.5 rem	2
12px	0.75 rem	3
16px	1 rem	4
20px	1.25 rem	5
24px	1.5 rem	6
32px	2 rem	8
40px	2.5 rem	10

Continue à toutes les tailles par défaut disponibles 12, 16, 20, 24, 32, 40, 48, 56, 64

7-Dimensionnement avec les classes w-* et h-* :

.w-* {}

.h-* {}

Tailles disponibles en REM:

0, 1, 2, 3, 4, 5, 6, 8, 10, 12,

16, 20, 24, 32, 40, 48, 56, 64,

Dimensionnement en pourcentages:

1/2... 1/{3, 4, 5, 6, 12)

Sizing utilities

screen, full

8-Padding et margins avec p-*, m-* , py-* et my-*:

Ces classes ajoutent un rembourrage et une marge à un élément en utilisant le système de numérotation Tailwind.

.p-* {}

.m-* {}

.p{x, y}-* {}

m{x, y}-* {}

9- Styliser du texte avec des classes utilitaires :

Le style de police est une partie importante de toute conception et Tailwind propose de nombreuses classes que nous pouvons utiliser pour styliser le texte sur notre applications. Il inclut même des classes utilitaires pour les transformations comme les majuscules.

Fonts:

.font-sans {} // Helvetica or similar

.font-serif {} // Times New Roman or similar

.font-mono {} // Monospace or similar

Sizing:

.text-xs {} // .75rem;

.text-sm {} // .875rem;

.text-base {} // 1rem;

.text-lg {} // 1.125rem;

.text-xl {} // 1.25rem;

.text-2xl {} // 1.5rem;

.text-3xl {} // 1.875rem;

.text-4xl {} // 2.25rem;

.text-5xl {} // 3rem;

.text-6xl {} // 4rem;

Text Align:

.text-left {}

.text-center {}

.text-right {}

.text-justify {}

Text Color:

.text-{color}-{shade (100-900)} {}

Styling:

.italic {}

.not-italic {}

Font Weight (Bold):

.font-hairline {} // 100

.font-thin {} // 200

.font-light {} // 300

.font-normal {} // 400

.font-medium {} // 500

.font-semibold {} // 600

.font-bold {} // 700

.font-extrabold {} // 800

.font-black {} // 900

Letter Spacing:

.tracking-tighter {} // -0.05em

.tracking-tight {} // -0.025em

.tracking-normal {} // 0

.tracking-wide {} // 0.025em

.tracking-wider {} // 0.05em

tracking-widest {} // 0.1em

Line Height/Spacing:

.leading-none {} // 1

.leading-tight {} // 1.25

.leading-snug {} // 1.375

.leading-normal {} // 1.5

.leading-relaxed {} // 1.625

.leading-loose {} // 2

Text Decorations:

.underline {}

.line-through {}

.no-underline {}

Text Transform:

.uppercase {}

.lowercase {}

.capitalize {}

.normal-case {}

10- Borders :

Ces classes coloreront, styliseront et ajouteront du rayon à n'importe quelle bordure ou coin.

.border {} // 1px

.border-0 {} // 0

.border-2 {} // 2px

.border-4 {} // 4px

.border-8 {} // 8px

Modifiers :

.border-{t, b, l, f}-*

Colors :

.border-{color}-{shade (100-900)}

Border Style :

.border-solid {}

.border-dashed {}

.border-dotted {}

.border-double {}

.border-none {}

Border Radius:

.rounded-none {} // 0

.rounded-sm {} // .125rem

.rounded {} // .25rem

.rounded-lg {} // .5rem

.rounded-full {} // 9999px

11- Display Modes :

La plupart des éléments sont des blocs d'affichage par défaut, mais cela peut facilement être modifié avec les classes d'affichage disponibles.

.block {} // block

.inline-block {} // inline-block

.inline {} // inline

.flex {} // flex

.inline-flex {} // inline-flex

.table {} // table

.table-row {} // table-row

.table-cell {} // table-cell

.hidden {} // none

12- Flexbox :

Tailwind utilise Flexbox pour la disposition des éléments sur le document. Flexbox est une propriété d'affichage css qui définit un flex container. Une fois qu'un conteneur a été affecté en tant que conteneur flexible, nous pouvons utiliser toutes les classes d'utilitaires d'alignement pour obtenir le look désiré.

.flex {}

Default direction - horizontal alignment:

.justify-start {}
.justify-center {}
.justify-end {}
.justify-between {}
.justify-around {}

Default direction - vertical alignment:

.items-stretch {}
.items-start {}
.items-center {}
.items-end {}
.items-baseline {}

Flex Direction :

.flex-row {}

.flex-row-reverse {}

.flex-col {}

.flex-col-reverse {}

Wrapping :

.flex-no-wrap {}

.flex-wrap {}

.flex-wrap-reverse {}

13- Responsive Design :

Toutes les applications modernes devraient pouvoir s'adapter de manière réactive à la taille de l'écran. Tailwind est un premier mobile framework, ce qui signifie que toutes les classes dont nous avons parlé jusqu'à présent sont destinées aux mobiles et se

succèdent jusqu'à bureau. Mais nous pouvons changer cela avec quelques modifications.

Default breakpoints:

[all] // 0px

.sm: // 640px

.md: // 768px

.lg: // 1024px

.xl: // 1280px

Default responsive classes:

.sm:bg-* {} // background color

.sm:w-* {} // width

.sm:h-* {} // height

.sm:p-* {} // padding

.sm:m-* {} // margin

.sm:font-sans {} // font family - sans, serif, mono

.sm:text-lg {} // font size - xs, sm, base, lg, xl, {2-6}xl

.sm:text-left {} // left, center, right, justify

.sm:text-{color}-{shade (100-900)} {} // text color

.sm:font-bold {} // font weight

.sm:tracking-tighter {} // letter spacing .sm:leading-tight {} // line spacing/height

.sm:uppercase {} // text transform

.sm:border-{color}-{shade (100-900)} // border color
.sm:border-{style} {} // border style .sm:border-{width} {} // border width .sm:rounded-{size} {} // border radius

.sm:{display} {} // block, inline, flex, table, hidden...

.sm:flex {} // display flex

.sm:flex-{col|row} {} // flex direction

14- Hover Modifier :

Nous pouvons puiser dans l'état de hover intégré dans CSS en utilisant le hover: modifier avec l'une des classes par défaut pour réaliser des designs interactifs.

Default responsive classes:

.hover:bg-* {} // background color

.hover:text-{color}-{shade (100-900)} {} // text color

.hover:font-bold {} // font weight .hover:border-{color}-{shade (100-900)} // border color

15- Focus Modifier :

L'ajout d'un état de focus est simple en utilisant le focus : modifier avec l'une des classes par défaut fournies par CSS vent arrière.

Default responsive classes:

.focus:bg-* {} // background color

.focus:text-{color}-{shade (100-900)} {} // text color

.focus:font-bold {} // font weight .focus:border-{color}-{shade (100-900)} // border color

16- Combination Modifiers :

Parfois, la conception peut nécessiter 2 modifications en même temps. Par exemple, vous devrez peut-être modifier le état de hover pour backgroud color mais uniquement dans md:size. Voyons comment y parvenir.

.md:hover:bg-{color}-{shade (100-900)} {} // hover background color

.md:focus:bg-{color}-{shade (100-900)} {} // focus background color

.md:hover:text-{color}-{shade (100-900)} {} // hover text color

.md:focus:text-{color}-{shade (100-900)} {} // focus text color

17-Autres :

Voici quelques autres utilitaires utiles que vous devriez connaître.

Box Shadows - responsive, hover, focus states:

.shadow-{size} {} // md, lg, xl, 2xl, inner, outline, none

Opacity - responsive, hover, focus states:

.opacity-{percent} // 100, 75, 50, 25, 0

Cursors – responsive:

.cursor-{style} // default, pointer, wait, text, move, not-allowed

Outline – focus:
.focus:outline-none // remove default browser outline styling

User Select – responsive:

.select-{style} // none, text, all, auto

Screen Readers - responsive, hover, focus, active:

.sr-only {} // visually hidden, present for screen readers

.no-sr-only {} // undo .sr-only

18-Configuration:

Tailwind CSS permet de personnaliser les paramètres par défaut en utilisant le fichier de configuration, vous donnant ainsi un contrôle total sur les classes utilitaires générées et vous permettant de créer des designs uniques en modifiant les valeurs par défaut.

Exemple :

```
module.exports = {
  content: ['./src/**/*.{html,js}'], // Spécifie les fichiers à analyser pour PurgeCSS (ici, les fichiers HTML et JavaScript dans le dossier src)
  theme: {
    // Définition des couleurs personnalisées
    colors: {
      'blue': '#1fb6ff',
      'purple': '#7e5bef',
      'pink': '#ff49db',
      'orange': '#ff7849',
      'yellow': '#ffc82c',
      'gray-dark': '#273444',
      'gray': '#8492a6',
      'gray-light': '#d3dce6',
    },
    // Définition des polices personnalisées
    fontFamily: {
      sans: ['Graphik', 'sans-serif'],
      serif: ['Merriweather', 'serif'],
    },
    // Extensions personnalisées pour les espacements et les bordures
    extend: {
      spacing: {
        '8xl': '96rem', // Ajoute une classe utilitaire '8xl' avec une valeur d'espacement de 96rem
        '9xl': '128rem', // Ajoute une classe utilitaire '9xl' avec une valeur d'espacement de 128rem
      },
      borderRadius: {
        '4xl': '2rem', // Ajoute une classe utilitaire '4xl' avec une valeur de bordure de 2rem
      }
    },
  }
```

19- Créer des classes personnalisées :

`@apply` :

`@apply` est une directive spécifique à Tailwind CSS qui permet de réutiliser des ensembles de classes utilitaires en créant vos propres classes personnalisées. Vous pouvez regrouper un ensemble de classes dans une nouvelle classe, puis appliquer cette classe personnalisée à un élément HTML. Cela permet de simplifier votre code et de favoriser la réutilisation des styles.

`@layer` :

`@layer` est une directive de regroupement de styles dans Tailwind CSS. Elle permet de définir un regroupement logique de styles, tels que les styles de base, les composants ou les utilitaires. Les directives `@layer` aident à mieux organiser votre code et à gérer la priorité des styles lorsque vous utilisez des plugins ou des extensions qui introduisent leurs propres styles.

Exemple :

@apply:

```
.btn {
  @apply px-4 py-2 rounded bg-blue-500 text-white;}
```

@layer :

```
@layer components {
  .btn-primary { @apply py-2 px-4 bg-blue-500 text-white font-semibold rounded-lg shadow-md hover:bg-blue-700 focus:outline-none ; }}
```