Framework CSS

Rechercher des informations:

Pour mener ma veille sur Tailwind CSS, j'ai utilisé différents outils et méthodes afin de collecter des informations sur les nouveautés et évolutions de ce framework CSS.

- **Moteur de recherche Google :**
  J'ai effectué des recherches avec les mots-clés:
  - ***"Tailwind CSS vs Bootstrap"***
  - ***"Nouveautés Tailwind CSS 4.0"***
  - ***"Avantages de Tailwind CSS"***
    Ces requêtes m'ont permis de comparer Tailwind CSS à d'autres frameworks et de mieux comprendre ses évolutions.
- **Daily.Dev :**
  J'ai également utilisé Daily.Dev, une plateforme de veille technologique qui centralise des articles et ressources issus de blogs spécialisés et de sites techniques.
- **Sites spécialisés :**
  J'ai consulté plusieurs sites pour approfondir mes recherches, notamment des blogs, documentation officielle afin de différencier tailwind CSS V3 et la V4 et selon mes connaissances également. Cependant, certains de sites que j'ai consultés ne sont plus accessibles.

2. Filtrage et analyse des informations

Après la collecte des données, j'ai procédé à une analyse, comme l'analyse de tailwindCSS et boostrap :

Les évolutions majeures de Tailwind CSS 4.0 :

- **Suppression de la dépendance à PostCSS et Autoprefixer.**
- **Introduction de la directive `@theme` pour une gestion simplifiée des variables CSS qui se rapproche de la variable ":root" de CSS.**
- **Amélioration des performances avec une compilation 10 fois plus rapide.**
- **Simplification de la configuration en supprimant `tailwind.config.js` dans certains cas.**

Comparaison avec Bootstrap :

tailwindCSS:

- **Tailwind CSS offre une approche plus utilitaire avec des classes prédéfinies, contrairement à Bootstrap qui propose des composants prêts à l'emploi.**
- **Meilleure personnalisation avec Tailwind CSS grâce à `@theme`.**
- **Réduction de la taille des fichiers CSS générés par Tailwind CSS 4.0.**
- **Réduction de l'espace de stockage.**

Boostrap:

- **Les fichiers CSS de Bootstrap sont plus volumineux en raison de la nécessité d'inclure tous les composants et styles par défaut.**
- **Plus de travail pour la personnalisations des composants, bien que boostrap propose des composants déjà tout préparés.**

Source:
https://www.tailflows.com/blog/tailwind-css-v4-va-tout-changer

https://geekly.blog/tailwind-css-v4-0/#:~:text=qui%20nous%20attend.-,Amélioration%20des%20performances,des%20performances%20allant%20de%20x3

https://tailwindcss.com

Synthèse

TailwindCSS est un framework du css permettant au développeur de personnaliser le design de leurs sites en restant au sein même du fichier html.

La dernière et meilleure version de tailwindCSS est la version alpha 4.0. Cette dernière version compile 10 fois plus vite que les anciennes versions, elle est complètement autonome et plus légère. Et les développeurs n'ont plus besoin de plugins externes comme postcss pour automatiser les opérations CSS.

Dans les anciennes versions de tailwindCSS les developpeur avait besoin d'installer plusieurs dépendances pour lancer un projet, la version alpha 4.0 de TailwindCSS obtient plus d'outils qui n'envisage plus d'avoir besoin d'installer des modules.

Avec la nouvelle fonctionnalité de tailwindCSS on peut réaliser des compositions comme *'group-has-focus:opacity-100'* pour détecter directement qu'un élément au sein d'un groupe est en état de focus. Alors que dans les

anciennes versions on devait utiliser la méthode de Javascript pour détecter l'élément parent qui contenait un enfant en focus.
comme ceci :

```
<div id="container" class="opacity-50">
        <input type="text" placeholder="Focus" />
</div>
<script>
document.querySelector('input')?.addEventListener('focus, () => {
        document.getElementByid('container').classList.add('opacity-100');
}

document.querySelector('input')?.addEventListener('focus, () => {
        document.getElementByid('container').classList.remove('opacity-100');
}
</script>
```

Dans la nouvelle version de TailwindCSS on peut maintenant définir des variables css dans le fichier css principal en utilisant "@theme{...}"

par exemple

```
@theme {
 --color-*: initial;
 --color-gray-50: #f8fafc;
}
```

Grâce à cette méthode, les développeurs peuvent gérer facilement les variables CSS dans le thème de leur projet. Dans les anciennes versions les variables étaient directement mis dans le fichier de configurations de "tailwind.config.js". La version 4.0 nous donne enfin la liberté de jongler directement dans le CSS.

La configuration du fichier tailwind.confi.js a été simplifié pour l'ajout de thèmes personnalisés par les utilisateurs que ce soit des couleurs, des polices etc…

Les évolutions majeures de TailwindCSS 4.0

Dans mes projets récents, j'utilise la directive `@theme` de TailwindCSS pour gérer efficacement les variables CSS, cela me permet de maintenir une cohérence visuelle tout en simplifiant les modifications, sans avoir à toucher au fichier `tailwind.config.js` comme c'était nécessaire dans les anciennes versions.

`@theme` est équivalent à `:root` en CSS pour gérer les variables

```css
@import 'tailwindcss';

@theme {
  --color-purple-500: #6b46c1;
}

.bg-custom {
  background-color: var(--color-purple-500);
}
```

```
<nav class="fixed top-0 left-0 w-full border-b border-gray-200 bg-custom text-white shadow-md z-50">
  <div class="max-w-screen-xl flex flex-wrap items-center justify-between mx-auto p-4">
    <a href="#" class="flex items-center space-x-3 rtl:space-x-reverse">
      <span class="self-center text-2xl font-semibold whitespace-nowrap dark:text-white">DN Père & Fils</span>
      <!-- image here maybe-->
    </a>
    <button data-collapse-toggle="navbar-default" type="button"
      class="inline-flex items-center p-2 w-10 h-10 justify-center text-sm text-gray-500 rounded-lg md:hidden hover:bg-gray-100 focus:outline-none focus:ri
      aria-controls="navbar-default" aria-expanded="false">
      <span class="sr-only">Open main menu</span>
      <svg class="w-5 h-5" aria-hidden="true" xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 17 14">
        <path stroke="currentColor" stroke-linecap="round" stroke-linejoin="round" stroke-width="2"
          d="M1 1h15M1 7h15M1 13h15" />
```

OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
tal@1.0.0 build
ailwindcss -i ./css/style.css -o ./dist/styles.css --watch
```

Find Aa

DN Père & Fils Home About contrat Contact

dans les anciennes versions de tailwind ont devaient rentrer dans le fichier `tailwind.config.js`, ce qui peut vite devenir embêtant quand plusieurs lignes se présentent.
pour les couleurs ou la taille des polices d'écriture.
Ceci me permet de personnaliser les couleurs selon ce que j'ai défini.
dans le fichier tailwind.config.js.

```js
module.exports = {
  content: ["./*.html"],
  theme: {
    screens: {
      sm: '480px',
      md: '768px',
      lg: '976px',
      xl: '1440px',
    },
    colors: {
      'blue': '#1fb6ff',
      'purple': '#7e5bef',
      'pink': '#ff49db',
      'orange': '#ff7849',
      'green': '#13ce66',
      'yellow': '#ffc82c',
      'gray-dark': '#273444',
      'gray': '#8492a6',
      'gray-light': '#d3dce6',
    },
    plugins: [],
  }
```

Par exemple je vais ajouter `'purple-400': '#000000.` purple-400 je l'ai défini en noir, c'est pour moi plus pratique que de regarder la documentation de tailwindCSS. Je n'ai besoin que de mémoriser le texte que j'ai spécifié.

```
'green': '#13ce66',
'yellow': '#ffc82c',
'gray-dark': '#273444',
'gray': '#8492a6',
'gray-light': '#d3dce6',
'purple-400': '#000000',
},
```

```
<div class="text-center my-6">
  <img class="mx-auto w-10" src="./images/courrier.png" alt="">
  <h1 class="font-bold text-4xl">
    Nous <span class="text-purple-400">contacter</span>
  </h1>
```

Nous contacter

Cependant le fichier pouvait vite devenir chargé, surtout lorsqu'il fallait gérer des variantes, ajouter des plugins etc…, dans la version 4.0 certaines configurations peuvent être effectuées directement dans le fichier CSS en utilisant la directive `@theme`, qui réduit donc la dépendance au fichier `tailwind.config.js` pour des personnalisations plus simples.

Avant la version 4.0 de tailwindCSS comme par exemple la version 3.x, nécessitait plusieurs dépendances comme PostCSS et Autoprefixer, ou bien le CLI. Pour l'installer il fallait effectuer la commande :
npm install -D tailwindcss@3 postcss autoprefixer.
npx tailwindcss init => Ce qui permet d'avoir un fichier tailwind.config.js

Avec la version 4.0 de tailwindCSS, le framework est devenu totalement autonome et peut s'installer plus rapidement et facilement et ne génère plus le fichier tailwind.config.js.
npm install tailwindcss.
Ajouter "@import "tailwindcss";" dans un fichier CSS.
Ajouter cette commande dans le terminal : npx @tailwindcss/cli -i ./src/input.css -o ./src/output.css --watch

Pour plus de clarté :

| **3.x et antérieures | PostCSS, Autoprefixer | `tailwind.config.js` obligatoire avec npx init | `npm install tailwindcss postcss autoprefixer`.**

| **4.0 | Aucune (autonome) | `tailwind.config.js` supprimé (remplacé par `@theme` dans le CSS) | `npm install tailwindcss`.**

Conclusion

En conclusion, ma veille sur Tailwind CSS a mis en lumière les évolutions significatives de la version 4.0, notamment la suppression de la dépendance à PostCSS, la suppression de la commande npx init qui enlève le fichier tailwind.config.js, le changement de beaucoup de variantes, l'introduction de la directive `@theme` et l'amélioration des performances de compilation. Ces changements témoignent de l'engagement de la communauté Tailwind à améliorer l'expérience de développement et à simplifier la personnalisation des styles.

La comparaison avec Bootstrap a également révélé des différences essentielles dans les approches des deux frameworks. Tandis que Tailwind CSS privilégie une méthode utilitaire permettant une personnalisation approfondie, Bootstrap se concentre sur des composants prêts à l'emploi, ce qui peut convenir à des projets nécessitant une mise en œuvre rapide afin de plus se concentrer sur le back-end.

Les avantages de Tailwind CSS, tels que le gain de temps lors du design et la réduction de la taille des fichiers CSS qui réduit la taille de l'espace disque, font de ce framework un choix attrayant pour les développeurs cherchant à optimiser leur flux de travail tout en bénéficiant d'une bonne flexibilité.

En outre, Tailwind CSS continue d'évoluer et de s'affirmer comme un outil puissant pour le développement web, et il est essentiel pour les développeurs de se tenir informés des nouvelles tendances et des améliorations de ce framework pour maximiser leur efficacité.