Leveling up my UI game with Tailwind CSS!

Key takeaways:

- **Tailwind basics**: Understanding the history and core concepts of Tailwind CSS.
- **Utility classes**: Exploring the powerful utility classes Tailwind offers and practicing them using Tailwind Play.
- **Flexbox integration**: Combining Tailwind with Flexbox for responsive and efficient layouts.
- **Project planning**: Discussing project ideas and setting up the development environment.
- **Component reusability**: Learning how to create and reuse custom components for efficient development.

I'm excited to apply these new skills to my projects and build visually stunning user interfaces.

Bishal Rijal

History of Tailwind CSS

Tailwind CSS was born out of frustration with the limitations of traditional CSS frameworks. Its creators, Adam Wathan, Jonathan Reinink, David Hemphill, and Steve Schoger, sought a more flexible and customizable approach to styling web applications.

Key points about Tailwind's history:

- **Utility-First Philosophy:** Unlike traditional frameworks that provide pre-built components, Tailwind emphasizes individual utility classes for styling elements directly.
- **Initial Release:** Tailwind was first released in 2019, quickly gaining popularity due to its unique approach and flexibility.
- **Just-in-Time (JIT) Compilation:** In 2021, Tailwind introduced JIT mode, which generates CSS on demand based on your actual usage, reducing file size and improving build times.
- **Growing Community and Adoption:** Tailwind's popularity has soared, with a large and active community contributing to its development and creating various plugins and extensions.

Tailwind's history reflects a shift towards more developer-centric and customizable CSS frameworks, offering greater control over the styling process.

Tailwind CSS vs. Bootstrap: A Comparison

Tailwind CSS and **Bootstrap** are two popular CSS frameworks, but they take significantly different approaches to styling web applications.

Approach

- **Tailwind CSS:** Utility-first framework, providing individual classes for various styles (e.g., bg-blue-500, text-center).
- **Bootstrap:** Component-based framework, offering pre-designed components (e.g., buttons, cards, navigation bars).

Customization

- **Tailwind CSS:** Highly customizable, allowing you to create unique designs from the ground up.
- **Bootstrap:** Offers customization options, but requires more effort to deviate significantly from the default styles.

Learning Curve

- **Tailwind CSS:** Requires a steeper learning curve, as you need to understand the utility classes and their combinations.
- **Bootstrap:** Generally easier to learn, as you can use pre-built components with minimal customization.

Performance

- **Tailwind CSS:** Can be more performant due to Just-in-Time (JIT) compilation, which generates only the CSS you use.
- **Bootstrap:** May include unused styles, which can potentially impact performance.

Use Cases

- **Tailwind CSS:** Ideal for projects that require highly customized designs and a deep level of control over styling.
- **Bootstrap:** Suitable for projects that need a consistent look and feel, and where rapid prototyping is important.

In summary:

- **Tailwind CSS** is great for those who want maximum flexibility and control over their designs.
- **Bootstrap** is a good choice for those who prefer a pre-built component library and a quicker development process.

The best choice for you depends on your specific project requirements and preferences.

Practice Tailwind: https://play.tailwindcss.com/

Tailwind CSS layer

The HTML code consists of two parents elements, each containing three child elements with the class box.

```
<div>
  <div class="m-4 h-20 w-20 bg-red-200">Hello</div>
  <div class="m-4 h-20 w-20 bg-red-200">Hi</div>
  <div class="m-4 h-20 w-20 bg-red-200">Bye</div>
</div>
```

Tailwind CSS Classes:

- **m-4:** Adds a margin of 4 units (based on your Tailwind configuration) to all sides of the element.
- **h-20:** Sets the height of the element to 20 units.
- **w-20:** Sets the width of the element to 20 units.
- **bg-red-200:** Applies a light red background color to the element.

```
@layer components {
  .box {
    @apply m-4 h-20 w-20 bg-red-200;
  }
}
```

Tailwind CSS Layer:

- **@layer components:** Defines a custom layer named "components" for organizing your Tailwind CSS styles.
- **.box:** Selects all elements with the class box.
- **@apply m-4 h-20 w-20 bg-red-200;:** Applies the specified utility classes to the .box elements.

Explanation:

The first element contains three plain elements with the class box. These elements have no explicit styling applied.

The second element contains three elements with the classes m-4 h-20 w-20 bg-red-200. These classes directly apply the desired styles (margin, height, width, and background color) to the elements.

The Tailwind CSS layer defines a custom component named .box and applies the same styles (margin, height, width, and background color) to all elements with that class.

In essence:

- Both elements will render the same content: three boxes with the same dimensions and background color.
- The first element achieves this styling through direct application of classes to the individual elements.

- The second element achieves this styling through the .box component defined in the Tailwind CSS layer, which is applied to each of the child elements.

The choice between these approaches depends on your preference and the complexity of your styling requirements. If you need to apply consistent styles to multiple elements, using a Tailwind CSS layer can be more efficient and maintainable.

Project Setup

Install Tailwind CSS: npm install tailwind -D tailwindcss

Execute Tailwind CSS: npx tailwindcss init

Convert Tailwind to CSS: npx tailwindcss -i ./styles/input.css -o ./styles/output.css --watch

This command uses the Tailwind CSS CLI to:

1. ***Process input file:*** *Reads the CSS file ./styles/input.css.*
2. ***Generate output file:*** *Creates a new CSS file ./styles/output.css containing the compiled Tailwind CSS rules based on the classes used in the input file.*
3. ***Watch for changes:*** *Continuously monitors the input file for changes. Whenever a change is detected, it recompiles the CSS and updates the output file.*

In simpler terms, this command automates the process of compiling your Tailwind CSS styles and keeping them up-to-date as you make changes to your input file.

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

@type {import('tailwindcss').Config}:

- This is a TypeScript annotation that specifies the type of the module.exports object. It indicates that the object is expected to conform to the Config interface defined in the tailwindcss package.

module.exports = { ... }:

- This line exports a JavaScript object that contains the configuration settings for your Tailwind CSS project.

content: ['*.{html,js}']:

- This property specifies the files and directories that Tailwind CSS should scan for class names to extract and generate the corresponding CSS rules. In this case, it will scan all files with .html or .js extensions.

theme: { extend: {} }:

- This property allows you to customize the default Tailwind CSS theme. The extend object is used to add or modify existing theme values. In this example, the theme is not extended, so it uses the default Tailwind CSS theme.

plugins: []:

- This property allows you to add custom plugins to your Tailwind CSS configuration. Plugins can extend Tailwind CSS's functionality or provide additional features. In this example, no plugins are used.

Overall, this configuration file:

- Specifies that Tailwind CSS should scan .html and .js files for class names.
- Uses the default Tailwind CSS theme.
- Does not use any custom plugins.

This configuration is a basic starting point for most Tailwind CSS projects. You can customize it further to suit your specific needs by modifying the content, theme, and plugins properties.

Some popular sites for Tailwind UI components:

1. https://tailwindui.com/
2. https://tailkit.com/
3. https://tailwindflex.com/
4. https://flowbite.com/
5. https://tailblocks.cc/

Bishal Rijal
GitHub: https://github.com/bishaldsrija08

Follow For More!

Bishal Rijal
GitHub: https://github.com/bishaldsrija08