Índice

Assuntos: TailWind CSS.

Abertura Alunos:
Apresentação pitch
Eloisa e Marilia.

React - Tailwind CSS

As bibliotecas de estilização em React são conjuntos de ferramentas que permitem aos desenvolvedores criarem e estilizarem interfaces de usuário para aplicativos React. Essas bibliotecas fornecem uma variedade de recursos como estilos pré construídos, sistemas de grade, componentes personalizados, e mais.

Bibliotecas comuns:

1. CSS Modules
2. Styled Components
3. Material-UI
4. Bootstrap
5. Tailwind CSS

Tailwind CSS é uma biblioteca de estilos utilitários que pode ser usada em conjunto com React para estilizar rapidamente os componentes sem escrever CSS personalizado.

Em vez de fornecer estilos pré-construídos, como muitas outras bibliotecas de estilização, o Tailwind CSS fornece classes utilitárias que podem ser usadas para definir estilos diretamente em seus elementos HTML ou em seus componentes React.

Por exemplo, em vez de escrever CSS personalizado para definir a largura de um elemento, você pode usar uma classe utilitária como **`w-1/2`** para definir a largura como metade do contêiner pai.

Essas classes utilitárias são definidas em um arquivo de configuração do Tailwind CSS e podem ser personalizadas para atender às necessidades específicas do seu projeto.

Vantagens de uso Tailwind:

1. Eficiência
2. Consistência
3. Flexibilidade
4. Responsividade
5. Manutenção

Instalação

No console, use NPM para instalar o Tailwind no seu projeto:

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

Adicione os caminhos a todos os seus arquivos de modelo em seu arquivo **tailwind.config.js**.

Exemplo:

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
```

```
        "./index.html",
        "./src/**/*.{js,ts,jsx,tsx}",
 ],
 theme: {
        extend: {},
 },
 plugins: [],
}
```

Adicione as **@tailwinddiretivas** para cada uma das camadas do Tailwind ao seu arquivo. **./src/index.css**

- **@tailwind base;**
- **@tailwind components;**
- **@tailwind utilities;**

Adicione o código adequado ao seu **app.tsx**.

Classes

Em Tailwind CSS, as classes são as unidades básicas que você pode usar para aplicar estilos a elementos HTML. Cada classe define um conjunto específico de estilos que podem ser aplicados a um elemento.

As classes em Tailwind são compostas por uma ou mais palavras-chave separadas por hífens. Cada palavra-chave representa uma propriedade CSS específica que pode ser aplicada a um elemento.

As classes são altamente configuráveis e modulares, permitindo que você personalize o estilo do seu site ou aplicativo sem precisar escrever CSS personalizado ou lidar com conflitos de estilos.

Componentes

Em Tailwind CSS em React, os componentes geralmente se referem aos elementos React que são estilizados usando as classes do Tailwind CSS. Esses componentes são criados usando a sintaxe JSX do React e são estilizados usando as classes do Tailwind.

A vantagem de usar componentes estilizados com Tailwind é que eles permitem que você crie interfaces de usuário consistentes e reutilizáveis, sem ter que escrever CSS personalizado para cada componente.

FlexBox

A classe flex em Tailwind CSS é utilizada para tornar um elemento HTML um contêiner flexível. Isso significa que os filhos desse contêiner poderão ser organizados em uma única linha ou em uma única coluna, dependendo da direção definida pelo desenvolvedor.

Por padrão, a direção do contêiner flexível é definida como "row", o que significa que os filhos são organizados em uma única linha. No entanto, é possível alterar a direção para "column", fazendo com que os filhos sejam organizados em uma única coluna.

Comandos

flex-row e flex-col

Utilizadas para definir a direção do contêiner flexível como "row" ou "column".

justify-start, justify-end, justify-center, justify-between e justify-around

utilizadas para definir o alinhamento horizontal dos filhos dentro do contêiner flexível.

items-start, items-end, items-center, items-baseline e items-stretch

utilizadas para definir o alinhamento vertical dos filhos dentro do contêiner flexível.

flex-wrap, flex-nowrap e flex-wrap-reverse

utilizadas para definir o comportamento de quebra de linha dos filhos dentro do contêiner flexível.

order-1, order-2, order-3, etc.

utilizadas para definir a ordem dos filhos dentro do contêiner flexível.

Grid

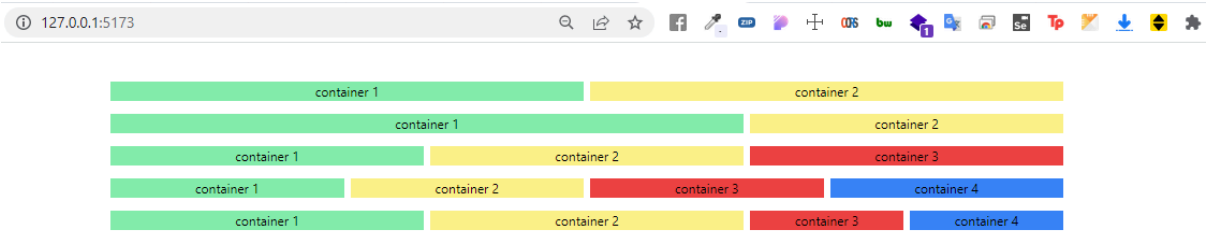
O grid do Tailwind é baseado em um sistema de grade responsivo de 12 colunas, que é facilmente personalizável para se adaptar às necessidades do projeto.

Por outro lado, o "Grid no Tailwind CSS" pode se referir ao uso específico de classes de grid no Tailwind CSS, que permitem definir as propriedades de layout do grid diretamente no HTML. Por exemplo, você pode usar as classes "**grid-cols-2**" e "**gap-4**" para definir um grid de duas colunas com um espaçamento de 4 pixels entre as colunas.

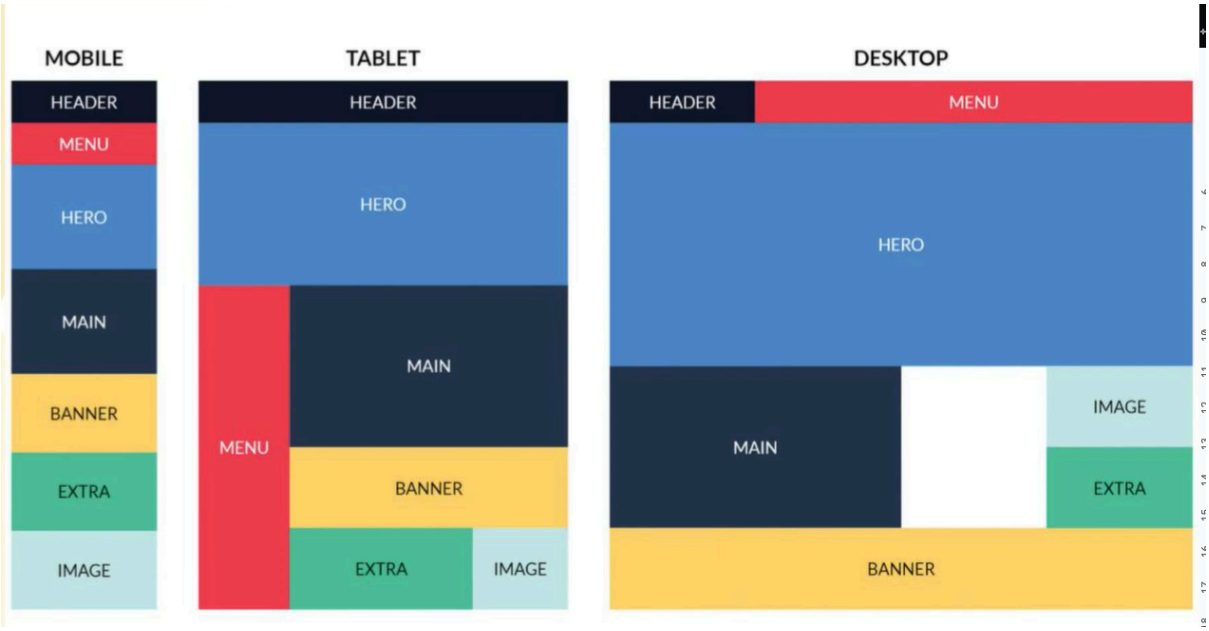
O grid permite que os desenvolvedores criem layouts mais flexíveis e responsivos, adaptando-se automaticamente aos diferentes tamanhos de tela e dispositivos.

Para usar o CSS grid, é preciso definir um container como um grid container e definir os elementos filhos como grid itens, podemos então usar as propriedades CSS para definir o tamanho e a posição dos itens dentro da grade.

Exemplo visual grid:



Layout Responsivo



Layout adaptativo que se adequa a diferentes dispositivos, todos os dados são adequados aos diferentes tamanhos de dispositvos.

1. Layouts flexíveis
2. Imagens responsivas

3. Tipografia responsiva
4. Menu e navegação
5. Performance

Comandos

**<div className = ‘grid grid-cols-12 grid-rows-5 gap-2 m-4 ‘> **

Exemplo de criação de um grid.

- Fazemos a criação de uma ** ** e inserimos as classes de Tailwind.
- **grid:** indica que faremos um grid
- **grid-cols-12:** indica que o grid terá 12 colunas.
- **grid-rows-5:** indica que o grid deve ter 5 linhas de tamanho.
- **gap-2:** indica que entre os espaços do grid teremos 2 pixels de distância.
- **m-4:** indica que haverá uma margem de 4 píxels.

**<div className = ‘col-span-4 bg-purple-150 > **

Exemplo de preenchimento visual de um grid.

- Fazemos a criação de uma ** ** e inserimos as classes de Tailwind.
- **col-span-4:** indica que esse espaço ocupará 4 colunas do container.
- **bg-purple-150:** define que a cor de fundo desse preenchimento será um tom de roxo.

React - Router

Bibliotecas de roteamento em desenvolvimento front-end são conjuntos de ferramentas e funções que permitem que um aplicativo de front-end gerencie as rotas do navegador de forma eficiente e confiável.

Essas bibliotecas geralmente são usadas em aplicativos de página única (SPA), que usam JavaScript para renderizar o conteúdo no navegador.

Bibliotecas comuns: React Router, Vue Router, Angular Router.

Essas bibliotecas de roteamento ajudam a gerenciar a navegação em um aplicativo SPA, permitindo que os usuários naveguem pelo aplicativo sem recarregar a página inteira a cada mudança de rota.

Além disso, as bibliotecas de roteamento fornecem recursos adicionais, como redirecionamentos, navegação programática e animações de transição, que ajudam a melhorar a experiência do usuário em um aplicativo de front-end.

React Router DOM é uma biblioteca de roteamento para aplicativos React. Ele fornece componentes que podem ser usados para definir rotas em seu aplicativo, como o **<Route>**

e o <**Switch**>. Esses componentes permitem que você defina quais componentes serão renderizados com base na URL atual do aplicativo.

Instalação

Digite no terminal:

yarn add react-router-dom

Construindo o Login

- Construa uma pasta chamada login dentro pasta paginas de seu projeto.
- Construa um componente chamado Login dentro da pasta navBar de seu projeto.
- Configure o Home e App.tsx de acordo.

Comandos

Há duas formas de navegação no router DOM, uma é o **useNavigate()**, outra o **Link**.

useNavigate é um hook fornecido pela biblioteca React Router que permite navegar programaticamente entre rotas em um aplicativo React.

é invocado **navigate('/outra-rota')**, redirecionando o usuário para a rota especificada **(/outra-rota), como visto abaixo**:

```
function handleClick() {
        navigate('/outra-rota');
}
 return (
        <div>
        <button onClick={handleClick}>Navegar</button>
        </div>
 );
```

A tag <**Link**> é uma componente fornecida pela biblioteca react-router-dom que permite criar links de navegação em um aplicativo React.

É importante notar que o atributo to da tag <Link> especifica a rota de destino para onde o usuário será navegado. O valor desse atributo pode ser uma string representando a URL da rota ou um objeto que define a rota com mais detalhes.

```
<div>
    <Link to="/outra-rota">  Ir para outra rota  </Link>
</div>
```

React - Componentes

Footer

O Footer é a seção da parte inferior da página e geralmente contém informações importantes, como informações de contato, links para as políticas de privacidade e termos de serviço, créditos e outras informações relevantes.

Ele também pode incluir links para recursos adicionais que os usuários possam achar úteis, como um blog ou uma seção de perguntas frequentes. O Footer ajuda a fornecer informações importantes ao usuário e a melhorar a credibilidade e transparência da aplicação.

Criação:

1. instalar a biblioteca phosphor-icons:
2. digite em seu terminal:

   ```
   npm install @phosphor-icons/react
   ```
3. Construa uma pasta chamada **components** dentro da pasta src de seu projeto.
4. Construa uma pasta chamada **footer** dentro pasta components de seu projeto.
5. Construa um componente chamado **Footer** dentro da pasta **footer** de seu projeto.
6. Construa o código de acordo.

NavBar

O Navbar é uma barra de navegação que geralmente fica no topo da página e permite que o usuário navegue pelos diferentes recursos e páginas da aplicação de forma fácil e intuitiva.

Ele geralmente inclui links para as principais seções do site (menu), bem como opções de pesquisa e login. O Navbar ajuda a melhorar a usabilidade, permitindo que os usuários naveguem rapidamente para as áreas da aplicação que desejam acessar.

1. Construa uma pasta chamada **navBar** dentro pasta **components** de seu projeto.
2. Construa um componente chamado **NavBar** dentro da pasta **navBar** de seu projeto.
3. Construa o código de acordo.

Home

A Home é a página principal da aplicação e geralmente é a primeira página que os usuários veem. Ela deve ser projetada para fornecer uma visão geral clara dos recursos e funcionalidades da aplicação, além de apresentar as informações mais relevantes aos usuários. Uma boa Home deve ser fácil de navegar e ajudar os usuários a encontrar rapidamente o que estão procurando.

Ícones Personalizados

No console digite npm **install @phosphor-icons/react**
Use o comando **npm run dev**

Links e afins

Cookbook, React, Tailwind, MD06:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/06.md

Cookbook, React, Flexbox Grid e Tailwind, MD07:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/07.md

Cookbook, React, Rotas Router Dom, MD08:

https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/08.md

Cookbook, React, NavBar Footer e Home, MD09:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/09.md

Cookbook, React, Criação de Rotas, MD10:
https://github.com/conteudoGeneration/cookbook_java_fullstack/blob/main/05_react/10.md

Documentação Tailwind CSS:
https://tailwindcss.com/docs

Conversor HTML para Tailwind:
https://transform.tools/html-to-jsx

Documentação Tailwind Vite:
https://tailwindcss.com/docs/guides/vite

Documentação ReactRouter:
https://reactrouter.com/en/main/start/tutorial

Dúvidas