



Curso de Tailwind CSS

do Básico ao Avançado



Por que Tailwind?

- Foco em **utility** e não componentes;
- **Pouco ou nenhum CSS** para atingir resultados excelentes;
- **Pouca sobreposição** de CSS, ao contrário de frameworks;
- Foco em **responsividade**;
- Podemos criar **componentes**;
- Facilidade em **customização**;



Propósitos do Tailwind

- **Diminuir o tempo** que passamos codificando CSS;
- Permitir **ampla liberdade** entre os componentes que criamos;
- Facilidade em **estender e customizar**;
- Acoplagem em diversos ambientes, com ou sem framework front-end;
- Dificuldade zero para começar com **mobile first**;



Instalação Node

- O **Tailwind** não necessita do **Node**, porém facilita muito a instalação do mesmo;
- E caso você utilize o Tailwind com algum framework com **React**, já vai agilizar a sua vida;
- Podemos seguir as instruções do site: www.nodejs.org



Instalação VS Code

- O **VS Code** é com certeza um dos editores mais utilizados atualmente;
- Ele também facilita muito a nossa vida com o **terminal integrado** e as suas extensões;
- Por estes e outros motivos, será o **editor utilizado no curso!**
- Podemos seguir as instruções do site: **code.visualstudio.com**



Instalação Tailwind

- Vamos agora instalar o **Tailwind**;
- Primeiramente precisamos instalar ele no projeto via **npm**;
- Depois vamos **criar um arquivo CSS** que vai conter as diretivas do Tailwind;
- Por último precisamos **buildar o arquivo** baseado no arquivo de configurações criado, **apontando para a saída** que é o arquivo final de CSS;



Instalação Live Server

- O **Live Server** é uma extensão excelente para o VS Code;
- Permite que as **alterações do front persistam na página** quando forem salvas;
- Isso vai agilizar o nosso trabalho;
- É opcional, mas **quebra um galho**;



Instalação Tailwind Intellisense

- O **Intellisense** é uma extensão excelente para o Tailwind;
- Vai nos ajudar na criação dos projetos, **auto completando** as classes do framework;
- E em alguns casos mostra um **preview** do que utilizaremos;



Como tirar o máximo proveito

- Faça todos os exemplos com o editor aberto e **codifique junto**;
- Crie **seus próprios exemplos** com o que foi aprendido na aula;
- Crie também **projetos pessoais**;
- Faça **anotações** dos pontos mais importantes;
- **Dica bônus:** assista a aula e depois execute;





Introdução do curso

Conclusão





Conceitos Fundamentais

Introdução da seção



Tudo é classe

- Estilizamos o nosso projeto todo por meio de **classes**;
- Temos classes desde **cor de fonte** até **pseudo seletores** (hover);
- Podemos adicionar classe para determinados **break points** (responsivo);
- Podemos também criar as **nossas classes** com base em classes;
- **Obs:** veremos detalhadamente cada parte do framework durante o curso, porém alguns exemplos apresentam classes antes de conhecermos elas a fundo;



Utility First

- A premissa do Tailwind é construir componentes complexos com um conjunto de **utilitários**;
- Ou seja, em vez de pegar um componente pronto ou codar muitas linhas de CSS, utilizamos um **conjunto de classes**;
- E desta maneira podemos criar diversas variações de componentes, de forma simples;
- Ou **componentes únicos** para o projeto;



Responsivo - Mobile first

- Todas as classes podem ser aplicadas a um **determinado breakpoint**;
- Os que vem com o framework são: **sm, md, lg e xl**;
- Lembrando que é **mobile first**, ou seja, as classes são aplicadas para resoluções acima destes breakpoints;
- Então, **não utilizamos nenhum breakpoint** para atingir o mobile;



Pseudo classes

- Podemos atingir as **pseudo classes** com Tailwind também;
- Um exemplo seria o **hover**, quando passamos o ponteiro do mouse em cima de um elemento;
- Desta maneira basta adicionar: **hover:classe**;
- E então ela **só será executada** quando preencher os requisitos;



Componentes

- Em Tailwind somos encorajados a **não utilizar** componentes prontos;
- Primeiramente desenvolvemos o que precisamos e depois podemos **transformar em um componente** (via apply);
- Essa abordagem é interessante pois nem sempre tudo é definido no CSS, talvez precisamos de uma estrutura de HTML diferenciada;





Conceitos Fundamentais

Conclusão





Construção de Layout

Introdução da seção



Container

- Um elemento que possui uma determinada largura, e que serve para incluir elementos dentro;
- O container com as classes de responsividade (sm, md, lg, xl), pode setar uma **max-width** no elemento;
- Podemos aplicar com a classe **container**;



Exercício 1

- Crie um container com cor de background verde;
- Quando atingir md, deve ficar azul;



Box sizing

- Com as classes **border-box** e **border-content**, podemos declarar como o elemento calcula o seu tamanho total;
- Com **border-box** teremos uma largura total somada com padding e borders;
- Com **box-content** teremos uma largura respeitando a medida e com padding e border passando a largura determinada;



Display

- Colocando classes com valores da propriedade **display**, como **block**, podemos controlar este comportamento do elemento;
- Então um elemento com classe **inline-block**, se comporta igual a um elemento com estilo de **display: inline-block**;
- As outras propriedades seguem a mesma lógica;



Exercício 2

- Crie um container com três divs no HTML;
- Deixe as divs internas inline utilizando display;
- Todas devem ter backgrounds diferentes;



Floats e clear

- Podemos controlar a propriedade **float** do elemento com classes **float-***;
- Onde * é o tipo de float que desejamos, por exemplo: **right ou left**;
- O **clear** segue o mesmo padrão, as classes tem o início de **clear-***;
- Onde * pode ser **both**, por exemplo;
- **Obs:** após o Grid e Flexbox, o float vem sendo pouco utilizado nos layouts;



Overflow

- O **overflow** é como controlamos quando o conteúdo de um elemento é muito grande para o mesmo;
- Podemos **criar ou eliminar o scroll**, por exemplo;
- Inserimos a classe **overflow-***, onde * é o valor da propriedade, como: none ou auto;
- Podemos controlar os **eixos** também com: **overflow-y-***



Posições e direções

- Podemos **controlar as posições** dos elementos por classes também;
- Neste caso o próprio nome da classe já é o da posição;
- Exemplos: **static, absolute, relative e etc**;
- Estes valores seriam relativos a: **position: absolute**;
- As direções também podem ser adicionadas via classe, exemplo: **top-2**;
- **Obs:** no Tailwind os números sempre são transferidos para rem, podemos converter para pixels com: **1rem = 16px** (valor default);



Exercício 3

- Crie um container com um altura determinada;
- Posicione uma div dentro deste container no canto inferior direito;



Visibilidade

- Podemos alterar a propriedade **visibility** com duas classes;
- **visible** = visibility: visible;
- **invisible** = visibility: hidden;
- Utilizamos para exibir ou esconder elementos;



Z-index

- Podemos controlar o **z-index** pelo Tailwind também;
- Esta propriedade controla a **sobreposição de elementos** na página;
- O valor a ser colocado na classe é **z-***;
- Onde * é o número de indexação, como **10, 20 ou 30**;



Exercício 4

- Crie 3 elementos;
- Cada um deve sobrepor o outro no HTML, por meio do z-index;
- Inclua cores diferentes;





Construção de Layout

Conclusão





Flexbox

Introdução da seção



Flex direction

- Podemos inserir o flexbox em um elemento com a classe **flex**;
- E controlar a direção dele com: **flex-row** e **flex-col**;
- Onde **flex-row** é para linhas;
- E **flex-col** para colunas;
- **Obs:** Lembrando que o container de elementos deve receber a classe flex;



Exercício 5

- Crie um container com flex;
- Layout colunar;
- Seis itens com cores diferentes;



Flex wrap

- O **wrap** é a propriedade que controla como os itens se adaptarão ao container;
- Com **flex-no-wrap** os itens tendem a ficar na mesma linha;
- Com **flex-wrap** os itens vão se encaixando em linhas abaixo, conforme o tamanho do container;



Flex

- Esta é a regra **flex para o elemento**, onde podemos controlar seu tamanho máximo e mínimo;
- Podemos inserir valores como **flex-1** para flex: 1 1 0%;
- Ou **flex-none** para flex: none;



Order

- Com order podemos controlar a ordem dos elementos;
- Inserimos a classe **order-x**;
- Onde x é igual ao valor da **propriedade order**;
- Exemplo: **order-1 = order: 1**;



Exercício 6

- Crie um container com flex;
- O container deve inverter a ordem depois da resolução lg;





Flexbox

Conclusão





Grid

Introdução da seção



Grid columnas

- Com o **grid** podemos criar uma estrutura muito fácil de página;
- A classe **grid-cols-x** define o número de columnas em um grid;
- Onde x é o **número de columnas** desejadas;
- Obs: a classe pai precisa conter a classe **grid**;



Exercício 7

- Crie um container de grid;
- Este container deve ter 3 colunas;
- E ao total 9 itens;
- As colunas devem alternar entre duas cores;



Tamanho das colunas

- Com a **col-span-x** podemos controlar o tamanho da coluna;
- Onde x é o tamanho, **baseado no número de colunas definido na classe pai;**
- Lembrando que essa classe vai nos elementos filhos do container de grid;



Exercício 8

- Crie uma estrutura grid semelhante a um site;
- Separado em duas colunas;
- Onde a primeira é uma barra de navegação lateral e a segunda é o conteúdo;
- O grid deve ter 4 colunas, e a segunda representar 3;
- Crie também uma barra de navegação superior, com o tamanho de todas as colunas;



Quantidade de rows

- Podemos também especificar a **quantidade de rows** que um grid possui;
- A classe para este fim é: **grid-rows-x**;
- Onde x é o **número de linhas** que o container possuirá;



Espaço entre colunas e linhas

- Podemos controlar o espaçamento das colunas e linhas com **gap**;
- Inserimos a classe **gap-x**, no container principal;
- Onde x é o **tamanho do espaçamento**;
- É possível também mudar o espaçamento pelo **eixo x ou y**;



Exercício 9

- Crie um grid com duas colunas e quatro itens;
- Insira um gap de 6 neste grid;





Grid

Conclusão





Customização

Introdução da seção



Como customizar o Tailwind

- Vamos adicionar as customizações no arquivo **tailwind.config.js**;
- Lá temos algumas seções:
- **theme**: estilos para o nosso projeto;
- **extend**: extensão das nossas classes do projeto;
- **plugins**: adicionar plugins de Tailwind externos;
- **prefix**: adiciona um prefixo as classes do Tailwind;



Customizando o tema

- A chave que iremos alterar é a **theme**;
- Aqui vamos **inserir regras que são parecidas com as do Tailwind**, gerando o efeito de estender o framework;
- Se colocarmos regras com o mesmo nome, iremos gerar um **override**, ou seja, substituição;



Definindo breakpoints

- Na chave **screens** de theme, podemos customizar o breakpoint;
- Podemos adicionar novos, apenas criando nomenclaturas diferentes, como por exemplo: **'tablet': '640px'**;
- Ou também colocar breakpoints para **max-width**, que é o contrário do que o Tailwind nos dá;



Adicionando cores e estendendo

- Na chave **colors** de theme, podemos adicionar novas cores, o que é muito normal para projetos com design mais avançado;
- O padrão é **nomedacor: hexadecimal**;
- Assim, **após o build** teremos acesso a nova cor;
- Podemos também criar uma estrutura de objeto, para **simular uma paleta de cores**;



Alterando o espaçamento

- Na chave **spacing**, podemos alterar o espaçamento do projeto;
- Podemos **definir nossos próprios**;
- Ou utilizar o **extend** para adicionar mais espaçamentos no padrão do Tailwind por rem, por exemplo;





Customização

Conclusão





Espaçamento e alinhamento

Introdução da seção



Alterando o espaçamento

- Utilizando a regra **justify-*** podemos controlar o espaçamento;
- O valor **start**, deixa os itens no começo;
- E **end** no fim;
- Podemos também centralizar com **center**;
- **Obs:** as regras de justify-* trabalham no eixo x;



Exercício 10

- Crie um container com seis itens;
- Eles devem estar centralizados;
- Alterne entre três cores



Alinhamento de conteúdo

- Utilizando o **content-*** podemos alinhar o conteúdo no eixo y;
- Com **center**, por exemplo, podemos deixar os elementos centralizados, em linhas;
- E com **end**, no fim do container;



Alinhamento de itens

- Utilizando **items-*** controlamos como os itens são dispostos ao longo do container;
- Podemos centralizar com **center**;
- Ou alongar do começo ao fim, com **stretch**;



Alinhamento e espaçamento

- Classe utilizada para **grid**;
- Utilizando **place-content-*** podemos controlar o espaçamento e alinhamento com a mesma regra;
- Utilizamos os valores semelhantes das regras anteriores, como: **start, end e center**;



Alinhamento e espaçamento de itens

- Classe utilizada para **grid**;
- Podemos controlar o alinhamento e espaçamento ao mesmo tempo dos itens com **place-items-***;
- Os valores também são semelhantes: **start, end, stretch e center**;



Padding

- Podemos adicionar padding de uma maneira muito simples no Tailwind, a classe **p-***, adiciona padding a todas as direções;
- E **px-*** ou **py-*** adiciona padding ao eixo x e ao eixo y, respectivamente;
- E para adicionar em uma direção específica podemos utilizar: **pt, pr, pl e bp**;



Exercício 11

- Crie um componente de botão;
- Adicione 4 de padding nas laterais;
- E 2 de padding na vertical;
- Extra: adicione uma cor de background;
- Extra 2: mude a cor de background com hover;
- Extra 3: adicione fonte em negrito;



Margin

- Podemos adicionar margem de uma maneira muito simples no Tailwind, a classe **m-***, adiciona margem a todas as direções;
- E **mx-*** ou **my-*** adiciona margem ao eixo x e ao eixo y, respectivamente;
- E para adicionar em uma direção específica podemos utilizar: **mt, mr, ml e mp**;



Exercício 12

- Crie uma div com um texto pequeno dentro;
- Coloque uma classe de w-20;
- Centralize a div na página, no eixo x, com margem;
- Adicione uma margem superior de 2;



Largura

- Podemos alterar a largura de um elemento com **w-***;
- Também é possível alterar a min-width, a classe é **min-w-***;
- E é claro a max-width, com: **max-w-***;



Altura

- Podemos alterar a largura de um elemento com **h-***;
- Também é possível alterar a min-height, a classe é **min-h-***;
- E é claro a max-height, com: **max-h-***;





Espaçamento e alinhamento

Conclusão





Tipografia

Introdução da seção



Tipo de fonte

- Podemos controlar o tipo de fonte do Tailwind com algumas classes:
- Sem serifa: **font-sans**;
- Com serifa: **font-serif**;
- Mono espaçada: **font-mono**;
- Para outras fontes, podemos estender a propriedade **fontFamily**;



Tamanho da fonte

- Podemos controlar o tamanho da fonte com **text-***;
- Onde * é o **tamanho da fonte** desejada;
- O tamanho em Tailwind é inserido por valores como: **sm, lg, xl**;



Exercício 13

- Crie um parágrafo com fonte monoespçada;
- Adicione um tamanho de fonte xl;
- Padding em todos os eixos de 4;
- E um background cinza;



Estilo e weight

- Com **italic** podemos deixar o texto em itálico;
- E com **font-*** podemos alterar o weight;
- Temos valores como: **light, normal, bold e black**;



Letter spacing e line height

- Letter spacing é o espaçamento entre letras, podemos controlar com: **tracking-***;
- Onde * é o valor que **modifica o espaçamento**;
- Já line height é o espaçamento entre linhas;
- Podemos controlar com: **leading-***;
- Onde * é o **valor de espaçamento**;



Exercício 14

- Crie um parágrafo e adicione texto em *itálico*;
- Aumente o espaçamento entre letras;
- E adicione a fonte `sm`;



List type e position

- Podemos controlar o tipo de lista com a classe: **list-***;
- Onde * será o tipo da lista, por exemplo: **disc**;
- E a posição da lista também é controlada por **list-***;
- Com os valores: **inside ou outside**;



Ajustando o placeholder

- Podemos também modificar o placeholder, com **placeholder-*** inserimos a cor;
- Onde * é a **cor** que vamos inserir no elemento;
- E também é possível **controlar a opacidade**;
- A classe é **placeholder-opacity-***;



Alinhamento de texto

- Podemos **alinhar o texto** para esquerda, centro e direita com Tailwind;
- A classe é: **text-***;
- Onde * podemos inserir valores como: **left, right e center**;



Cor de texto

- Podemos alterar a cor do texto com **text-***;
- Onde * é a **cor** que vamos inserir;
- Um detalhe para color é que temos também a **shade da cor**, que vai de 100 a 900;
- E é claro, podemos **customizar completamente** as cores;



Exercício 15

- Crie um parágrafo que parece com uma mensagem de erro;
- Fundo vermelho;
- Texto em vermelho e negrito;
- Extra: Adicione bordas vermelhas também;
- Extra 2: arredonde um pouco o canto das bordas



Decoration e transform

- Com decoration podemos **adicionar alguns estilos a textos**, como: underline e line trough;
- As classes são: **underline e line-trough**;
- Com transform podemos **alterar a case do texto**, ou seja, minúsculas e maiúsculas;
- Utilizamos classes como: **lowercase e uppercase**;





Tipografia

Conclusão da seção





Backgrounds

Introdução da seção



Exibição de background

- Podemos controlar como a imagem de background se comporta;
- A classe **bg-x** pode ser utilizada para isso;
- Esta classe controla a propriedade **background-attachment**;
- Temos as opções de **fixed**, **local** e **scroll**;



Cor de background e opacity

- Podemos controlar a cor de background com **bg-***;
- Onde * é a **cor** que vamos inserir;
- A opacidade **bg-opacity-*** controla a opacidade do background;
- Onde * é a intensidade da opacidade do elemento de background;



Posição e repetição

- Com **bg-*** podemos também controlar a posição;
- Onde * é a **posição** que queremos inserir, como top, left e bottom;
- Com **bg-repeat** podemos controlar a forma de repetição da imagem de fundo;
- Podemos também controlar o eixo da repetição;



Tamanho do background

- Podemos controlar o tamanho do background com **bg-***;
- Onde * é o tamanho que o background vai representar;
- Temos valores como **auto e cover**;



Imagem de background

- Podemos fazer gradients com a classe de background image;
- A classe é **bg-gradient-***, onde podemos criar um degradê;
- Podemos também definir uma imagem de background no **arquivo de configuração**;
- E inserir esta imagem via classe;





Backgrounds

Conclusão da seção





Bordas

Introdução da seção



Raio da borda

- Podemos adicionar raio as bordas com a classe **rounded-***;
- Onde * é o **tamanho do raio**;
- Outras possibilidades são: **elementos pílulas e circulares**;
- Para estes elementos vamos utilizar **rounded-full**;



Largura e cor de borda

- Podemos alterar a largura da borda com a classe **border-***;
- Onde * é o spacing que determina a **largura**;
- Já a cor é adicionada por **border-***;
- Onde * neste caso é uma **shade de cor**;



Opacidade e estilo da borda

- Podemos alterar a opacidade de uma borda com a classe **border-opacity-***;
- Onde * é a proporção de **opacidade do elemento**;
- O estilo é alterado por **border-***;
- Onde * é o tipo de estilo da borda, algumas opções são: **solid, dashed, double, none**;



Divide

- Divide é uma **utility de borda** que é utilizada nos containers para dividir elementos;
- Utilizamos com **divide-***;
- Onde nesta mesma classe podemos escolher o **eixo, cor, largura e opacidade do divide**;
- Que no fim das contas é uma **borda**;



Box Shadow

- Com o box shadow podemos inserir um **sombreamento** no elemento;
- A classe utilizada é **shadow-***;
- Onde * é a **tonalidade** do sombreamento que vamos inserir;





Bordas

Conclusão da seção





Interatividade

Introdução da seção



Aparência

- Podemos remover a **aparência default** de alguns elementos com o Tailwind;
- Utilizamos para isso a classe **appearance-none**;



Cursor do mouse

- Podemos alterar o **cursor** também com Tailwind;
- Vamos utilizar a classe **cursor-***;
- Onde * é o **tipo de cursor** (ponteiro) que queremos inserir, como por exemplo: pointer;



Removendo o outline

- O outline é **uma borda** que é inserida em alguns navegadores quando acessamos um input;
- Para remover este recurso utilizamos a classe: **outline-none**;



Removendo pointer events

- Podemos **remover os eventos de clique** do HTML;
- A classe que realizará este efeito é: **pointer-events-none**;
- Caso precise voltar o evento: **pointer-events-auto**;



Modificando o resize

- Resize é uma **propriedade do textarea**, que permite redimensionar o elemento;
- Podemos controlar como ele é exibido com: **resize-***;
- Onde * é o **tipo de resize** que vamos adicionar ao elemento;



Removendo seleção

- Podemos também **remover a seleção de um elemento** pelo usuário;
- A classe utilizada é **select-***;
- Onde * é o **tipo de seleção** permitida sobre o elemento alvo;





Interatividade

Conclusão da seção





Tailwind V2

Introdução da seção



Corrigindo problema de instalação

- Devido às diversas dependências de Tailwind e suas versões independentes o modo de instalação mudou um pouco;
- **Ao longo do curso** crie os projetos desta maneira:
`npm install tailwindcss@latest postcss@latest autoprefixer@latest`
- Esta forma vai fazer com o que o seu projeto rode normalmente o Tailwind já na **versão 2** =)



Cores novas

- Foi adicionada uma **tonalidade 50** para todas as cores já existentes do Tailwind;
- E também temos agora **22 cores** (antes eram 10);
- Estas novas cores devem ser importadas de **tailwind.colors** no arquivo de configuração;
- E também devemos configurar o nome delas;



Dark Mode

- O **dark mode** é uma funcionalidade muito presente nos sistemas atualmente e foi adicionada ao Tailwind;
- Para habilitar precisamos adicionar a configuração **darkMode** o valor de **media ou class**;
- O valor de media corresponde a **configuração do sistema operacional do usuário**;
- Vamos ver na prática;



Dark Mode Manual

- Podemos também ativar o Dark Mode de **modo manual**;
- Vamos precisar trocar o valor de darkMode para **class**;
- E inserir uma classe dark na tag **html** do nosso HTML;
- Podemos fazer isso com **JavaScript**;
- Vamos ver na prática!



Outros recursos do Dark Mode

- Podemos aplicar dark mode a classe de **hover**;
- E também condicionar a **media queries** com: md, lg e xl, por exemplo;
- Vamos ver na prática!



Ring Utilities

- Uma forma de **contornar o elemento com uma borda**;
- Semelhante ao border, porém **não ocupa espaço na tela**;
- Podemos adicionar a um **evento de focus**, por exemplo;
- E o ring pode ser **interno ou externo**;



Tailwind CSS Forms

- Um pacote dos criadores do Tailwind para agilizar o desenvolvimento de formulários;
- Vamos precisar instalar o pacote com npm: **npm install @tailwindcss/forms**
- E também configurá-lo em **plugins** no arquivo de configuração;
- Está nos estágios iniciais, porém possui funcionalidades interessantes



Line height padrão

- Agora cada **font-size** vem com um **line-height** padrão;
- Ou seja, quanto maior a fonte, maior a altura da linha contabilizada na página;
- Podemos alterar isso com a classe **leading-***;



Mais opções para...

- **Espaçamentos:** 0.5, 1.5, 2.5, 72, 80, 96
- **Fontes:** 7xl, 8xl, 9xl
- **Opacidade:** 5 e 95;
- Estes valores aumentam nossas opções para estas configurações;



Apply funciona com qualquer coisa

- Antes o **apply** não funcionava com alguns recursos, como o **hover**;
- Agora podemos **utilizar com qualquer regra**, segundo a documentação;
- Ou seja: hover, focus e etc;



Funcionalidades para text overflow

- Agora temos novas opções para lidar com limitação de texto;
- **overflow-ellipsis** para cortar o texto com reticências;
- **overflow-clip** para apenas cortar o texto;





Tailwind V2

Conclusão da seção

