Sumário

[03Preparando o ambiente 2](#_Toc134625534)

[05ATV - Iniciando o projeto 13](#_Toc134625535)

[06 ATV - Nome do arquivo 14](#_Toc134625536)

[07 ATV - Criando seu arquivo HTML principal 15](#_Toc134625537)

[08Para saber mais: criação de pastas 16](#_Toc134625538)

[09O que aprendemos? 19](#_Toc134625539)

[Nessa aula, você aprendeu como: 19](#_Toc134625540)

[02 - 01Preparando o ambiente 20](#_Toc134625541)

[03Para saber mais: documentação do HTML 21](#_Toc134625542)

[A importância da documentação 21](#_Toc134625543)

[02.05 ATV Adicionando uma imagem 23](#_Toc134625544)

[02.06 ATV - Título 25](#_Toc134625545)

[02.07Quirks Mode e Live Server 26](#_Toc134625546)

[02.08Quirks Mode 31](#_Toc134625547)

[02.09Para saber mais: extensões do VSCode 32](#_Toc134625548)

[02.10Para saber mais: estruturando uma página HTML 33](#_Toc134625549)

[A estrutura básica do HTML 33](#_Toc134625550)

[02.11O que aprendemos? 35](#_Toc134625551)

[Nessa aula, você aprendeu: 35](#_Toc134625552)

[03. 02 Preparando o ambiente 36](#_Toc134625553)

[03 .03Projeto no Figma 38](#_Toc134625554)

[Transcrição 38](#_Toc134625555)

[03.04Tags semânticas 42](#_Toc134625556)

[Transcrição 42](#_Toc134625557)

[03.05Desenvolvendo o HTML 47](#_Toc134625558)

[Transcrição 47](#_Toc134625559)

[03. 06Para saber mais: tags semânticas 57](#_Toc134625560)

[03. 07Sobre os metadados 58](#_Toc134625561)

[03. 08Usando tags semânticas 60](#_Toc134625562)

[ALURA - Mergulhe em Tecnologia! 60](#_Toc134625563)

[03. 09Diferença entre âncora e botão 62](#_Toc134625564)

[03. 10O que aprendemos? 63](#_Toc134625565)

[Nessa aula, você aprendeu: 63](#_Toc134625566)

[04. 01Projeto da aula anterior 64](#_Toc134625567)

[04. 02Como funciona o CSS 65](#_Toc134625568)

[Transcrição 65](#_Toc134625569)

[04. 03Incluindo CSS na página 69](#_Toc134625570)

[04. 04Criando o CSS 70](#_Toc134625571)

[Transcrição 70](#_Toc134625572)

[04. 05 ATV Estilizando o HTML com CSS 75](#_Toc134625573)

[04. 06Cor de fundo 76](#_Toc134625574)

[04. 07Faça como eu fiz: estilize seu HTML 77](#_Toc134625575)

[04. 08O que aprendemos? 78](#_Toc134625576)

[Nessa aula você aprendeu: 78](#_Toc134625577)

[04. 01Projeto da aula anterior 79](#_Toc134625578)

[04. 02 Como funciona o CSS 80](#_Toc134625579)

[Transcrição 80](#_Toc134625580)

[04. 03Incluindo CSS na página 84](#_Toc134625581)

[04. 04 Criando o CSS 85](#_Toc134625582)

[Transcrição 85](#_Toc134625583)

[04. 05 ATV - Estilizando o HTML com CSS 90](#_Toc134625584)

[04. 06Cor de fundo 91](#_Toc134625585)

[04. 07Faça como eu fiz: estilize seu HTML 92](#_Toc134625586)

[04. 08O que aprendemos? 93](#_Toc134625587)

[Nessa aula você aprendeu: 93](#_Toc134625588)

[05. 01Projeto da aula anterior 94](#_Toc134625589)

[05. 02Cores no CSS 95](#_Toc134625590)

[Transcrição 95](#_Toc134625591)

[05. 03Para saber mais: escolhendo as cores do projeto 100](#_Toc134625592)

[Coolors 100](#_Toc134625593)

[Adobe Color 100](#_Toc134625594)

[Color Hunt 101](#_Toc134625595)

[Color Tool - Material Design 102](#_Toc134625596)

[05.04 ATV Cor de fundo 103](#_Toc134625597)

[05. 05Destacando o texto 104](#_Toc134625598)

[Transcrição 104](#_Toc134625599)

[05. 06Para saber mais: destacando o texto 108](#_Toc134625600)

[05. 07Projeto final do curso 110](#_Toc134625601)

[05. 08Desafio: compartilhe seu projeto com o mundo 111](#_Toc134625602)

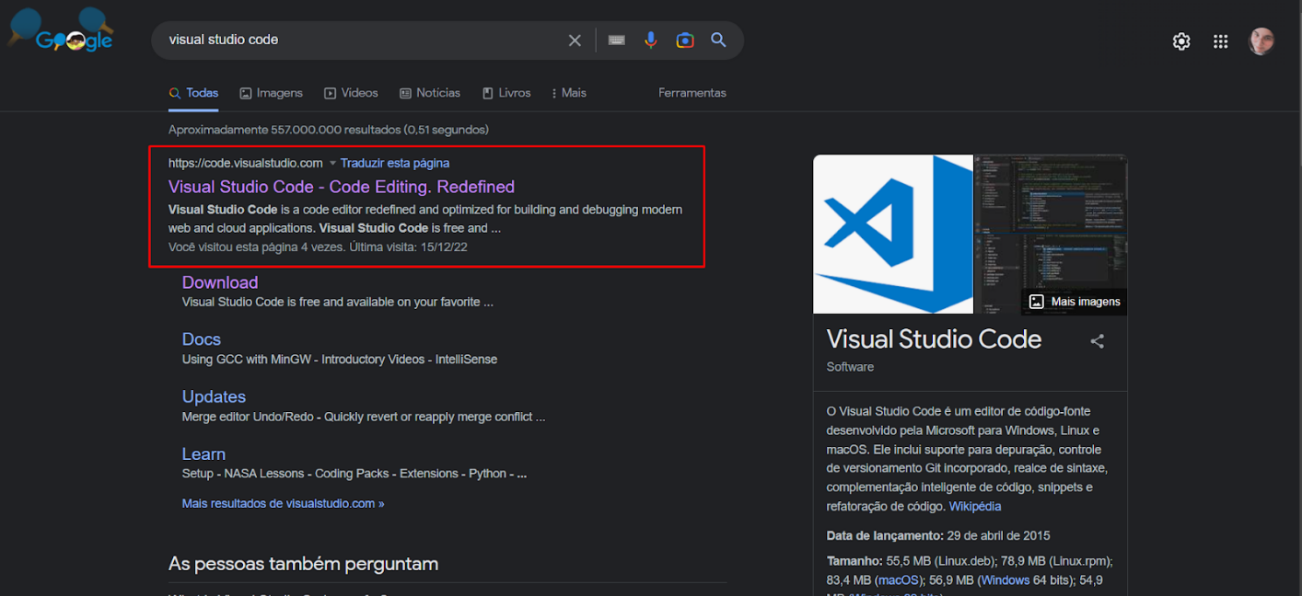
[05. 09O que aprendemos? 112](#_Toc134625603)

[Nessa aula, você aprendeu como: 112](#_Toc134625604)

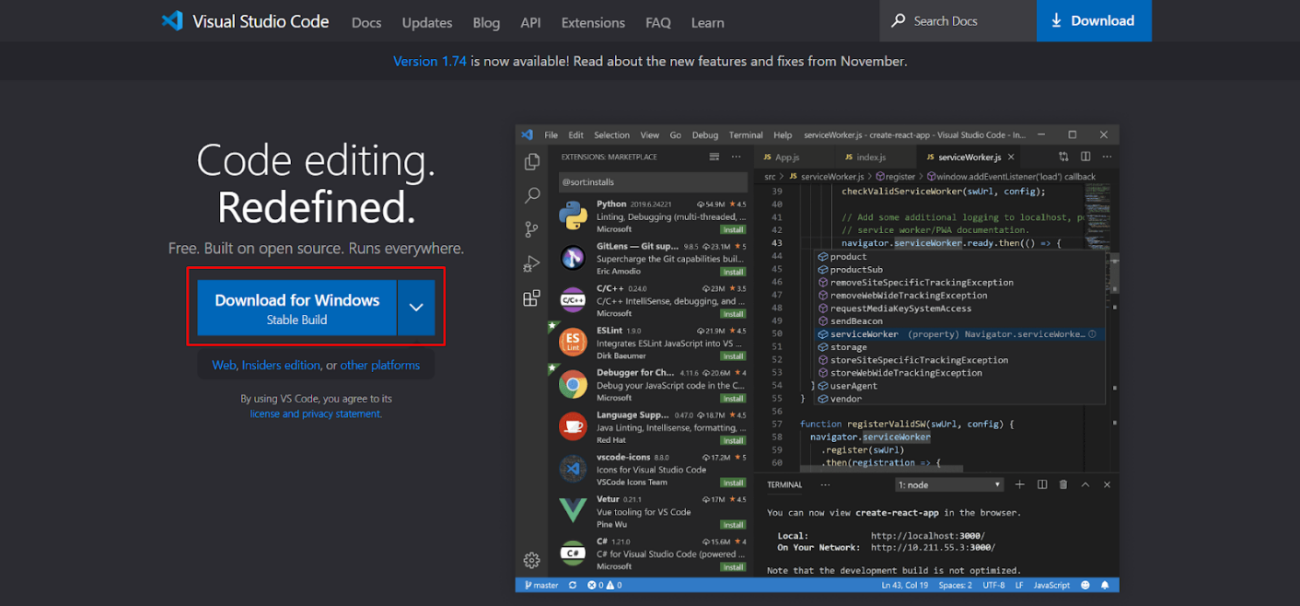
# 03Preparando o ambiente

No curso, usaremos uma ferramenta chamada [Visual Studio Code](https://code.visualstudio.com/), que é um editor de código desenvolvido pela Microsoft para Windows, Linux e macOS. Caso queira acompanhar os instrutores com as mesmas configurações, reserve um tempinho para a instalação do mesmo.

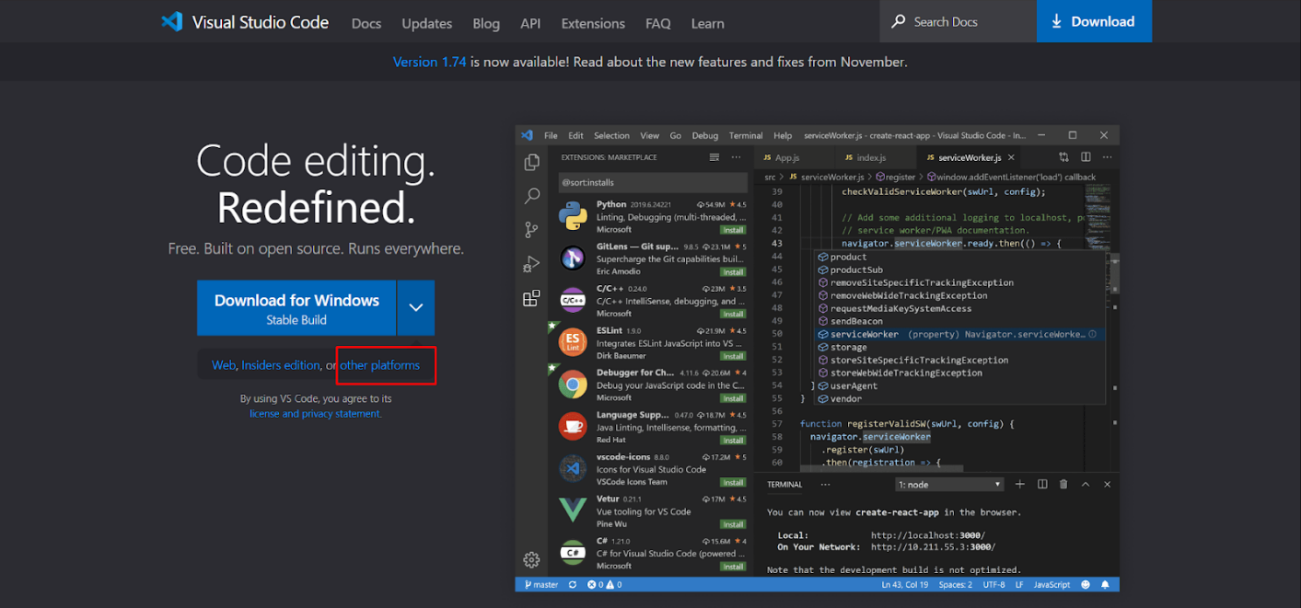
Para baixar o VSCODE, você pode procurar no Google por Visual Studio Code e clicar no primeiro link que aparece:

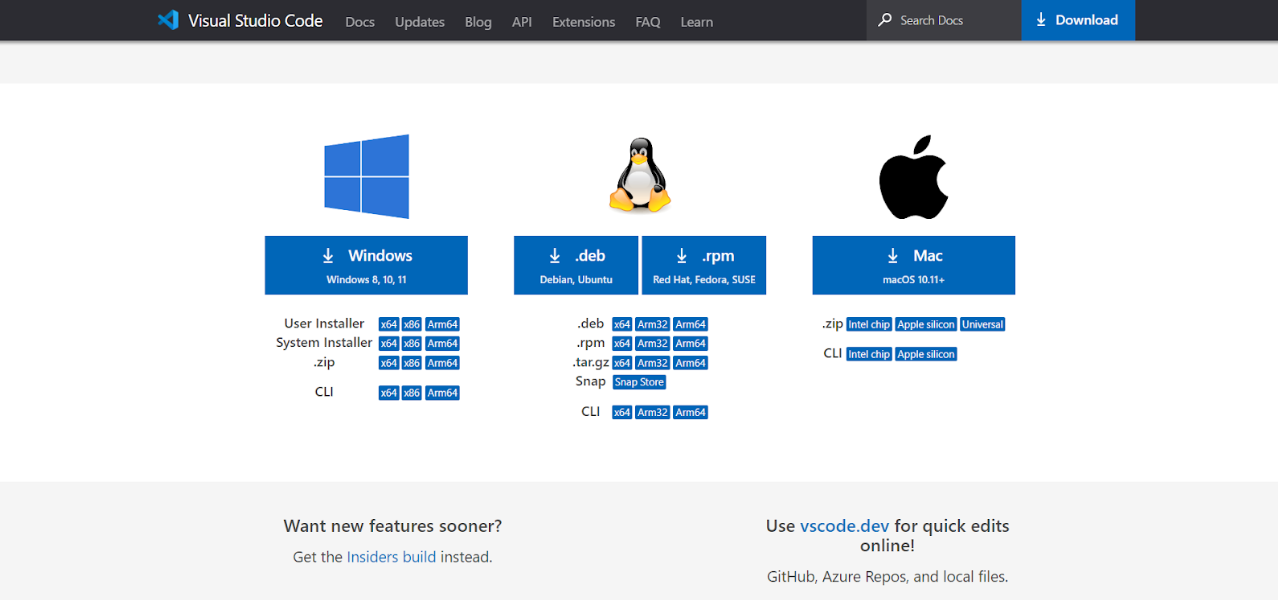


Se o seu sistema operacional for Windows você pode clicar para baixar no botão **Download for Windows**:

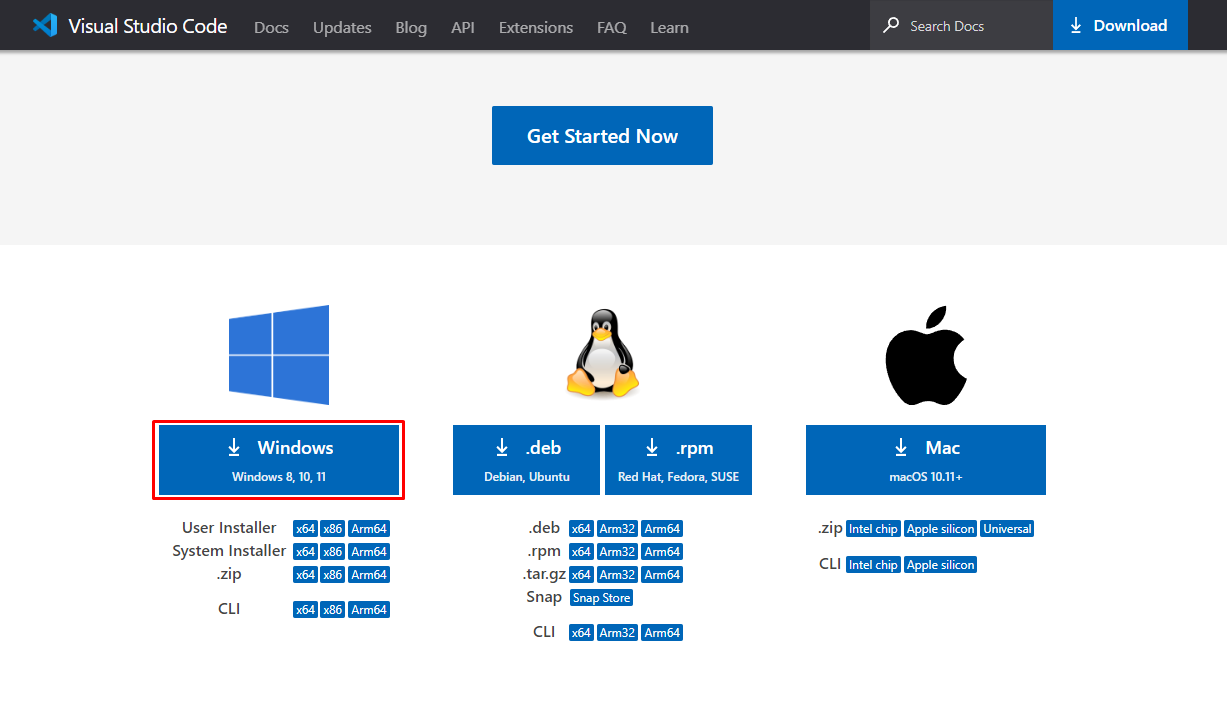


Senão você pode clicar logo abaixo em [other platforms](https://code.visualstudio.com/#alt-downloads) que irá abrir uma página com opção de download para **Linux** e **macOS** também:

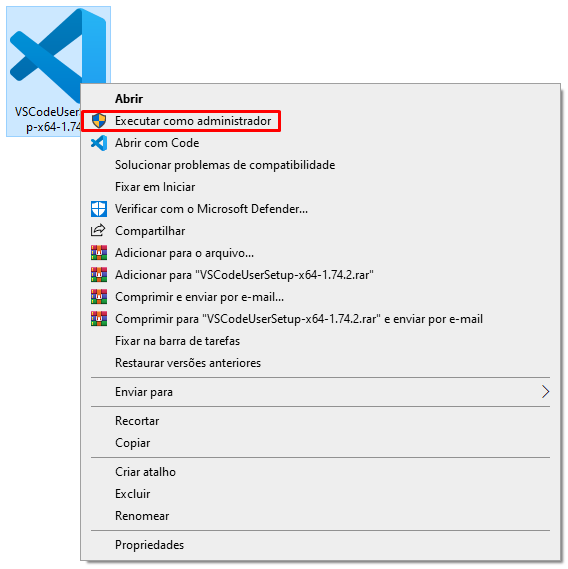




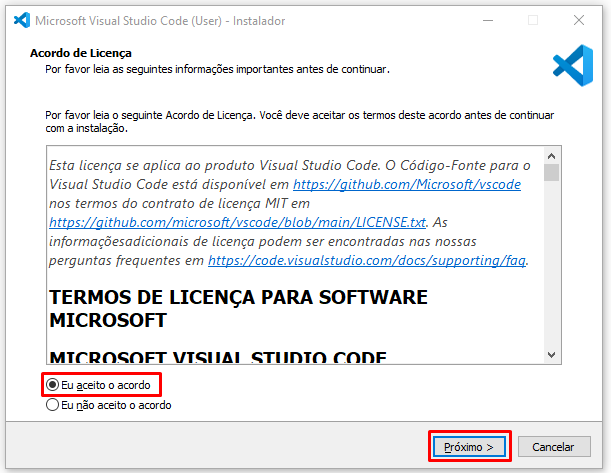
Se o seu sistema operacional for o Windows 8, 10 ou 11, siga os seguintes passos para instalação:



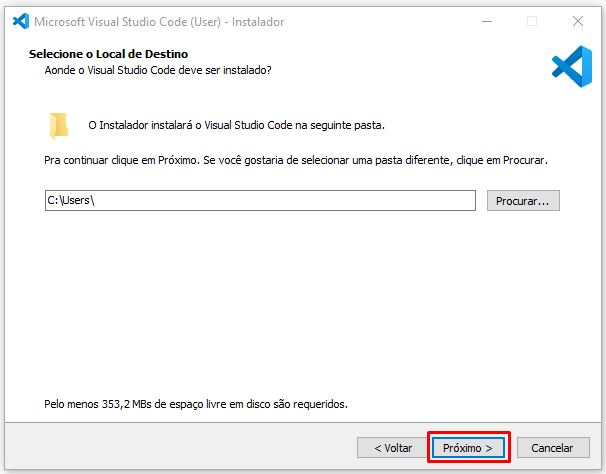
Após clicar na opção de baixar para Windows, o instalador do Visual Studio Code vai para a pasta de Downloads do seu computador. Acesse a pasta, localize o arquivo de instalação, clique com o botão direito sobre ele e “Execute como administrador”.



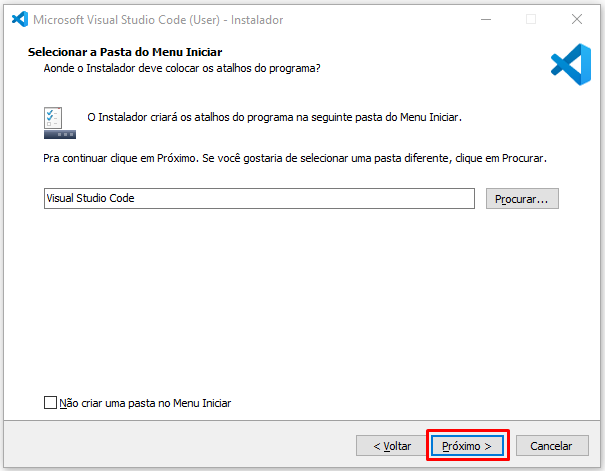
Após aceitar as permissões do Windows para a instalação, o instalador será iniciado e nessa etapa você deve aceitar o acordo de licença para poder prosseguir.



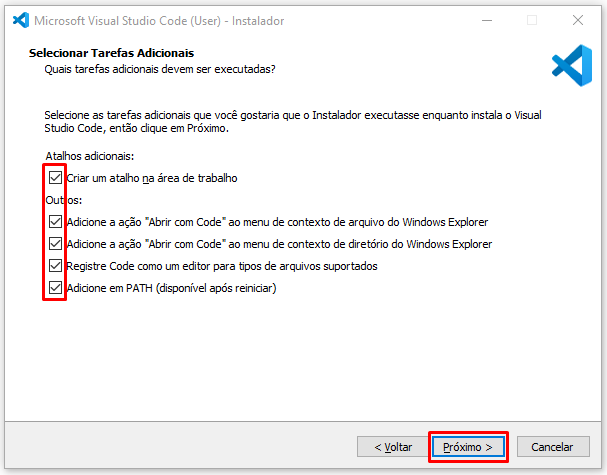
A próxima tela é para você selecionar o local de destino, ou seja, onde o Visual Studio Code será instalado no seu computador. Caso prefira, pode deixar como está sem alterar nada e clique em “Próximo”. Antes de prosseguir com a instalação, é importante lembrar que para poder instalar o Visual Studio Code o seu computador tem que ter o espaço necessário livre.



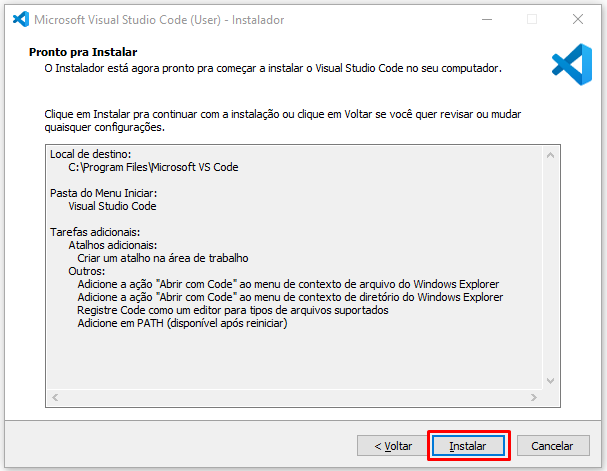
A tela seguinte serve apenas para você decidir se o Windows criará ou não uma pasta do Visual Studio Code no Menu Iniciar, ou seja, no menu que encontramos quando clicamos no símbolo do Windows na barra de tarefas do computador ou até mesmo no nosso teclado. Você pode renomear a pasta se quiser e prosseguir.



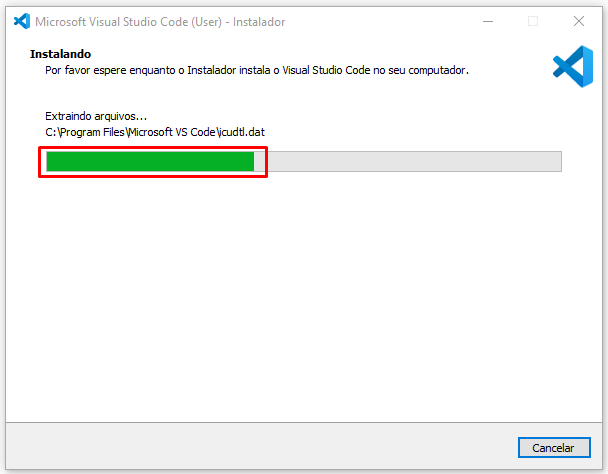
Essa próxima etapa é a seleção de tarefas adicionais que o Visual Studio Code será responsável ou capaz de fazer em seu computador. Recomendamos que marque todas as opções para poder desfrutar das diversas funcionalidades da ferramenta.



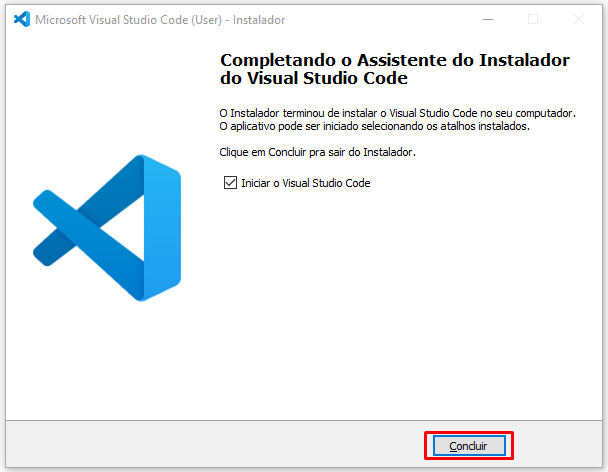
Pronto, chegamos no final da configuração de instalação e agora podemos clicar na opção “Instalar” para executar a última etapa do processo, a instalação de fato.



Na próxima tela podemos ver o progresso da instalação, ou seja, da extração dos arquivos baixados para a pasta de destino que selecionamos. Após a barra de extração se completar, podemos prosseguir para a última tela.



Chegamos na última tela, essa é apenas para informar se o Visual Studio Code foi instalado com sucesso. Você pode marcar a opção “Iniciar o Visual Studio Code” para abri-lo após o fechamento do instalador. Finalmente, ao clicar em “Concluir” o instalador será fechado e todo o processo finalizado. A partir disso você pode usar o VSCode normalmente.



Caso seu ambiente não seja Windows e tenha dúvidas durante a instalação ou no decorrer do curso, pode contar conosco criando um tópico no fórum ou interagindo no nosso servidor do [servidor do Discord](https://discord.gg/QeBdgAjXnn). Também não deixe de ajudar outros colegas. Vamos construir juntos essa grande comunidade da Alura? :)

# 05ATV - Iniciando o projeto

As boas práticas na organização de um projeto podem ser consideradas um dos pilares fundamentais nas etapas iniciais do desenvolvimento. É muito importante iniciarmos da forma correta. Levando em consideração o que vimos anteriormente, podemos afirmar que:

Parte superior do formulário

* Alternativa correta



É uma boa prática usarmos títulos não descritivos nas subpastas para que possamos englobar o máximo de material possível por pasta de forma organizada e lógica.

Independente de ser a pasta principal ou uma subpasta é importante que você sempre use títulos que deixem claro qual o conteúdo dentro daquela pasta. Lembre-se que o foco não é a quantidade de arquivos que você consegue guardar/acumular no mesmo lugar, mas sim se esses arquivos estão onde devem estar.

* Alternativa correta



É uma boa prática mantermos nossos projetos sempre na área de trabalho, pois este é o ponto de acesso mais fácil no sistema e de melhor organização.

Não existe um lugar específico para criação de pastas, por isso a criação de pasta nem sempre será na área de trabalho, crie onde você achar melhor.

* Alternativa correta



É uma boa prática criarmos pastas e subpastas que explicitem e organizem os arquivos de código de forma lógica.

Nem sempre seus projetos terão apenas dois ou três arquivos, a depender do nível de complexidade, podemos chegar na casa das centenas de arquivos. Portanto, é uma boa prática separarmos arquivos por categorias claras e objetivas fazendo uso de pastas e subpastas.

# 06 ATV - Nome do arquivo

No mundo fantasia do Pokemon, treinadores pokemon decidem criar um site eletrônico, o Pokémart.com: um site de comércio eletrônico onde os treinadores poderiam comprar e vender itens, como Pokébolas, Potions e Berries. Para começar, por enquanto, seguindo nosso padrão eles precisam criar um arquivo chamado:

Parte superior do formulário

* Alternativa correta



google.html

Esse é um nome válido de html mas não é o padrão que utilizamos quando acessamos uma página.

* Alternativa correta



index.css

Veremos como funcionam arquivos css mas não é a nossa página

* Alternativa correta



index.html

É a página principal que o navegador carrega ao visitar o site.

* Alternativa correta



google.js

Arquivos js contem código Javascript mas não é o foco dessa aula.

# 07 ATV - Criando seu arquivo HTML principal

Ao criar um novo projeto é comum seguirmos um padrão para nomear seu arquivo principal e para estruturar a hierarquia de arquivos. Das alternativas abaixo assinale a que apresenta a **hierarquia correta** de um projeto HTML:

Parte superior do formulário

* Alternativa correta



Pasta do projeto

* + index.html
  + arquivos do projeto

O termo “index” significa “índice”, pois essa página principal vai conter os links que redirecionam para as outras páginas do seu projeto. Sendo assim, esse nome facilita com que esse arquivo seja reconhecido como “padrão” dentro de sua pasta e será a primeira página carregada sempre que seu projeto for aberto.

* Alternativa correta



Arquivos do projeto

* + index.html
  + pasta do projeto

É uma boa prática criar uma pasta do projeto para guardar os arquivos de maneira organizada, por isso tanto o index.html como os outros arquivos devem estar dentro dessa pasta.

* Alternativa correta



Index.html

* + pasta do projeto
  + arquivos do projeto

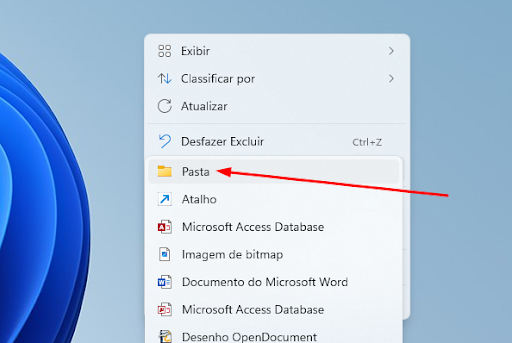
Não é possível criar uma pasta dentro de um arquivo como foi feito aqui com a pasta do projeto dentro do index.html.

# 08Para saber mais: criação de pastas

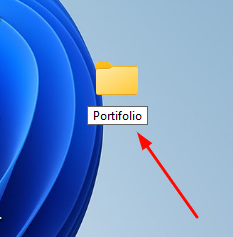
Geralmente um projeto de Front-end possui vários arquivos sendo eles (HTML, CSS, Javascript etc). Quando iniciamos precisamos criar uma pasta para organizar os nossos arquivos, pois não queremos que nenhum se perca, não é mesmo? Vamos lá:

Para criar o projeto basta criarmos uma pasta no computador, pode ser na área de trabalho mesmo:

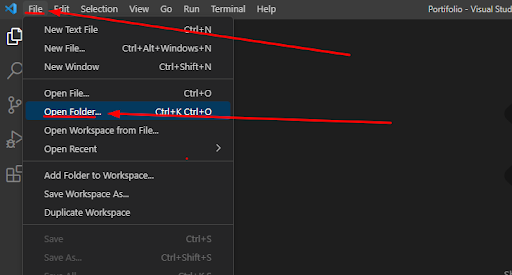
Na área de trabalho clique com o botão direito do mouse na parte vazia da tela, nas opções clique em **novo** depois em **pasta** para criar uma nova pasta projeto:



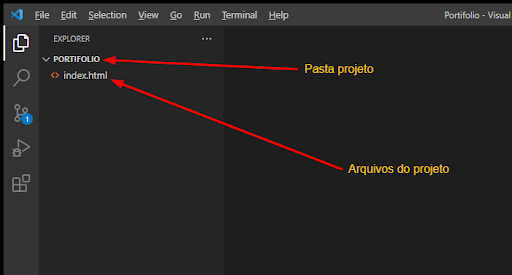
Depois definimos o nome da pasta que vai ser o mesmo nome do nosso projeto:



E agora basta abrir essa pasta dentro do VSCode para criar os arquivos nela e começar a codificar:



No VSCode podemos visualizar a pasta com o nome do projeto e os arquivos do projeto:



# 09O que aprendemos?

## Nessa aula, você aprendeu como:

* Diferenciar a criação de um arquivo no Google Docs e no HTML;
* Fazer download e instalar o Visual Studio Code;
* Criar pasta e abrir no editor de código.

# 02 - 01Preparando o ambiente

Nos próximos vídeos, os instrutores irão utilizar uma imagem de exemplo. Você pode baixar essa imagem [Link aqui](https://github.com/alura-cursos/html-logo/archive/refs/heads/main.zip).

**Logotipo, Ícone

Descrição gerada automaticamente**Parte inferior do formulário

Documentação: <https://www.w3schools.com/html/html_intro.asp>

# 03Para saber mais: documentação do HTML

## A importância da documentação

**O que é:**

A [documentação](https://www.w3schools.com/html/html_intro.asp) é um guia que toda pessoa desenvolvedora deve levar a sério no dia a dia, é através dela que aprendemos como funcionam as linguagens de programação e também ferramentas e bibliotecas no mundo da tecnologia.

**Sua importância:**

A documentação é muito importante no aprendizado e no desenvolvimento de aplicações. Afinal quem melhor que a pessoa que criou a ferramenta para nos orientar sobre as suas funcionalidades não é mesmo?

**Quando devemos utilizar:**

Devemos ler a documentação sempre que precisamos saber a estrutura de um método, ou quando queremos saber algum comando ou recurso de uma biblioteca, ou até mesmo quando esquecemos certa funcionalidade e precisamos relembrar.

**Outra forma de ajuda:**

Existem também as comunidades de tecnologia e programação que são bem úteis para tirarmos nossas dúvidas e aprendermos mais, como por exemplo a [Stackoverflow](https://stackoverflow.com/) que é uma das maiores comunidades de ajuda sobre programação e tecnologia atualmente, é uma comunidade onde você vai encontrar dúvidas sobre quase todas as linguagens de programação entre outras ferramentas, super recomendável acessá-la.

Aqui na Alura tem um artigo muito legal sobre comunidades, você pode conferir acessando [Comunidades Front-End](https://www.alura.com.br/artigos/principais-comunidades-front-end). Você pode acessar também a documentação do HTML que foi apresentada na aula através do link da w3schools. A [w3schools](https://www.w3schools.com/html/html_intro.asp) também é uma ótima opção para quem precisa aprender algum método novo ou consultar exemplos sobre determinada linguagem, é um site bem completo e de fácil compreensão.

# 02.05 ATV Adicionando uma imagem

Nessa aula, aprendemos que para inserir uma imagem em sua página é necessário utilizar a tag <img>. Pensando nisso, considere o código a seguir:

<!DOCTYPE html>

<html>

<head>

<title>Portfólio</title>

</head>

<body>

<h1>Isso é um título</h1>

<p>Isso é um parágrafo</p>

<img scr="html.png" alt="Logo do HTML 5">

</body>

</html>COPIAR CÓDIGO

Assinale as alternativas corretas a respeito do código acima:

Parte superior do formulário

* Alternativa correta



A imagem só será exibida caso exista um arquivo de imagem chamado “html.png” salvo dentro da pasta do projeto.

É comum principalmente no início, errarmos o caminho da imagem e não conseguirmos visualiza-las. Por isso, é essencial que o caminho dentro do atributo src corresponda ao nome do arquivo de imagem.

* Alternativa correta



Apenas o texto alternativo será exibido e a imagem estará indisponível, porque o atributo src da tag <img> foi escrito incorretamente.

A imagem não será localizada e não conseguiremos visualizá-la na página, pois escrevemos scr ao invés de src no atributo da tag . O atributo src vem do termo em inglês “source”, que significa fonte, pois indica o caminho onde a imagem está localizada.

* Alternativa correta



A imagem não será exibida pois a tag <img> não foi fechada corretamente com </img>, o que causará um problema no código.

A <img> é uma das tags que não precisa de fechamento, portanto o </img> não deve ser adicionado.

* Alternativa correta



O uso do atributo alt é uma boa prática, mas não fará diferença na exibição da página nesse caso específico.

Como houve um problema na exibição da imagem, o alt foi o único recurso na página para descrever a imagem que não apareceu, por isso é sempre importante usar o atributo alt mesmo não sendo obrigatório, por ser uma boa prática para a acessibilidade e manutenção da página.

# 02.06 ATV - Título

Na página web que você está criando, você deseja criar um título "Marvel Universe". Qual o código HTML correto para realizar essa tarefa?

Parte superior do formulário

* Alternativa correta



<p>Marvel Universe</p>

O parágrafo <p> é usado para texto normal, não para títulos de páginas.

* Alternativa correta



<head>Marvel Universe</head>

A tag <head> é usada para incluir informações adicionais na página, como meta informações de SEO e links para outras folhas de estilo, não para definir o título principal da página.

* Alternativa correta



<h3>Marvel Universe</h3>

O cabeçalho <h3> é uma opção válida para um título, mas não é a melhor opção para um título principal de página.

* Alternativa correta



<title>Marvel Universe</title>

A tag <title> é usada para definir o título da guia que aparece no navegador, não o título principal da página.

* Alternativa correta



<h1>Marvel Universe</h1>

O cabeçalho <h1> é a melhor opção para um título principal de página e é comumente usado para esse fim.

# 02.07Quirks Mode e Live Server

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/html-css-ambiente-arquivos-tags/task/118941/next)

**Rafaela:** Temos uma página Web mais bonita, com todos os nossos elementos. Porém, ainda podemos brincar com esse código.

**Guilherme:** Exatamente. Quando seguimos a documentação de uma linguagem, sentimos uma certa desconfiança: será que se tirarmos uma propriedade indicada, conseguiremos ter o mesmo resultado?

Vamos alimentar a nossa curiosidade e mexer no código, tirando e colocando elementos de volta no lugar. Por exemplo, vamos retirar a primeira linha: o <!DOCTYPE html>.

**Rafaela:** Vamos. Esta linha foi indicada como importante para indicar a versão do HTML.

**Guilherme:** Isso. Vamos retirá-la e salvar o código.

<html>

<head>

<title>Portfolio</title>

</head>

<body>

<h1>Isso é um título</h1>

<p>Isso é um parágrafo</p>

<img src="html.png" alt="Logo do HTML 5">

</body>

</html>COPIAR CÓDIGO

**Rafaela:** Retornaremos ao navegador e atualizaremos a nossa página. Veremos que nada foi alterado. Aparentemente tudo está funcionando.

**Guilherme:** Aparentemente, contudo, houve sim uma alteração.

Existe uma forma de verificar o código HTML e CSS de qualquer página da internet. Para isso, na tela do navegador, clicaremos com o botão direito e selecionaremos a opção "Inspecionar". Neste instante, será aberta uma aba à direita da página Web, onde existe uma barra de menus superior. Nela, selecionaremos o menu "Elements". No interior deste menu será exibido o código-fonte da nossa página index.html.

Vamos acessar esse código-fonte e perceber que, quando passamos o cursor por cima de uma tag, o navegador realça o elemento correspondente diretamente na página Web. Vamos conferir se todos os elementos que codamos estão na página: <html>, <head>, <title>, <body>, <h1>, <p> e <img>.

Aparentemente, está tudo igual. Ou seja, a retirada do <!DOCTYPE html> não mudou a parte visual da nossa página. Contudo, **houve uma alteração na forma como o navegador enxerga essa página**. Podemos ver a alteração dentro da própria aba "Inspecionar", selecionando o ícone de balão de comentário, localizado à direita da barra de menus superior. Este ícone aponta as issues da página. Após a seleção, será aberta uma aba secundária denominada "Issues", na parte inferior da aba "Inspecionar", a qual exibirá a seguinte mensagem:

Page layout may be unexpected due to Quirks Mode

Podemos traduzi-la como "O layout da página pode ser inesperado devido ao Modo Quirks". Ou seja, a página está em "modo peculiaridades". Não sabemos o que isso significa. Vamos clicar nesta mensagem para entender mais a fundo.

Abaixo da mensagem será exibida uma explicação mais detalhada do problema apontado,a qual apresentaremos uma versão resumida e traduzida:

Um ou mais documentos desta página foram abertos em um modo diferente do esperado. O Modo Quirks existe por razões históricas. Caso ele tenha sido acionado de forma não intencional, [adicione o <!DOCTYPE html>](https://developer.chrome.com/pt/docs/lighthouse/best-practices/doctype/) para renderizá-la sem o Modo Quirks.

**Afinal, o que é o Modo Quirks?** Vamos acessar o link disponibilizado acima, o qual nos direcionará para uma nova página. Na parte superior dela veremos o título "A página não tem o doctype HTML, acionando assim o Modo Quirks". Esta página possui várias informações e links sobre o Modo Quirk, o modo Standards, códigos-fonte, etc. Recomendamos que você a leia posteriormente.

Vamos resumir o problema: antigamente existiam dois grandes navegadores, o NetScape e o Internet Explorer. Quando codávamos, precisávamos informar para qual navegador a página estava sendo construída. Hoje em dia isso não é mais necessário, a não ser que seja algo intencional — como em casos de projetos antigos.

Na [página sobre Quirks Mode e Standards Mode](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Quirks_Mode_and_Standards_Mode) existe a seção "Como os navegadores determinam qual modo usar?", onde há uma explicação sobre a questão da determinação de modos. Nela veremos que, para informarmos ao navegador que utilizaremos o HTML5, precisamos adicionar o <!DOCTYPE html>.

**Rafaela:** Atualmente, todos os navegadores sabem utilizar o HTML5.

**Guilherme:** Isso mesmo. Vamos sair do modo peculiaridades.

**Rafaela:** Vamos fechar as abas de explicação e a aba "Inspecionar". Em seguida retornaremos ao VS Code.

**Guilherme:** Para sair desse modo, pressionaremos "Ctrl + Z" para recuperar o DOCTYPE que havíamos deletado.

<!DOCTYPE html>

<html>

<head>

<title>Portfolio</title>

</head>

<body>

<h1>Isso é um título</h1>

<p>Isso é um parágrafo</p>

<img src="html.png" alt="Logo do HTML 5">

</body>

</html>COPIAR CÓDIGO

Salvaremos esse código e voltaremos ao nosso projeto aberto no navegador. Nele pressionaremos "F5" para atualizar a página e abriremos novamente a aba "Inspecionar". Veremos que, à direita da barra de menus, não existe mais o ícone de issues.

Existe um ponto que incomoda em nossa página: **a necessidade de atualizá-la sempre que alteramos o código**.

**Rafaela:** Vamos adicionar uma **extensão** ao nosso editor de código para que ocorram atualizações automáticas na nossa página a cada salvamento. Acessaremos a coluna lateral esquerda do nosso VS Code, onde selecionaremos o ícone "Extensions" ("Extensões"). Podemos acessá-lo também por meio do atalho "Ctrl + Shift + X".

Neste momento, o explorador de arquivos será substituído por um campo de busca. Nele digitaremos "live server" e faremos a busca. Nos resultados abaixo do campo, buscaremos a opção "Live Server", a qual possui milhões de downloads.

**Guilherme:** Muitas pessoas a utilizam.

**Rafaela:** Nós, inclusive. Após a seleção, essa extensão será aberta na tela principal do VS Code, onde clicaremos no botão "Install". Após o término da instalação, Ela será habilitada automaticamente. Caso contrário, basta clicar no botão "Enable". Em seguida fecharemos a aba dessa extensão.

Existem extensões para milhares de tarefas, inclusive para embelezar sintaxes de linguagens específicas. Podemos explorá-las e utilizá-las sempre que quisermos.

**Guilherme:** Como a extensão funciona?

**Rafaela:** Agora temos um botão na barra inferior do VS Code denominado "Go Live". Como o arquivo HTML selecionado, clicaremos neste botão. Uma aba de Alerta de Segurança será aberta pelo nosso sistema, na qual selecionaremos a opção "Permitir acesso". Neste momento será aberta uma aba com a nossa página HTML no nosso navegador.

**Guilherme:** Vamos manter as duas abas abertas. Veremos que ambas estão rodando de locais diferentes: a antiga está rodando no local do arquivo, enquanto a nova (com live server ativo) exibe a URL com um número constituído de 10 dígitos.

Retornando ao VS Code, alteraremos o parágrafo adicionando "interessante" ao final da frase.

<p>Isso é um parágrafo interessante</p>COPIAR CÓDIGO

Salvaremos esse código e voltaremos ao navegador. Na aba nova, veremos a nossa alteração.

**Rafaela:** Na aba antiga, a mudança só aparecerá se atualizarmos a página.

**Guilherme:** A principal vantagem do live server é não precisar atualizar a página a todo momento.

**Rafaela:** O único passo necessário é salvar o arquivo no editor.

# 02.08Quirks Mode

Como vimos nessa aula, o uso do <!DOCTYPE html> é muito importante para que o navegador utilize o modo padrão e não o “Quirks Mode”. Pensando nisso, assinale a alternativa correta:

Parte superior do formulário

* Alternativa correta



O <!DOCTYPE html> é responsável por evitar o Quirks Mode pois informa ao navegador que o arquivo que está sendo exibido é do tipo HTML e não um arquivo com outra extensão, como por exemplo um arquivo de texto, que seria TXT.

O papel principal do <!DOCTYPE html> é informar ao navegador sobre a versão do HTML, no caso, a versão 5 e não necessariamente a extensão desse arquivo.

* Alternativa correta



Evitamos o Quirks Mode utilizando o <!DOCTYPE html> para declarar a versão do HTML, porém essa declaração não funciona se não escrevermos “DOCTYPE” em maiúsculas e html em minúsculas.

O <!DOCTYPE html> funciona normalmente se adotarmos outra maneira de declarar, como por exemplo: <!doctype html>, <!DOCTYPE HTML> e até <!DoCtYpE hTmL>. O que acontece é que existe um acordo de boas práticas para utilizar o <!DOCTYPE html> como padrão. .

* Alternativa correta



O Quirks Mode é o modo em que o navegador adapta páginas web que estão em versões antigas para que funcionem, o que pode quebrar sua página em HTML 5. Portanto, esse modo deve ser evitado através do uso do <!DOCTYPE html>.

O Quirks Mode (modo peculiaridade) adapta as páginas antigas que não estão na versão 5 do HTML para que funcionem normalmente, causando quebras nas páginas HTML 5. Para evitá-lo basta aplicar o <!DOCTYPE html> antes de abrir a tag <html>, fazendo com que o navegador reconheça a versão 5 do HTML e funcione em modo padrão.

# 02.09Para saber mais: extensões do VSCode

Agora que você conheceu o Live Server, que tal explorar mais algumas extensões que podem facilitar seu dia a dia no desenvolvimento com o VSCode?

Para isso, preparamos o artigo [Extensões VS Code: descubra quais são as mais usadas](https://www.alura.com.br/artigos/extensoes-vs-code-descubra-as-mais-usadas), para que você conheça outras extensões e escolha aquelas que preferir para auxiliar em seus estudos!

# 02.10Para saber mais: estruturando uma página HTML

## A estrutura básica do HTML

Para criar um arquivo HTML funcional devemos seguir um padrão de construção, utilizando um conjunto de elementos, ou seja, os hipertextos, que se conectam entre si formando a página. Como já visto anteriormente, os elementos HTML ou também chamados de tags HTML, são utilizados para informar ao navegador que tipo de estrutura é essa que está sendo construída, podendo ser títulos, parágrafos, imagens, links, entre outros.

Um exemplo prático dessa estrutura básica seria:

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

</body>

</html>COPIAR CÓDIGO

É importante entender como funciona e o que faz cada elemento dentro da estrutura básica do HTML, e para isso deixamos como recomendação o artigo [O que é o HTML e suas tags? Parte 1: estrutura básica](https://www.alura.com.br/artigos/o-que-e-html-suas-tags-parte-1-estrutura-basica?utm_source=gnarus&utm_medium=timeline). Lembrando que esse artigo compõe uma série de cinco artigos sobre o que é HTML e suas tags, os quais podem ser encontrados aqui na plataforma da Alura e são perfeitos para desenvolvedores HTML de primeira viagem.

# 02.11O que aprendemos?

## Nessa aula, você aprendeu:

* A importância da documentação W3S;
* O que é HTML e porque é considerada uma linguagem de marcação;
* Estruturar um documento HTML com tags e elementos;
* A utilidade da introdução <!DOCTYPE html>;
* Diferença entre a metainformação presente no <head> e o conteúdo presente no <body> de uma página HTML;
* Criar textos alternativos (alts) para uma imagem;
* Acessar a Developer Tools (Ferramentas para Desenvolvedores) de um navegador;
* Quirks mode (modo peculiaridade);
* Utilizar extensões no Visual Studio Code (Live Server por exemplo).

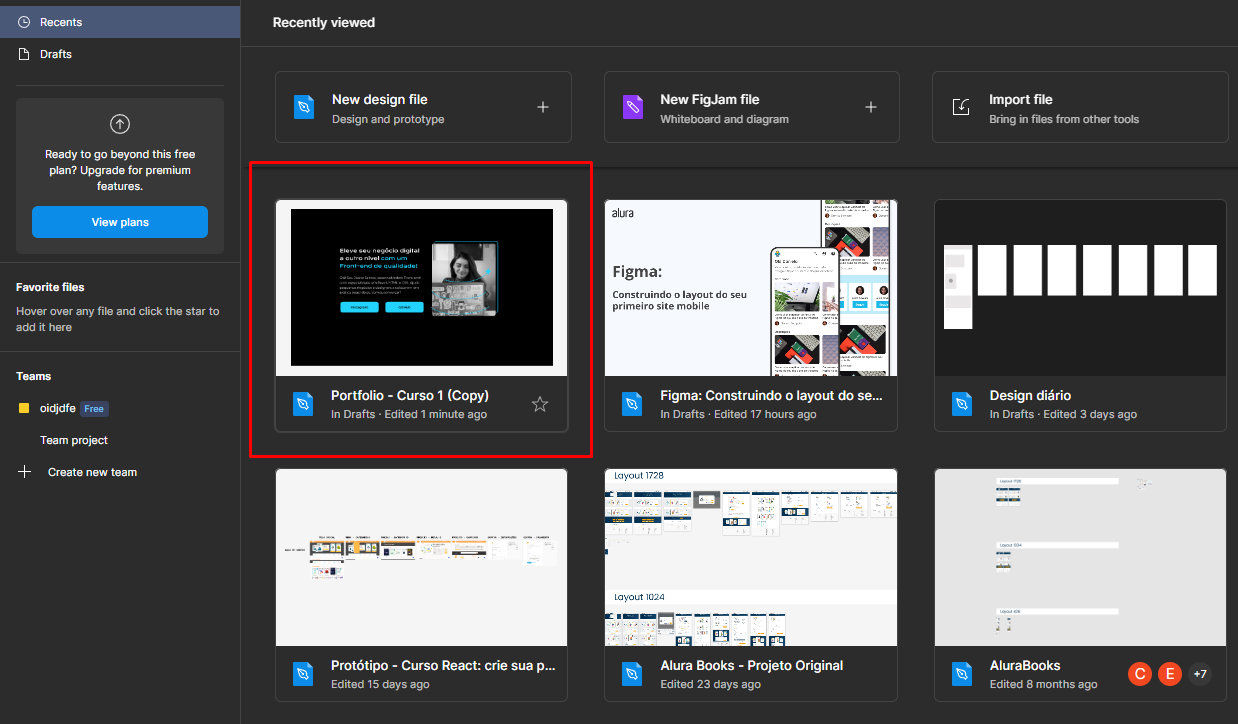
# 03. 02 Preparando o ambiente

As próximas aulas serão baseadas no Figma, que é uma ferramenta de prototipagem de interfaces, onde vamos construir o layout da página do Portfólio. Para acessá-lo, clique [aqui](https://www.figma.com/file/4EKKCbr5rS93RWP7kRjXIz/Portfolio---Curso-1?node-id=0%3A1&t=l5RuWCe75yFNoWEr-0).

Para ter a mesma visualização que a da aula, primeiramente é necessário criar uma conta ou fazer login em uma existente. Para isso, entre no site do [Figma](https://www.figma.com/) e clique em *Log In* (caso já tenha conta) ou em *Sign Up* (para cadastrar nova conta).

Depois de estar com uma conta conectada, é a hora de criar uma cópia do layout para a sua conta do Figma, assim garantindo o acesso de edição. Para fazer isso, entre no link do layout original que está acima e no menu superior da plataforma, clique no nome do arquivo: Portfolio - Curso 1. Irão abrir algumas opções, clique em *“Duplicate to your drafts”*.





Fazendo um duplo clique nele você conseguirá abrir o projeto. É com esse arquivo que você irá acompanhar as aulas.

E para demonstrar como uma pessoa desenvolvedora Front-end utiliza o Figma, separamos [esse Alura+ com a Rafa Ballerini](https://cursos.alura.com.br/extra/alura-mais/como-front-end-utiliza-o-figma-c858) onde você aprenderá tudo o que é necessário para analisar um documento do Figma para transformá-lo em código, desde a exportação de imagens e ícones até a estilização com CSS.

# 03 .03Projeto no Figma

## Transcrição

**Rafaella:** Já temos todas as ferramentas para começarmos nosso projeto de fato, pois até agora só fizemos testes, aprendemos como escrever títulos e configuramos o VSCode para nosso desenvolvimento.

Como pessoas desenvolvedoras, criamos uma página de portfolio com estilos de cores e fontes próprias, ou nos baseamos em algum modelo pronto?

**Guilherme:** Geralmente, o mercado e o cotidiano de quem trabalha com desenvolvimento é dividido em dois tipos de profissionais: Há a pessoa que faz o **desenho da página** e indica onde estarão as imagens, ícones, cores e tamanhos, e que passa para a segunda pessoa que irá **codificar o design** com HTML, CSS, React, JavaScript e etc.

**Rafaella**: Em nosso caso, somos este segundo tipo de profissional que recebe um projeto já desenhado e tem a tarefa de codar a página web.

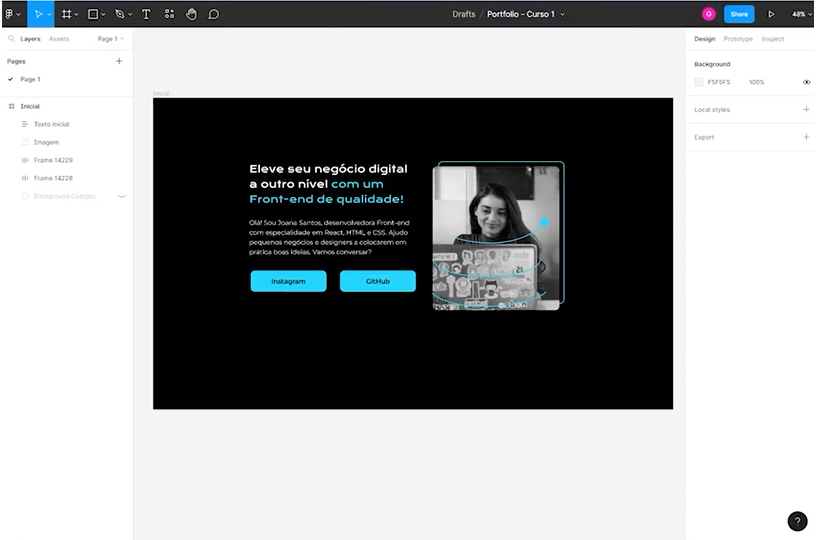
A pessoa que desenha a página "do zero" faz toda uma **pesquisa** sobre ideias e direções para construir o design, estudando os porquês de cada elemento possuir uma cor e um formato com base na importante Experiência do Usuário ou **UX/UI**.

**Guilherme:** A Isa, nossa designer aqui da Escola de Front-End da Alura produziu o layout que estamos trabalhando, e ela de fato desenhou e pesquisou bastante para isso.

**Rafaella:** Na prática, existem várias ferramentas que essas pessoas utilizam para construir páginas e que servem como base para profissionais de desenvolvimento.

Uma das mais utilizadas é o **Figma**, justamente o que estamos usando para desenvolver.

A Isa já nos enviou o [link do projeto no Figma](https://www.figma.com/file/4EKKCbr5rS93RWP7kRjXIz/Portfolio---Curso-1?node-id=0%3A1&t=l5RuWCe75yFNoWEr-0) e conseguimos ter todo o acesso às informações necessárias para codificarmos as páginas neste curso.



**Guilherme:** Se observarmos a tela inicial, os espaçamentos, a imagem, as cores e os demais elementos foram elaborados a partir de estudos.

Se quisermos saber mais sobre layouts, sobre o Figma, criação de telas e outros assuntos das áreas de **Experiência de Usuário** e **Interface de Usuário**, ou User Experience (UX) e User Interface (UI) em inglês, há diversos cursos em formações muito interessantes aqui da **Alura**.

Nosso desafio é o segundo passo do processo, pois alguém já fez o design da página e sua estrutura, mas como podemos trabalhar a partir das informações do Figma?

**Rafaella:** Primeiramente, quando abrirmos o projeto através do link disponibilizado neste curso, precisaremos ter o cadastro na plataforma para acessar todos os dados.

Além de conseguirmos visualizar a página já montada, pegaremos as informações ao clicarmos nos elementos. Por exemplo, clicando no botão de "Instagram", abriremos uma aba lateral direita contendo três abas no topo: "Design", "Protótipo" ou "Prototype" e "Inspecionar" ou "Inspect".

Basicamente utilizaremos mais a primeira e a última aba durante nosso trabalho.

Na aba de "Design", encontraremos informações sobre largura em "W:" e altura em "H:", o formato do botão, as cores que são utilizadas, entre outras.

Já na aba "Inspect", teremos outras formas de valores para as mesmas propriedades. Inclusive, na parte "Code", há um código CSS, mas não veremos isso agora.

Portanto, conseguiremos obter todos os dados necessários nesta ferramenta chamada Figma.

**Guilherme:** Nosso desafio é codar o HTML das telas já construídas.

**Rafaella:** Exatamente, as transformaremos em **páginas web** utilizando tanto o **HTML** quanto o **CSS**.

Faremos a **estrutura dos elementos** com o primeiro, e aplicaremos a **estilização** com o segundo, adicionando cores, formatos, fontes, posicionamento e etc.

Pensando de fato no HTML e observando a página inicial no Figma, teremos o título principal no topo do bloco de código à esquerda. Abaixo, há um parágrafo seguido de dois botões, além da foto de nossa desenvolvedora da Alura à direita, mas é interessante termos uma imagem própria para nosso portfólio.

**Guilherme:** Inclusive poderemos alterar o conteúdo textual, as cores e outras propriedades se quisermos.

A seguir, vamos começar a trabalhar com o HTML!

# 03.04Tags semânticas

## Transcrição

**Rafaella:** Iremos iniciar nosso projeto e **transformar o design do Figma em código**.

**Guilherme:** Vamos abrir o arquivo index.html no VSCode. Sempre que criarmos uma página HTML, devemos escrever <!DOCTYPE html>, <html>, <head>, <body> e fechá-los?

**Rafaella:** Existem alguns atalhos que nos permitem iniciar um documento .html com essa estrutura básica.

Para entendermos, excluiremos todo o conteúdo do arquivo, salvaremos e notaremos que não há mais nada da página no navegador, lembrando que ainda estamos rodando o Live Server.

Em seguida, digitaremos apenas um ponto de exclamação ! na primeira linha do documento e exibiremos uma janela de título "Emmet Abbreviation" contendo todo um código. Quando simplesmente apertarmos a tecla "Enter", toda a estrutura básica do HTML aparecerá.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

</body>

</html>COPIAR CÓDIGO

Também perceberemos que há mais coisas do que antes, como as três informações <meta> e lang="en". Entenderemos o que são essas novas tags.

**Guilherme:** Isso é tão comum que criaram este atalho para gerar o texto automaticamente.

Fazendo uma primeira leitura, ao posicionarmos o cursor sobre lang="en", exibiremos uma caixa de diálogo com um texto explicando que este trecho define a língua oficial do elemento que estamos trabalhando, que neste caso está em inglês.

**Rafaella:** Utilizaremos em português e escreveremos o conteúdo usando os acentos específicos que não são usados com "en", então substituiremos por "pt-br" para especificar que é do Brasil.

**Guilherme:** Na quarta linha, encontraremos a primeira tag <meta> seguido de charset="UTF-8".

**Rafaella:** O **UTF-8** é a **codificação dos caracteres** mais usada no mundo, pois consegue trazer todos os acentos que comentamos, além de interpretar corretamente as outras línguas.

**Guilherme:** É **Unicode**, como se fosse um grande dicionário de caracteres que usaremos na aplicação.

**Rafaella:** Se não utilizarmos o UTF-8, pode ser que não consigamos exibir acentos como ~ ou circunflexo ^ corretamente em alguns navegadores.

**Guilherme:** Entendi! E na quinta linha com a segunda <meta> que tem http-equiv="X-UA-Compatible" e content="IE=edge", é algo para o **Microsoft Edge**?

**Rafaella:** Exato, esta é uma configuração para sempre utilizarmos sua versão mais recente neste navegador especificamente.

Caso estejamos rodando o código no Edge, é importante que coloquemos esta informação <meta>, além de que a pessoa usuária final que vai usar nossa aplicação pode estar rodando nele também, principalmente se forem versões anteriores.

**Guilherme:** Já na sétima linha, temos o <title> que já conhecemos, contendo Document. Na sexta linha com o terceiro <meta> que tem name="viewport" e o content que inclusive fica com a parte final encoberta pelo limite da tela do VSCode.

Se apertarmos as teclas "Alt + Z" com o cursor sobre esta linha, iremos fazer com que este trecho passe para a linha seguinte e fique totalmente visível dentro do espaço de texto.

Aparentemente, o conteúdo de content parece dizer respeito à largura e ao tamanho da imagem.

**Rafaella:** Exato, o nome "viewport" é o próprio **dispositivo** que acessará nossa página, e obtém sua **largura** com width=device-width para depois gerar a **escala** da página com initial-scale, que sempre iniciará com 1.0 com base na tela, seja de celular, Ipad, computador, entre outras.

Com isso, não teremos o problema dos elementos ficarem posicionados de forma errada e nem da fonte ter um tamanho desproporcional.

**Guilherme:** Cientes disso, poderemos começar alterando o conteúdo do <title>.

**Rafaella:** Escreveremos Portfolio mesmo, que é o nosso projeto. Salvaremos e abriremos o navegador para exibirmos este nome na aba. Como é uma **metainformação**, não aparecerá no corpo da página. Agora começaremos a desenvolver todos os elementos dela.

**Guilherme:** Mas só iremos escrever HTML diretamente?

**Rafaella:** Talvez sim, e há pessoas que fazem isso apesar de **não** ser recomendado, pois sempre teremos que pensar e planejar o que iremos desenvolver em nossa página web.

Por exemplo, é interessante analisarmos o Figma e separar nossas tarefas em partes, como pelo cabeçalho da página no topo, ou o rodapé na base, e há também o conteúdo principal no centro da tela, como no nosso caso.

No HTML, utilizamos algumas tags de maneira **semântica** para separarmos de fato os elementos da página conforme as estruturas que temos. A que seguiremos será pelo cabeçalho, conteúdo principal e o rodapé.

**Guilherme:** Se acessarmos a própria plataforma da Alura, encontraremos esses elementos. Temos o menu onde fazemos o login, o conteúdo principal ao centro e o footer na base.

**Rafaella:** Às vezes precisaremos descer ou scrollar para o final da página, mas este é o básico.

**Guilherme:** Então vamos criar esta estrutura! Como cada um desses elementos são feitos por tags que possuem conteúdos internos, elas devem ser abertas com < e fechadas com />.

**Rafaella:** Como são tags estruturais, sim.

Primeiramente, no topo da página, teremos o menu com **<header>** que significa "cabeçalho", que é **diferente** de <head>. A inseriremos dentro de <body>, abrindo e fechando na mesma linha.

Em seguida, teremos o conteúdo principal em **<main>** na linha seguinte, que é a própria tradução do inglês.

Dentro desta tag, iremos colocar o título, o parágrafo, os botões e a imagem.

Na linha seguinte, teremos o <footer> relativo a "rodapé", também abrindo e fechando.

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Portfolio</title>

</head>

<body>

<header></header>

<main></main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Guilherme:** Com isso, teremos a estrutura básica do HTML, e vamos começar com o <main>.

# 03.05Desenvolvendo o HTML

## Transcrição

**Rafaella:** Vamos **começar a codar** dentro do nosso **<main>**.

O primeiro passo é **entender** o que precisamos fazer em nossa página web e **planejar** o que iremos estruturar.

**Guilherme:** Observando a tela inicial do projeto no Figma, notaremos dois blocos de texto um em cima do outro, dois botões lado a lado logo abaixo de "Instagram" e "GitHub", e uma imagem à direita desses elementos.

**Rafaella:** Uma boa sugestão é começar a trabalhar com os elementos **da esquerda para direita, de cima para baixo**. Então começaremos pelo título, depois pelo parágrafo, seguido pelos botões, e por fim colocaremos a foto.

Vamos fazer isso no nosso arquivo index.html no VSCode.

Abriremos uma nova linha dentro da tag <main> e começaremos com o título usando **<h1>**, em que "h" remete a "heading".

Voltaremos ao Figma e clicaremos várias vezes sobre o texto para o selecionarmos inteiro, para depois copiarmos o escrito com "Ctrl + C" e colarmos dentro de <h1> com "Ctrl + V".

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Portfolio</title>

</head>

<body>

<header></header>

<main>

<h1>Eleve seu negócio digital a outro nível com um Front-end de qualidade!</h1>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Rafaella:** Outro detalhe importante é que, ainda na parte do título, o trecho "com um Front-end de qualidade!" possui uma cor diferente que o destaca do começo do texto.

Para fazer essa diferenciação, existe uma tag semântica em nosso próprio HTML que indica ao navegador que deve destacar essa parte.

**Guilherme:** É para deixarmos este trecho mais em forte evidência mesmo.

**Rafaella:** Para isso, antes desta parte de destaque, aplicaremos a **<strong>**, que é justamente a palavra em inglês para "forte", e a fecharemos antes do fechamento da <h1>, pois a tag está dentro dela.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

<h1>Eleve seu negócio digital a outro nível<strong>com um Front-end de qualidade!</strong></h1>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Rafaella:** Salvaremos esse código e veremos o resultado no navegador.

**Guilherme:** Visualmente, não há diferença entre os trechos, mas o navegador já sabe que a parte importante deve ser destacada.

**Rafaella:** Trabalharemos com essas questões visuais quando entrarmos na parte de **estilização**.

Em seguida, observando o Figma, teremos o nosso parágrafo que também copiaremos e, logo abaixo da </h1> no index.html, adicionaremos a tag **<p>** de "parágrafo" que já conhecemos para então colarmos o texto dentro dela.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

<h1>Eleve seu negócio digital a outro nível<strong>com um Front-end de qualidade!</strong></h1>

<p>Olá! Sou Joana Santos, desenvolvedora Front-end com especialidade em React, HTML e CSS. Ajudo pequenos negócios e designers a colocarem em prática boas ideias. Vamos conversar?</p>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Guilherme:** Lembrando que o escrito pode mudar, afinal nem todo mundo se chama "Joana Santos". Vamos salvar e ver o resultado.



**Rafaella:** Com isso, já teremos as duas partes de texto em nossa página, e agora vamos colocar os dois botões. Mas o que são de fato?

**Guilherme:** Excelente pergunta; quando clicamos em "Instagram" ou "GitHub", o esperado é que nos redirecionem para outro lugar. Mas esta questão de **redirecionamento** para outra página geralmente **não é o comportamento de um botão**.

Existe uma tag do HTML chamada <Botao> que é usada quando estamos trabalhando com formulário, por exemplo. Semanticamente, neste nosso caso, usaremos a tag âncora **<a>**.

**Rafaella:** Certo, então a adicionaremos após </p> e sabemos que ela abre e fecha também. Mas o que colocamos como conteúdo?

**Guilherme:** Colocaremos o link. Se voltarmos ao código, o que escrevermos dentro dela como um texto é o que será exibido na tela, ou seja, Instagram. Inseriremos outra tag <a> abaixo e adicionaremos Github para salvarmos e vermos o resultado.'

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

<h1>Eleve seu negócio digital a outro nível<strong>com um Front-end de qualidade!</strong></h1>

<p>Olá! Sou Joana Santos, desenvolvedora Front-end com especialidade em React, HTML e CSS. Ajudo pequenos negócios e designers a colocarem em prática boas ideias. Vamos conversar?</p>

<a>Instagram</a>

<a>Github</a>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Guilherme:** Na página no navegador, notaremos que aparece o texto "Instagram Github".

**Rafaella:** Mas não está clicável, e parece apenas um outro parágrafo.

**Guilherme:** Está exatamente igual, e não é esse comportamento que esperamos, pois queremos poder clicar e nos redirecionar para o Instagram ou o Github.

Poderemos colocar o Instagram da Rafaella para exemplificar.

**Rafaella:** A tag âncora é bastante semelhante à propriedade que estávamos colocando de imagem com **src**, que vem de source ou "fonte", e que ser para buscar o caminho para encontrá-la.

Em <a>, vamos usar um hiperlink que nos permite clicar e ir para outra pagina. Escrevendo **href** entre a e > na abertura da primeira tag, e já exibiremos uma caixa com uma descrição.

Depois do sinal de igual =, escreveremos a URL de destino entre aspas para onde vamos nos redirecionar. Neste exemplo, será "https://instagram.com/rafaballerini", depois salvaremos e veremos o que vai acontecer.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

//código omitido

<a href="https://instagram.com/rafaballerini">Instagram</a>

<a>Github</a>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO



**Guilherme:** Já mudou e visualmente está diferente.

**Rafaella:** Já se tornou um elemento clicável com o destaque sublinhado em azul, e quando clicamos, nos redirecionaremos para a página do Instagram que colocamos em href.

**Guilherme:** No botão de "Github", pode colocar o meu mesmo como exemplo para vermos.

**Rafaella:** Teremos o mesmo formato usando href igual a "https://github.com/guilhermeonrails". Vamos salvar e ver o resultado no navegador.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

//código omitido

<a href="https://instagram.com/rafaballerini">Instagram</a>

<a href="https://github.com/guilhermeonrails">Github</a>

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Rafaella:** Quando clicarmos em "Github" também já clicável e sublinhado em azul, iremos para a página do GitHub do Guilherme.

**Guilherme:** Revisando o que fizemos, construímos tudo isso dentro do <main> até agora.

Na tag <strong>, pegamos o texto em destaque no Figma, mas visualmente não há diferença no navegador. Mesmo não tendo alterado nada, já indicou que é algo a ser destacado.

Também criamos o parágrafo com <p>, além dos dois links que redirecionam para o Instagram e GitHub, que não são literalmente dois botões.

Deixaremos essas duas âncoras <a> para outros sites com aparência de botão apenas, como está na tela no Figma.

**Rafaella:** Exatamente. Nosso último passo é inserir a fotografia da direita, e conseguimos fazer seu download direto do Figma, afinal não a encontraremos no Google.

Mas poderemos usar qualquer outra imagem que tivermos salvas no computador.

Para pegarmos direto do Figma, clicaremos sobre o elemento que queremos baixar e, na aba lateral direita chamada "Design", teremos a opção de "Export" onde clicaremos no botão de "Export Imagem" para fazermos o download.

Abriremos a pasta em que o arquivo foi salvo e poderemos renomear se quisermos, como apenas "imagem" mesmo para padronizarmos.

Clicando e arrastando a imagem para dentro do VSCode, criaremos o arquivo imagem.png na lista lateral esquerda chamada "Explorer", da mesma forma que fizemos com o HTML 5.

Para a adicionarmos no código, inseriremos a tag **<img>** que já conhecemos e sabemos que é única, então não precisamos abrir e fechar.

A propriedade usada para encontrar a imagem é a **src** que será igual a "imagem.png" entre aspas, pois já está na mesma pasta do projeto, bastando só escrever o nome do arquivo.

Além disso, usaremos outra importante propriedade chamada **alt** que diz respeito ao texto alternativo que descreve a imagem, como "Foto da Joana Santos programando" por exemplo.

Vamos Salvar e ver o resultado no navegador.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header></header>

<main>

//código omitido

<a href="https://instagram.com/rafaballerini">Instagram</a>

<a href="https://github.com/guilhermeonrails">Github</a>

<img src="imagem.png" alt="Foto da Joana Santos programando">

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO



**Rafaella:** Ainda não está como o projeto do Figma porque ainda temos muito o que estilizar, e é o que faremos a seguir.

# 03. 06Para saber mais: tags semânticas

Quando começamos um arquivo HTML, há uma estrutura padrão que é usada em qualquer projeto. É importante saber quais são as tags que precisam ser implementadas e entender suas funções dentro do código. Para facilitar esse processo, utilizamos as **tags semânticas**, que são tags descritivas sobre o conteúdo que armazenam, como é o caso das tags <header>, <main> e <footer>, que conhecemos nessa aula. Elas servem tanto para otimizar a leitura pelos navegadores, como pelas pessoas desenvolvedoras que vão fazer a manutenção do código.

Para aprender mais sobre as tags que fazem parte da base de um arquivo HTML, você pode ler a documentação MDN [“Semântica”](https://developer.mozilla.org/pt-BR/docs/Glossary/Semantics) e conhecer outros elementos semânticos disponíveis para tornar o seu código mais claro, seja para outras pessoas programadoras, para navegadores ou mecanismos de buscas.

# 03. 07Sobre os metadados

Quando construímos a estrutura básica de tags do HTML, inserimos meta tags dentro da tag <head>. As tags meta determinam os metadados, que são as informações sobre dados de um documento HTML. Logo abaixo, podemos visualizar os metadados que colocamos no código do projeto Portfolio.

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">COPIAR CÓDIGO

Dadas as afirmações abaixo, selecione as alternativas corretas sobre metadados:

Parte superior do formulário

* Alternativa correta



O metadado http-equiv="X-UA-Compatible" content="IE=edge" é utilizado para configurar a página web no Internet Explorer, para que ela esteja sempre em sua versão mais recente.

Exatamente esse metadado é utilizado para configurar a página para ser renderizada como em outra versão do Internet Explorer. Caso não definirmos esse metadado, o Internet Explorer, especificamente, pode encontrar problemas de compatibilidade, por isso é importante utilizarmos essa tag meta pois o usuário final pode estar utilizando este navegador.

* Alternativa correta



O metadado viewport é usado para informar a página que a pessoa usuária está abrindo o site em um dispositivo móvel.

Informamos para o navegador as medidas da tela para definir a largura que será renderizada na página, mas não remetemos qual o dispositivo que está usado.

* Alternativa correta



O metadado charset="UTF-8" é empregado para repassar aos navegadores qual é o formato de codificação de caracteres utilizado naquele documento.

Ao utilizarmos a tag meta charset="UTF-8”, estamos comunicando aos navegadores que a codificação daquela página é a UTF-8, o padrão de codificação mais utilizado, também conhecido por Unicode. Com esse padrão as informações são unificadas ao redor do mundo e os navegadores renderizam todos os conteúdos a partir das regras estabelecidas pelo UTF-8.

# 03. 08Usando tags semânticas

Vimos nas aulas que para escrever um título, um parágrafo, precisamos respeitar uma estrutura básica do HTML. Diante disso, qual das alternativas abaixo contempla a estrutura correta para obtermos como resultado um título e dois parágrafos conforme o exemplo abaixo:

## ALURA - Mergulhe em Tecnologia!

#### **Você vai estudar, praticar, discutir e se aprofundar em uma plataforma que respira tecnologia.**

#### **Mergulhe com profundidade e navegue em outras áreas de Tecnologia. Profissional em T.**

Parte superior do formulário

* Alternativa correta



<title>ALURA - Mergulhe em Tecnologia!</title>

<p>Você vai estudar, praticar, discutir e se aprofundar em uma plataforma que respira tecnologia.</p>

<p>Mergulhe com profundidade e navegue em outras áreas de Tecnologia. Profissional em T</p>

A tag <title> define o título do documento e é mostrado na barra de título do navegador ou na aba da página, não usamos para criar um título em destaque. Esta tag é utilizada apenas dentro da tag <head>, e não da tag <body>.

* Alternativa correta



<h1>ALURA - Mergulhe em Tecnologia!<h1>

<p>Você vai estudar, praticar, discutir e se aprofundar em uma plataforma que respira tecnologia.<p>

<p>Mergulhe com profundidade e navegue em outras áreas de Tecnologia. Profissional em T<p>

As tags desta alternativa não foram fechadas corretamente com **/**. É muito importante não esquecermos desse detalhe para evitarmos erros no resultado final.

* Alternativa correta



<h1>ALURA - Mergulhe em Tecnologia!</h1>

<h2>Você vai estudar, praticar, discutir e se aprofundar em uma plataforma que respira tecnologia.</h2>

<p> Mergulhe com profundidade e navegue em outras áreas de Tecnologia. Profissional em T </p>

Usamos a tag <h2> para subtítulos, ela não tem a mesma força que o <h1>, mas ainda assim dá um destaque maior para o texto em comparação com as tags de corpo do texto. Sendo assim, não é a tag que precisamos usar para obter o resultado desejado.

* Alternativa correta



<h1>ALURA - Mergulhe em Tecnologia!</h1>

<p>Você vai estudar, praticar, discutir e se aprofundar em uma plataforma que respira tecnologia.</p>

<p> Mergulhe com profundidade e navegue em outras áreas de Tecnologia. Profissional em T </p>

Isso mesmo, para criarmos um título em destaque abrimos a tag com <h1> e fechamos com </h1>. E para adicionar parágrafos usamos as tags <p> também fechando com </p>.

# 03. 09Diferença entre âncora e botão

Vimos que para desenvolver diferentes funcionalidades do HTML precisamos entender o comportamento das tags, e que, embora muitas vezes duas ou mais tags sejam parecidas em nome ou finalidade, cada uma é específica à certa situação. Dessa maneira, pensando nas tags <button> de botão e <a> de âncora, marque a alternativa correta que indique a diferença entre elas:

Parte superior do formulário

* Alternativa correta



As duas tags são diferentes em nome, porém idênticas em finalidade, já que ambas são usadas para indicar que existe um botão clicável pronto para executar uma ação.

As tags têm nomes e finalidades diferentes. Lembre-se que em HTML, cada tag tem a sua funcionalidade e precisamos saber o momento certo de utilizá-las.

* Alternativa correta



A tag <button> é diferente da tag <a>, pois, além da semântica, a finalidade também é outra. Usamos <button> para criar um botão de ação e <a> para indicar um link.

As tags são diferentes em ambos os aspectos e é necessário saber utilizar cada uma na sua função correta. Enquanto <button> pode ser utilizada para ações como envios de formulários, a tag <a> não possui essa funcionalidade, já que seu papel é apenas redirecionar o usuário para diferentes urls.

* Alternativa correta



As duas tags são completamente diferentes, pois enquanto a <button> indica um link clicável que leva o usuário para outra página, a <a> é responsável pela criação visual de um botão.

Apesar das tags serem de fato diferentes, as funções de cada uma não foram apresentadas da forma correta.

Parte inferior do formulário

# 03. 10O que aprendemos?

## Nessa aula, você aprendeu:

* Como consultar o layout do projeto no Figma;
* Escrever o código base do arquivo HTML, usando as tags semânticas que fazem parte da estrutura básica do arquivo;
* A função de cada tag meta.

# 04. 01Projeto da aula anterior

Aqui você pode [baixar o zip da aula 03](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/archive/refs/heads/aula_3.zip) ou [acessar os arquivos no Github](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/tree/aula_3)!

# 04. 02Como funciona o CSS

## Transcrição

**Guilherme:** Já criamos um HTML semântico, e escrevemos as estruturas e as marcações de todo o conteúdo que queremos realizar, mas não está visualmente bem finalizado.

**Rafaella:** Os elementos já estão estruturados, mas falta colocar as cores e formatos que fazem uma página web.

**Guilherme:** Isso faz muita diferença, pois há anos atrás, a parte visual não era tão desenvolvida como atualmente, tanto que há uma área profissional que se dedica somente a esta parte de criação.

O desafio desta aula é deixarmos a página mais bonita e parecida com o projeto no Figma, com as cores, tamanhos, formas e posicionamentos corretos.

**Rafaella:** Já compreendemos que o HTML é uma **linguagem de marcação** enquanto o CSS é uma **linguagem de estilização**, portanto **não** são linguagens de programação.

Vamos começar a estilizar nossa página.

**Guilherme:** Sempre quando formos aprender algo novo, é interessante consultarmos a **documentação**. Na barra de busca do navegador, digitaremos "CSS w3s" para acessarmos o [link sobre CSS no W3Schools](https://www.w3schools.com/css/) que aparece na lista de resultados.

Este tutorial é bem simples e possui vários exemplos. Há outras documentações, mas esta é bastante objetiva.

Clicando na opção "CSS Introduction" na lista lateral esquerda do site, abriremos o texto de introdução. Poderemos alterar para o português clicando no ícone de planeta ao lado da lupa na barra superior de opções da cor preta, para depois clicarmos em "Selecionar idioma" e escolhermos "português".

* O HTML é a sigla para "Hyper Text Markup Language" em inglês, ou "**Linguagem de Marcação de HiperTexto**".
* O CSS é a sigla para "Cascading Style Sheet" ou "**Folhas de Estilo em Cascatas**" em português.

**Guilherme:** O CSS descreve como esses elementos HTML devem ser exibidos na tela, mas não necessariamente é só a cor, mas também o posicionamento deles, tamanho, peso da fonte e diversos outros detalhes visuais importantes na construção final da página.

A introdução da documentação diz que o CSS "economiza" bastante trabalho, pois pode controlar o layout de várias páginas web de uma só vez.

Além disso, também são armazenados em arquivos **.css**, da mesma forma que o HTML é armazenado em arquivos **.html**.

Vamos ver uma [demonstração de seu uso neste link](https://www.w3schools.com/css/css_intro.asp).

**Rafaella:** Mais adiante na introdução, encontraremos a parte de "Demonstração CSS - Uma página HTML - Vários estilos!" com um mesmo arquivo .html relacionados com arquivos .css diferentes, ou seja, com estilizações diferentes aplicados a um mesmo HTML.

Quando clicamos na opção de "Folha de estilo 1" à esquerda e abaixo de "Bem-Vindo à minha página inicial", teremos uma composição de cores, posições, formas e tamanhos específica, e se clicarmos no "Folha de estilo 2" em seguida, teremos outra composição, e assim por diante.

Todas as alterações são estabelecidas nos arquivos .css, mesmo contendo o mesmo conteúdo textual. A última opção de "Sem estilo de folha" é a mais básica e bem parecida com a nossa página atual do projeto.

Portanto, já conseguimos entender como funciona na prática e como escrevemos em CSS.

**Guilherme:** Semelhante ao HTML, existem palavras que são específicas para realizar alguma ação.

Da mesma forma que comentamos que não é necessário aprendermos todas as tags em um primeiro curso de HTML, também não é possível aprender todas as **propriedades** de todas as estilizações que existem em CSS de uma só vez. Então focaremos nas **principais**.

Na parte de "Por que usar CSS?" Teremos um exemplo de código contendo body que abre e fecha chaves {}, e dentro delas, escreveremos as propriedades que vamos precisar e que veremos de forma gradativa ao longo das aulas.

**Rafaella:** Em seguida, teremos a parte de "CSS resolveu um grande problema".

Antigamente, tínhamos apenas o HTML que apenas descreve o conteúdo de uma página web. Ao longo do tempo, aumentou a necessidade de fazer com que as telas fiquem mais visualmente agradáveis e eficientes.

Primeiramente era apenas para passar informações e **marcar** o que é cada elemento, e aos poucos as estilizações foram aumentando. Conforme está no texto,

"O HTML NUNCA foi destinado a conter tags para **formatar** uma página da web!"

Justamente para resolver este problema, o World Wide Web Consortium (**W3C**) removeu a formatação de estilo da página HTML e criou o CSS para essa responsabilidade.

**Guilherme:** Inclusive comenta que:

"Quando tags como <font> e atributos de cor foram adicionados à especificação do HTML 3.2, começou um 'pesadelo' para os desenvolvedores da web"

Antes, tínhamos tags de marcação e de estilização juntas em um único arquivo, que o deixava bem grande e confuso. Então resolveram isolar essas questões.

Um arquivo ficou responsável pelas informações e marcação do texto que vai aparecer na página, enquanto outro contém os estilos.

Mais adiante, na parte de "CSS economiza muito trabalho!", temos algo que é muito relevante:

"As definições de estilo são normalmente salvas em arquivos .css **externos**".

**Rafaella:** Ou seja, temos o arquivo .html e há outro separado com a extensão .css.

**Guilherme:** Também diz que "Com um arquivo de folha de estilo externo, você pode alterar a aparência de um site inteiro alterando apenas um arquivo!"

**Rafaella:** É exatamente este arquivo que trabalharemos a seguir.

# 04. 03Incluindo CSS na página

No vídeo anterior, você aprendeu que as “Folhas de Estilos em Cascata” (CSS) descrevem um conjunto de regras de formatação que controlam a aparência de uma página da internet. Pensando nisso, marque a alternativa correta que apresente a boa prática recomendada na hora de utilizar o CSS:

Parte superior do formulário

* Alternativa correta



É recomendado escrever o CSS internamente no arquivo HTML através da tag <style> dentro do <head> da página. Isso permite organizar melhor o nosso código.

Essa forma não é recomendada, pois o CSS Interno é carregado toda vez que um site é atualizado, o que irá aumentar o tempo de carregamento.

* Alternativa correta



É recomendado criar um arquivo CSS externo com extensão .css e incluí-lo na estrutura head do HTML. Assim, conseguimos ter uma estrutura de estilos universal para várias páginas.

O CSS externo é o mais indicado e recomendado, pois além dos seus arquivos HTML ficarem com uma estrutura mais limpa e tamanho menor, os mesmos estilos do arquivo .css podem ser usados em várias páginas.

* Alternativa correta



O ideal é utilizar o atributo style em elementos do HTML, para estilizar nosso conteúdo de uma forma simples e direta. Essa maneira é conhecida como *Inline*.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual, o que leva tempo e gera retrabalho.

# 04. 04Criando o CSS

## Transcrição

**Rafaella:** Vamos criar o **arquivo CSS** e estilizar nossa página web. Para isso, clicaremos no primeiro ícone do lado direito do nome da pasta "PORTFOLIO" na aba lateral "Explorer".

Mas qual nome iremos dar?

**Guilherme:** Damos o nome de index.html para a página principal da aplicação, e para o **estilo principal**, geralmente nomeamos apenas como style.css, que é "estilos” em inglês.

Com isso, teremos nosso primeiro arquivo CSS.

Voltando ao Figma, notaremos que há vários detalhes como cores dos elementos e fundo, posicionamentos, tamanhos e outras características, então não conseguiremos fazer tudo de uma só vez.

Uma sugestão é começarmos pela cor de fundo e pela cor do texto.

**Rafaella:** É uma boa ideia, porque começamos de algo que engloba todos os elementos até irmos especificando cada um deles.

Conforme a documentação do CSS que vimos anteriormente, conseguimos definir qual a tag que queremos alterar. Neste caso, como estamos utilizando a página inteira, conseguiremos usar a <body>, que é justamente onde inicia o corpo da página logo abaixo das metainformações.

Na primeira linha em style.css, escreveremos body e abriremos as chaves {} em seguida.

**Guilherme:** Com a tecla "Enter", abriremos o bloco e todos os estilos que aplicaremos ficarão entre as chaves.

A ideia do código em "cascata" remete ao sentido de cima para baixo, e "desceremos" em cada elemento conforme avançamos.

Na aula anterior, comentamos que não aprenderemos todas as propriedades de uma só vez, então focaremos em algumas específicas.

**Rafaella:** Há uma que é relacionada a cor do plano fundo, chamada **background-color:**, que é este mesmo termo em inglês.

**Guilherme:** Quando estamos escrevendo background, o próprio editor de código exibe as opções possíveis em uma lista.

**Rafaella:** Conseguiremos autocompletar da mesma forma que fizemos no HTML, subindo e descendo com as setas do teclado para passar pelos sufixos disponíveis.

Em nosso caso, queremos o background-color:, então o selecionaremos e apertaremos a tecla "Tab" para autocompletar. Em seguida, o editor preencherá com a estrutura que usaremos para aplicarmos o valor da propriedade.

Como observamos no Figma, a cor do background é preta, então bastará escrevermos black e salvaremos o arquivo.

body {

background-color: black;

}COPIAR CÓDIGO

**Rafaella:** No navegador, notaremos que o código não funcionou porque a cor de fundo continua branca como antes.

**Guilherme:** Isso se deu porque algo de errado aconteceu na **comunicação** entre o arquivo CSS e o HTML.

**Rafaella:** Exatamente, pois simplesmente não ligamos um ao outro, e precisaremos sempre linkar o CSS no HTML para que seja utilizado e os estilos sejam de fato aplicados.

Abrindo o index.html, iremos às metainformações do <head> e usaremos uma tag chamada **<link>**, a qual também possui algumas propriedades que serão inseridas dentro para indicarmos o arquivo CSS que iremos usar.

Primeiro, escreveremos a **rel** que irá indicar qual a relação do documento HTML com o outro que traremos, que no caso será **"stylesheet"** que significa "folha de estilo".

Em seguida, colocaremos onde este arquivo está usando href igual ao nosso próprio "style.css" entre aspas. Por fim, fecharemos a tag apenas com > e salvaremos o arquivo para vermos o resultado.

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Portfolio</title>

<link rel="stylesheet" href="styles.css">

</head>

//código suprimidoCOPIAR CÓDIGO

**Rafaella:** No navegador, veremos a cor de fundo preta desta vez.

**Guilherme:** Sim, porém todo o texto sumiu porque é da mesma cor do background, mas é possível vê-lo quando o selecionamos com o cursor do mouse, então ainda está na página.

**Rafaella:** Vamos mudar apenas a cor do texto. Em nosso arquivo style.css no editor de código, usaremos a propriedade **color:**.

**Guilherme:** Então teremos o background-color que diz respeito à cor de fundo, e apenas color: que altera a cor do texto.

**Rafaella:** Vamos definir como branco usando o valor white, do mesmo jeito que aparece no projeto do Figma.

body {

background-color: black;

color: white;

}COPIAR CÓDIGO

**Rafaella:** Salvando e atualizando nossa página no navegador, conseguiremos exibir o texto em branco sobre um fundo preto, tornando a leitura possível.



**Guilherme:** Só os links que ainda aparecem em azul e sublinhados na base esquerda da imagem da programadora.

**Rafaella:** Mais adiante iremos estilizá-los especificamente, mas por enquanto já conseguimos aplicar alguns estilos.

# 04. 05 ATV Estilizando o HTML com CSS

Para que os estilos em CSS sejam aplicados nos elementos HTML, a folha de estilos precisa estar conectada corretamente no documento HTML através de uma tag <link>, caso contrário, os estilos não serão aplicados.

Das alternativas abaixo, selecione aquela que apresenta a forma correta de escrever a tag <link> e o local que deve conter essa tag:

Parte superior do formulário

* Alternativa correta



<header>

<link rel="stylesheet" href="nomedoarquivo.css">

</header>

A tag <link rel="stylesheet" href="nomedoarquivo.css"> deve ser adicionada dentro do <head> no HTML, e não na tag <header>.

* Alternativa correta



<head>

<link rel="stylesheet" href="nomedoarquivo.css"></link>

</head>

A tag link não possui o fechamento </link>.

* Alternativa correta



<head>

<link rel="stylesheet">

</head>

É essencial que a tag link possua o atributo rel="stylesheet", que determina que a relação desse link com o HTML é de folha de estilo, além do atributo href="nomedoarquivo.css", que irá determinar qual é esse arquivo de estilo. Sem ele, o HTML não encontra o arquivo CSS.

* Alternativa correta



<head>

<link rel="stylesheet" href="nomedoarquivo.css">

</head>

O HTML não reconhece a página de estilo automaticamente, portanto é necessário adicionar a tag <link rel="stylesheet" href="nomedoarquivo.css"> dentro do <head> para que a estilização seja aplicada.

# 04. 06Cor de fundo

Qual das alternativas troca a cor de fundo para vermelho (red)?

Parte superior do formulário

* Alternativa correta



body {

background-color: blue;

color: red;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

Color é a cor do texto.

* Alternativa correta



body {

color: blue;

font-color: red;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

color é a cor do texto

* Alternativa correta



body {

background-color: red;

color: blue;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

Perfeito, background-color é o atributo de cor de fundo.

* Alternativa correta



body {

back-color: red;

color: blue;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

back-color não existe.

# 04. 07Faça como eu fiz: estilize seu HTML

Nessa aula, aprendemos que com o CSS podemos selecionar elementos do HTML e atribuir propriedades do CSS dentro de chaves para que esses elementos sejam estilizados.

Fizemos isso atribuindo à tag <body> uma cor de fundo preta, através da propriedade background-color e alteramos a cor do texto para branco utilizando a propriedade color, da seguinte maneira:

body{

background-color: black;

color: white;

}COPIAR CÓDIGO

Agora que você já sabe como criar uma folha de estilo para sua página HTML, é hora de aplicar o que aprendeu nessa aula. Manipule o estilo da tag <body>, teste novas cores para o fundo e para o texto e deixe o projeto com a sua cara! =)

Não se esqueça de aplicar a tag <link rel=”stylesheet” href=”style.css”> dentro do <head> em seu arquivo HTML, para que sua estilização funcione corretamente.

# 04. 08O que aprendemos?

## Nessa aula você aprendeu:

* O que é CSS (Cascading Style Sheets);
* Estilização na prática;
* Propriedades CSS;
* Criar um arquivo externo para estilizar a página;
* Integrar o arquivo CSS ao arquivo HTML.

# 04. 01Projeto da aula anterior

Aqui você pode [baixar o zip da aula 03](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/archive/refs/heads/aula_3.zip) ou [acessar os arquivos no Github](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/tree/aula_3)!

# 04. 02 Como funciona o CSSParte inferior do formulário

## Transcrição

**Guilherme:** Já criamos um HTML semântico, e escrevemos as estruturas e as marcações de todo o conteúdo que queremos realizar, mas não está visualmente bem finalizado.

**Rafaella:** Os elementos já estão estruturados, mas falta colocar as cores e formatos que fazem uma página web.

**Guilherme:** Isso faz muita diferença, pois há anos atrás, a parte visual não era tão desenvolvida como atualmente, tanto que há uma área profissional que se dedica somente a esta parte de criação.

O desafio desta aula é deixarmos a página mais bonita e parecida com o projeto no Figma, com as cores, tamanhos, formas e posicionamentos corretos.

**Rafaella:** Já compreendemos que o HTML é uma **linguagem de marcação** enquanto o CSS é uma **linguagem de estilização**, portanto **não** são linguagens de programação.

Vamos começar a estilizar nossa página.

**Guilherme:** Sempre quando formos aprender algo novo, é interessante consultarmos a **documentação**. Na barra de busca do navegador, digitaremos "CSS w3s" para acessarmos o [link sobre CSS no W3Schools](https://www.w3schools.com/css/) que aparece na lista de resultados.

Este tutorial é bem simples e possui vários exemplos. Há outras documentações, mas esta é bastante objetiva.

Clicando na opção "CSS Introduction" na lista lateral esquerda do site, abriremos o texto de introdução. Poderemos alterar para o português clicando no ícone de planeta ao lado da lupa na barra superior de opções da cor preta, para depois clicarmos em "Selecionar idioma" e escolhermos "português".

* O HTML é a sigla para "Hyper Text Markup Language" em inglês, ou "**Linguagem de Marcação de HiperTexto**".
* O CSS é a sigla para "Cascading Style Sheet" ou "**Folhas de Estilo em Cascatas**" em português.

**Guilherme:** O CSS descreve como esses elementos HTML devem ser exibidos na tela, mas não necessariamente é só a cor, mas também o posicionamento deles, tamanho, peso da fonte e diversos outros detalhes visuais importantes na construção final da página.

A introdução da documentação diz que o CSS "economiza" bastante trabalho, pois pode controlar o layout de várias páginas web de uma só vez.

Além disso, também são armazenados em arquivos **.css**, da mesma forma que o HTML é armazenado em arquivos **.html**.

Vamos ver uma [demonstração de seu uso neste link](https://www.w3schools.com/css/css_intro.asp).

**Rafaella:** Mais adiante na introdução, encontraremos a parte de "Demonstração CSS - Uma página HTML - Vários estilos!" com um mesmo arquivo .html relacionados com arquivos .css diferentes, ou seja, com estilizações diferentes aplicados a um mesmo HTML.

Quando clicamos na opção de "Folha de estilo 1" à esquerda e abaixo de "Bem-Vindo à minha página inicial", teremos uma composição de cores, posições, formas e tamanhos específica, e se clicarmos no "Folha de estilo 2" em seguida, teremos outra composição, e assim por diante.

Todas as alterações são estabelecidas nos arquivos .css, mesmo contendo o mesmo conteúdo textual. A última opção de "Sem estilo de folha" é a mais básica e bem parecida com a nossa página atual do projeto.

Portanto, já conseguimos entender como funciona na prática e como escrevemos em CSS.

**Guilherme:** Semelhante ao HTML, existem palavras que são específicas para realizar alguma ação.

Da mesma forma que comentamos que não é necessário aprendermos todas as tags em um primeiro curso de HTML, também não é possível aprender todas as **propriedades** de todas as estilizações que existem em CSS de uma só vez. Então focaremos nas **principais**.

Na parte de "Por que usar CSS?" Teremos um exemplo de código contendo body que abre e fecha chaves {}, e dentro delas, escreveremos as propriedades que vamos precisar e que veremos de forma gradativa ao longo das aulas.

**Rafaella:** Em seguida, teremos a parte de "CSS resolveu um grande problema".

Antigamente, tínhamos apenas o HTML que apenas descreve o conteúdo de uma página web. Ao longo do tempo, aumentou a necessidade de fazer com que as telas fiquem mais visualmente agradáveis e eficientes.

Primeiramente era apenas para passar informações e **marcar** o que é cada elemento, e aos poucos as estilizações foram aumentando. Conforme está no texto,

"O HTML NUNCA foi destinado a conter tags para **formatar** uma página da web!"

Justamente para resolver este problema, o World Wide Web Consortium (**W3C**) removeu a formatação de estilo da página HTML e criou o CSS para essa responsabilidade.

**Guilherme:** Inclusive comenta que:

"Quando tags como <font> e atributos de cor foram adicionados à especificação do HTML 3.2, começou um 'pesadelo' para os desenvolvedores da web"

Antes, tínhamos tags de marcação e de estilização juntas em um único arquivo, que o deixava bem grande e confuso. Então resolveram isolar essas questões.

Um arquivo ficou responsável pelas informações e marcação do texto que vai aparecer na página, enquanto outro contém os estilos.

Mais adiante, na parte de "CSS economiza muito trabalho!", temos algo que é muito relevante:

"As definições de estilo são normalmente salvas em arquivos .css **externos**".

**Rafaella:** Ou seja, temos o arquivo .html e há outro separado com a extensão .css.

**Guilherme:** Também diz que "Com um arquivo de folha de estilo externo, você pode alterar a aparência de um site inteiro alterando apenas um arquivo!"

**Rafaella:** É exatamente este arquivo que trabalharemos a seguir.

# 04. 03Incluindo CSS na página

No vídeo anterior, você aprendeu que as “Folhas de Estilos em Cascata” (CSS) descrevem um conjunto de regras de formatação que controlam a aparência de uma página da internet. Pensando nisso, marque a alternativa correta que apresente a boa prática recomendada na hora de utilizar o CSS:

Parte superior do formulário

* Alternativa correta



É recomendado escrever o CSS internamente no arquivo HTML através da tag <style> dentro do <head> da página. Isso permite organizar melhor o nosso código.

Essa forma não é recomendada, pois o CSS Interno é carregado toda vez que um site é atualizado, o que irá aumentar o tempo de carregamento.

* Alternativa correta



É recomendado criar um arquivo CSS externo com extensão .css e incluí-lo na estrutura head do HTML. Assim, conseguimos ter uma estrutura de estilos universal para várias páginas.

O CSS externo é o mais indicado e recomendado, pois além dos seus arquivos HTML ficarem com uma estrutura mais limpa e tamanho menor, os mesmos estilos do arquivo .css podem ser usados em várias páginas.

* Alternativa correta



O ideal é utilizar o atributo style em elementos do HTML, para estilizar nosso conteúdo de uma forma simples e direta. Essa maneira é conhecida como *Inline*.

Este tipo de CSS não é realmente recomendado, já que cada tag HTML precisa ser estilizada de maneira individual, o que leva tempo e gera retrabalho.

# 04. 04 Criando o CSSParte inferior do formulário

## Transcrição

**Rafaella:** Vamos criar o **arquivo CSS** e estilizar nossa página web. Para isso, clicaremos no primeiro ícone do lado direito do nome da pasta "PORTFOLIO" na aba lateral "Explorer".

Mas qual nome iremos dar?

**Guilherme:** Damos o nome de index.html para a página principal da aplicação, e para o **estilo principal**, geralmente nomeamos apenas como style.css, que é "estilos” em inglês.

Com isso, teremos nosso primeiro arquivo CSS.

Voltando ao Figma, notaremos que há vários detalhes como cores dos elementos e fundo, posicionamentos, tamanhos e outras características, então não conseguiremos fazer tudo de uma só vez.

Uma sugestão é começarmos pela cor de fundo e pela cor do texto.

**Rafaella:** É uma boa ideia, porque começamos de algo que engloba todos os elementos até irmos especificando cada um deles.

Conforme a documentação do CSS que vimos anteriormente, conseguimos definir qual a tag que queremos alterar. Neste caso, como estamos utilizando a página inteira, conseguiremos usar a <body>, que é justamente onde inicia o corpo da página logo abaixo das metainformações.

Na primeira linha em style.css, escreveremos body e abriremos as chaves {} em seguida.

**Guilherme:** Com a tecla "Enter", abriremos o bloco e todos os estilos que aplicaremos ficarão entre as chaves.

A ideia do código em "cascata" remete ao sentido de cima para baixo, e "desceremos" em cada elemento conforme avançamos.

Na aula anterior, comentamos que não aprenderemos todas as propriedades de uma só vez, então focaremos em algumas específicas.

**Rafaella:** Há uma que é relacionada a cor do plano fundo, chamada **background-color:**, que é este mesmo termo em inglês.

**Guilherme:** Quando estamos escrevendo background, o próprio editor de código exibe as opções possíveis em uma lista.

**Rafaella:** Conseguiremos autocompletar da mesma forma que fizemos no HTML, subindo e descendo com as setas do teclado para passar pelos sufixos disponíveis.

Em nosso caso, queremos o background-color:, então o selecionaremos e apertaremos a tecla "Tab" para autocompletar. Em seguida, o editor preencherá com a estrutura que usaremos para aplicarmos o valor da propriedade.

Como observamos no Figma, a cor do background é preta, então bastará escrevermos black e salvaremos o arquivo.

body {

background-color: black;

}COPIAR CÓDIGO

**Rafaella:** No navegador, notaremos que o código não funcionou porque a cor de fundo continua branca como antes.

**Guilherme:** Isso se deu porque algo de errado aconteceu na **comunicação** entre o arquivo CSS e o HTML.

**Rafaella:** Exatamente, pois simplesmente não ligamos um ao outro, e precisaremos sempre linkar o CSS no HTML para que seja utilizado e os estilos sejam de fato aplicados.

Abrindo o index.html, iremos às metainformações do <head> e usaremos uma tag chamada **<link>**, a qual também possui algumas propriedades que serão inseridas dentro para indicarmos o arquivo CSS que iremos usar.

Primeiro, escreveremos a **rel** que irá indicar qual a relação do documento HTML com o outro que traremos, que no caso será **"stylesheet"** que significa "folha de estilo".

Em seguida, colocaremos onde este arquivo está usando href igual ao nosso próprio "style.css" entre aspas. Por fim, fecharemos a tag apenas com > e salvaremos o arquivo para vermos o resultado.

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Portfolio</title>

<link rel="stylesheet" href="styles.css">

</head>

//código suprimidoCOPIAR CÓDIGO

**Rafaella:** No navegador, veremos a cor de fundo preta desta vez.

**Guilherme:** Sim, porém todo o texto sumiu porque é da mesma cor do background, mas é possível vê-lo quando o selecionamos com o cursor do mouse, então ainda está na página.

**Rafaella:** Vamos mudar apenas a cor do texto. Em nosso arquivo style.css no editor de código, usaremos a propriedade **color:**.

**Guilherme:** Então teremos o background-color que diz respeito à cor de fundo, e apenas color: que altera a cor do texto.

**Rafaella:** Vamos definir como branco usando o valor white, do mesmo jeito que aparece no projeto do Figma.

body {

background-color: black;

color: white;

}COPIAR CÓDIGO

**Rafaella:** Salvando e atualizando nossa página no navegador, conseguiremos exibir o texto em branco sobre um fundo preto, tornando a leitura possível.



**Guilherme:** Só os links que ainda aparecem em azul e sublinhados na base esquerda da imagem da programadora.

**Rafaella:** Mais adiante iremos estilizá-los especificamente, mas por enquanto já conseguimos aplicar alguns estilos.

# 04. 05 ATV - Estilizando o HTML com CSS

Para que os estilos em CSS sejam aplicados nos elementos HTML, a folha de estilos precisa estar conectada corretamente no documento HTML através de uma tag <link>, caso contrário, os estilos não serão aplicados.

Das alternativas abaixo, selecione aquela que apresenta a forma correta de escrever a tag <link> e o local que deve conter essa tag:

Parte superior do formulário

* Alternativa correta



<header>

<link rel="stylesheet" href="nomedoarquivo.css">

</header>

A tag <link rel="stylesheet" href="nomedoarquivo.css"> deve ser adicionada dentro do <head> no HTML, e não na tag <header>.

* Alternativa correta



<head>

<link rel="stylesheet" href="nomedoarquivo.css"></link>

</head>

A tag link não possui o fechamento </link>.

* Alternativa correta



<head>

<link rel="stylesheet">

</head>

É essencial que a tag link possua o atributo rel="stylesheet", que determina que a relação desse link com o HTML é de folha de estilo, além do atributo href="nomedoarquivo.css", que irá determinar qual é esse arquivo de estilo. Sem ele, o HTML não encontra o arquivo CSS.

* Alternativa correta



<head>

<link rel="stylesheet" href="nomedoarquivo.css">

</head>

O HTML não reconhece a página de estilo automaticamente, portanto é necessário adicionar a tag <link rel="stylesheet" href="nomedoarquivo.css"> dentro do <head> para que a estilização seja aplicada.

# 04. 06Cor de fundo

Qual das alternativas troca a cor de fundo para vermelho (red)?

Parte superior do formulário

* Alternativa correta



body {

background-color: blue;

color: red;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

Color é a cor do texto.

* Alternativa correta



body {

color: blue;

font-color: red;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

color é a cor do texto

* Alternativa correta



body {

background-color: red;

color: blue;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

Perfeito, background-color é o atributo de cor de fundo.

* Alternativa correta



body {

back-color: red;

color: blue;

font-family: 'Comic Sans MS', sans-serif;

font-size: 16px;

line-height: 1.5;

}

back-color não existe.

# 04. 07Faça como eu fiz: estilize seu HTML

Nessa aula, aprendemos que com o CSS podemos selecionar elementos do HTML e atribuir propriedades do CSS dentro de chaves para que esses elementos sejam estilizados.

Fizemos isso atribuindo à tag <body> uma cor de fundo preta, através da propriedade background-color e alteramos a cor do texto para branco utilizando a propriedade color, da seguinte maneira:

body{

background-color: black;

color: white;

}COPIAR CÓDIGO

Agora que você já sabe como criar uma folha de estilo para sua página HTML, é hora de aplicar o que aprendeu nessa aula. Manipule o estilo da tag <body>, teste novas cores para o fundo e para o texto e deixe o projeto com a sua cara! =)

Não se esqueça de aplicar a tag <link rel=”stylesheet” href=”style.css”> dentro do <head> em seu arquivo HTML, para que sua estilização funcione corretamente.

# 04. 08O que aprendemos?

## Nessa aula você aprendeu:

* O que é CSS (Cascading Style Sheets);
* Estilização na prática;
* Propriedades CSS;
* Criar um arquivo externo para estilizar a página;
* Integrar o arquivo CSS ao arquivo HTML.

# 05. 01Projeto da aula anterior

Aqui você pode [baixar o zip da aula 04](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/archive/refs/heads/aula_4.zip) ou [acessar os arquivos no Github](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/tree/aula_4)!

# 05. 02Cores no CSS

## Transcrição

**Guilherme:** Já conseguimos alterar tanto a cor de fundo quanto a cor do texto, mas ainda há um detalhe importante.

No Figma, será que a cor branca que colocamos com white é realmente esta?

Abrindo o projeto no Figma e selecionando o bloco de texto, notaremos que há duas cores: um valor 22D4FD para azul claro e F6F6F6 para o branco, e não white.

Mas isso tem uma diferença, afinal não faz sentido dizermos no cotidiano que outros objetos de cor branca são da cor "F6F6F6", como uma camisa por exemplo, dizemos que é "branca". Mas para o computador, faz sentido.

**Rafaella:** Vamos usar esses valores obtidos do Figma em nossas propriedades no arquivo style.css. Copiaremos "F6F6F6" e colaremos no lugar de white em color: para salvarmos e vermos o resultado.

body {

background-color: black;

color: F6F6F6;

}COPIAR CÓDIGO

No navegador, notaremos que o texto está em preto, da mesma cor que o fundo, e não ficou visível.

**Guilherme:** Sabemos que o texto ainda está no corpo da página quando selecionamos com o cursor do mouse.

**Rafaella:** Está com a cor padrão que é preto igual ao fundo, do mesmo jeito que estava antes de aplicarmos estilos no CSS. Ou seja, apenas o que escrevemos de valor para a cor branca não está funcionando.

**Guilherme:** Não reconheceu porque temos que adicionar um sinal quando queremos representar cores de outra forma, que é a cerquilha ou hashtag # antes do valor da cor.

body {

background-color: black;

color: #F6F6F6;

}COPIAR CÓDIGO

**Rafaella:** Salvando e atualizando a página no navegador, conseguiremos deixar o texto em branco sobre o fundo preto.

**Guilherme:** Então teremos um problema, pois conseguimos representar cores no HTML e estilizando no CSS de maneiras diferentes.

Na barra de busca no navegador, pesquisaremos por "cores no CSS" e, logo no topo da lista de resultados do buscador, aparecerá a **Especificação CSS Level 1**.

| **Especificação** | **Palavra-chave** | **valores hex RGB** |
| --- | --- | --- |
|  | gray | #808080 |
| **CSS** Level 1 | white | #ffffff |
|  | marron | #800000 |
| red | #ff0000 |  |

Portanto, existem formas diferentes de gerar nossas cores.

Acessando a [documentação oficial do Mozilla neste link](https://developer.mozilla.org/pt-BR/docs/Web/CSS/color_value), encontraremos uma tabela mais completa de cores, nomes e códigos.

Repararemos que podemos usar **palavras-chave** para as cores como o black e white que fizemos antes. Mas também poderemos representá-las com **Notação Hexadecimal RGB** que sempre começam com a cerquilha # seguido de seis números ou letras.

**Rafaella:** O Hexadecimal usa números de zero até nove e da letra "A" até "F". Por exemplo, o preto que é a ausência de cor é "000000", e o branco que é a junção de todas as cores é "ffff".

Todas as outras cores estão neste intervalo e sua numeração representa a proporção de vermelho, verde e azul que possuem, ou seja, red, green e blue que compõe o padrão RGB.

**Guilherme:** Vamos fazer um teste então. Eu vou escolher a cor "purple" por palavra-chave.

**Rafaella:** E eu vou outra com valor "00ffff" pelo RGB.

**Guilherme:** De volta ao código style.css, vamos usar a cor purple em background-color: e #00ffff em color:

body {

background-color: purple;

color: #00ffff;

}COPIAR CÓDIGO

**Guilherme:** Vamos salvar e ver como ficaram essas cores aleatórias na página do navegador.



**Guilherme:** Não ficou tão ruim, mas a versão que a pessoa designer elaborou para nosso projeto é muito melhor. De volta ao nosso código no VSCode, voltaremos com as cores black e #F6F6F6 do nosso projeto.

Isso acontece porque a junção das cores de **forma harmônica** é baseada em estudos, boas práticas e conceitos da área.

Deixaremos um **desafio** para pesquisar e escolher as cores que poderiam ficar interessantes, e há várias ferramentas para isso.

Pesquisando no navegador, digitaremos "roda de cores adobe" na barra de busca e clicaremos no [link do **Adobe Colors**](https://color.adobe.com/pt/create/color-wheel).

Esta Roda de Cores produz paletas com boa interação entre si que ajudam na escolha dos estilos, e há várias regras de harmonia de cores como "Análogo", "Monocromático", "Tríade", "Complementar"< entre outras. Vale a pena explorar, mas há outros também.

**Rafaella:** Continuando, poderemos deixar a cor black como padrão substituindo-a pelo Hexadecimal #00000 em background-color.

body {

background-color: #000000;

color: #F6F6F6;

}COPIAR CÓDIGO

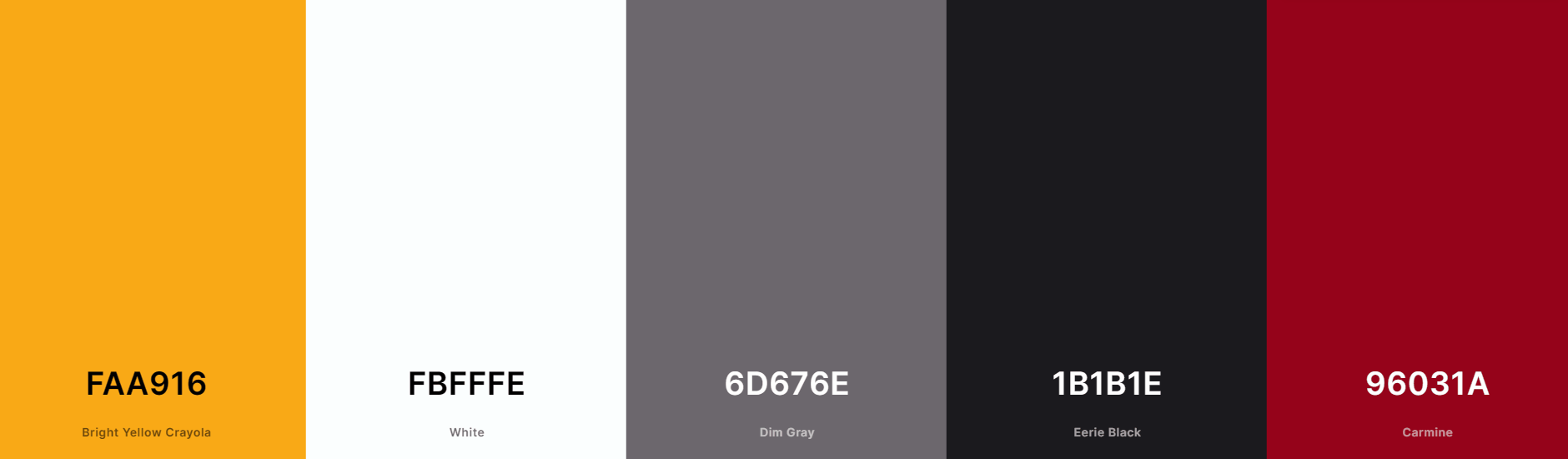
**Rafaella:** Salvaremos e rodaremos a aplicação sem problemas.

# 05. 03Para saber mais: escolhendo as cores do projeto

Chegou a hora de colocar as mãos na massa. Desafiamos você a escolher cores para o seu projeto, pois uma paleta é essencial para providenciar uma boa experiência de usuário e enriquecer a identidade de sua página. E para te ajudar com esse desafio, separamos alguns sites.

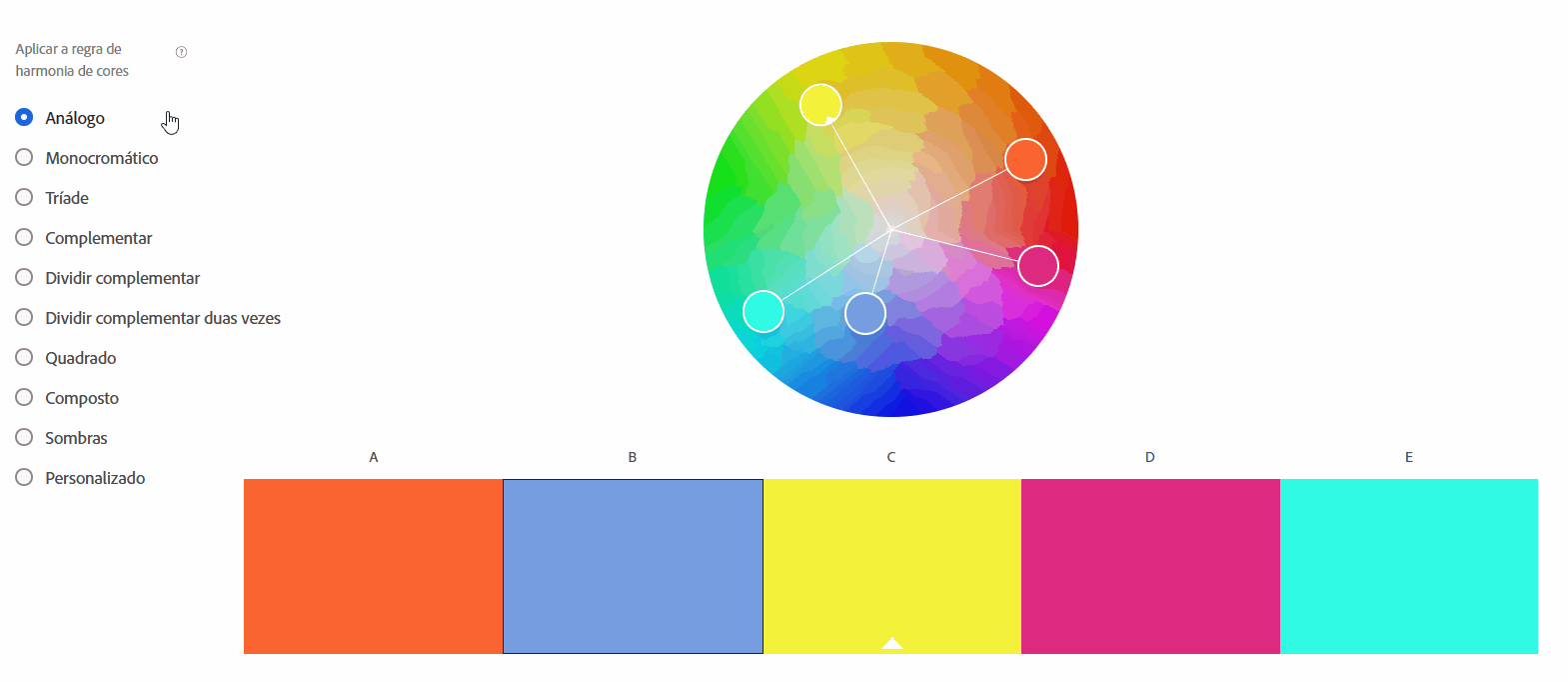
## Coolors

O [Coolors](https://coolors.co/) possui uma interface bem clara. Com a barra de espaço do seu teclado, você consegue criar várias combinações, e uma das funções mais legais é a opção travar, que você pode usar se gostar de apenas uma cor, e assim que você clica nela, consegue continuar elaborando outras combinações levando em conta a cor que você escolheu.



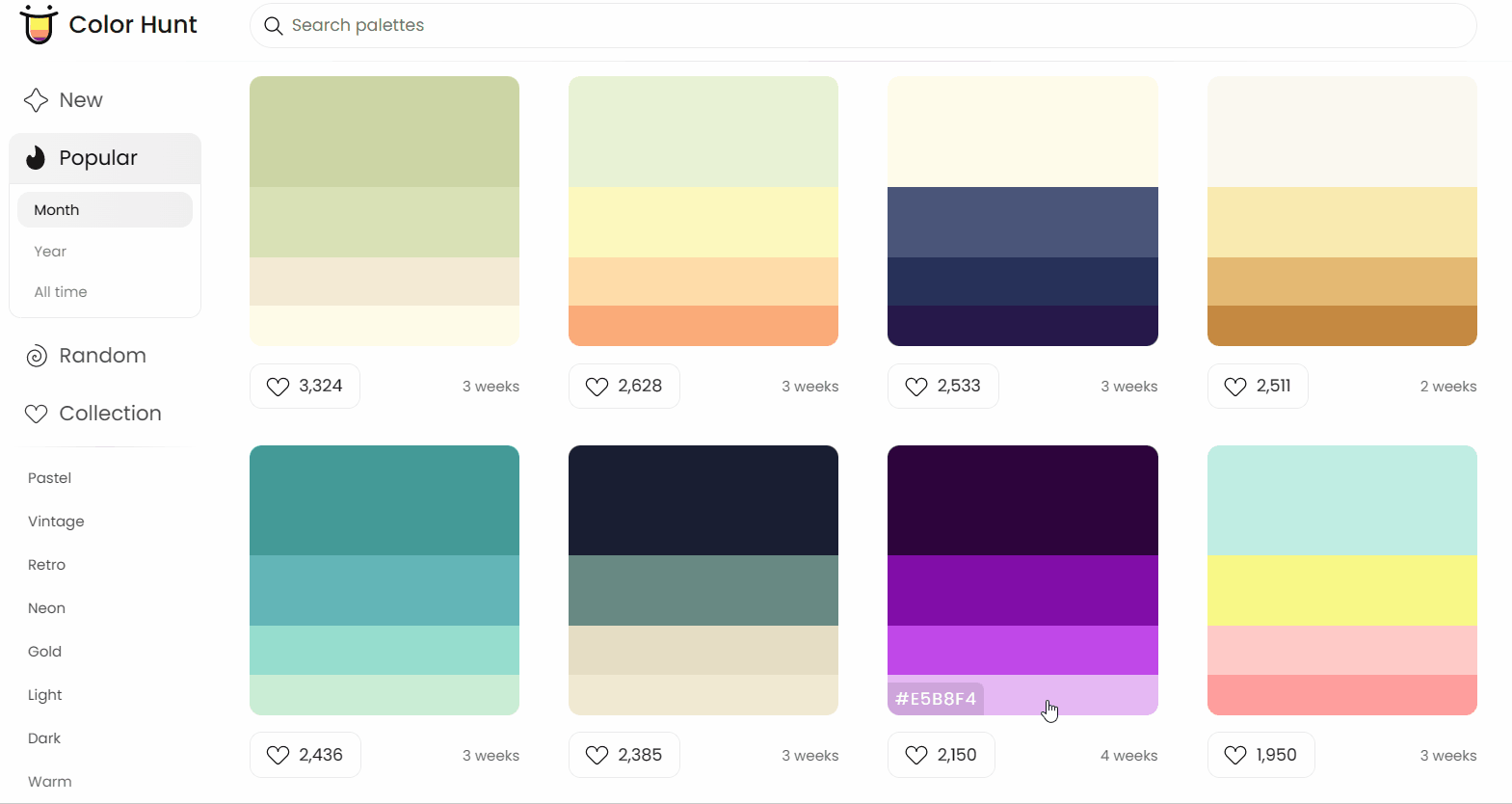
## Adobe Color

O [Adobe color](https://color.adobe.com/pt/create/color-wheel) apresenta um Color Wheel (roda de cores) que pode ser ajustado de maneiras variadas para obter uma harmonia de cores, e você pode aplicar diversas regras de harmonia de cores, como o modo análogo, monocromático, tríade, complementar, quadrado, composto, entre outros.



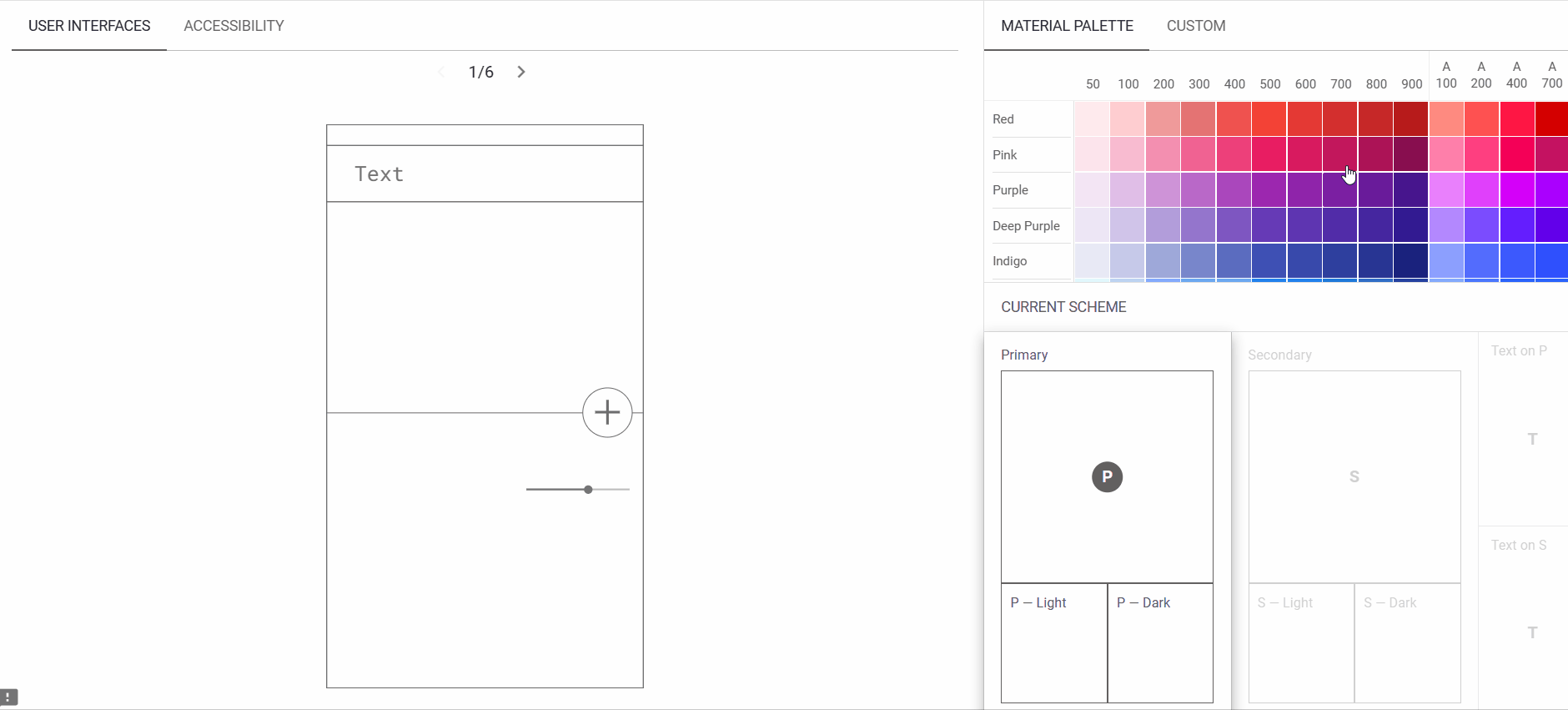
## Color Hunt

O [Color Hunt](https://colorhunt.co/) dispõe de diversas paletas elaboradas. Você consegue encontrar a combinação que mais te agrada e consegue buscar por palavras-chave como pastel, vintage, neon e assim por diante. E caso não encontre nenhuma que te agrade, você consegue criar a sua própria paleta clicando nos três pontinhos do canto superior direito da página.



## Color Tool - Material Design

O [Color Tool](https://m2.material.io/resources/color/#!/?view.left=0&view.right=0&primary.color=6002ee) é ótimo para criar, compartilhar e aplicar paletas de cores à interface do usuário, bem como é possível medir o nível de acessibilidade de qualquer combinação de cores na aba accessibility.



São opções incríveis, não é? Se você conhece algum outro site nesse estilo pode indicar no fórum ou no [Discord da Alura](https://discord.gg/QeBdgAjXnn) para que outras pessoas estudantes também conheçam.

Agora é com você! Encontre a paleta de cores que mais te agrada e aplique no projeto para deixá-lo com a sua carinha :D

# 05.04 ATV Cor de fundo

Utilizamos o CSS para alterar os estilos dos elementos, como por exemplo a cor de fundo, fontes, margens, etc.

Pensando nisso, qual atributo do CSS utilizamos quando queremos alterar a cor de fundo do <body> para a cor branca?

Parte superior do formulário

* Alternativa correta



body{ cor-de-fundo: #FFFFFF;}

Todos os atributos do CSS são escritos em inglês e não em português!

* Alternativa correta



body{ background: #FFFFFF;}

O background serve para configurar todas as propriedades do plano de fundo em uma declaração mais implícita.

* Alternativa correta



body{ color: #FFFFFF; }

O atributo color é utilizado para alterar a cor do texto!

* Alternativa correta



body{ background-color: #FFFFFF;}

Utilizamos o atributo background-color para mudarmos a cor de fundo de um elemento.

# 05. 05Destacando o texto

## Transcrição

**Rafaella:** Olhando para o nosso design, já aplicamos os estilos de plano de fundo e decidimos a cor principal das nossas fontes.

Porém, o trecho "com um Front-End de qualidade!" em destaque no título está na cor azul.

Por isso definimos a tag <strong> em index.html para destacarmos essa parte. Como utilizaremos o CSS para estilizar com essa outra cor?

**Guilherme:** No arquivo style.css, pegamos a body que queríamos, abrimos chaves e dentro colocamos as duas propriedades.

Já que strong também é uma tag, poderemos pegá-la e indicar qual a cor iremos usar nesse trecho do texto com color:.

Pegaremos o valor hexadecimal dela no projeto do Figma, que é #22D4FD e colaremos na propriedade.

body {

background-color: #000000;

color: #F6F6F6;

}

strong {

color: #22D4FD;

}COPIAR CÓDIGO

**Rafaella:** Vamos salvar e voltar à página atualizada no navegador para vermos o resultado.



**Guilherme:** Funcionou, mas se voltarmos ao Figma, notaremos que, durante todo o projeto, a maioria dos textos que aparecem possuem a cor branca #F6F6F6.

Já o trecho "com um Front-end de qualidade!" foi definido com <strong>. Mas se a colocarmos em outra parte do projeto, também irá alterar a cor?

**Rafaella:** Vamos testar para saber. No index.html, copiaremos a tag de destaque do trecho e a colaremos para destacar o trecho com as três tecnologias citadas no texto do parágrafo em <p>, "React, HTML e CSS". O restante ficará em branco mesmo.

<!DOCTYPE html>

<html lang="pt-br">

//código omitido

<body>

<header</header>

<main>

<h1>Eleve seu negócio digital a outro nível <strong>com um Front-end de qualidade!</strong></h1>

<p>Olá! Sou Joana Santos, desenvolvedora Front-end com especialidade em <strong>React, HTML e CSS</strong>. Ajudo pequenos negócios e designers a colocarem em prática boas ideias. Vamos conversar?

</p>

//código omitido

</main>

<footer></footer>

</body>

</html>COPIAR CÓDIGO

**Rafaella:** Vamos salvar e analisar o resultado no navegador.



**Rafaella:** Veremos que a <strong> deixou na mesma cor azul claro que havíamos definido no CSS, portanto toda vez que ela for aplicada, terá a mesma cor.

**Guilherme:** Certo, mas não é o que queremos, pois teoricamente, se nosso projeto continuar evoluindo e o código aumentando, todas as partes de texto que tiverem essa tag terão a cor que definimos.

**Rafaella:** Caso queiramos depois criar um subtítulo com outro trecho em destaque em uma cor diferente, não poderemos usar a <strong>.

**Guilherme:** Exatamente, então precisaremos de uma forma para dizer que somente uma dessas tags usará o azul claro. Mas faremos isso mais adiante.

# 05. 06Para saber mais: destacando o texto

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/html-css-ambiente-arquivos-tags/task/120616/next)

Você aprendeu a destacar o texto utilizando a tag <strong>, porém existe outra tag que também é muito utilizada para isso que é a tag <span>, diferente da strong a span não deixa em negrito por padrão mas é uma ótima forma de marcar trechos do texto em HTML. Veja:

**HTML:**

<h1>Formação de <span>Front-end</span></h1>COPIAR CÓDIGO

**CSS:**

h1{

font-weight: bold;

}

span{

color: #22D4FD;

border: 1px solid #22D4FD;

padding: 10px;

}COPIAR CÓDIGO

**Explicando o código acima:**

font-weight: bold; (Texto em negrito)

color: #22D4FD; (Cor do texto azul)

border: 1px solid #22D4FD; (Borda com 1 px de largura, do tipo sólida e cor da borda azul)

padding: 10px; (Espaçamento interno de 10px)

**RESULTADO:**



# 05. 07Projeto final do curso

Aqui você pode [baixar o zip da aula 05](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/archive/refs/heads/aula_5.zip) ou [acessar os arquivos no Github](https://github.com/alura-cursos/Portifolio-HTML-e-CSS/tree/aula_5)!

# 05. 08Desafio: compartilhe seu projeto com o mundo

Agora que você conheceu HTML e CSS e criou a base do seu portfólio, que tal customizar esse projeto e deixar com a sua cara?

Você pode personalizar o título e adicionar informações sobre você no parágrafo de texto, colocar sua foto e os links para suas redes sociais, além de alterar o CSS trabalhando com as cores e destaques. Solte sua criatividade!

Além disso, queremos muito ver seu progresso nesse curso! Por isso fique à vontade para compartilhar seu projeto conosco publicando em suas redes sociais como [Instagram](https://www.instagram.com/aluraonline), [Twitter](https://twitter.com/AluraOnline) e [LinkedIn](https://www.linkedin.com/school/aluracursos). Você pode fazer isso da maneira que preferir, seja através de fotos, stories, vídeos etc. E não se esqueça de marcar o perfil da Alura, estaremos de olho nos projetos!

# 05. 09O que aprendemos?

## Nessa aula, você aprendeu como:

* Utilizar as cores no CSS;
* Utilizar as cores hexadecimais no CSS;
* Utilizar paleta de cores de terceiros;
* Alterar as cores de fundo e dos textos;
* Extrair a cor do Figma para utilizar no CSS;
* Destacar o texto e alterar a cor do texto em destaque.

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário

Parte inferior do formulário