**CURSO EM VÍDEO – HTML / CSS3 – MÓDULO 3:**

**Git e GitHub:** São plataformas para **repositório** e **versionamento** de código.

**Repositório local (git):** Por vários motivos precisamos ter backups de versões de nossos projetos. Para isso usamos o gerenciador de versões (nós indicamos quando queremos salvar determinada versão), tornando possível voltar a usar alguma versão a qualquer momento. Git (mesmo criador do Linux) é um software 100% gratuito que guarda versões de um código **dentro do seu computador. Git 🡪 Mantém versões de software em seu computador**

**Repositório remoto / online (GitHub):** Comprado pela Microsoft. Faz um backup de versionamentos de forma gratuita na nuvem (internet). Ele permite que outra pessoa ou cliente consiga ver o projeto de um website, apenas com um link. **GitHub 🡪 Mantém versões de software na nuvem.** A grande sacada do GitHub é trabalhar em equipe, onde mais pessoas podem trabalhar na mesma ou outra versão de um projeto com você.

* Quando mandamos algo do git (repositório local) para o github (repositório remoto) damos o nome de **PUSH.**

Curso completo / gratuito de GitHub.

**Criando repositório:** A primeira etapa é criar um repositório local (disco rígido) de sua máquina. Depois ele permitira publicar (fazer pushing) para o github (remoto).

\*Dica: na hora de criar repositórios prestar atenção em nomes de pastas, subpastas e arquivos.

**Gerenciamento de repositórios:** Menu do canto superior esquerdo você acessa seus repositórios, ao fazer alterações nos arquivos é preciso fazer um **commit to main** é uma manifestação de modificação. E depois clicar em Push origin no github desktop. **COMMITS** basicamente é criar versões para o código.

**Repositório local < > repositório remoto:** Apagar o local (repositor > remove...), não garante apagar o remoto (settings > delete this repositor) e vise versa.

**Como hospedar um site (gratuitamente) usando o GitHub:** O GitHub consegue hospedar páginas que usam, somente, HTML, CSS e JS. Caso precise usar PHP, node.js entre outros é necessário um serviço de hospedagem.

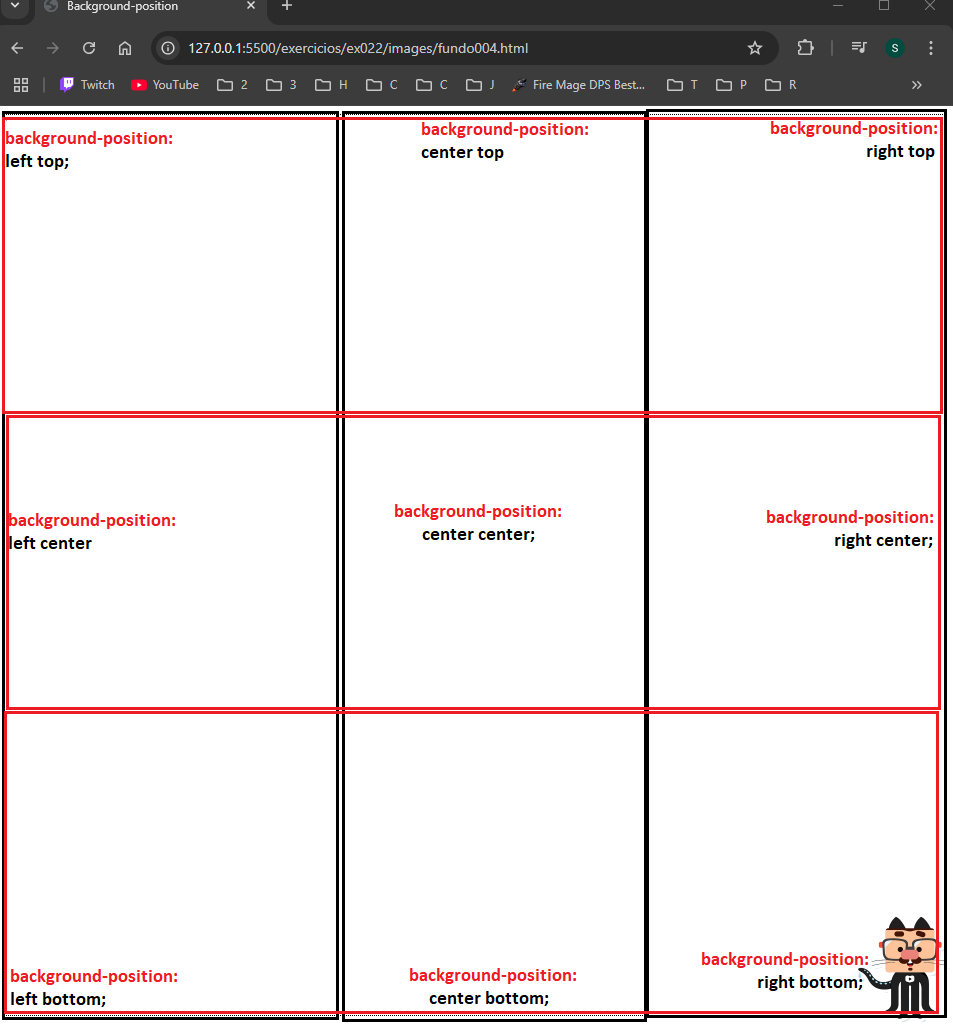
**CLONANDO PROJETOS DO GITHUB:** Basta abrir um repositório e clicar em <> Code > Open with GitHub Desktop e em Choose escolher uma pasta para salvar.

Ideal para trabalhar em uma máquina diferente e “não segura”. Basta clonar o projeto pelo Github Desktop, depois file > options > git > save. Abrir com VScode e fazer todas alterações necessárias. Antes de sair dar push para o repositório online. Fechar tudo > GitHubDesktop > remove repositories + reclicle Bin. File > Options >Sign Out + Git remover dados. Esvaziar lixeira. Desse modo o projeto será atualizado online e limpo da máquina local “não segura” e sem deixar rastro.

**Background-image: url(‘endereço.png’);** É possível usar uma imagem na propriedade background-image no css, podendo inclusive passar valores de arquivos do tipo .jpg, .png, .gif entre outras e também endereços da web (que ficam refém dessa imagem continuar existindo na web). **Por padrão** uma imagem começa a ser esparramada dentro de uma <div> da mesma maneira que lemos (encostada no topo e na esquerda).

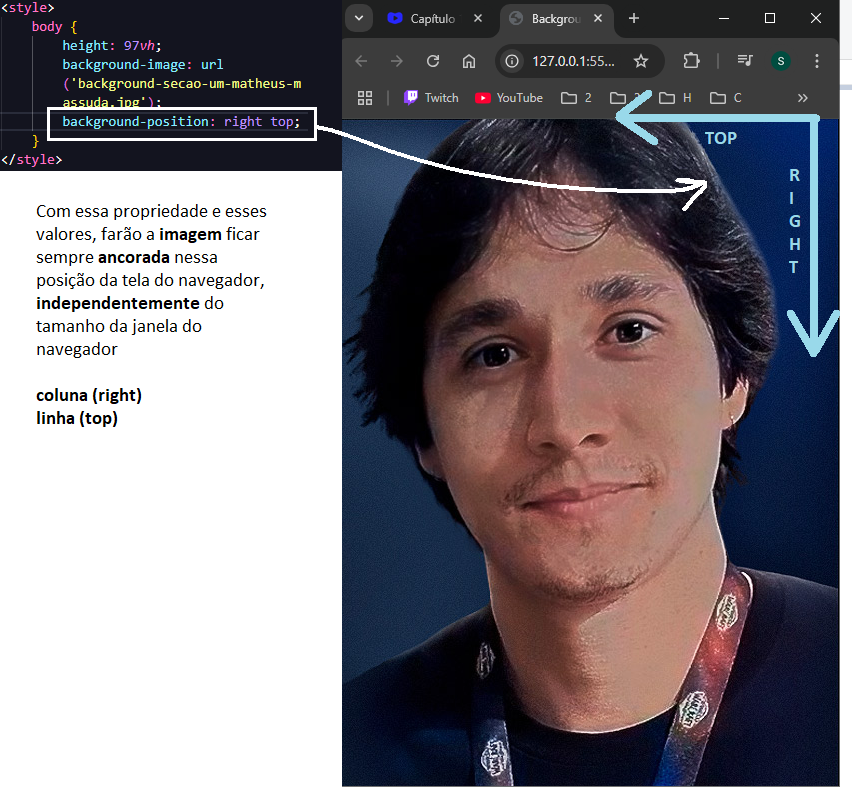
**Background-repeat: repeat;** **REPETIÇÃO:** Essa propriedade sempre causa repetição da imagem para ela completar a tela por causa que por padrão ela passa também (ocultamente) a propriedade **background-repeat: repeat;**

* **No-repeat:** não repete (deixa apenas uma no ponto de partida “canto superior esquerdo”).
* **Repeat:** repete até completar a tela inteira.
* **Repeat-x:** repete no primeiro eixo x (horizontal).
* **Repeat-y:** repete no primeiro eixo y (vertical).

 **Background-position: COLUNA LINHA;** Posiciona a imagem dentro de uma div ou dentro de um espaço imaginário como body. Além disso podemos passar quaisquer valores dentro desse parâmetro para posicionar a imagem da maneira que desejamos como, por exemplo: left top; left center; left bottom; center top; right bottom; Teoricamente podemos dividir a imagem em **9 posições diferentes** com essas combinações.

**Background-size: 100px;** Altera o “tamanho” visual da imagem. Se apenas um valor for passado ele irá repetir esse tamanho para largura e altura.

\***IMPORTANTE:** Se colocarmos o body { height: 100vh;} e colocarmos o background-position: center center; ele irá se ajustar para se posicionar literalmente no centro da tela do navegador. Quando passamos height: 98vh; estamos determinando que a tela de visualização terá 98% da tela, e se passarmos os valores center center em background-position toda vez que mudarmos o tamanho da view port a imagem irá se ajustar para center center. A imagem irá **ancorar** no posicionamento que pedirmos, independentemente do tamanho da janela. Uma outra dica importante é também pré estabelecer uma cor de fundo para o site, pois quando um site é carregado, primeiramente se carrega o conteúdo do site (texto e outros códigos leves e simples). Portanto a experiência do usuário pode ser melhor se tiver uma cor pré estabelecida caso a imagem ainda não tenha sido carregada



**Background-size: auto;** Existe uma maneira de deixar a página “mais ou menos” responsiva para a imagem se ajustar ao tamanho da tela independente do dispositivo que ela for aberta. Por padrão o parâmetro background-size vem com o valor: auto;

1. **CONTAIN:** A imagem será 100% exibida, sem distorção (com proporções originais) e sem corte de pedaços. Com esse valor ele força a imagem acontecer 100% na tela, no entanto ele irá sobrar espaço na tela do navegador a depender do tamanho da janela do navegador.
2. **COVER:** Ele cobre a tela inteira, mesmo que para isso tenha que dar uma cortadinha. É o valor mais utilizado, geralmente. O que pode ser feito é usar {background-size: cover;} e também usar {background-position: coluna linha;} para focalizar a imagem no ponto mais importante e que deve ser exibido para diferentes tamanhos de tela. Nesse caso ele ocupará 100% da View Port, mesmo que para isso não use 100% da imagem.

**Background-attachment:** Cria um vínculo com o background e o local que esse parâmetro está:

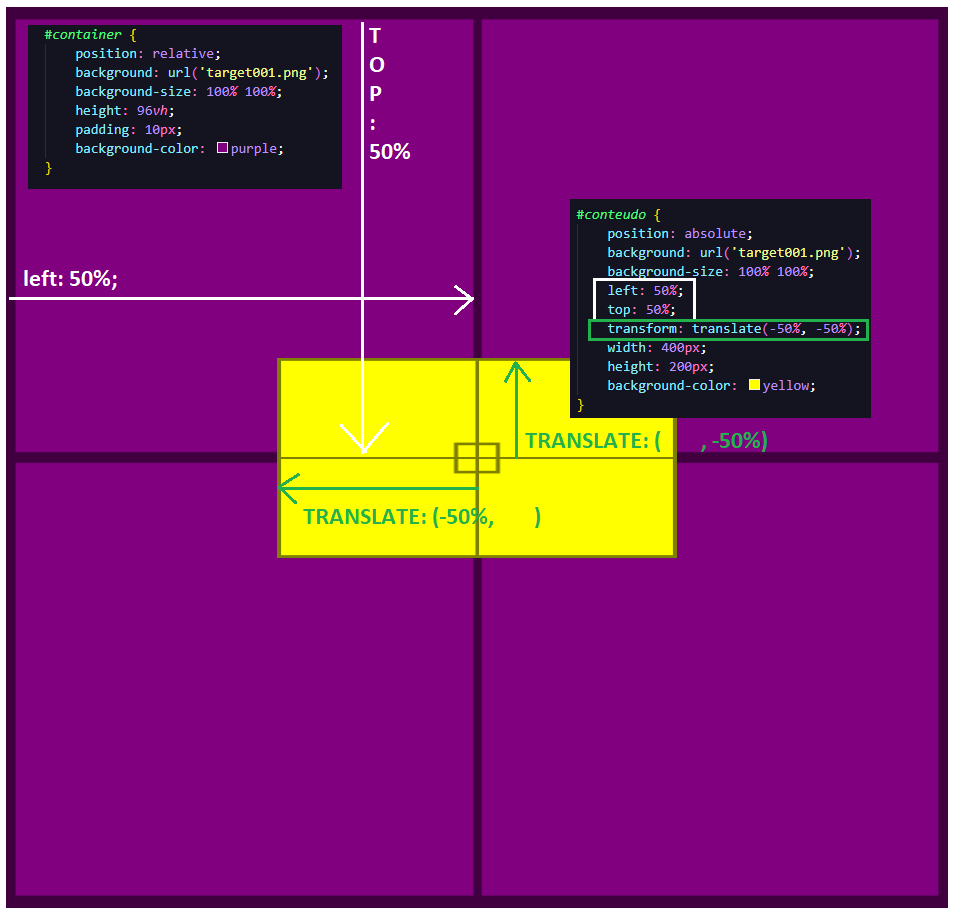
1. **Scoll:**
2. **Fixed:** Faz o background ficar fixo no local que passamos esse parâmetro, quando escrolamos o conteúdo o conteúdo irá andar, mas o background atrás dele com valor fixed ficará fixo.

**\*SHORT-HAND BACKGROUND:** COLOR > IMAGE > POSITION > [SIZE] > ATTACHMENT. O size está na regra, mas ainda não está funcionando em todos os navegadores.

Background: {black url(‘images/wallpaper002.jpg’) center center no-repeat cover fixed;}

**\*POSICIONAMENTOS CSS:** Maneiras de posicionar um elemento horizontalmente e verticalmente dentro de uma caixa.

* **Horizontal:** Possível com **margin: auto**; (coloca sobras para direita e esquerda).
* **Vertical:** É possível ativar as propriedades **left: 50px; e top: 80%;** com valores que podem ser em px, porcentagem, entre outros, quando se passa a propriedade **position: absolute;** para o conteúdo e **position: relative;** para o container. No entanto, quando se usa essas duas propriedades juntas, a posição absolute e relative, o ponto de referência do conteúdo é literalmente o canto superior esquerdo dele. Para juntar o centro do conteúdo com o centro do container é precisar fazer uma transformação no conteúdo com a propriedade **transform: translate (-50%, -50%);** O que estamos dizendo aqui é: faça um movimento de translação do objeto com 50% para -x e 50% para -y).



**Efeito paralaxe:** Basta adicionar o position: fixed;

**Tabelas em HTML:** As tabelas devem ser usadas apenas para se criar **tabelas, NÃO** se deve usar a tag <table> para layout de site. A hierarquia de uma tabela em HTML é: <table> (tabela) 🡪 <tr> (table row = linha de tabela) 🡪 <th> (table header = cabeçalho de tabela) 🡪 <td> (table data = dado de tabela). **Tabelas são as melhores opções para organizar dados tabulados.**

**Border-collapse:** Uma nova propriedade para tabelas em css que permite retirar o mini espaço padrão que existe entre os dados da tabela.

**Bordas:** É possível adicionar bordas em uma tabela apenas passando a propriedade border nas tags <td> e passando uma borda mais grossa em <table>, por exemplo.

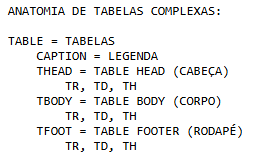
Obs: As tags <tr> <td> não precisam ser fechadas, mas é bom fechar.

**Alinhamento de tabelas:** Dados de tabelas, geralmente são dados ‘string’, ou seja, escritos em texto.

**HORIZONTAL:** Para alinhá-los horizontalmente, basta passar a propriedade: **text-align** com algum dos valores**: left // center // right.**

**VERTICAL: :** Para alinhá-los verticalmente, basta passar a propriedade: **vertical-align:** com algum dos valores: **top // middle // bottom**

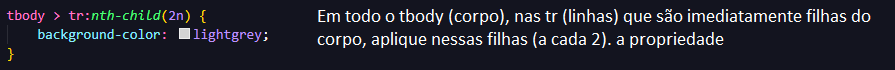
**HIERARQUIA (ANATOMIA) DE TABELAS:**

**TRABALHANDO COM TABELAS SEMÂNTICAS (COMPLEXAS):** Os navegadores são preparados para entender a tabela de uma maneira **semântica,** ou seja, uma maneira organizada. A maneira organizada de se escrever uma tabela é determinando ser uma tabela **<table>** e depois organizando essa tabela em mini “seções” **<thead> (cabeçalho) // <tbody> (corpo) // <tfoot> (rodapé).** Inclusive, independentemente de como escrevemos a ordem de <thead> <tbody> e <tfoot> no HTML5 o navegador já coloca em ordem quando a página é exibida.

**\*IMPORTANTE: TD e TH (SIGNIFICADO)** TH’s são usados para **títulos** de dados, por exemplo: estados, nomes, idade, sexo. TD’s são **dados** dos títulos, respectivamente: São Paulo, Matheus, 26, masculino. Além disso é **muito importante**  passar uma propriedade nas TH’s que é: <th scope: col> </th> , assim estamos declarando que para esse título os dados dele estarão no escopo da **coluna.** O valor para a propriedade scope é um valor de **onde está o dado relativo a ele.** A propriedade scope não tem resultado visual, mas é muito importante para o significado e motores de busca.

**scope= “col” // scope=”row” TH SEMPRE TEM ESCOPO: para colocar escopo em uma th de uma tabela devemos nos perguntar como os dados desse título estão dispostos: estão dispostos em linhas ou colunas? É somente uma linha ou coluna ou um grupo de linhas e colunas?**

**Efeito zebrado:** É possível passar uma classe em cada linha que queremos o efeito zebrado, mas isso seria péssimo em funcionalidade. Por isso é mais rápido e útil fazer com um pseudo elemento: Dentro dos parênteses é possível passar tipos de propriedades: (1n), (2n), (3n)... (125n) que são numéricas, ou podemos passar **(odd**) para zebrar as **ímpares** ou (**even**) para zebrar as **pares.** Inclusive, além de zebrar é possível passar outras propriedades de CSS para estilizar.



**CABEÇALHOS FIXOS EM TABELAS:** Para deixar o cabeçalho de uma tabela fixo, precisamos fazer duas coisas: 1) tornar o comportamento da tabela position: relative e 2) tornar o comportamento do cabeçalho da tabela como position: sticky; Além disso precisamos passar top: -1 e background-color: igual a cor que deixamos o cabeçalho estático. Esse recurso ainda não funciona em todos os navegadores.

**Mesclagem de células em tabelas HTML:** Mesclar células em tabelas de HTML é muito fácil, desde que você entenda o que está sendo mesclado. Quando vamos mesclar células dentro de uma **mesma linha,** na verdade estamos mesclando colunas e o contrário: quando vamos mesclar células dentro de uma **mesma coluna**, na verdade estamos mesclando linhas. Portanto se quero mesclar itens que estão lado a lado, devo usar dentro da tag o atributo **colspan=”n°”** e para mesclar dentro de uma mesma coluna devemos usar dentro dessa tag o atributo **rowspan=”n°”.** Sendo “n°” o número de células que desejamos mesclar.

**ZEBRANDO COLUNAS - Personalizando COLUNAS (COL) e GRUPOS DE COLUNAS (COLGROUP):** Surgiu a tag <colgroup> com as filhas <col> (uma para cada coluna de sua tabela) que permite fazer isso.

**Tabelas RESPONSIVAS com <div> e overflow:** Para algumas telas pequenas as tabelas podem acabar causando uma rolagem **indesejada** para lateral ou vertical da tela, geralmente a rolagem mais indesejada é a **HORIZONTAL**. Para resolver isso é muito siples, basta **enveloparmos a tag <table>** em uma **div** com um **#id=”container”** e no estilo de css passar as seguintes propriedades: **div#container { overflow-x: auto;}.** Além disso essa propriedade também possui variações overflow-y (eixo y) e overflow que contém os valores: **auto, scroll e hidden.**