**Módulo 1 - HTML5 e CSS3**

**Curso em vídeo – Guanabara**

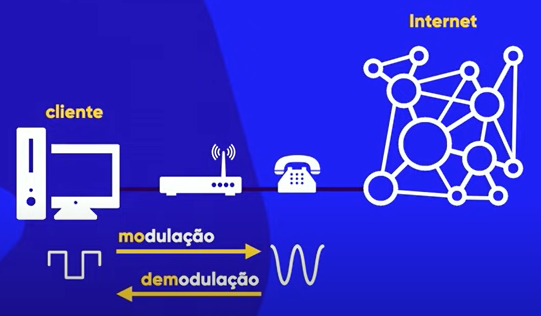
Github.com/gustavoguanabara

Gustavoguanabara.github.io

**História da Internet**

* **Protocolo TCP/IP** (Transmission Control Protocol/Internet Protocol): surgiu com o intuito de padronizar a comunicação entre os computadores da **ARPANET** nos estados unidos, em computadores específicos de bases militares espalhados pelo país. Em resumo hoje em dia é o conjunto de protocolos que permite a comunicação entre computadores e servidores, formando a Internet
* O nome internet então surgiu da ARPANET, que foi uma rede criada com o intuito de facilitar a comunicação e transferência de dados entre bases militares nos estados unidos, com o intuito de proteger os dados
* Hoje em dia o mundo inteiro é conectado por cabos submarinos que permitem essa troca de informações, permitindo a existência da internet. A comunicação entre continentes é feita predominantemente pelos cabos, já a conexão pra dentro do país pode ser feita por diferentes meios como satélites e outros.
* Documentário recomendado: como a internet funciona: <https://www.youtube.com/watch?v=TNQsmPf24go>
* **HTTP** (Hyper Text Transfer Protocol)
* **HTML** (Hyper Text Markup Language)
* **WWW** (World Wide Web)
* Dentro da internet então temos diversos servidores; o www é uma rede de servidores especializados em http

**Como a internet funciona**

* O computador funciona em linguagem de máquina, entende apenas código binário, representados por sequências de 0 e 1, chamados de bits
* **Byte**: sequência de 8 bits, é a porção mínima para que se represente algum dado. Ex: 01000001 – representa a letra A. Para ver a representação de cada letra você pode pesquisar pelo código multibyte **UTF-8**; é um código que utiliza de 1 a 4 bytes pra representar uma diversidade de símbolos, letras, acentuações, pontos, emojis, etc.
* Daí temos os múltiplos dos bytes
* 1 KiloByte (KB) = 1024 bytes
* 1 MegaByte (MB) = 1024 KB
* 1 GigaByte (GB) = 1024 MB
* 1 TeraByte (TB) = 1024 GB e assim por diante
* Obs – o computador não trabalha com a base 10, trabalha com base binária, 2, então o 1024 é devido a isso, 210 = 1024
* Nota: ao escrever MB você está se referindo a MegaBytes; ao escrever Mb você está se referindo a Megabits. Mas muitas vezes escrevem errado isso por aí. MB é geralmente referido a armazenamento, por exemplo HD, pen drive, etc; já Mb é mais utilizado pra transmissão, por exemplo a sua rede de internet em casa é de 60Megabits, “ah uma rede de internet mto rápida é de 1 giga (gigabit)”
* Modem: aparelho para realizar a Demodulação das ondas que chegam da internet até o computador e a Modulação das ondas q saem do computador pra internet
* Cliente-Servidor: os servidores são computadores que disponibilizam serviços, arquivos e outras coisas por meio da internet (netflix, instagram, youtube, etc), e os clientes são os computadores que acessam esses servidores.
* Cada servidor é identificado por uma numeração (ex: 3.224.112.47)
* Você também ao se conectar à internet recebe uma numeração (ex: 201.17.81.22), isso é o endereço IP. Ao desligar e ligar o modem novamente ele pega já outro endereço IP. Os endereços IPs de cada site também mudam com frequência
* [www.iplocation.net](http://www.iplocation.net) – site para descobrir qual o endereço IP
* DNS (domain name system) – serve para relacionar um domínio (ex: instagram.com) ao seu endereço IP. Os domínios então facilitam na hora que vamos pesquisar por sites, ao em vez de procurar por endereço IP. Ao acessar um site pelo domínio então o Modem lança o domínio para o DNS que dá o endereço IP para que então o Modem procure pelo endereço IP na internet.

**O que é Domínio e Hospedagem**

* Domínio: nome que identifica um site
* Hospedagem: local (servidor) onde o site vai estar armazenado
* URL (uniform resource locator – localizador de recursos padrão/uniforme): utilizado para ir para um lugar específico, uma página exata de um dado site.
* **www.github.com/gustavoguanabara**:
* todo esse nome seria a url
* **www**: subdomínio. O subdomínio nem sempre é www, ex: gustavoguanabara.github.io, o subdomínio é o gustavoguanabara
* **github.com**: domínio.
* Dentro do domínio tem o TLD (Top Level Domain), nesse caso o “**.com**”. Mas existem também gTLD (Generic Top Level Domain), como “**.net**”. “**.io**”, “**.info**”, “**.gov**”. ccTLD: country code (código de país): “**.br**”, “**.ca**”, etc
* **/gustavoguanabara**: Caminho

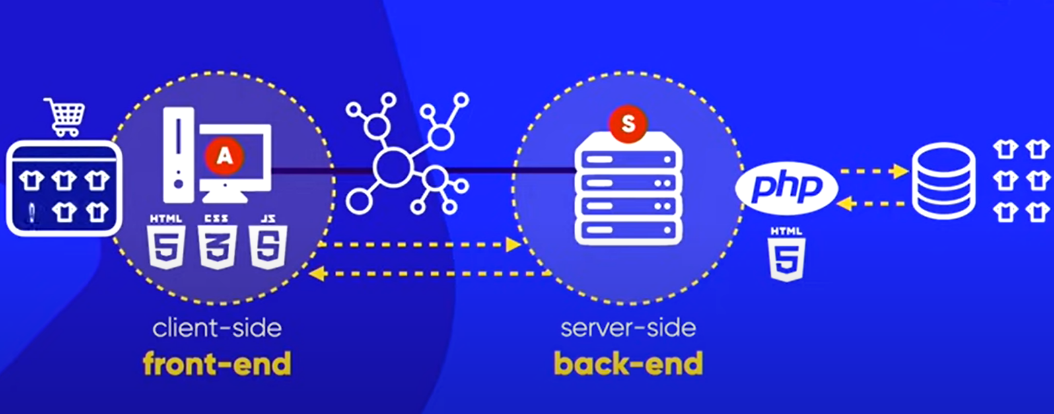
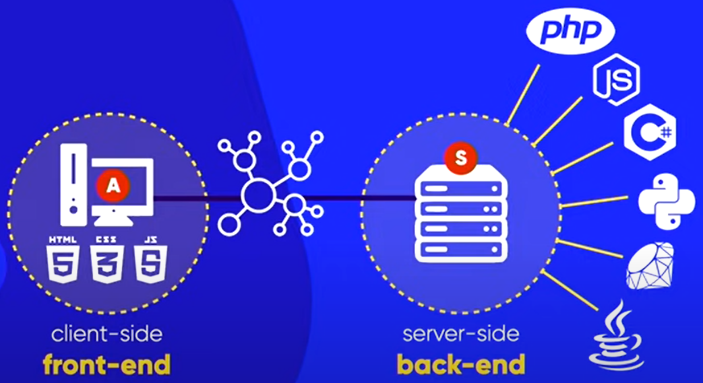
**A diferença entre HTML, CSS e Javascript**

* HTML – HyperText Markup Language – Linguagem de Marcação de Hipertexto (Hipertexto são textos que ao serem clicados direcionam para outro local). É uma linguagem focada no conteúdo, que pode ser texto, imagem, vídeo, tabela, etc
* CSS – Cascading Style Sheets – Folhas de Estilo em Cascata (em cascata pois uma vai modificando a característica da outra). Linguagem focada no design, dando um atrativo visual para a página, configurando coisas como cores, sombras, tamanhos, posicionamento, etc
* Javascript: linguagem de programação, que será responsável pela responsividade e interatividade do site com o usuário (menus, animações, popups, validações, etc)
* HTML:
* No HTML são utilizadas diversas Tags, como por exemplo <h1></h1> para fazer com que o texto seja um título. Algumas como essa tem fechamento, outras não
* Toda Tag pode também ter parâmetro e valores, como mostrado abaixo
* Estrutura básica de documento HTML:



* CSS:
* No CSS geralmente é colocado uma tag do html como seletor, é aberto e fechado chaves e dentro dessas chaves então são escritas as declarações, compostas por uma propriedade e o valor. Ao final de cada declaração é preciso colocar ponto e virgula
* Ao acessar um site então, após passar por todas aquelas etapas mencionadas antes, domínio, endereço IP, DNS, modem, rede, servidor, o servidor então disponibiliza uma cópia do HTML, CSS daquele site para o seu computador, o google chrome então é responsável por fazer esse processamento e análise desse código para gerar o resultado visual que aparece na sua tela

**Front-End, Back-End e Full Stack**

* O Front-end é responsável pelas tecnologias que irão rodar do lado do cliente, por exemplo o html, css e javascript, que após serem enviados pelo servidor, o navegador do cliente é que vai ser o responsável por interpretar essa informação. Então é algo mais focado na parte visual, no usuário, na experiência dele, na interatividade
* Back-end: responsável pelas tecnologias que irão rodar do lado do servidor. Por exemplo bancos de dados com informações de login e senha; num e-commerce dados a respeito de estoques, vendas etc
* Full Stack: Desenvolvedor com experiência em ambas áreas, Front e Back

**Instalando todas as Ferramentas**

* Google Chrome
* Visual Studio Code

**TAGs**

* Ao colocar uma ! e da enter vai aparecer as tags iniciais do HTML. Importante mudar a linguagem para português caso o site for ser escrito em português
* **<h1></h1>**
* Título 1
* Também tem h2, 3, 4, 5, 6
* H vem de “headings”, que signific titulo
* **<p></p>**
* Parágrafo
* **<hr>**
* Horizontal Line – cria uma linha fina que atravessa a tela horizontalmente
* **<br>**
* Break – Quebra de linha
* Para fazer uma quebra de linha maior com mais espaço não é recomendável usar essa tag várias vezes, em CSS ele vai mostrar mais
* **&lt e &gt**
* Servem pra quando vc quer escrever uma tag mas como texto, pra que ela apareça para o leitor ao em vez de ser interpretada como tag, ex: &lt p &gt; isso fará aparecer <p> na tela, ao em vez do visual studio interpretar como um paragrafo
* Lt: less than – menor que
* Gt: Greater than – maior que
* **<!-- -->**
* Abre um comentário que será mostrado somente no código, não na página
* Existem tags para **caracteres especiais**, como: &reg; &copy; &trade; mas tem mais detalhes no pdf do curso (capítulo 5 – página 4), são chamadas de HTML Entities, se pesquisar acha. Também tem nesse site:

<https://www.w3schools.com/charsets/ref_utf_symbols.asp>

* Para colocar **emojis: &# + código do emoji** ; e vai aparecer, o código do emoji vc pode pegar no **emojipedia.org**; vc coloca o código sem o “U+” q aparece no começo
* **Img: <img src="" alt="">**
* Tag para carregar uma imagem. Src é de Source (origem do arquivo) e alt de alternative, um texto que explica melhor a imagem. Serve para melhorar a indexação do site em mecanismos de buscas e também para deficientes visuais, pra leitores de tela
* Digite apenas “img” e dê enter, que o vscode carrega os parâmetros
* Na parte de src=“” se vc colocar no meio das aspas e clicar **Ctrl+Espaço** vai aparecer as opções de imagem na pasta do arquivo, e vc também pode selecionar subpastas dentro dessa pasta, caso entre em subpasta antes do nome do arquivo vai aparecer o nome da subpasta + /
* Direitos autorais de imagem: Ao pesquisar no google imagens vc pode ir em ferramentas e direitos de uso e colocar **“marcadas para reutilização com modificação”**, vai mostrar imagens de pessoas que deixaram a imagem disponível pra ser usada e modificada. As Marcadas para reutilização também podem ser usadas comercialmente mas nn alteradas
* Sites pra imagens: **Unsplash, pexels**
* **Gimp:** programa de manipulação de imagem, gratuito
* Geralmente na web utilizamos imagens em formato **JPEG ou PNG**. JPEG (jpg) é um formato com alta compactação de imagem, fazendo os arquivos ficarem mais leves. A PNG foi criada pela WWW, surgiu pra substituir o formato gif, com a diferença de q o png não permite animações, mas é um formato de imagem que permite transparência no fundo. E se nn me engano a qualidade é um pouco melhor, nn é tão compactada quanto o jpeg. Mas ele falou que é melhor deixar pra usar png só quando precisar de transparência de fundo
* É importante tomar cuidado com o tamanho dos arquivos de imagem, pra nn deixar o carregamento do site mto lento
* Para pesquisar por imagem de **alta resolução** no google coloque em **“tamanho grande”**. Também tem um filtro específico pra procurar por imagens com transparência, que são as em png
* **Redimensionamento:** Ao pegar uma imagem grande é importante redimensiona-la para o tamanho que vc quer, fazendo isso o arquivo fica menos pesado, da pra fazer pelo gimp. É importante pois no css vc pode redimensionar a imagem porém não altera o tamanho do arquivo
* **Favicon:** ícone favorito, é o ícone que aparece na aba de um site. Site de ícones grátis: IconArchive. Site para converter arquivo png em ícone: Favicon.io. Alguns navegadores carregam favicons em formato de imagem, mas em geral carrega melhor se o arquivo estiver em formato “.ico”
* Para adicionar um Favicon:
* Na parte de Head, do html, acima de title, escreva “link”, então selecione “link:favicon” e dê enter. Na parte de href=””, apague o texto e aperte ctrl+espaço para aparecer os arquivos da pasta
* No rel=”” coloque: "shortcut icon"
* Após o href, coloque: type="image/x-icon"
* **Lorem:** escreve várias palavras sem sentido pra preencher espaço. Vc pode escrever a quantidade de palavras que quer após o lorem, ex: lorem50 escreve 50 palavras
* <b></b>
* Com essa tag vc coloca uma sentença em negrito, mas somente isso, não dá um significado para a sentença
* **<strong></strong>**
* Com essa tag além de a sentença ficar em negrito, a sentença é reconhecida como importante, algo enfático
* <i></i>
* Coloca uma sentença em itálico, mas sem dar um significado
* **<em></em>**
* Coloca uma sentença em itálico e dá ênfase a ela
* <u></u>
* Colocava a sentença sublinhada
* **<ins></ins>**
* Coloca a sentença sublinhada, mas é considerado como um texto inserido, usado pra dar ênfase e indicar que ele foi adicionado depois
* **<del></del>**
* Tag para texto deletado; o texto aparecerá riscado
* **<address></address>**
* Identifica uma sentença como sendo um endereço
* De toda forma é melhor deixar pra colocar negrito, itálico e sublinhado no css, se nn for pra dar sentido ou ênfase a sentença. O HTML serve mais pra dar sentido ao conteúdo, como identificar um trecho como um endereço
* **Selecionando uma sentença, apertando Ctrl + Shit + p, e selecionando “wrap with abreviation”, vc pode já escrever uma tag pra envelopar a sentença**
* <mark></mark>
* Marca uma sentença, como se fosse marca texto; marca em amarelo, mas pra estilizar é melhor usar css. Se for algo pontual vc pode colocar o css dentro dessa tag: <mark style=“background-color: ;”>. Mas depois será dado mais detalhes no css
* <big></big> e <small></small>; são tags para colocar palavras maiores ou menores, mas é mais recomendável css, e a tag big ficou obsoleta
* **<sup></sup> e <sub></sub>**
* O sup é pra texto sobrescrito (x2), e o sub pra texto subscrito (H2O)
* **<code></code>**
* Deixa a fonte monoespaçada; quando todas as letras ocupam o mesmo tamanho de espaço. Indicado pra ler códigos
* **<pre></pre>**
* Significa texto pré-formatado. Ao utilizar essa tag as quebras de linhas que vc separar com enter serão apresentadas como linhas separadas em vez de aparecer tudo numa linha só. E o texto vai aparecer como outra fonte semelhante ao do comando code
* É bom pra gerar quebras de linha ao em vez de usar um monte de comando br
* É bom pra separar as linhas dentro da tag <code>
* **<q></q>**
* Quote – Citação. O texto é reconhecido como uma citação, e é colocado entre aspas
* **<Blockquote></blockquote>**
* Usado pra fazer uma citação mais longa, com as palavras do autor
* Dentro do blockquote vc pode colocar cite= pra colocar um link referenciando de onde vem a citação, quem vai reconhecer isso são os mecanismos de busca. <blockquote cite=”<https://........>”>
* **<abbr></abbr>**
* Serve pra abreviações. Colocando essa tag o sistema reconhece como uma abreviação e da também pra colocar o título completo da abreviação que poderá ser exibido na página quando o usuário passar o mouse por cima da sigla
* <abbr title="HyperText Markup Language">HTML</abbr>
* **<bdo></bdo>**
* Tag para inverter o texto. bidirectional override
* Bdo dir=”rtl” – direction= right to left
* **<ol></ol>**
* Ordered List
* **<li></li>** - List Item. Na vdd o </li> no final se tornou opcional no html5, da pra escrever só com o <li> no inicio se quiser
* <ol **type=**””> fornece tipos de ordenação, numero, letras maiúsculas minúsculas etc. Se quiser que a lista inicie em um numero ou letra especifico coloque por exemplo <ol type=”1” **start=”4”**>; caso o tipo seja letra no start também use numeração pra se referir ao 4° elemento, por exemplo se type for A e vc quiser começar pela letra C, no start coloque 3 pois C é a 3ª letra do alfabeto
* **<ul></ul>**
* Unordered list
* Tem que colocar li também
* No type, vc pode colocar: circle, disc, square. Disc é o padrão
* Obs: é possível colocar listas dentro de listas
* **<dl></dl>**
* Lista de Definição
* Sempre lembrar que essas marcações são muito importantes para **optimizar os mecanismos de buscas** até o seu site
* Dentro da lista vc vai colocar o **<dt></dt>** que é o termo de definição, e o **<dd></dd>** que é a descrição do termo
* **<a href=""> </a>**
* “a” vem de âncora. É a tag usada para inserir links. Href é a url completa
* Mas para que o link abra uma outra aba é preciso colocar também o parâmetro **target=“\_blank”,** sendo importante também colocar **rel=”external”** pra indicar que é um link pra um site externo. Ex:

<a href="https://gustavoguanabara.github.io/" target="\_blank" rel="external">clicando aqui</a>

* O atributo **target** suporta os seguintes valores:

‣\_blank vai abrir o link em uma nova janela em branco

‣ \_self vai abrir o link na janela ou frame atual (padrão)

‣ \_top vai desfazer todos os frames e abrir o destino no navegador completo

‣ \_parent similar ao uso do \_top em uma referência à janela mãe nome-do-frame caso esteja usando frames, indicar o nome da janela a abrir

* Como o uso de frames é uma técnica quase em desuso, vamos nos basear apenas nas duas primeiras opções
* Já o atributo **rel** aceita os seguintes valores:

‣next indica que o link é para a próxima parte do documento atual

‣prev indica que o link é para a parte anterior do documento atual

‣author indica que é um link para o site do autor do artigo atual

‣external indica que é um link para outro site que não faz parte do site atual

‣ nofollow indica que é um link para um site não endossado, como um link pago

* Outro atributo útil é o **hreflang=””,** que indica o idioma da pagina associada ao link, servindo pra softwares de tradução
* Vc também pode gerar um link interno, pra navegar por diferentes páginas do seu site por exemplo

<p>Esta é a 1ª página, para acessar a 2ª página <a href="pag002.html" rel="next">>clique aqui</a>

* Apertando Ctrl+Espaço dentro das “” o próprio programa lhe mostra os arquivos da pasta disponíveis. Caso haja uma subpasta também irá mostrar. É importante fazer esse processo pra referenciar arquivos de subpastas pois se só colocar o nome não vai pegar
* Agora se você tiver querendo referenciar um arquivo de uma pasta anterior é preciso colocar **../** dentro das “” pra aparecer os arquivos da página anterior
* O rel pode ser next ou previous para indicar uma página seguinte, ou pra indicar que vai voltar pra 1ª página ou uma página anterior
* **Link pra download**: referencie um arquivo dentro de uma das pastas e coloque o atributo download, e também type = “application/pdf” pra arquivos em pdf, pra zip é /zip e pra outros tipos tem que pesquisar no site iana.org/assignments/media-types/ mas na apostila também tem alguns exemplos

<a href="livro/10 - Ligações em toda parte.pdf" download="10 - Ligações em toda parte.pdf" type="application/pdf">Baixar livro em PDF</a>

**Imagens dinâmicas**

* Todo o seu conteúdo deve se adaptar a tela do aparelho utilizado pelo usuário, seja notbook ou celular. É possível fazer isso com CSS, porém fazendo pelo CSS o tamanho do arquivo que será carregado não muda, já usando o HTML conseguimos selecionar o arquivo exato que será exibido
* Ex de tags pra utilizar imagens dinâmicas:

<picture>

        <source media="(max-width: 750px)" srcset="Imagens/Img-p.png" type="image/png">

        <source media="(max-width: 1050px)" srcset="Imagens/Img-m.png" type="image/png">

        <img src="Imagens/Img-g.png" alt="Imagem Flexível">

    </picture>

* **<picture></picture>**
* É a tag que vamos utilizar pra criar imagens dinâmicas
* É importante criar as imagens já prontas com os tamanhos que vc vai utilizar, pode utilizar o gimp pra isso
* Dentro da tag Picture vc vai colocar de forma ordenada de cima pra baixo da menor pra maior das imagens
* A última imagem, que é a maior, é a que vai ser aberta por padrão, as demais são abertas caso o tamanho máximo total da tela seja menor que X pixels
* **<source media=”” srcset=”” type=””>**
* A tag source vai nos permitir inserir uma imagem dinâmica, o atributo media=”(max-width = X px)” vai definir que caso a tela seja menor que esse tamanho essa será a imagem que vai ser exibida.
* O atributo srcset linka a imagem que vai ser exibida (obs – dentro desse atributo nn da pra usar Ctrl+Espaço, então apenas digite)
* E o atributo type define qual o tipo de imagem (media type): image/png, image/jpg, etc
* Obs: Você pode até colocar o valor exato da imagem (ex: 1000px) na propriedade max-width, mas vai perceber que um valor ligeiramente acima (1050 por exemplo) vai gerar resultados mais interessantes, pois evita o surgimento da barra de rolagem lateral
* **Centralizando uma imagem:**

.container {

    display: flex;

    justify-content: center; /\* Centraliza horizontalmente \*/

    align-items: center; /\* Centraliza verticalmente \*/

    height: 100vh; /\* Altura da janela de visualização \*/

}

Considerando que você atribuiu a classe **container** dentro da tag **picture**. Ou pode simplesmente usar o próprio Picture no css pra estiliza-lo, caso queira que todas as pictures tenham aquela configuração

Também da pra fazer assim:

img{

    width: 100%;

}

**Áudio**

* **<audio></audio>**
* Formatos de áudio: MP3, OGG, WAV (wav é muito pesado, não é recomendável)
* Forma simples de adicionar áudio (utilizando atributos):

<audio src="Media/3AM lyrics.mp3" controls autoplay></audio>

Controls – mostra o controle na tela que pode ser clicado pra tocar a musica, e autoplay faz a musica iniciar sozinha sem precisar clicar no botão

Porém essa forma é um pouco limitada, inclusive quando coloca sem o controls o autoplay nn funcionou

* Forma correta de se colocar (utilizando a tag **<source>**):

<audio preload="metadata" autoplay controls loop>

        <source src="Media/3AM lyrics.mp3" type="audio/mpeg">

        <source src="Media/3AM lyrics.ogg" type="audio/ogg">

        <source src="Media/3AM lyrics.wav" type="audio/wav">

        <p>Infelizmente seu navegador não consegue reproduzir áudio</p>

        <a href="" download="" type="">Clique aqui pra fazer o download do arquivo MP3</a>

    </audio>

* Usaremos a tag **source** dentro da tag áudio. Ela possui os atributos **src** e **type**, que indicam o nome do arquivo e o media type.
* O navegador tentará abrir o primeiro arquivo, caso ele nn tenha compatibilidade ele tenta com o segundo e assim por diante, e caso não consiga abrir nenhum será exibida a mensagem disponibilizando o download
* Além disso é importante colocar atributos dentro da tag **áudio**:
* **Preload**=”” define a forma como o áudio vai ser pre-carregado

- auto: não recomendável, o site só irá carregar quando carregar o arquivo completo do áudio, o que pode atrasar o carregamento da página

- metadata: carrega apenas algumas características previamente (como nome do arquivo, etc), mas o áudio em si só será carregado quando o usuário clicar no botão (ai quando o usuario clicar pode demorar um pouco pra carregar dependendo do tamanho do arquivo), ou caso esteja em autoplay carregará no inicio

- none: não será carregado nenhum dado do arquivo previamente

* **Autoplay** – faz a musica começar a tocar automaticamente quando o site for aberto
* **Controls** – aparece o controle pra dar play, ajustar o tempo etc
* **Loop** – faz a musica ser reiniciada automaticamente quando terminar de tocar

**Vídeos**

* O site pexels pode ser usado pra baixar vídeos também
* Formatos: MP4, WEBM (levinho, qualidade baixa), OGV
* HandBrake: programa gratuito, vc pode utilizá-lo para converter o seu vídeo para outros formatos e deixar mais leve
* Tag: **<video></video>**

<video width="500" poster="imagens/limoes-capa.png" controls>

        <source src="midia/meu-video.mp4" type="video/mp4">

        <source src="midia/meu-video.m4v" type="video/mp4">

        <source src="midia/meu-video.webm" type="video/webm">

        <source src="midia/meu-video.ogv" type="video/ogg">

        <p>Seu navegador não tem compatibilidade com vídeos</p>

    </video>

* É basicamente o mesmo que áudios, só se atente para os media types
* **Width:** ajusta o tamanho que o vídeo aparecerá na tela. Porém o arquivo será carregado com o tamanho original do mesmo jeito
* **Poster:** serve pra inserir uma imagem de capa pro vídeo
* Vc também pode usar o autoplay e loop
* Observação: porém essa é a forma de se colocar vídeos hospedados no próprio servidor, isso pode acabar levando a um alto consumo de banda caso seu site tenha muitos acessos por exemplo, o que pode levar a um alto custo de hospedagem
* **Adicionando vídeos de serviços de hospedagem de vídeo:**
* Youtube: clique em compartilhar um vídeo, e em **incorporar**, assim ele vai te fornecer um código HTML pronto, é só colar no seu código, e então ao visualizar o vídeo o usuário estará consumindo dados do youtube e não do seu servidor
* Outra consideração importante é que colocando o vídeo no youtube e linkando no site isso amplia ainda mais a abrangência que o algoritmo do google vai dar
* Outra plataforma é o **Vimeo**, que permite que você ajuste se o vídeo pode ser visto por qualquer pessoa ou se vai ser privado (o Guanabara utiliza no studonauta). Ele tem uma qualidade melhor q a do youtube, mas o algoritmo não é tão bom, e pode demorar um pouco pra carregar e dar travadinhas

**Estilos CSS inline**

* Estilo CSS inline – refere-se a uma técnica de CSS onde aplicamos os parâmetros de CSS isolados em uma linha específica. Ex:

<body style="background-color: aquamarine; font-family: Arial, Helvetica, sans-serif;">

<h1 style="color: blueviolet;">Capítulo 1</h1>

* Porém é uma forma limitada de aplicar CSS, pois se restringe a uma única parte do código

**Estilos CSS interno** **(local)**

* É melhor do que fazer inline, e é mais indicado pra caso vc esteja fazendo uma página simples e não precisar aplicar configurações pra páginas múltiplas.
* Consiste em implementar a tag **<style></style>** dentro da tag head do código
* Ex:

<style>

        body{

            background-color: lightblue;

            font-family: Arial, Helvetica, sans-serif;

            font-size: 20px;

        }

        h1{

            color: darkblue;

            background-color: white;

        }

        h2{

            color: darkred;

        }

        p{

            text-align: justify;

        }

</style>

* Pontos negativos: as vezes o CSS feito dessa forma pode acabar ocupando até mais espaço do que o código html em si; outra coisa é que pra aplicar em múltiplas páginas vc precisaria copiar e colar todo o código css pra todas as páginas

**CSS externo**

* Essa é a melhor forma e mais recomendada de se fazer o CSS, vc cria um arquivo isolado (ex: “style.css”) que armazena todo o código CSS e que pode ser linkado pra configurar páginas diferentes
* Pra linkar um arquivo HTML a um CSS, vc vai inserir a tag **link** dentro de **head**. Ao escrever link vc já vai perceber que vai aparecer diferentes opções, vc clica em **link:css**, e vai carregar o seguinte:

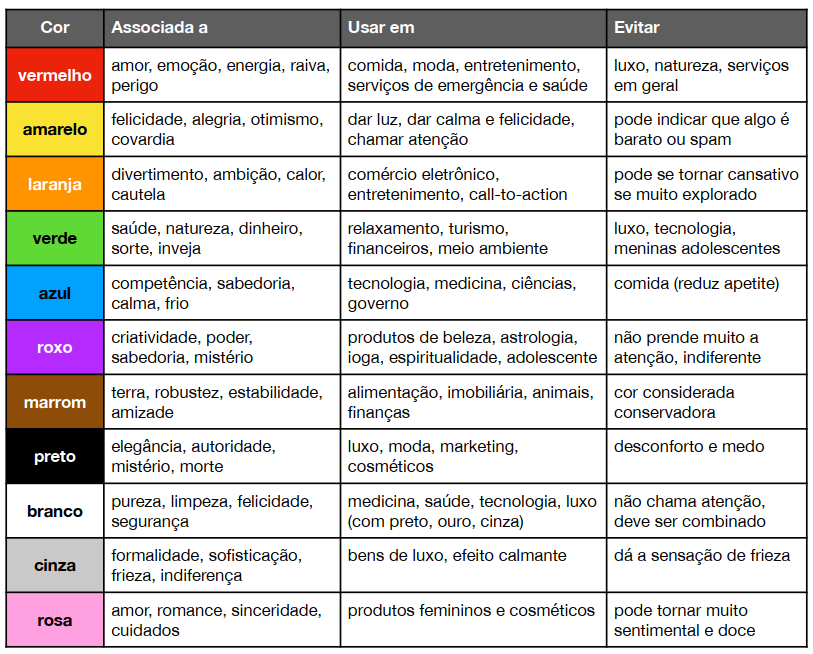
<link rel="stylesheet" href="style.css">

* Em **href=””** vc vai inserir o nome do arquivo CSS
* Outra coisa importante é inserir a regra **@charset “UTF-8”;** no início do arquivo CSS. Essa linha não é obrigatória e normalmente nem vai aparecer na maioria dos seus arquivos de configurações de estilo, mas caso você comece a ter problemas de compatibilidade com alguns caracteres, acentuação e tals, saiba que ela existe.
* Obs – é possível inserir mais de um link css dentro de um arquivo html, caso vc tenha códigos separados de CSS específicos pra determinadas páginas por exemplo. Da também pra usar link e ao mesmo tempo usar a tag style dentro do head pra dar configurações específicas pra uma determinada página

**Módulo 2 - HTML5 e CSS3**

**Psicologia das Cores**

* A harmonia das cores e a apresentação visual do seu site é fundamental pra transmitir uma sensação boa ao usuário e despertar maior interesse nele
* Azul: é a cor com maior taxa de aprovação e a menor taxa de desaprovação. Remete a harmonia, segurança, profissionalismo, integridade
* Mas muita atenção ao seguir guias de cores e artigos, pois eles não devem ser considerados como uma verdade absoluta para todos os mercados e situações. Considere as recomendações, mas não se prenda a elas.



**Representando Cores em CSS**

* **Representação por nome:** Dentro de algum parâmetro de cor vc pode clicar **Ctrl + Espaço** para visualizar as cores prontas disponíveis. Mas é um pouco limitado pois existem inúmeras outras possibilidades de cores.
* No entanto, se vc passar o mouse por cima da cor, vc pode então alterar a cor da forma como quiser, mudando a cor, o tom e a transparência, e da até pra selecionar se quer rgb, hsl ou hexadecimal

<h2 style="background-color: blue; color: white;">Exemplos de Cores</h2>

* **Representação por código Hexadecimal**: **#000000** - toda cor é formada por uma combinação de Vermelho, Verde e Azul (**RGB – Red, Green, Blue**).
* Esse código com 6 dígitos representa quanto de cada cor está presente na cor em questão:
* Os 2 primeiros dígitos representam a cor Vermelho,
* Os 2 digitos do meio representam a cor Verde,
* E os 2 digitos finais representam a cor Azul
* A cor: #0000ff representa o azul puro. O F é o digito mais significativo dentro da base hexadecimal
* No gimp vc pode verificar as paletas de cores RGB clicando no iconizinho pintado nas ferramentas do canto superior esquerdo

<h2 style="background-color: #0000ff; color: #ffffff;">

* **Representação pelo RGB**:
* Como mencionado, cada 2 caracteres hexadecimais representam 1 cor
* Porém, um número com 2 caracteres no sistema hexadecimal quando passados para um sistema decimal, podem representar um número que varia de **0-255**, que é a numeração que representa cada cor em **RGB**
* Exemplo:

- azul: (0, 0, 255)

- branco: (255, 255, 255)

* Para representar no CSS, vc escreve: **rgb(0, 0, 255)**

<h2 style="background-color: rgb(0, 0, 255); color: rgb(255, 255, 255);">Exemplos de Cores</h2>

* **Representação pela função HSL**
* Todas as cores apresentam as características: Matiz (hue), Saturação e Luminosidade, que são abreviadas na sigla **hsl**

**Harmonia de Cores**

* Circulo cromático de cores
* Cores primárias: Vermelho, Amarelo e Azul
* Cores secundárias: laranja, verde, roxo
* Cores frias / Cores quentes
* Paleta de cores: 3-5 cores
* Cores complementares: cores que tem maior contraste: ex: roxo + amarelo
* Cores análogas: são as cores imediatamente ao lado, vizinhas, tem uma boa harmonia
* Vc pode criar por exemplo 3 cores análogas + 1 complementar
* Cores análogas relacionadas: 2 cores vizinhas e + 1 cor semivizinha pulando 1 casa
* Cores intercaladas: 3 cores semivizinhas que pulam 1 cor. Outro exemplo são as cores triádicas: quando vc pega 3 cores equidistantes pulando 3 cores entre elas
* Monocromia: uma paleta de cores muito semelhantes

**Paleta de Cores**

* Use o **adobe color** para escolher paletas de cores. É possível criar paletas com diferentes tipos de harmonia de cores
* Na opção de extrair tema é possível extrair as cores presentes em uma imagem, ou logo
* Em Explorar também da pra ver ideias de paletas
* Outro site legal também é o **palleton**, também pra visualizar paletas, e é possível também montar algumas coisas com a paleta selecionada, como exemplos de sites, animações
* Outro é o **coolors**, que te dá ideia de paletas aleatórias, e vc pode de repente escolher 1 cor dessas e então montar paletas baseadas nessa cor ou em 2 cores

**Capturando cor da tela**

* É possível tirar uma captura de tela e ir no gimp pra pegar a cor, mas não é mto indicado até pq pode mudar um pouco a tonalidade com a captura
* **CollorZilla** – use essa extensão do google chrome para capturar uma cor em um determinado site

**Degradê com CSS**

* **/\* \*/ - serve pra adicionar comentários em CSS**
* **\* - adiciona configurações globais do CSS**
* Para criar um degradê em CSS, basta adicionar o seguinte dentro das chaves do body:

background-image: linear-gradient(to right, white, blue);

background-attachment: fixed;

* Daí vc pode colocar **to right, to left, to top ou to bottom**, ou vc pode definir a angulação, por exemplo: **45deg**
* O background-attachment: fixed é pra fixar o degradê msmo q vc role pela página
* Outra coisa importante é colocar dentro das configurações globais height de 100%:

\* {

   height: 100%

  }

* É possível extrair o gradiente de uma imagem pelo adobe color, e copiar as cores e ir colocando no vscode
* Outra opção também é colocar como **radial-gradient**, e no lugar da direção colocar **circle**
* Vc também pode definir a porcentagem do quanto cada cor vai ocupar na tela, colocando **X%** após a cor, ex: **#ED4F4A 40%** (com espaço mesmo)

**Tipografia**

* Glifo: é o nome dado a uma letra, todas as letras são um glifo, também podem ser chamados de caracteres
* Fonte: corresponde ao conjunto de todos os glifos de uma família
* Algumas fontes tem diferentes famílias tipográficas, ex: light, normal, seminegrito, etc, mas a maioria tem pelo menos o normal e o negrito
* Anatomia do tipo:
* Altura-x: corresponde a altura de todas as letras minúsculas, que é igual a altura de x
* Altura das maiúsculas
* Ascendente: corresponde a altura extra de letras que tem vazamento pra cima, como por exemplo b, k, l
* Descendente: altura extra pra baixo: j, g, q
* Corpo: corresponde a altura total, juntando ascendentes, descendentes, maiusculas
* **Serifa**: é o pezinho, um pequeno tracinho no final da letra, ela cria uma linha imaginária ao longo das letras, que guia o olhar e facilita a leitura
* Componentes anatômicos geométricos:
* Haste: é um traço reto na vertical presente em muitas letras, como A, b, l, k
* Filete: é o traço reto na horizonta, presente por exemplo no A
* Arco: curvaturas, por exemplo na letra n
* Esporão: é um tracinho pra baixo, na base da letra, como se fosse fincar ela. Algumas fontes tem, outras não, tem na letra b, em algumas fontes
* Vértice: uma setinha, como por exemplo o topo da letra A
* Terminal: um elemento que não se prende em outro lugar, ex: a pontinha de cima do r
* Componentes anatômicos:
* Braço: por exemplo o tracinho de cima da letra k
* Perna: o tracinho de baixo do k
* Pé: a base da letra, ex: A, k. é onde existem as serifas, quando presentes
* Espinha: é a curva dupla, na letra s
* Barriga: letra b
* Olho: quando há uma área interna fechada numa letra, ex: q, b, e, g
* Orelha: ex: a pontinha superior direita na letra g, algumas fontes tem outras não
* Cauda: ex: g, q
* Categorias de fontes:
* Serifada
* Sans-serif – não serifada
* Monoespaçada: todas as letras tem a mesma largura
* Handwriting: fontes que simulam a caligrafia humana
* Display: fontes comemorativas, são chamativas
* Geralmente ao definir uma fonte colocamos mais de uma fonte pro caso de o navegador não ter uma das fontes especificadas, ex:

font-family: Arial, Helvetica, sans-serif;

* A arial seria a primeira a tentar ser usada, caso nn tenha ai tenta a helvética e assim por diante
* Chamamos isso de Safe Combination. Se quiser exemplos pode pesquisar por CSS web safe font combinations – no web3school
* Na parte de font-family em vez de colocar uma fonte específica vc também pode colocar um tipo de fonte, como sans-serif, serif, monospace, e então o próprio navegador que definirá qual a fonte padrão usada em cada tipo
* **Tamanho de fonte:**
* **Font-size**
* Medidas absolutas: cm, in, px (pixel)
* Medidas relativas: em (altura M maiusculo de uma fonte), ex (altura x de uma fonte), rem (é tipo em, mas relativo ao root, a fonte que tá configurada no body), vw (view width – tamanho de tela), vh, %
* Normalmente utilizamos o **em**, mas o px também é recomendado
* Geralmente o tamanho padrão é 16px. 16px normalmente corresponde a 1em
* **Peso da fonte:**
* **Font-weight**
* Pode ser **lighter, normal, bold, bolder**
* Ou de 100-900 (quanto maior mais grosso)
* Somente algumas fontes tem essas variações
* **Font-style:** pode ser normal ou itálico
* **Text-decoration:** underline (sublinhado)
* **Shorthand font**:
* Condensa as configurações padrões de fonte, na seguinte ordem:

**- font-style -> font-weight -> font-size -> font-family**

* Ex:
* font: italic bolder 3em "work Sans", sans-serif;
* Outro exemplo – vc pode omitir alguma das configurações se quiser:
* font: bold 2em "work Sans";
* Google fonts: vc pode pegar estilos que não estão presentes no vscode. É possível pesquisar por categorias. Pra colocar a fonte no código é só ir em **embed** e depois em **import**, dai basta colocar o código no style.
* Também é possível adicionar outras fontes fora do google fontes:
* Um site legal pra pegar fontes é: **dafont.com**. Daí vc baixa a fonte que quer e coloca na pasta junto com o arquivo html. Na vdd o ideal é criar uma subpasta só pras fontes
* Daí vc coloca a regra: **@font-face** dentro do style
* No **font-family** vc vai definir qual vai ser o nome da fonte, pode escolher um nome de sua preferência
* No **“src: url()”** vc coloca entre aspas o nome exato do arquivo, e depois vc pode colocar o **format()**, que é o formato da fonte, e depois vc pode colocar ainda outro url com outro format da mesma fonte, pro caso de o primeiro nn servir no navegador:

- tipos de format():

- opentype(otf)

- truetype(ttf)

- embedded-opentype

- truetype-aat (Apple Advanced Typography)

- svg

Ex:

src: url("Brownist.otf") format("opentype");

* Também pode colocar o **font-weight** e **font-style**, e acho q se quiser pode colocar mais atributos também
* **Fonts Ninja**: Extensão pra capturar uma fonte em um determinado site
* Pra detectar fontes dentro de imagens:
* Whatfontis.com
* Fontsquirrel.com
* Myfonts.com
* **Alinhamento**:
* Por padrão o alinhamento do site feito no vscode será a esquerda
* **Text-align:** é a tag de alinhamento, pode ser **left, right, center ou justify**
* **Text-indent:** faz a endentação dos parágrafos, vc determina por quantidade de pixels, ex: text-indent: 30px.
* **Line-height:** define o espaçamento entre linhas, 1em, 1.5em, 2em, etc

**Seletores Personalizados**

* Os seletores personalizados suprem a necessidade de fazer formatações diferentes para itens semelhantes, como por exemplo paráfragos, títulos
* Para isso vc precisa criar uma identificação para o elemento que vc quer personalizar diferente dos demais, ex:
* <h1 id="principal">Criando sites com HTML e CSS</h1>
* **Id=””** – coloque esse atributo dentro da tag e escolha um nome para idetificá-lo
* Daí no CSS vc pode personalizar adicionando **# +** o nome do identificador, ex:

h1#principal {

}

* Identificadores:
* O que for **id** no HTML, no CSS vai ser #
* O que for **class** no HTML, no CSS vai ser **.**
* Entretanto, dentro do documento HTML vc só pode ter 1 elemento com o msmo id, segundo regra do W3C. Então por exemplo aquele id “principal” não poderia ser usado em outro título
* **Class**: com o atributo class vc pode identificar vários elementos com uma mesma classe, ex:

<h1 class="principal">Aprendendo CSS</h1>

<h1 class="principal">Criando sites com HTML e CSS</h1>

* Mas no CSS lembre de usar o ponto **.** para identificar
* Outra coisa importante é não dar nome a classe de acordo com a forma, a cor, dê nomes com base na função do elemento. Até pq se precisar alterar a cor vc vai ter q alterar os nomes das classes
* Um detalhe é que no CSS vc nn precisa colocar **h2.principal{}**, vc pode só colocar **.principal{}**
* E daí vc pode usar a classe pra diferentes elementos, parágrafos, trechos do texto, títulos etc
* **<span></span>**
* Serve pra vc selecionar um trecho específico do texto, daí vc pode adicionar uma classe a ele
* Também é possível adicionar mais de uma classe a um mesmo elemento, ex:

<h2 class="intermediario destaque">

Obs – não é necessário separar por vírgula; atribui as classes intermediário, e destaque

Já se for colocar um ID + CLASS:

<h2 id="principal "class="intermediario">

Mas o ID sobrepõe as configurações de classe

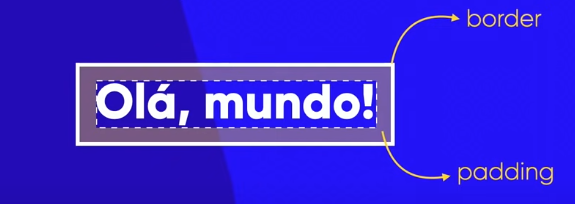
**Pseudo-classes**

* As pseudoclasses são referenciadas no CSS por 2 pontos “**:**”
* Elas devem estar relacionadas a um elemento ou a uma classe e servem pra relacionar o elemento ou classe a um determinado estado, se ele ta ativo, marcado, vazio, habilitado, etc
* Colocando por exemplo: **div:** no CSS vão aparecer algumas opções de pseudoclasse
* Por exemplo:
* **Hover** – serve pra alterar o estado quando o usuário passar o mouse por cima
* div:hover{
* background-color: yellow;
* }
* Irá fazer com que os objetos div fiquem com fundo amarelo quando passar o mouse por cima
* Outro exemplo é fazer aparecer texto quando passar por cima, mas verifique o arquivo Hover no Ex020
* **Transition-duration:** 0.5s; - é o tempo que levará pra fazer a transição no hover, é legal pra dar uma suavizada no efeito
* **>** - referente a children, ex: div > p{}
* **:visited**
* Pseudoclasse referente a links já visitados:
* a:visited{
* color: darkblue
* }
* **Text-decoration**:
* **None** – retira sublihados e outras decorações
* **Underline** – aciciona sublinhado
* **:active**:
* Muda a característica quando estiver ativo
* Ex: links – vc pode mudar a cor quando o link for clicado
* **Pseudoelemento:**
* Serve pra alterar o conteúdo de um elemento, classe etc, diferente da pseudoclasse que altera o estado
* O pseudoelemento é identificado com 2x 2 pontos: “**::**” ex:
* a::after{
* content: "[link]";
* }
* Irá adicionar “[link]” no final
* Também pode usar **::before** pra adicionar texto no início
* Um pseudoelemento também pode ser usado pra classes, ex:

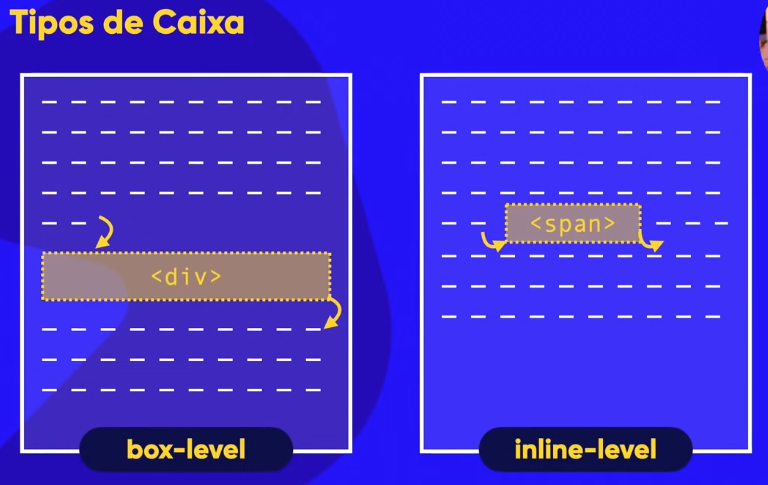
.especial::after{}

**Modelo de Caixas (Box Model)**

* É aplicado ao CSS e HTML e ajuda a organizar melhor o site
* Caixas é uma analogia para ilustrar melhor a forma como as coisas aparecem no site. Por exemplo, o título h1 é uma caixa, o conteúdo dentro dele que aparecerá na tela pode ser demarcado com **altura (height) e largura (width)**.
* Também sendo possível traçar uma linha imaginária ao redor da caixa, percorrendo as alturas e larguras, essa linha chamamos de **border**.
* Também é possível criar um espaço entre o conteúdo e a borda, deixando o conteúdo mais afastado da borda, esse espaço chamamos de **padding** (preenchimento/alcochoamento)



* **Margin**: seria o espaço externo, além da borda, que permite que a borda de diferentes “caixas” não se toquem
* **Outline**: linha que fica ao redor da borda, no espaço externo, é como um contorno da borda
* **Tipos de caixa:**
* **Box-level:** conteúdos desse tipo ocupam a largura inteira da tela, sendo necessário quebras de linhas pra coloca-lo junto a outros elementos. Um exemplo é a tag <**div>**.
* **Inline-level:** elementos desse tipo podem ser alinhados na msma linha que outros elementos. Um exemplo é o **<span>**.



* Por padrão esses são os elementos que são considerados box-level e inline-level, porém no CSS vc pode alterar isso usando **display**, ex:

**H1{**

**Display: inline;**

**}**

E automaticamente o h1 se torna em um elemento inline. Da msma forma da pra transformar um elemento inline em block usando **display: block;**

* Também é possível utilizar um elemento com características mistas, **display: inline-block;**, mas ele nn se aprofundou mto
* Ao abrir uma pagina no google, clicar com o botão direito do mouse e clicar em inspecionar vai aparecer as informações da página, onde é possível ver o modelo de caixas e as configurações CSS de cada elemento. Essa ferramenta de dev tools ajuda mto a realizar pequenos ajustes em tempo real pra vc ver quais configurações ficam melhor pra página
* **User agent** – é referente ao navegador. User agente slylesheet são as configurações CSS padrões do navegador que são colocadas em elementos sem CSS definido
* **Display: block** – é referente a um elemento que é do tipo box level
* **Diplay: in-line block** – elementos do tipo in-line level
* Todas aquelas características mencionadas podem ser alteradas no CSS:
* Height
* Width
* Border
* Padding
* Margin
* Outline
* Para centralizar um box-level:
* **Margin: auto**
* Todas as características apresentam várias opções de configurações, ex: top, right, left, bottom, style, width, etc
* Para simplificar é preciso usar **shorthands**:
* **Border: width style color;**

Ex: border: 10px solid blue;

**Obs- lembre de colocar ponto e vírgula “;” no final das linhas**

O **outline** também é configurado nessa ordem

* O shorthand pras características que tem “top, bottom, left, right” é seguir no sentindo horário, ex:

**Padding: top, right, bottom, left;**

* Outra coisa é que, caso todas essas 4 opções forem iguais, tipo, vc for fazer um padding com a msma largura em todas as 4 direções, basta colocar:

**Padding: 10px;**

* Outra opção também é configurar os pares top-bottom e right-left com a msma espessura:

**Padding: 10px 20px;**

* Em relação a margem, caso vc queira o box centralizado, mas queira alterar as margens de cima de de baixo, vc pode fazer assim:

**Margin: 20px auto 30px auto**

**Tags de Agrupamento**

* **<div></div> e <span></span>**
* Antigamente se usava div pra separar todo o conteúdo do site, mas atualmente são utilizadas tags semânticas, que além de servirem como div elas dão sentido, semântica ao conteúdo que tem dentro delas
* **<header></header>**
* Cabeçalho
* **<main></main>**
* Conteúdo principal
* **<footer></footer>**
* Rodapé
* **<nav></nav>**
* É referente a “navegação”, onde vc coloca links
* **<section></section>**
* **<article></article>**
* **<aside></aside>**
* As tags podem também serem colocadas uma dentro da outra
* E daí vc pode personalizar o conteúdo de tags específicas no CSS
* **Sombras:**
* **Box-shadow**: 1px 1px 1px black;
* Os 4 itens em sequencia são: deslocamento horizontal, vertical, espalhamento e sombra
* Mas por padrão vc pode sempre colocar 3x 1px e black como sombra
* É possível configurar uma sombra pelo dev tools de forma mais prática e visual, após ter colocado a sombra, no dev tools vc vai ver um quadradinho ao lado e box-shadow, dai vc clica e abre o menu de configuração de sombra
* Pra fazer a sombra nn ficar tão escura vc pode mudar a transparência do preto, é recomendável que teja em 50% pra baixo
* É possível criar sombras internas também, colocando “inset” logo após o box-shadow, antes de definir os px

**Caixas com vértices arredondados**

* **Border-radius:** 5px;

vc pode configurar dessa forma, configurando os 4 vértices ao msmo tempo

* **Border-radius:** 10px 20px 30px 40px;

também é possível configurar cada borda individualmente

* **Border-radius:** 10px 20px;

Dessa forma vc configura os pares de cantos top-left+bottom-right e top-right+bottom-left

* Também da pra configurar por % em vez de px, colocadno 50% vc cria uma bola, caso o elemento a ser arredondado for um quadrado

**Bordas Decoradas**

* border: 20px solid black;
* border-image-source: url("Borda.png");
* border-image-slice: 27;
* border-image-repeat: repeat (ou pode ser stretch);
* Da também pra simplificar, com shorthand:
* border-image: url("Borda.png") 27 repeat;
* Obs - é preciso ter uma borda no início pra que vc possa configurar
* Mock Flow: site pra criar a estrutura do site, é tipo o figma

**Variáveis em CSS**

* Primeiro é necessário definir uma área dentro do arquivo CSS onde você pode declarar variáveis:

:root{

    --cor0: #c5ebd6

--cor1: #83e1ad

--cor2: #3ddc84

}

* Definindo as cores como variáveis por exemplo, ao colocar um “background-color” no css, lá no final das opções que aparecem, vai aparecer essas cores que foram definidas como variáveis
* Ao colocar no código, a variável é implementada da seguinte forma:

**Color: var(--color1);**

* Isso é muito importante pro caso de você precisar mudar uma paleta de cores de repente, pra não precisar ir mudando em cada partezinha do CSS, dessa forma você só precisa mudar no **root**
* Também é possível fazer isso pra fontes e outras coisas
* **Configuração global:**
* É possível criar uma configuração global no CSS para definir uma característica específica pra todos os elementos:

\*{

    margin: 0px;

    padding: 0px;

}

* **Window Resizer**: Extensão no google que te permite colocar o site em tamanhos variados para testar a responsividade
* **Responsividade:**
* É possível ajustar uma largura mínima e máxima pra o corpo por exemplo, definindo **min-width e max-width**
* Outra coisa também é colocar uma imagem com **width: 100%**, pois daí a imagem vai se ajustando a cada variação do tamanho
* **Ajustando listas:**
* **List-style-position**: inside - faz com que as bolinhas fiquem dentro do box
* **Columns:** 2 - cria 2 colunas, pode ser mais se quiser
* **list-style-type:** '\2714\00A0\00A0'; - muda o símbolo dos itens da lista, tem que pegar o código na emojipedia. O código nesse caso é só o 2714, poderia colocar só “/2714”, mas os 00A0 servem pra dar um espaçamento
* **Ajustando Vídeos:**

.container{

   background-color: green;

    margin: 0px -15px 30px -15px;

    padding: 20px;

    padding-bottom: 50%;

    position: relative;

}

.container>iframe{

    position: absolute;

    top: 5%;

    left: 5%;

    width: 90%;

    height: 90%;

}

**Módulo 3 - HTML5 e CSS3**

**REVISÃO**

* **HTML**
* Sequencia de texto aleatório, com quantidade de palavras especificas
* Como envelopar uma parte do código com uma tag?
* Tags a se saber:
* Títulos, parágrafos,
* Quebra de linha, linha horizontal
* <>, caracteres especiais, emojis
* Comentário em html
* Enfases, abreviação, texto inserido e deletado (sublinhado/riscado)
* tag de endereço, de citação (curta e longa),
* Como coloco texto sobrescrito e subscrito? ² e 2
* Tag pra fonte monoespaçada e texto pré-formatado
* tag pra inverter texto
* Listas ordenadas e não ordenadas e seus simbolos, item de listas, e como coloco pra iniciar a partir de uma dada numeração/letra?
* Listas de definição e seus itens
* Favicon
* Imagem: como inserir, colocar descrição, como achar imagens em pastas, como criar imagens dinâmicas? E como fazer com que 1 mesma imagem ajuste de tamanho de acordo com a tela
* Como insiro links? Como diferencio Links da própria pagina e links externos, e como faço pra abrir em outra aba? Como especifico a linguagem do link? Como coloco links pra download?
* Como insiro áudios? Como inserir botão controlador pra iniciar ou parar musica? Como fazer pra musica se tocar sozinha? Como faço pra deixar a musica tocando em looping? Como mudar se o áudio vai ser carregado junto com o site ou só quando clicar nele?
* Como insiro vídeos? Como definir uma imagem de capa? Controles? Como colocar vídeos do youtube?
* **CSS**
* Explique as formas de fazer CSS
* Regra utf-8
* Como adicionar comentários em CSS
* Como adicionar configurações globais
* Explique as 4 formas de representar cores em CSS
* Como crio um degradê de cores em CSS?
* Como inserir uma fonte; o que é safe combination; como modificar: tamanho, peso, itálico, sublinhado; qual a ordem do shorthand pra fontes? Como inserir fontes do google fontes? Como inserir fontes baixadas?
* Como alinhar texto? Com fazer a indentação dos parágrafos? E mudar as alturas entre as linhas?
* Fale sobre seletores personalizados e os seus tipos. Como colocar mais de 1 classe em um mesmo elemento? E colocar 1 id e 1 class em um msmo elemento?
* Tag neutra para selecionar um trecho de texto e poder colocar uma classe ou id nele
* O que são pseudoclasses? E como usar? Cite 3, referente a passar mouse por cima, e links
* Explique pseudoelementos e como usá-los
* Explique o modelo de caixas, tipos de caixa, e as 6 características de uma caixa; como mudar o tipo de caixa no css? Como centralizar uma caixa? Fale sobre os shorthands pras características das caixas. Como fazer sombra na caixa. Como fazer caixas com vértices arredondados
* Tags semânticas: cabeçalho, rodapé, pra links, seção, artigo, texto aparte
* Como colocar variáveis em CSS?