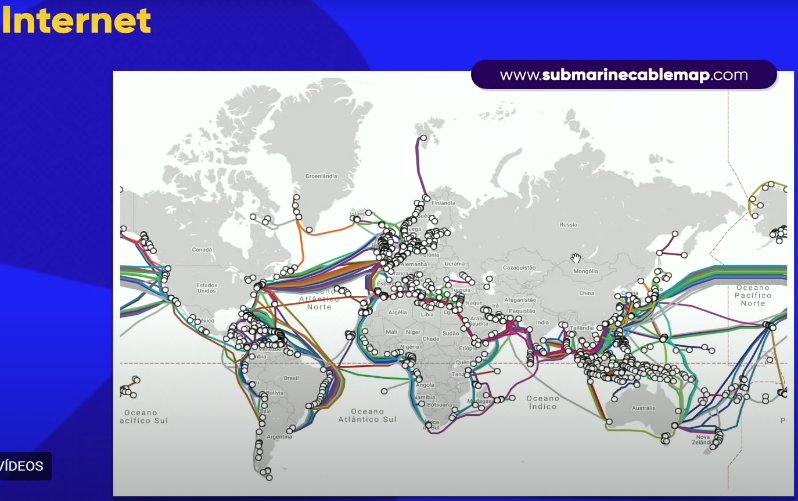
**CURSO EM VÍDEO – HTML5 E CSS3:**

**Leituras recomendadas:**

* Mark pilgrim O’realy (html5)
* Eric Meyer O’realy (css o guia definitivo)
* Eric Meyer O’realy (css o grid layout)
* Estelle Wey O’reale (flexbox in css)
* HTML5 e CSS3 Elizabeth castro alta books
* Jon Duckett Alta books
* Eva Heller GG (psicologia das cores)
* Ellen Lupton gg (Pensar com tipos)

**HTML5:**

**HISTÓRIA DA INTERNET: O QUE É E COMO SURGIU A INTERNET:** Nasceu na guerra fria. Depois do lançamento do satélite “Sputnic” pela Rússia os EUA, criou a ARPANET uma “pequena rede de internet” para manter backup de dados entre as duas costas litorais dos EUA. Os únicos 4 computadores existentes na época eram de fabricantes diferentes e usavam linguagens diferentes.

 Para resolver isso criaram o **protocolo NCP,** que era uma linguagem “padronizada” que permitia a comunicação entre as máquinas. Era tão ruim que precisava interromper toda a rede (na época era a cabo) para que apenas 2 máquinas transmitissem dados. **NCP > TCP/IP > AS DE HOJE.**

**ARPANET** se difundiu em 3: 1)MILNET (militar) 2) NSFNET (National scientst.. universidades) 3) COMERCIAL

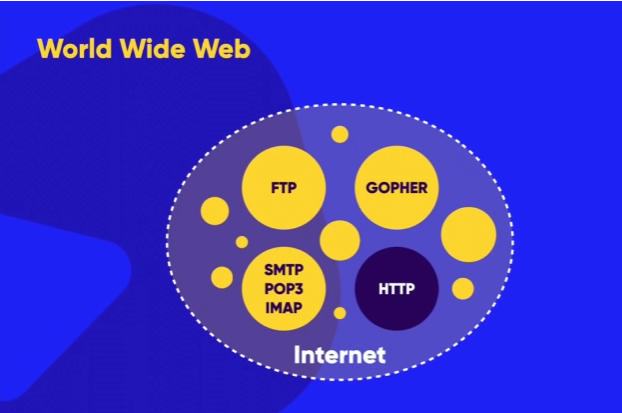
**INTERNET:** Junção de todas as redes de internt.

As conexos inter continentas são feitas por cabos subterrâneos

**CLIENTE X SERVIDOR:** Minha máquina (Cliente) faz uma requisição de um vídeo para a máquina do fornecedor (Servidor Youtube) > o TCP/IP do youtube que possui esse arquivo de vídeo, quebra em vários fragmentos menores para não sobrecarregar a rede mundial e me manda tudo desorganizado, se sumir algo ele manda de novo, chegando em minha máquina o TCP/IP faz a reconstrução do vídeo, imagem ou arquivo.

Em 1993 (Tim Berners-Lee) criou o http:// e o html.

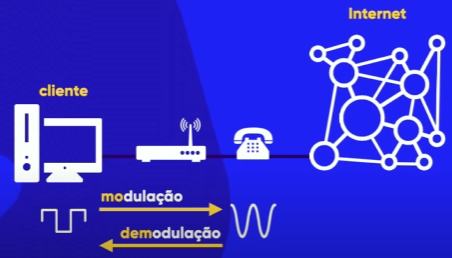
**World Wide Web (Rede de Alcance Mundial):** É sistema de informações interligados que permite circulação de dados na internet. O **HTTP** é uma **sub rede** (servidor) dentro da grande rede de Alcance Mundial internet responsável pela rede do HTTP:// -> Protocolo de Transferência de Hyper Texto (HTML)



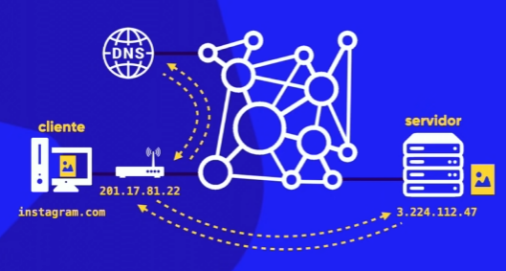
**Como a internet funciona:** Computador funciona com sinais de zero (0) e um (1), na verdade, são ondas de sinais elétricos, topo (1) e vale (zero), ou dígitos binários, binary digit (bit). O mínimo de bits necessários para formar uma informação são “8”, por exemplo, = 01000001 que representa a letra “A”. Quando apertamos a tecla a, vai para a memória do computador a sequência de bits 01000001.

Cada letra tem uma representação em bytes (tabela UTF-8 – código multibyte que suporta acentuação).

**8 bits = 1 byte // 1024 bytes = 1KB // 1024 KB = 1MB // 1024 MB = GB // 1024 GB = 1TB // 1024 TB = 1 PB // 1024 PT = 1EB // 1024 EB = ZB**

**1024 EB = 1 YB. MB = megabYtes <> Mb = megabits**

O cliente **(computador)** entende apenas sinais de ondas “**quadradas**” enquanto a rede telefônica/internet entende apenas ondas **senoidais,** para isso é instalado um **modem** de alguma operadora para fazer o serviço de **modulação** (transformação de onda quadrada em onda senoidal) e o servido de **demodulação** transformação de onda senoidal para onda quadrada.



Quando fazemos uma requisição na internet ele busca pelo IP do servidor (<https://www.iplocation.net/ip-lookup>) . Esse número é cadastrado por um **DNS – DOMAIN NAME SYSTEM** (Sitstema de nome de domínio) ex: instagram.com, quando digito o site do instagram ele faz consulta de IP no DNS e o DNS devolve, permitindo ir para o servidor e buscar dados (fotos, vídeos).

**Rotas:** são caminhos que o roteamento pega para transferir dados.

**Domínio e Hospedagem: Domínio** = Nome que identifica seu site (é único e pago anualmente com **TLD – Top Level Domain** ou **GTLD** – Generical Top Level Domain: .io e **CCTLD –** Country Code Top Level DOmain .br .uk .us) / **Hospedagem** = Local onde o site está armazenado (outra máquina e paga mensalmente escolhido por espaço, memória, recursos “tipo de suporte”).

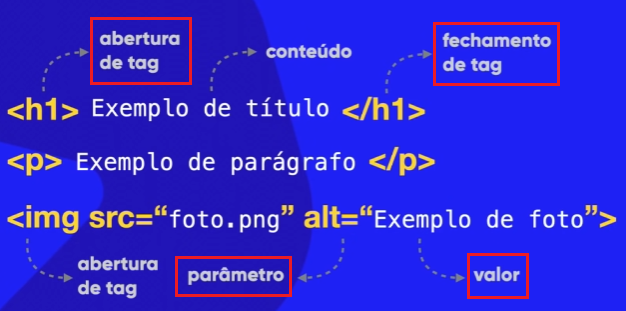
**URL (Uniform Resource Locator):** Ele serve para indicar um único lugar.[www.github.com/matheusmassuda](http://www.github.com/matheusmassuda)

**Como funcionam a HTML e as CSS:** Linguagens de desenvolvimento. **HyperText Markup Language** (linguagem de marcação de hiper texto). Hypertexto são ancoras sensíveis, markup são marcações da linguagem. **Cascading Style Sheets** (Folhas de estilo em cascata), em cascata significa que uma folha vai alterando o estilo padrão da outra.

\***HTML:** Focado em **conteúdo:** textos, imagens, vídeos, áudios, tabelas, listas, artigos...

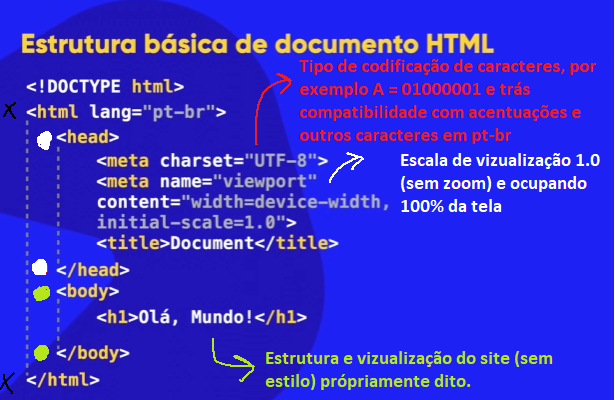
\***CSS:** Focado em **design/estilo:** cores, tamanhos, posicionamento, estilos, sombras, tamanho, sobreposições...

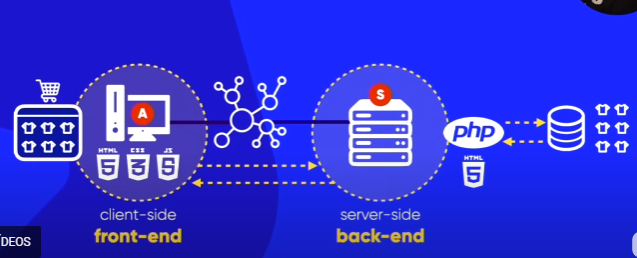
\***JS:** Focado em **interatividade:** menus, animações, popups, validações**,** funções...

**Tags:** São estruturas que permitem marcar textos no HTML. Tag de abertura “<p>” e tag de fechamento “</p>”. A maioria tem tag de abertura e de fechamento, mas outras tem apenas a de abertura como <img>.

**Seletores:** selecionam por meio da abertura de “{“ e fechamento “}” quais os valores que eu quero dar para determinadas propriedades do css.

**Estrutura básica de HTML:** Preciso eu dizer para o meu navegador: “Vou desenvolver meu site em HTML5”, ou seja, o tipo do meu documento é HTML.

**Obs:** O navegador é responsável de ler as tecnologias cliente-side (HTML, CSS, JS).

**Front-end, Back-end, Full stack:** Uma vaga para **front-end** significa vaga para desenvolvedor **client-side,** ou seja, um desenvolvedor que desenvolve para o cliente usando tecnologias como HTML, CSS, JS. Ele deve entender um pouco sobre dsign, psicologia das cores, e experiência do usuário. Por outro lado, uma vaga para desenvolvedor do lado do servidor: **server-side** é uma vaga de **back-end** que desenvolve usando tecnologias preocupadas com o funcionamento do servidor como (php, JS – Node-JS, c#, Python, Ruby, Jaba).

**\*Funcionamento:** Cliente entra no site (requisição p/ servidor) > php busca num grande banco de dados do servidor > php gera um página HTML com informações > Manda HTML, CSS, JS p/ cliente > Cliente vê na tela.

**Full stack:** Atua nas duas áreas.

**HTML5:** O HTML é lido por um navegador de acordo com a **funcionalidade** de cada tag, ou seja, cada tag possui um modo de mostrar (display) seu conteúdo no navegador. Por exemplo algumas tags pulam linha automaticamente, outras não. Algumas tags apresentam o texto um pouco maior ou menor, outras colocam funcionalidade de “negrito” <b>, outras marcam que determinado conteúdo é um artigo ou uma seção.

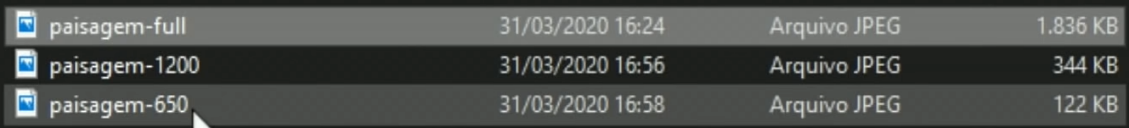
**Nunca** use <br><br><br><br><br> em desenvolvimento profissional. Deixe o layout e espalhamentos por conta do css.

**Formatos de imagem para Web:** Qual formato de imagem devo usar para web? (.jpg; .png; .gif; .tif;). A resposta é: depende. O melhor formato para **alta compactação** (redução do tamanho do arquivo) é .jpg; (cirado pelo W3C), mas que não permite transparência. O .gif; é um formato que permite transparência e animação. O formato .png; permite transparência, mas não tem a mesma compactação alta do .jpg. **Obs:** na maioria das vezes para imagens de alta qualidade e tamanho é necessário usar o .jpg; pois se uma imagem tiver 20MB não há 3g que consiga carregar.

**Lógica ao usar imagens:**

* **Background:** .jpg 🡪 uma imagem grandona, porém perde qualidade com zoom.
* **Imagens flutuantes:** .png 🡪 Qualidade muito melhor e sem fundo, porém grande.

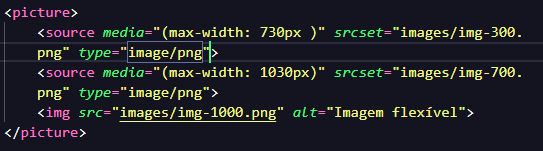
**Tamanho e Resolução de imagens:** Para que o arquivo não fique muito grande (pesado) o recomendado de tamanho para uma imagem é de 1200 até 1500 px de **largura** e 50 até 72 de **resolução.** Um programa gratuito que pode ser usado para fazer isso é o **Gimp.**

\***Pq é tão importante reduzir o tamanho da imagem:** O css também pode reduzir o tamanho da imagem, no entanto, o peso da imagem continua o do original (do arquivo buscado pelo HTML), então esse redimensionamento de imagem deve ser feito antes de subir a imagem para o site com HTML. Arquivos pesados deixam os sites lentos e sites lentos não aparecem no Google com a mesma facilidade (é como se o algoritmo do Google punisse a entrega de um site pesado).

**Carga de imagens com HTML5:** Para carregar imagens primeiro já escolha com antecedência o tamanho e dimensionamento de imagens, se a imagem original full for 1200px por 1200px e com sobra de pixels transparentes, use ferramentas no Gimp ou Photoshop para cortar espaços em vazio e redimensionar imagem, por exemplo, para 185px para 185px. **Isso economizará muita largura de banda.**

**Pastas, subpastas e URLS externas:** Podemos pegar a fonte de uma imagem de diversas maneiras. Na própria pasta junto com index.html (não recomendado), em uma subpasta img (recomendado), e de uma URL externa com o “carregar endereço da imagem” (não recomendado pois qualquer alteração no servidor fonte dessa imagem vai impactar o carregamento dela em nosso site).

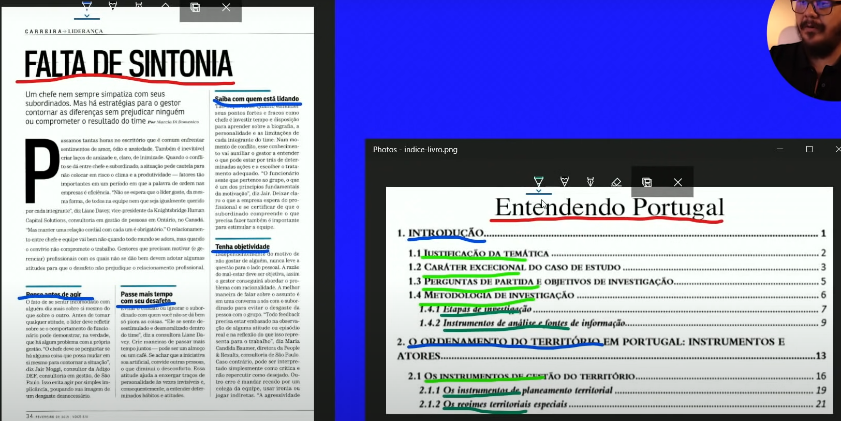
**Tamanhos e resoluções de imagens para celular:** 300 x 300px // 700 x 700px // 1000 x 1000px com resolução variando de 50 a 80.

<Picture> + <source> permite criar imagens dinâmicas que se lê: <img> carregue a imagem 1000px, <source> mas para largura máxima de até 1030px carregue a imagem de 700px. Com o <source> mais de cima ainda: para largura máxima de até 730px carregue a imagem de 300px.

Essa ferramenta é muito importante para **bounce rate “taxa de rejeição”** do usuário.

**\*FavIcon:** É a imagem de ícone que fica na aba do navegador junto com o nome do site (de preferência usar extensão .ico). Obs: para criar, transformar um .png em .ico pode-se usar o favicon.io (site que transforma).

**Títulos e suas hierarquias:** h1; h2; h3; h4; h5; h6; São hierarquias de importância dentro dos títulos de um site. H1 só devemos ter 1 por página. **Shortcuts** para escrever a imagem a baixo no VSCode com h1 até h4 🡪 h1+(h2(h3\*4+h4\*2))\*2 🡪 Crie um h1 e por duas vezes crie uma estrutura composta por 1 h2 e embaixo 4 h3 e 2 h4

**De h1 até h6:** É HTML5 semântico e serve para dar hierarquia de textos. É possível alterar tamanho de texto depois na CSS3.

**Semântica em HTML5:** Semântica é o sentido das palavras e interpretação de sentenças. O significado das palavras, em oposição a sua forma. HTML5 não se foca na forma (como o texto ou o resultado aparecerá no viewport), HTML5 se foca na semântica ou significado. **HTML5 > SEMÂNTICA // CSS3 > ESTILO (FORMA).**

**Negrito / Destaque e Itálico:** Para destacar um texto dando significado a ele usamos **<Strong>** (leitores de tela lêem com mais ênfase). Podemos também destacar o texto usando **<b>,** no entanto, ela tem apenas forma (aparece mais escura) e não possui semântica (significado). Para deixar um texto apenas na forma de itálico sem significado semântico podemos usar **<i>,** no entanto, se quisermos deixar um texto em itálico com significado semântico devemos usar **<em>.**

**\*Dica:** segundo a W3C - **World Wide Web Consortium** que é a principal organização de padronização da World Wide Web devemos usar a **HTML** para **semântica** (significado)e a **CSS** para **forma** (estilização). O grande objetivo da W3C é dar acessibilidade e compatibilidade entre diferentes plataformas e dispositivos. Portanto é inteligente usar **<Strong>** e **<em>** no lugar de <b> e <i>.

**Textos pequenos e grandes com <small> e <big>:** Teoria é de que textos pequenos estejam relacionado com textos pequenos de contratos por isso ainda não é obsoleta, a tag big está obsoleta.

**~~Deletado~~ e Inserido:** Podemos dar o significado de texto deletado com **<del>** e o significado de texto inserido com **<ins>.** A tag de forma para fazer o sublinhado é <u>, mas provavelmente ficará obsoleta.

**Outras formatações:** Fontes mono-space são fontes com mono espaçamento, a largura de um caractere é igual entre todos os caracteres de uma palavra.

Esse comando na html <code> permite escrever textos como se eles fossem códigos, com fonte mono-space.

**<code>:** Permite colocar a formatação da fonte em **mono-space**, porém tudo na mesma linha como um parágrafo.

**<pre>:** Pré formated text, permite levar a formatação usada **(com espaços – tab, tabulações e quebra de linhas)**, literalmente igual foi escrito na HTML.

**<q>:** Quotation (citação), é usada para citar (literalmente) termos ditos ou escritos por outras pessoas.

**<blockquote>:** É uma citação mais longa com deslocamento para lateral esquerda que inclusive aceita o parâmetro cite =”link” onde pode ser colocado o link com autor e/ou site de referência.

**<abbr>:** Abreviations. Permite colocar explicações sobre o termo que está abreviado com o “hoover” do mouse. Além de ser **semântica** essa tag deixa a **experiência** do usuário mais rica.

**Listas no HTML:**

**<ol>: Listas ordenadas –** Para criar uma lista a onde a ordem importa e pode-se acrescentar novos itens nela sem perder a numeração da ordem podemos criar um <ol> “Ordered List”. Na ordered list podemos passar parâmetros de contagem com type=”1;A;a;I;i;” e quando ela deve começar a contar start=”1”.

**<ul>: Lista NÃO ordenada –** Para criarmos uma lista não ordenada, onde a ordem não importa, podemos usar <uL> “unordered list”. Os parâmetros da lista não ordenada são apenas três type=”disc; circle; square”

**LISTAS E MENUS:** É possível colocar uma lista dentro da outra usando <ul> <ol> e <li’s> para criar menus com várias opções dentro de cada item.

**Listas de Definição com <dl><dt><dd>:** Podemos criar um termo com sua definição logo a baixo e para mecanismos de busca e SEO é muito importante usar esse tipo de lista. O Google também usa definições de listas de definição na hora de encontrar e entregar algum conteúdo.

**LINKS:** Hiperlinks são âncoras que levam para outros sites. O Google é o maior buscador do mundo, em segundo lugar o maior buscador do mundo é o Youtube (que também é do Google). O motor de buscas do Google é baseado em links. Vídeo de um engenheiro do Google 2010 falando sobre o que seria o SEO: <https://www.youtube.com/watch?v=BNHR6IQJGZs>

**Parâmetros considerados obrigatórios:** para <a> são: href=” “ target=”\_blank” rel=”external”, target=”\_self” eles indicam, respectivamente, URL, abertura em nova guia e relação a um site externo, abertura na mesma guia.

**Links internos:** São links que interligam nossas próprias páginas. Os parâmetros de links internos (dentro das próprias pastas de documentos) são: herf=” “ rel=”next/prev”. **Next** para próxima página e **prev** (previous) para a anterior. Quando colocarmos os parâmetros rel com valor “next” ou “prev” estamos forçando o Google a entender qual é a próxima página e a anterior de nosso site.

**Links para downloads:** Alguns navegadores podem reconhecer um arquivo e fazer download direto, no Chrome e alguns outros navegadores é preciso colocar dois parâmetros para forçar o download acontecer diretamente que são: download=”nome-do-arquivo.extensao” type=”application/pdf”. Com esses parâmetros o navegador já solicita o download do arquivo.

**Parâmetro nofollow:** Podemos usar o parâmetro rel com valor nofollow para fazer o Google não indexar (por não conhecermos 100% a fonte ou quando é um site patrocinado que está pagando para estar ali com link).

**Áudios:** Os formatos de áudio compatíveis com os navegadores são, geralmente: .mp3; .wav; .ogg;

**<áudio>** permite parâmetros como **src**=”arquivo.mp3” **controls** (ativa mixer de áudio) **autoplay** (liga o som quando página é carregada) **loop** (roda o áudio novamente quando ele acabar), **preload** (já começa a mostrar informações no mixer).

\*Por questão de carregamento do site e SEO é prioridade carregar arquivos leves. Para áudios siga a hierarquia **MP3 (+ leve) > OGG (+/- leve) > WAV (pesado).** Você pode usar MP3 para reprodução e WAV para download caso o usuário queira.

**Vídeos:** Em HTML5 os tipos de arquivos mais comuns para vídeo são: .mp4 e .m4v; .webm; .ogv; Um software muito bom para **conversão de vídeos** com diferentes formatos (web, gmail...) e com diferentes quantidades de quadros é o **HandBraker** (conversor de formato de vídeos).

Conversor para diferente tipo de arquivo com HandBraker.

**\*COMPATIBILIDADE:** Quando trabalhamos com mídias (VÍDEOS, IMAGENS, ÁUDIOS) é recomendado termos em nossa máquina ou servidor todos os tipos de arquivos possíveis e pedir, por meio de HTML, que o navegador tente rodar as versões dele. Isso por que não são todos os navegadores que têm compatibilidade com todos os tipos de arquivos, então podemos procurar a compatibilidade de cada browser online ou como uma forma de aumentar a compatibilidade entre eles, oferecer vários tipos de extensões. No caso dos **vídeos**, os tipos de arquivos mais comuns de serem lidos são: **.mp4; .m4v; .ogv; .webm;**

\***CONSUMO DE BANDA:** Quando hospedamos mídias (vídeos, imagens e áudios) diretamente em nosso servidor, para cada pessoa que acessa e faz requisição ou baixa nossa mídia, nosso consumo de banda do servidor aumenta e esse consumo é limitado de acordo com a quantidade que pagamos pelo serviço de hospedagem. Existe um migué que podemos fazer que é: **hospedar vídeos a partir de serviços de hospedagem de vídeo (twitch, youtube, facebook, instagram, vímeo...) 🡪 YOUTUBE > COMPARTILHAR > INCORPORAR > CÓDIGO HTML5**

**CSS3:** Cascading Style Sheets (Folhas de estilo em cascata)**.** Se por um lado o HTML5 dá semântica (sentido) e estrutura à página web, por outro, as CSS3 dão **forma (sentido, dsign, perfumaria e alteram o visual).** Existem 3 maneiras de aplicar css em uma página web:

1. **Inline:** Passando o parâmetro dentro da própria tag, por exemplo: style=”background-color: black;” a grande desvantagem disso é que o intuído do css3 existir é fazer um efeito em cascata, e o formato **inline** faz o desenvolvedor ter que colocar esse parâmetro em cada tag que ele desejar esse estilo. Além de ser chato e trabalhoso isso **polui** o código HTML (que deve ser o mais sintético possível), além disso ainda dificulta manutenção do código. **[MENOS VERSÁTIL – MAIS HIERÁRQUICO].**
2. **Local / Interno:** Ele é feito dentro de <head> e pode ser logo após <title>. Lá abriremos uma <style> e dentro dela vamos escrever **seletores** onde faremos **declarações = parâmetro + valor,** por exemplo: <style> body { background-color: blue;} </style>. A vantagem de usar dessa maneira é que ela já funciona em cascata, ou seja, se tivermos uma declaração para <p> todos os <p> serão afetados. **Desvantagens:** código muito extenso, dificulta chegar e ver o HTML5, além de que se tiver muitas páginas que precisam ser alteradas, esse trecho precisará ser copiado e colado no <head> de todas as páginas. **[MÉDIA VERSATILIDADE – MÉDIA HIERARQUIA].**
3. **Style.css:** É um arquivo exclusivo de estilos que pode ser linkado em html dentro de <head> provavelmente embaixo de <title>. Pode ser chamado **link:css** e resulta: <link rel=”stylesheet” href=”style.css”>. **[MAIS VERSÁTIL – MENOS HIERÁRQUICO].**

**ATALHOS VSCode:**

* **Ctrl + Shift + P:** Realiza última tarefa (no caso deixar: wrap with abbreviation – envelopar com abreviação).
* **Ctrl + Space:** Seleciona opções de arquivos para href=” “ e src=” “
* **Tab:** Adiciona “tabulação” com a quantidade de código desejado.
* **Shift + Tab:** Remove “tabulação” com a quantidade de código desejado.
* **Alt + cursor mouse:** Cria vários cursores permitindo apagar várias coisas de uma só vez.