# **Curso HTML/CSS Módulo 1**

Professor Gustavo Guanabara

No HTML5, tomar cuidado com a **semântica**, ou seja, o significado das **tags**. Semântica é muito importante, pois o próprio Google dá preferência na busca para sites com semântica. Existem tags que ainda funcionam, mas só são para estilo, e hoje, estilo é totalmente com CSS. Sempre preferir tags com semântica.

## **Cabeçalho/Head**

**Meta**: acentuação e tamanho do viewport (apenas no head);

**Title**: título da página que vai aparecer no navegador.

Texto

Descrição gerada automaticamente

## **Inserindo comentários**

**HTML**: <!-- Só de colocar isso no VS, ele já adiciona a tag do comentário, que completa é **<!-- texto -->**.

**CSS: /\* texto \*/**

## **Usando uma tag como texto**

Se quisermos mencionar a frase “p e /p”, para que não vire uma tag, devemos colocar assim: **&lt;p&gt e &lt;/p&gt**. Isso significa less than (menor que) e greater than (maior que).

## **Símbolos especiais**

Conseguimos adicionar diversos símbolos no html, basta digitar **&** e o código do símbolo. Alguns exemplos:

Tabela

Descrição gerada automaticamente

## **Emojis**

Ver em [www.emojipedia.org](http://www.emojipedia.org), buscar o emoji e embaixo o código do que queremos inserir. Sempre colocar inicialmente **&#x** e o código que tá lá no site. Exemplo: **&#x1F436** (cachorrinho).

## **Favicon**

Pequena imagem do site que o identifica mais facilmente no navegador. Formato ICO. Ir no site <https://favicon.io/> ou <https://www.iconarchive.com/> e baixar de lá, colocar na mesma pasta. Para inserir, ir na parte do **head** do código, acima do title, digitar link e escolher a opção favicon. Mudar **o** href, colocar o caminho do favicon:



## **Negrito**

A tag **<strong>** significa que o termo delimitado possui força dentro da frase. Logo, ele aparecerá em negrito. A tag <b> também funciona, mas não tem semântica, é apenas uma marca de forma, não está informando ao html que é um texto importante;

## **Itálico**

A tag **<em>** significa que queremos dar ênfase ao termo. Logo, ele aparecerá em itálico. A tag **<i>** também funciona, mas não tem semântica, é apenas uma marca de forma, não está informando ao html que é um texto importante;

## **Ênfase a endereços**

A tag **<adress>** significa que queremos dar ênfase a um endereço, e estamos informando ao navegador que aquilo é um endereço. No caso, aparecerá em itálico, e talvez até conseguiremos abrir no Google Maps em outras plataformas;

## **Texto sublinhado**

A tag **<mark>** deixa como se fosse com caneta marca texto;

**Dica de envelopamento de um texto**

Ctrl shift P: ir em “Emmet: wrap with abbreviation”, dar enter, e depois colocar a tag que queremos, tipo Strong. Muito mais fácil quando esquecemos de colocar uma frase numa tag.

## **Texto menor**

A tag **<small>** deixa o texto menor. A tag <**big>** deixava o texto maior, mas parou de existir;

## **Texto riscado**

A tag **<del>** deixa o texto com o risco ao meio;

## **Texto sublinhado**

A tag **<ins>** deixa o texto com o risco ao meio. A tag **<u>** também funciona, mas não tem semântica, é apenas uma marca de forma, não está informando ao html que é um texto importante;

## **Texto sobrescrito**

A tag **<sub>** é usada para deixar acima, tipo x2.

## **Texto subscrito**

A tag **<sup>** é usada para deixar acima, tipo h2o.

## **Destacando códigos**

A tag **<code>** deixa o texto com uma fonte monoespaçada, que é aquela utilizada em códigos e que tem o mesmo tamanho para cada letra. A tag **<pre>** serve para deixar o código com os espaçamentos à esquerda, bem certinho.

## **Citações simples**

A tag **<q>** deixa o texto entre aspas, de forma semântica (pois até daria para colocar as aspas manualmente, mas nesse caso é uma fala ou citação de terceiros). É uma citação simples, na mesma linha.

## **Citações completas**

A tag **<blockquote>** deixa o texto um recuo à esquerda, como se fosse um parágrafo de um livro. Já a tag **<cite>** serve para citar quem falou ou quem é o dono de tal citação, não há diferença nenhuma visualmente, mas é semântica e faz com que os mecanismos de busca identifiquem que essa frase é de tal livro ou pessoa;

## **Abreviações**

A tag **<abbr>** deixa um pontilhado embaixo do texto, e ela abre de forma que a gente consiga informar no **title** o que significa aquela sigla quando encosta o mouse:



Texto

Descrição gerada automaticamente

## **Texto invertido**

A tag **<bdo>** deixa o de forma invertida. Há 2 tipos: **bdo l (left),** que vai deixar o texto de left to right; e a **bdo r (right),** que vai deixar o texto de right to left. A opção **bdo r** deixa o texto invertido, bdo l não faz nada:



## **Listas**

### **Listas ordenadas**

A tag **<ol>** faz uma ordered lists (ol), que são aquelas listas onde a ordem dos itens é algo muito importante. A tag **<li>** (list item)serve para identificar cada item. Dá para configurar o marcador da lista com a tag **<type>**,ou seja, informar se é uma lista numérica **(1)**, algarismo romano maiusculo **(I),** algarismo romano minúsculo **(i),** lista com letra maiúscula **(A**) ou letra minúscula **(a).** Dá para usar também a tag **<start>,** informando de onde a lista vai começar:

Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente

### **Listas não ordenadas**

A tag **<ul>** faz uma unordered lists (ol), também chamadas de listas com marcadores, que são aquelas onde a ordem dos itens não influenciará no significado da lista. A tag **<li>** (list item)serve para identificar cada item. Dá para configurar o marcador da lista com a tag **<type>**,ou seja, informar se a lista deve ser marcada com círculo pintado **(disc),** círculo vazado **(circle)** ou quadrado **(square)**:

Texto

Descrição gerada automaticamente Texto

Descrição gerada automaticamente

### **Juntando listas**

Dá para juntar várias listas:

Texto

Descrição gerada automaticamente Texto, Carta

Descrição gerada automaticamente

### **Listas de definição**

A tag **<dl>** faz uma definition list, que basicamente é um dicionário, onde temos os termos e as descrições. A tag **<dt>** (definition term)representa cada termo, e a **<dd>** (definition description) é cada descrição. Dá para configurar o marcador da lista com a tag **<type>**,ou seja, informar se a lista deve ser marcada com círculo pintado **(disc),** círculo vazado **(circle)** ou quadrado **(square)**:

Texto

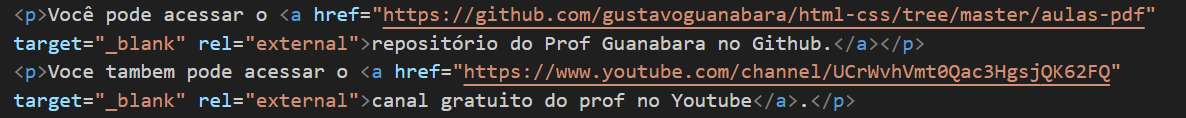
Descrição gerada automaticamente**Interface gráfica do usuário, Texto

Descrição gerada automaticamente**

## **Links**

### **Link externo (site de fora)**

A tag **<a>** faz um link com um site externo, ou seja, uma url., que basicamente é um dicionário, onde temos os termos e as descrições. O atributo **<href>** serve para colocar o próprio link; já o **<target>,** opcional,mas importante, serve para que o navegador abra uma nova janela **(\_blank),** ou na própria página **(\_self)**. O atributo **<rel> serve** para demonstrar se o site é externo (**external**), próxima parte de um site (next), parte anterior do site (**prev**), para um site do autor do próprio site (**author**), ou quando precisamos colocar um link mas não sabemos se é um link oficial, confiável ou não queremos dar um aval para aquele contéudo **(nofollow)**:

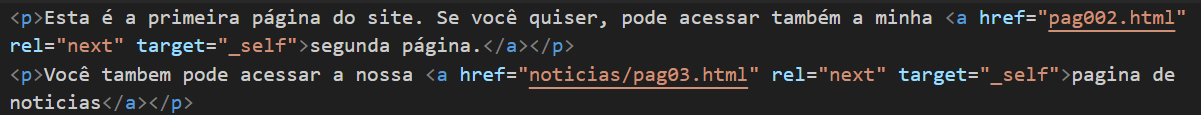


### **Para criar um link**

Apertar Ctrl e clicar em cima do link que colocamos!

### **Link interno (outra página dentro do site)**

Mesmas tags acima, mas no atributo **href** coloca-se o nome da página que foi criada (tipo index2.html, devendo estar na mesma pasta), no **target** vai ser o **\_self,** e no **rel** vai ser **next:**

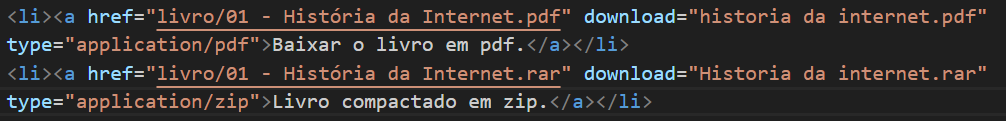


Se tiver uma terceira página de um site mas que esteja em outra pasta, se quiser colocar a referência para a primeira, devemos informar que tem que acessar uma “pasta acima” através de **../ :**



### **Link para downloads**

Mesmas tags acima, mas no atributo **href** coloca-se o nome do arquivo para fazer o download. Um novo atributo é o **download,** queé o nome do arquivo, e o atributo **type,** que no caso de um pdf, devemos colocar “application/pdf”. Para saber o que escrever dentro do **type,** acessar<https://www.iana.org/assignments/media-types/media-types.xhtml>:



Aqui vão alguns media types bem usados no nosso dia-a-dia:

* application/zip
* text/html
* text/css
* text/javascript
* video/mp4
* video/H264
* video/JPEG
* audio/aac
* audio/mpeg
* font/ttf
* image/jpeg
* image/png

## **Imagens**

Não é correto utilizar imagens da internet nos nossos sites, principalmente profissionalmente. Devemos contratar um fotografo ou ilustrador, tirar a própria foto, ou até contratar uma imagem nos bancos de imagens, como iStockPhoto, DepositPhotos, ShutterStock, etc. Mas no google, conseguimos ter acesso à imagens gratuitas procurando assim: Google imagens > Ferramentas > Marcar **“Licenças Creative Commons”.** Não ir emLicenças comerciais ou outras, pois essas imagens têm licenças que não são Creative Commons e podem ser de sites disponíveis sem custos financeiros ou de sites comerciais que exigem pagamento. Abaixo alguns sites para pegar imagens gratuitamente:

• UnSplash

• Pexels

• FreePik

• Rawpixel

• Pixabay

• Libreshot

• Wikimedia Commons

Basicamente existem 2 tipos principais de imagens para web, e devemos nos atentar quando utilizar cada um:

* **JPEG (criado em 1983, gerenciado por Joint Photographics Experts Group):** formato com imagens extremamente leves e compactos, o que é muito bom quando estamos fazendo um site que sempre deve ser leve e rápido, mas tem que tomar cuidado na hora de configurar o tamanho da compactação para que não fique horrível e toda borrada, pois a qualidade não é tão boa. Se quisermos colocar uma foto gigante no fundo da tela, usar esse formato;
* **PNG (criado em 1996, gerenciado por Portable Network Graphics)**: muito bom pois permite transparência, é leve, mas não tanto quanto o JPEG; quando quisermos uma imagem flutuando na tela com fundo transparente, é esse que vamos utilizar.

De forma resumida, na hora de escolher o formato de imagem para o seu site, opte sempre pelo formato **JPEG** com uma compactação entre 30% e 50%. O formato PNG só deverá ser usado quando precisarmos de transparência na foto.

Sobre o **tamanho das imagens**, a regra de ouro nesses casos é: use imagens do tamanho certo! Se precisar de uma imagem com 200 pixels de largura, gere um arquivo exatamente com esse tamanho. **Não ficar salvando arquivos gigantes** e diminuindo o tamanho da imagem com códigos, até porque não vai ficar mais leve se mudar a largura dela no seu CSS, o contrário, o navegador vai levar um tempão para carregar o arquivo do servidor e vai exibir a imagem minúscula na tela.

Para adicionar imagens, usar a tag **<img>.** O primeiro será onde a imagem está **(src=source, origem)**, que pode estar junto com o arquivo index, ou em outra pasta, ou mesmo num navegador. Depois, é o texto alternativo da imagem **(alt=alternative, alternativo),** que serve para quando um visitante for deficiente visual, o navegador vai ser capaz de descrever que tipo de foto está sendo mostrada ali. É importante para a acessibilidade.

Usar ctrl + espaço para achar o local da imagem:

**Texto

Descrição gerada automaticamente**

## **Imagens dinâmicas (que se adaptam)**

São imagens que se adaptam de acordo com o tamanho pré-definido da view port (janela). Por exemplo, temos 3 imagens: uma de 1000px x 1000px (G); outra de 700px x 700px (M); e outra de 300px x 300px (P).

Inserir a tag **<picture>** (indica que vai vim uma imagem logo a seguir) e, dentro dela, **<img>,** inserindoaimagem G, que é a imagem base que se quer carregar. Antes da tag img, inserir a tag **source** **mediatype,** que serve para informar alguns parâmetros de tamanho. Essa tag funciona assim: dentro dela, temos o atributo **media,** que serve para dizer o tamanho máximo ou mínimo da imagem; o outro atributo **srcset,** vamos colocar as imagens secundárias, ou seja, as imagens que vai carregar se os parâmetros não atenderem para carregar a imagem G. Por último, atributo **type,** informando o tipo da imagem (jpeg ou png):

Tela de computador com letras brancas

Descrição gerada automaticamente

Resumindo: até 750 px, vai carregar a imagem P; se passar disso e até 1050px, carrega a M; se passar, carrega a G e mantém nela. Tem que seguir a sequência, no caso a P primeiro, que é a menor, até a G, a maior.

Isso é muito útil para a UX experience, pois o site fica mais leve e mais bonito, não fica aquela imagem enorme se o usuário abrir no celular por exemplo.

## **Áudios**

Áudios gratuitos**: Ir no YouTube > Perfil > YouTube Studio > do lado esquerdo, Biblioteca de áudio.** Dá pra fazer o download!

Para adicionar um áudio no site, usar a tag <**audio>.** Informar o atributo **src**, que é a fonte, ou local onde está, e colocar o atributo **controls autoplay:**

****

Outra forma de inserir áudios de um jeito que não dê problema em nenhum navegador é: inserir normalmente a tag **<audio>,** e dentro dele colocar o atributo **source**. No source, colocar o **src,** que é onde o arquivo está, e o atributo **type,** que é o tipo de áudio (pode ser mpeg, ogg, wav). Sendo assim, o que estiver primeiro é o que o navegador vai tentar reproduzir; se não conseguir, vai para o próximo, e assim por diante. O atributo **preload** da tag **áudio** é bem importante, pois ele que vai indicar se o áudio deve ser carregado ou não. Se informar **auto**, o site vai carregar o áudio inteiro assim que abrir a página (bem perigoso, pois vai ficar pesado); se colocar **metadata**, vai carregar só algumas informações do áudio, e tem o **none,** que não vai carregar nada. Temos também o atributo **loop,** que recomeça o áudio quando ele chega no final:

Texto

Descrição gerada automaticamente

## **Vídeos**

**Dica de programa para fazer edição de vídeos**

**HandBrake**, baixado na máquina

### **Vídeos em hospedagem própria (localmente na própria máquina)**

Temos diversos formatos, dentre eles, o **m4v** (mais recomendado), **mp4, ogg e webm**. Aqui, o ideal é que se tenha vários formatos para inserir todos, igual fizemos com as imagens dinâmicas e com os áudios, para se caso o navegador não suportar um, vai pro outro.

A tag para inserir vídeo é **<video>.** Existem vários atributos, dentre eles, o **width**, que vai indicar a largura que o vídeo vai ter na tela. O atributo **poster** configura uma imagem que vai aparecer como uma capa, enquanto o usuário não aperta o play (capa do vídeo). O atributo **controls** vai configurar se os controles do vídeo vão aparecer na parte inferior da mídia. Por padrão, os controles não aparecerão, mas basta colocar a palavra **controls** na tag **<video>.** O atributo **autoplay** diz para o navegador se o vídeo vai começar a tocar automaticamente.

Texto

Descrição gerada automaticamente

O problema de colocar assim, localmente, é que o tráfego de dados fica absurdo de caro para o cliente, pois está no servidor local. Vamos ver outras formas muito melhores e mais baratas.

### **Vídeos com hospedagem no YouTube**

Para pegar um site do **YouTube**, existe a opção de incorporar o vídeo no nosso código. Temos que faze esse passo a passo: Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

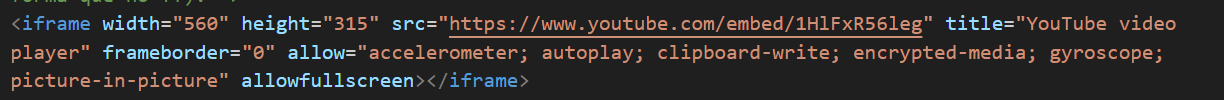
Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Só colar esse código com a tag **<iframe>** no html, mas dá pra mudar algumas coisas também, tais como largura e altura:



Nesse caso, não consome os dados do servidor, e sim do YouTube. De grande valia, pois se o cliente tiver uma conta no Yt que é de graça e colocar os vídeos dele lá, dá pra usar só a tag **iframe** mesmo.

### **Vídeos com hospedagem no Vimeo**

Bom porque dá para deixar os vídeos privados, por exemplo, num site de assinatura de cursos (estudonauta), o usuário só consegue chegar até o vídeo se assinar. Para incorporar, fazer o passo a passo abaixo:

Para vídeos do próprio usuário:



Para vídeos de outras pessoas:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

## **Tipos de CSS**

**CSS** quer dizer, *Cascading Style Sheet*, ou **folha de estilo em cascata,** e serve para formatar visualmente a página, seja em cores, fontes, espaços, formas, etc. Dá pra inserir de 3 formas, sendo a última a mais recomendada:

### **CSS inline**

A **CSS** **inline** deve ser inserida linha por linha, funciona, mas não é recomendável, além de deixar o código “poluído”, pois não há separação de html e css. Deve inserir a tag **<style>** em cada linha, e depois as declarações de estilo:

Texto

Descrição gerada automaticamente com confiança média

### **CSS interna/local**

Diferente do inline, a **interna/local** fica separada do html, fica na tag **head.** É funcional, mas só é recomendada se tiver apenas 1 página no site. Deve inserir a tag **<style> debaixo do title do head,** e tudo que for colocada dentro dela, que serão os seletores (h1, p, h2, etc), através de declarações (cada linha), vai ser estilo:

Texto

Descrição gerada automaticamente

### **CSS externa**

Também tem seletores (h1, h2, p) e declarações (background color, font, font-size) igual o interno, mas o **CSS externa** é o melhor, pois o estilo fica totalmente separado do html. Deve inserir o link da css **debaixo do title do head:**

Texto

Descrição gerada automaticamente

A página CSS, a parte, será assim (colocar o @charset “UTF-8”, sempre, para não dar problema com acentos):

Texto

Descrição gerada automaticamente

# **Curso HTML/CSS Módulo 2**

## **Representação das cores**

Existem 3 formas de representar cores por CSS:

### **Representação por nome**

Blue, white, yellow, red.. é muito simples, indicado apenas se for um site simples, ou mesmo para estudo:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

### **Representação por código hexadecimal (rgb)**

Lembrar que **hexadecimal** quer dizer base hexa, ou seja, os algarismos são **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.** São utilizados 6 dígitos que representam 65 milhões de cores.

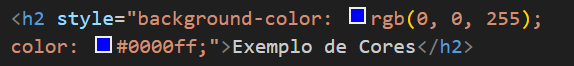
Todas as cores são derivadas de **vermelho (R), verde (G), e azul (B).** A representação fica assim: os 2 primeiros para vermelho (R), os 2 do meio verde (G), e os 2 últimos azul (B), formando o famoso **RGB**. Sendo que f = tudo da cor, e 0 = nada da cor. Cor preta é 000000, e branco ffffff. Cor vermelha pura é ff0000; cor verde pura é 00ff00; e cor azul pura é 0000ff.

Cada cor pode ir de 0 à 255 na escala:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Na CSS, pode colocar de 2 formas o RGB:



Informando: **rgb (0, 0, 255)** ou **#0000ff.** Dá para colocar transparência na cor também, aí adiciona a letra a e um dígito a mais (rgba).

### **Representação por características das cores (hsl)**

H = heel (matiz, cor) / S = saturation (saturação) / L = luminosity (luminosidade).

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Heel vai de 0 à 255; Saturation vai de 0 a 100%; e Luminosity de 0 à 100%. Também vira hsla se colocar transparência.

## **Tipos de cores e combinações**

Existe o **círculo cromático,** que serve para harmonizar as cores. Temos várias divisões de cores e coisas a observar:

**Cores primárias:** amarelo, azul e vermelho. Importante para gerar outras cores;

**Cores secundárias:** laranja, violeta e verde. Geradas pelas cores primárias;

**Cores terciárias:** cor primária + cor secundária. Ex: amarelo-esverdeado (sempre o nome da primária primeiro):

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Temperatura das cores:** frias e quentes:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores complementares:** maior contraste, cor oposta no círculo. Serve para ser utilizado em fundo e cores de fonte, para aparecer bastante:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores análogas:** 3 cores vizinhas, irmãs, não tem contraste tão grande, ótimo para criar paletas:

Ícone

Descrição gerada automaticamente

**Cores análogas e uma complementar:** ótimo para criar uma paleta, pois sempre precisamos de uma cor com bastante contraste. 3 cores análogas, e 1 complementar:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores análogas relacionadas**: ótimo para criar uma paleta também sem tanto contraste. 2 cores análogas, pula uma, pega a próxima:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores intercaladas**: 3 cores, escolhe 1 cor, pula uma, pega a próxima, pula, pega a próxima novamente. Fica um pouco dura, Guana não indicou muito:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores triádicas:** 3 cores,outra forma de intercalar, mas aqui pula-se de 3 em 3:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores em quadrado:** 4 cores,outra forma de intercalar, mas aqui pula-se de 2 em 2:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Cores tetráticas:** 4 cores, mas deve-se formar um retângulo, pegando sempre a complementar:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

**Monocromia:** 1 cor só, apenas mexendo na saturação e luminosidade. É um perfeito gradiente:

Gráfico, Gráfico de explosão solar

Descrição gerada automaticamente

## **Criando paletas de de cores**

O site <https://color.adobe.com/pt/create/color-wheel> é gratuito e ajuda muito a criar paletas.

Dá para criar todos os tipos de cores lá, e extrair tema de imagens (só arrastar pra lá), gradientes. Na aba “**Explorar**” e “**Tendências**”, aparecem milhares de sugestões de combinação de paletas.

O site <https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF> também é muito bom, dá pra ver até um pré visualização de um site na aba “Exemplo”.

O site <https://coolors.co/dde392-afbe8f-7d8570-646f58-504b3a> também é ótimo para inspirações, basta apertar espaço e ele vai gerando diversas paletas.

## **Capturando cores**

A extensão **Colorzilla** do Google Chrome consegue identificar qualquer cor da página ou imagem.

## **Configurações globais em CSS**

É necessário colocar algumas configurações antes de iniciar um trabalho de CSS. Para aplicar

Desenho em preto e branco

Descrição gerada automaticamente com confiança baixaNecessário para o gradiente pegar em toda a **altura** da tela.

Outra coisa que ajuda **muitooo** para não dar problemas, é definir tudo sem padding e margin. Para isso, basta colocar:

\* {

    margin: 0px;

    padding: 0px;

}

## **Gradientes em CSS**

Temos que usar a propriedade **background-image:**

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Primeiro informar o tipo de gradiente, que pode ser **linear-gradient**, que deixa faixas, ou **radial-gradient**, que deixa em forma de círculo.

Depois, colocar o sentido do gradiente: **to top, 45deg, 180deg, to left** para **linear,** e **circle** para **radial**.

O % depois da cor indica o quanto da cor vai aparecer.

## Texto Descrição gerada automaticamente**Propriedades diversas em CSS**

1

2

3

4

5

6

7

8

9

10

1 – Definir a **fonte** a ser utilizada: o dispositivo vai colocar a primeira fonte informada, se não encontrar, vai pra segunda, se não, vai pra terceira;

2 – Definir **cor de fundo;**

3 – Definir **arredondamento da borda;**

4 – Definir a **sombra da caixa**: o primeiro é o quanto vai andar para o lado; depois para baixo; e por último, o quanto ela vai se espalhar; por fim, a cor;

5 – Definir a **largura da caixa;**

6 – Definir o **padding**, ou espaçamento do texto com relação a caixa;

7 – Definir **onde a caixa vai se encaixar na página**; auto é no meio, e se adapta à tela;

8 – Definir a **cor do texto;**

9 – Definir o **alinhamento do texto:** center, justify, left; right;

10 – Definir a **sombra do texto:** o primeiro é o quanto vai andar para o lado; depois para baixo; e por último, o quanto ela vai se espalhar; por fim, a cor.

## **Fontes em CSS**

A grande questão são as fontes que nem sempre o dispositivo vai conseguir ler, para isso, devemos sempre colocar várias fontes:

font-family: 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

Nesse caso, ele vai primeiro tentar a Trebucht, se não achar, vai na Lucida, e assim por diante. Sempre terminar a declaração com **serif, sans-serif, monospace,** pois se não achar nenhuma, ele vai colocar qualquer uma que tiver no dispositivo.

## **Tamanhos de fontes em CSS**

Para a propriedade **font-size,** devemos nos atentar a algumas questões. Existem 2 tipos de medidas:

**Medidas absolutas:** cm, mm, in (polegada), px (pixel), pt (ponto - não se recomenda em tela), pc (paica - nao se recomenda em tela);

**Medidas relativas:** em (relativa ao tamanho atual da fonte), ex, rem, vw, vh, %. ex (relativo a altura x da fonte), rem, vw, vh, %.

O mais recomendado pelo W3C é o **em** ou **px.** Normalmente, o tamanho normal de uma fonte é **16px**. Sendo assim, 1em = 16px. A medida **em** é multiplicadora, se colocarmos 0.5em = 8px, 2em = 32px. A melhor a ser utilizada é o **em**, pois ela é baseada em porcentagem e serve para qualquer tela.

font-size: 1em;

## **Peso e estilos de fontes em CSS**

Para a propriedade **font-weight,** que define o peso de uma fonte, ou seja, o quanto ela ficará magrinha ou gordinha, devemos utilizar no font-family um tipo de família que tenha todas essas fontes. Temos as opções **lighter, normal, bold, bolder,** e também dá pra informar de **100 a 900** (apenas algumas fontes aceitam).

font-weight: bold;

Para a propriedade **font-style,** conseguimos colocar o estilo da fonte. Temos as opções **italic, normal, oblique.**

Para a propriedade **text-decoration,** conseguimos colocar tracejados a fonte. Temos as opções **dashed, dotted, double, underline etc.**

Contudo, temos que ver se a família da fonte instalada tem todas essas opções, algumas não tem itálico por exemplo.

## **Shorthand Fonte**

Dá pra aplicar todas as configurações em 1 só linha pra facilitar: a ordem é: **style,** depois **weight**, depois **size**, depois o **Family**:

font: italic bolder 3em 'Trebuchet MS', 'Lucida Sans Unicode', 'Lucida Grande', 'Lucida Sans', Arial, sans-serif;

## **Inserindo fontes externas sem baixar**

Acessar o site <https://fonts.google.com/> para ter acesso ilimitado as fontes. Achar a fonte, e ir em **@import**, e copiar o código de lá no css:

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Colar logo embaixo do **style**, que indica o início do CSS interno, ou logo no **início do código** se for CSS externo**:**

<style>

        @import url('https://fonts.googleapis.com/css2?family=Dancing+Script&display=swap');

        @import url('https://fonts.googleapis.com/css2?family=Work+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800&display=swap');

        @import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');

## **Inserindo fontes externas baixadas**

Site <https://www.dafont.com/pt/> também muito bom, bastante fontes diferentes.

Observar sempre os glifos (letras) da fonte, se tem todos que queremos, tais como acentos, tiu etc. E verificar se tem regular, bold, italic.

Fazer download das fontes e colocá-las na mesma pasta do index.html, ou criar uma pasta de fonts também, tanto faz. Temos alguns tipos de fontes: **otf (opentype), ttf (truetype)**, **embedded-opentype, truetype-aat, svg.** O indicado é sempre baixar 2 tipos (otf e ttf são os mais utilizados), e importá-los no código, pois caso o navegador não ler um, tem o outro, através da propriedade **@font.face:**

@font-face {

            font-family: 'Best Valentina';

            src: url('fonts/Best Valentina TTF.ttf') format('truetype'), url('fonts/Best Valentina OTF.otf') format('opentype');

            font-weight: normal;

            font-style: normal;

        }

Dar o nome no **font-family** que quiser, que vai ser o nome que vamos utilizar no texto; no **src**, a **url** é exatamente o nome da fonte como aparece na pasta; no **format,** definir o formato (truetype, opentype); dá para colocar as 2 fontes na mesma linha (otf e ttf), primeiro informar a url e format de uma, depois da outra. Depois, só usar a fonte com o nome dado:

      font-family: 'Best Valentina', Times, serif;

## **Capturando fontes de textos**

A extensão **Fonts Ninja** do Google Chrome consegue identificar qualquer fonte da página.

## **Capturando fontes de imagens**

Dá para tentar achar uma fonte de uma imagem nesses sites, mas não são 100% confiáveis:

<https://www.whatfontis.com/?s2c#croptext>

<https://www.fontsquirrel.com/>

<https://www.myfonts.com/?gclid=Cj0KCQjwho-lBhC_ARIsAMpgMofPsxq5ds-Je8y7G2Tl30xRjG_Fc9kgB11yfSc6Vh9XiEvyFv43yzoaAo3qEALw_wcB>

## **Alinhamento de textos**

Existem 4 tipos de **alinhamento de texto:** center, justify, left; right. O padrão é a esquerda. A propriedade **text-indent** afasta o início do texto:

text-indent: 30px;

## **Seletores personalizados**

Utilizamos quando temos 2 seletores iguais (dois h2 por exemplo), mas queremos editar apenas 1. Por padrão, todos os seletores são genéricos. Ou seja, se aplicar um estilo ao h2, todos os h2 pegarão aquele estilo.

### **ID (#)**

É um identificador que o diferencia dos outros, colocado logo após o seletor no html:

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

<h1>Aprendendo HTML</h1>

O ideal é utilizar apenas um determinado ID para um seletor, segundo o W3C. Ou seja, não podemos mais colocar id=”principal” para outro seletor. Ele até funciona, mas vai contra as boas práticas.

Na CSS, para fazer o estilo desse id, utilizamos “#”. Daria até para colocar só o #principal, sem o h1 também:

h1#principal {

    text-align: center;

    background-color: green;

    color: white;

}

### **Class (.)**

É a classe que o diferencia das outras, colocado logo após o seletor no html. Podemos usar class em vários seletores, diferente do id:

<h2 class="intermediario">HTML intermediário</h2>

<h2 class="intermediario">CSS intermediário</h2>

Na CSS, para fazer o estilo dessa class, utilizamos “.”. Dá pra para colocar só o .intermediario, sem o h2 também:

.intermediario {

    color: rgb(148, 148, 31);

    font-weight: ;

}

Dá também para utilizar 2 classes, não precisa separar com vírgula:

 <h2 class="avançado destaque">CSS avançado</h2>

E dá pra colocar um id e uma class para um seletor, lembrando que as configurações do id sempre vão se sobrepor às da class.

## **Pseudo-Classes em CSS**

Elas devem estar relacionadas a um elemento ou uma classe ou div, e estão relacionados ao estado de um desse elemento: se está ativo, marcado, vazio, habilitado etc. Usamos o “:” depois do seletor. Tem vários tipos, pode ser **:hover, :visited, :active, :checked, :empty e :focus.**

### **Hover (:) – Quando encosta o mouse**

Por exemplo, :**hover** faz com que ao passar o mouse em cima, fique uma configuração diferente (abaixo a cor de fundo da div fica em amarelo):

div:hover {

            background-color: yellow;

        }

Outro exemplo a seguir, podemos utilizar “>” para indicar que tal coisa é filho de outra. No exemplo abaixo, o parágrafo que está dentro da div, é filho dela, e a propriedade **display: none** esconde o texto:

div > p {

            display: none;

        }

Mas se quisermos fazer com que o texto apareça quando encostar o mouse em cima, podemos usar a **hover** assim, usando o **display: block**:

div:hover > p {

            display: block;

            color: white;

            background-color: red;

            width: 300px;

        }

Isso é um protótipo de um menu, onde a gente pode encostar o mouse e aparecer as opções.

### **Active (:) – Quando clica em cima (no exato momento)**

Quando clica num link, no exato momento, dá pra fazer alterações também com o **:active:**

a:active {

            color: yellow;

        }

### **Visited (:) – Quando já clicou em cima de um link**

Por padrão, um link em HTML fica violeta quando **visitado**. Porém, podemos fazer alterações nessas configurações, através de **:visited:**

a:visited {

            color: green;

        }

## **Pseudo-Elementos em CSS**

É uma palavra-chave adicionada às declarações de um seletor e que permitem que você formate um pedaço específico do elemento referenciado. Os principais pseudo-elementos usados são: **::before, ::after, ::first-letter, ::first-line.**

### **Before (::) – Adicionando antes de um elemento**

Coloca algum parâmetro antes de um elemento. Abaixo por ex., lê-se “Antes de todos os links, vai aparecer as configurações...”:

a::before {

            content: '⇒ ';

            text-decoration: none;

            font-weight: normal;

            color: grey;

        }

No **content,** pode-se colocar um texto ou símbolos, isso é o que vai aparecer exatamente no código:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Mesmas configurações para o **after.**

## **Modelos de caixas**

**Caixas** são elementos, seletores, h1, p, a, vídeo etc.

**Aninhamento** é colocar uma caixa dentro da outra no HTML, ou seja, organizar os elementos um dentro do outro e com certa ordem.

Existem 2 tipos de **caixa**:

**Box level:** uma caixa que ocupa toda largura onde está inserida, e quebra o texto (h1, h2, h3, h6, p, vídeo, tags semânticas). Ou seja, ela não continua na mesma linha do código quando é inserida. Mais famoso é a **div.**

**Inline level:** uma caixa que não quebra a linha, ela acompanha o conteúdo (a, code, button). Ou seja, ela continua na mesma linha do código quando é inserida. Podemos citar o **span,** que não quebra a linha.

### **Facilitando a vida do programador**

Quando inserimos alguns seletores num código HTML e configuramos com CSS, o navegador mostra as medidas utilizadas no **“Inspecionar elemento” (botão direito):**

Diagrama

Descrição gerada automaticamente

Olhe como está no código (o mesmo h1 do print acima):

h1 {

            background-color: lightpink;

            height: 300px;

            width: 300px;

            border: 10px solid brown;

            padding: 10px 10px 10px 10px;

            margin: 20px auto 40px auto;

            outline: 5px dashed salmon;

        }

**Outline:** é praticamente uma borda, mas fica da borda real para a margem. Ela pega o espaço da margem. É a parte traçejada:

Forma, Quadrado

Descrição gerada automaticamente

## **Mais shorthands**

Os famosos atalhos, temos para vários tipos:

### **Borda**

Dá para aplicar todas as configurações em 1 só linha pra facilitar: a ordem é: **largura,** depois **estilo da borda** e depois **cor**:

border: 10px solid brown;

### **Padding**

Dá para aplicar todas as configurações em 1 só linha pra facilitar: em sentido do relógio, primeiro a **largura** do **topo, depois direita, embaixo e esquerda**. Ou se for tudo igual, pode colocar só uma vez a largura. Se colocar só 2 valores, ele vai configurar o de cima e o debaixo iguais, e da esquerda e direita iguais:

padding: 10px 10px 10px 10px;

### **Margem**

Dá para aplicar todas as configurações em 1 só linha pra facilitar: em sentido do relógio, primeiro a **largura** do **topo, depois direita, embaixo e esquerda**. Ou se for tudo igual, pode colocar só uma vez a largura. Se colocar só 2 valores, ele vai configurar o de cima e o debaixo iguais, e da esquerda e direita iguais:

margin: 20px auto 40px auto;

Lembrar que **margin** **auto** é para **centralizar a caixa**.

### **Outline (borda externa)**

Dá para aplicar todas as configurações em 1 só linha pra facilitar: a ordem é: **largura,** depois **estilo da borda** e depois **cor**:

outline: 5px dashed salmon;

## **Transformando inline em box level**

Dá pra transformar um elemento inline em box level, e vice-versa, através do **display**. No exemplo abaixo temos um link:

a {

            display: block;

Visualmente ficou assim, transformou em box level (ou seja, não fica mais na mesma linha dos outros):

Uma imagem contendo Retângulo

Descrição gerada automaticamente

## **Grouping tags**

A linguagem HTML padrão tinha apenas duas tags de agrupamento genérico: **a <div> e a <span>.** A diferença básica entre elas é que a primeira é um elemento agrupador do tipo block-level e o segundo é inline-level. Com o surgimento da HTML5, surgiram as tags **semânticas** **de** **agrupamento**. Isso não significa que as <div> e <span> (agora chamadas de não-semânticas) deixaram de existir ou ficaram obsoletas, mas seu uso agora faz menos sentido, pois temos tags para dividir as partes do nosso documento HTML. A seguir vamos conhecer cada uma delas.

### **Header**

Cria áreas relativas a cabeçalhos. Pode ser o cabeçalho principal de um site ou até mesmo o cabeçalho de uma seção ou artigo. Normalmente inclui títulos <h1> - <h6> e subtítulos. Podem também conter menus de navegação.

### **Nav**

Define uma área que possui os links de navegação pela estrutura de páginas que vão compor o website. Um <nav> pode estar dentro de um <header>.

Calendário

Descrição gerada automaticamente com confiança média

### **Main**

É um agrupador usado para delimitar o conteúdo principal do nosso site. Normalmente concentra as seções, artigos e conteúdos periféricos.

### **Section**

Cria seções para sua página. Ela pode conter o conteúdo diretamente no seu corpo ou dividir os conteúdos em artigos com conteúdo específicos. Segundo a documentação oficial da W3C, “uma seção é um agrupamento temático de conteúdos, tipicamente com um cabeçalho”.

### **Article**

Um artigo é um elemento que vai conter um conteúdo que pode ser lido de forma independente e dizem respeito a um mesmo assunto. Podemos usar um <article> para delimitar um post de blog ou fórum, uma notícia, etc.

### **Aside**

Delimita um conteúdo periférico e complementar ao conteúdo principal de um artigo ou seção. Normalmente um conteúdo <aside> está posicionado ao lado de um determinado texto ou até mesmo no meio dele, exatamente como fizemos no bloco de texto apresentado anteriormente, falando sobre “MÚLTIPLOS NÍVEIS”.

Texto

Descrição gerada automaticamente

### **Footer**

Cria um rodapé para o site inteiro, seção ou artigo. É um conteúdo que não faz parte diretamente do conteúdo nem é um conteúdo periférico (o que caracterizaria um <aside>), mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

## **Sombras**

Cuidado com sombras, não é necessário exagerar, sempre colocar transparência. Dá para usar o **box-shadow:** o primeiro é o quanto vai andar para o lado; depois para cima/baixo; e por último, o quanto ela vai se espalhar; por fim, a cor:

box-shadow: 2px 2px 4px rgba(0, 0, 0, 0.315);

## **Cantos arredondados**

Deixar os cantos arredondados para não ficar tão grosseiro. Só 1 medida aplica para todos; 4 medias é um para cada canto; 2 medidas considera: superior esquerda + inferior direita, e superior direita + inferior esquerda:

border-radius: 15px 0px;

Fica uma bola!:

border-radius: 50%;

## **Bordas decoradas**

Guana não gosta muito, realmente bem feio:

Quando vai colocar imagem, deixar transparente:

border: 12px solid transparent;

border-image: url('borda.png') 27 repeat;

Dica do professor – Site para desenhar wireframes:

**MockFlow**

## **Transição em CSS**

Ao encostar num link, para dar aquela “demoradinha”, no caso abaixo 0,5 seg:

transition-duration: 0.5s; /\*qdo mexer o mouse por cima, leva 0.5 seg de transição.\*/

## **Altura de linhas em CSS**

Espaçamento entre uma linha e outra:

line-height: 2em; /\*altura entre linhas.\*/

## **Configurando marcadores de listas em CSS**

Para deixar os marcadores do lado de dentro da margin, utilizar a configuração abaixo:

list-style-position: inside; /\*deixar as bolinhas para dentro da caixa.\*/

Fica assim:

Diagrama

Descrição gerada automaticamente

Outra coisa, para deixar com os marcadores acima, pegar o código da emojipedia que se quer usar e utilizar assim (mas não é compatível com todos os navegadores!):

list-style-type: '\2714\00A0\00A0';/\*esse é o código do emoji que substituiu as bolinhas, os 00A0 são espaços depois. Nem sempre funciona para todos os navegadores.\*/

## **Colunas com CSS**

columns: 2; /\*dividir a lista\*/

## **Variáveis em CSS**

Utilizadas para facilitar nossa vida, pois guardam valores para serem utilizados mais para a frente. O campo **root** ajuda muito, devemos inseri-lo logo no início do código CSS, antes mesmo das configurações globais. Ai colocar as variáveis, no caso abaixo são as **--cor0:, --cor1:, --font-padrao:**, e assim por diante. Sempre iniciar com --. No exemplo abaixo, definimos variáveis para cores e fontes:

:root {

    --cor0: #c5ebd6;

    --cor1: #83e1ad;

    --cor2: #3ddc84;

    --cor3: #2fa866;

    --cor4: #1a5c37;

    --cor5: #063d1e;

    --font-padrao: Arial, Verdaba, Helvetica, sans-serif;

    --fonte-destaque: 'Bebas Neue', cursive;

    --fonte-android: 'Idroid', cursive;

}

Agora fica mais fácil definir as configurações para cada seletor, pois ao invés de colocar cor tal para cada um, é só colocar o nome da variável. Por exemplo:

body {

    background-color: var(--cor0);

    font-family: var(--font-padrao);

}

## **Responsividade em sites**

Para evitar rolagens laterais e imagens cortadas, devemos pensar em responsividade, ou seja, uma forma do nosso site se adaptar a diferentes telas e ainda assim ficar apresentável.

Pensar também na forma do texto, ou seja, em como ele vai ser visto pelas pessoas, Não é legal um site que ocupe muito espaço de tela, pois as pessoas podem ter dificuldade para ler; o ideal é que tenha um caixa atras de todos as outras caixas, e que ela delimite:

Uma imagem contendo Aplicativo

Descrição gerada automaticamente

No caso da tag **main,** que é a onde está todo o conteúdo em branco acima, devemos colocar larguras mínima e máxima e auto (este ultimo para centralizar). Quando tiver 1000px, a caixa vai parar de crescer:

main {

    min-width: 300px;

    max-width: 1000px;

margin: auto;

Para as imagens, para que eles se adaptem na tela, colocar largura:

main img {

    width: 100;

}

E também lembrar que lá no html, dá pra colocar 2 imagens, uma para dispositivos grandes e outros pequenos:

<picture>

                <source media="(max-width: 670px )" srcset="imagens/dan-droids-pq.png" type="image/png">

                <img src="imagens/dan-droids.png" alt="Dandroids, primeiro mascote do Android">

            </picture>

O caso abaixo, tínhamos uma imagem muito grande (apenas 1), então, para diminuir, foi criado uma class pequena e colocado as configurações de **largura máxima, display bloco (para virar uma caixa), e margin auto para centralizar:**

main img.pequena { /\*só foi criada para diminuir a img, ficou mto grande após tornar o site responsivo.\*/

    max-width: 350px;

    display: block; /\*necessário para fazer com que a tag img vire um bloco, e que possa ser centralizada na tela.\*/

    margin: auto;

}

## **Responsividade em vídeos**

O grande problema em vídeos é que como eles já vem prontos do Yt (na maioria das vezes pelo menos), fica difícil configurar seu tamanho sem perder qualidade. Sendo assim, precisamos fazer alguns macetes para deixar ele agradável na tela. O que podemos fazer?

1. Colocá-lo dentro de uma caixa, no caso uma **div**, para que a gente consiga editá-lo;
2. Usar posições absolutas e relativas, separando os seletores:
3. div.video {
4. background-color: var(--cor5);
5. margin: 0px -20px 30px -20px;
6. padding: 20px;
7. padding-bottom: 58%;
8. position: relative;
9. }
10. div.video > iframe {
11. position: absolute;
12. top: 5%; /\*so aparece quando coloca posição absoluta\*/
13. left: 5%; /\*so aparece quando coloca posição absoluta\*/
14. width: 90%;
15. height: 90%;
16. }

Sugestão de um colega do Yt, só não ficou com a caixa atras (div), mas ficou mto bom!

Texto, Carta

Descrição gerada automaticamente