**BITS E BYTES**

Cada byte é formado por 8 bits, esses 8 bits são formados por 0 e 1 (Binary Digit - dígitos binários). O 0 representa negativo e o 1 representa positivo.

1 **K**B (**quilo**byte) ➜ 2\*\*10 = 1.024

1 **M**B (**mega**byte) ➜ 2\*\*20 = 1.048.576

1 **G**B (**giga**byte) ➜ 2\*\*30 = 1.073.741.824

1 **T**B (**tera**byte) ➜ 2\*\*40 = 1.099.511.627.776

1 **P**B (**peta**byte) ➜ 2\*\*50 = 1.125.899.906.842.624

1 **E**B (**exa**byte) ➜ 2\*\*60 = 1.152.921.504.606.846.976

1 **Z**B (**zetta**byte) ➜ 2\*\*70 = 1.180.591.620.717.411.303.424

1 **Y**B (**yotta**byte) ➜ 2\*\*80 = 1.208.925.819.614.629.174.706.176

**IMPORTANTE: Letra “B”:**

Letra “B” ➜ byte(s) | Associado a **ARMAZENAMENTO**.

Letra “b” ➜ bit(s) | Associado a **TRANSMISSÃO**.

Faz diferença SIM! Então, cuidado!

**Definição de termos de Internet:**

**Cliente ➜ pode ser um celular, computador e até mesmo uma TV.**

**Internet ➜ um local virtual gigante com todos os servidores do mundo.**

**Modem ➜ modulação e demodulação, são essas duas palavras que formam seu nome e além disso mostra sua função.**

**DNS ➜ sigla de Domain Name System. É o que converte o nome de um domínio para o IP correspondente a esse nome. Todo site tem um IP e ele é fundamental para acessar esse site. Seria muito cansativo e complicado ficar digitando um IP para acessar um site toda vez. Além disso, os IPs mudam a todo momento. Então o DNS facilita muito, sendo necessário pesquisar por um nome ao invés de um monte de números.**

**Servidor ➜ é onde está hospedado dados em arquivos, sendo eles imagens, vídeos, etc. Quando solicitado, envia uma cópia de arquivos para o Cliente.**

**Domínio ➜ nome que identifica um site. Não existem domínios iguais.**

**Hospedagem ➜ local onde um site vai estar armazenado.**

**URL ➜ é o endereço que você digita para entrar em um site na Internet. A sigla URL significa *Uniform Resource Locator*, ou “Localizador Uniforme de Recursos”.**

**TLD ➜ o *Top Level Domain* faz parte de um Domínio, exemplos de TLDs são: .com, .net, gov.br, .io, .edu. Eles representam a instituição. Existem tipos de TLDs, veja:**

**GTLD ➜ são TLDs genéricos, como .com, .net. .io, .org, .online, .store, .gov, etc.**

**ccTLD ➜ são as TLDs de países (coutry code). Exemplos são: .com.br, .edu.fr, co.us, .es, .tv, .jp.**

**HTML ➜ *HYPERTEXT MARKUP LANGUAGE* (Linguagem de Marcação em Hipertexto). Linguagem focada em conteúdo, podendo ser imagens, vídeos, textos, tabelas, etc.**

**CSS ➜ CASCADING STYLE SHEETS (Folhas de Estilo em Cascata). Linguagem focada em Design, como cores, sombra, posições, tamanhos, etc.**

**JavaScript ➜ abreviado em JS, é uma linguagem de programação. Em um site é responsável pelas interações.**

Um computador entende apenas a base 2 (0 e 1 – Binary Digit) e quando o Cliente digita um domínio, como [*www.instagram.com*](http://www.instagram.com),ele passa para o servidor DNS, que procura esse domínio em uma lista de nomes e ao fim, passa o IP do site desejado, no caso o Instagram. Assim, ele passa esse IP para o Cliente e assim é possível acessar o site.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Conteúdos em HTML**

**Alguns exemplos antes de começar:**

**<h1> título </h1>**

**<p> parágrafo </p>**

Abertura de uma tag, título e o fechamento da tag.

**<img src=”foto.png” alt=”Exemplo de foto”>**

Abertura de tag, src e alt são parâmetros e “Exemplo de foto” é um valor.

**Estilos em CSS**

**h1 {**

**font-family: Arial;**

**font-size: 20pt;**

**color: orange;**

**}**

Tudo entre os colchetes é chamado de declaração, o “h1” é o **seletor**, font-family, font-size e color são chamados de **propriedade,** e após os dois pontos é chamado de **valor**.

**Estrutura básica de um em HTML**

<!DOCTYPE html>

<html lang=*"pt-BR"*>

<head>

    <meta charset=*"UTF-8"*>

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

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

    <title>Nome da Aba</title>

</head>

<body>

</body>

</html>

A área **“head” é a área de configurações**, já a área **“body” é o corpo, o site em si**. **UTF-8 é a configuração das letras**, para que os acentos sejam mostrados corretamente. A segunda linha “meta” é a exibição do site, sua escala padrão, ocupação etc. O “***title***” é o título.

**HTML**

Vamos ver um pouco de HTML (ou muito, sei lá).

**SEMÂNTICA**

“Semântica é o significado dos vocábulos, por oposição a sua forma”. O que importa no **HTML** é o significado, deixe a forma para o **CSS**. Como exemplo:

*Samuel mora em <u>Cachoeira de Teraquente, 231<u>.*

A tag ***<u>*** serve para destacar o texto, servia, só que de forma genérica. É mais interessante escrever da seguinte forma:

*Samuel mora em <address>Cachoeira de Teraquente, 231</address>.*

Dessa forma, o texto se tornou literalmente um endereço, podendo até mesmo acessar o Google Maps com um clique, caso programado!

**Negrito / Destaque**

A palavra **"negrito"** está escrita em negrito! Foi utilizada a tag <b> (não semântico)

Já aqui a palavra **"negrito"** está utilizando a tag <strong> (semântico)

**Itálico / Ênfase**

Escrito de forma *genérica*, com a tag <i> (não semântico)

Já nesse parágrafo, o *itálico* é por conta da tag <em> (semântico)

**Marcador**

A palavra está marcada! Utilizado a tag <mark>.

**Sublinhar**

Texto sublinhado. Foi utilizado a tag <ins>.

**Riscar texto (texto descartado)**

Este texto está ~~riscado~~, utilizado a tag <del>!.

**Caracteres Pequenos (fórmulas sla)**

H2O, o caractere pequeno foi escrito com a tag <sub>.

105, escrito com a tag <sup>.

**TAGS OBSOLETAS**

Com o tempo, uma tag pode ‘morrer’, simplesmente deixar de funcionar, isso pode acontecer por vários motivos, pode ser que exista outra que é melhor ou ela é muito mais **forma** que **significado**. A grande diferença da HTML4 para a HTML5 é justamente a **semântica** e, por conta disso, muitas tags se tornaram **obsoletas**.

**Exemplos de tags obsoletas:**

~~basefont~~

~~big~~

~~blink~~

~~center~~

~~font~~

~~marquee~~

~~multicol~~

~~nobr~~

~~spacer~~

~~tt~~

**Texto Invertido**

Use a tag ***<bdo>*** (*bi-directional override*), no parâmetro “***dir***” coloque a direção: “***rtl***” = *right to left* | “***ltr***” = *left to right*.

    <p>

        <bdo dir=*"rtl"*>Olá, Mundo!</bdo>

    </p>

**Comentários**

**<!—TEXTO -->** ➜ em documentos HTML.

**/\* TEXTO \*/** ➜ em documentos CSS.

**Manter Escrita da Forma Digitada**

Utilizando a tag ***<pre>,*** é possível manter um parágrafo/título da forma que você digitou, com espaços e quebras de linha.

**Escrita de Código**

Talvez surja a necessidade de compartilhar seus códigos em seu site, pra isso existe uma tag: ***<code>***.

***Código de exemplo:***

    <h1>

        <pre>   Ímpar ou Par em Python</pre>

    </h1>

    <pre>

        <code>

            n = int(input('Digite um número: '))

            if n % 2 == 0:

            print(f' - O número {n} é PAR!')

            else:

            print(f' - O número {n} é ÍMPAR!')

        </code>

    </pre>

**Títulos**

**<h1> a <h6>** ➜ maior **<h1>**, menor **<h6>**. O “h” significar *headings*, que traduzindo ficar “título” mesmo.

**Hierarquia dos Títulos**

O título principal deve ser o maior, ou seja, o ***h6***. Os subtítulos desse ***h6*** devem ser ***h5***, os subtítulos desse ***h5*** devem ser ***h4***, e assim por diante.

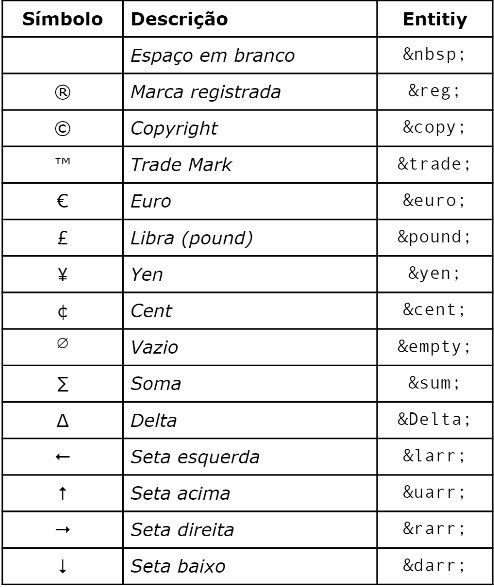
Não há nenhum problema em haver dois h6 em um texto, mas não deve ser dentro de um ***h6*** e deve ser bem organizado para não confundir o leitor.

**Parágrafos**

**<p>** ➜ cria um parágrafo.

**<br>** ➜ quebra de linha dentro de um parágrafo. Essa tag significa “*break row*”, ou “quebra de linha”

**HTML Entities**



**Emojis**

Para utilizar um emoji basta saber seu *Codepoints*, no caso, pode ser encontrado no site [www.emojipedia.org](http://www.emojipedia.org).

Transcrevendo o emoji referente ao *Codepoints* U+1F600 (carinha feliz) para ser escrito em HTML, basta trocar U+ por &#x (minúsculo mesmo) e ao final acrescentar um ponto e vírgula “;”.

**Citações**

Para fazer uma citação, use a tag ***<q>***, uma referência direta a “quote”. É útil para fazer citações no meio de parágrafos!

***Código de exemplo:***

    <p>

        Já foi dito: <q>Yuumi não é Champion!</q>

    </p>

**Citar um Bloco Inteiro**

Para isso use a tag <blockquote cite=’ ’>, o parâmetro “cite” é deve ser uma URL de um site.

***Código de exemplo:***

<blockquote cite=*"https://www.youtube.com/channel/UCs6bahfNFeK51gkuwYIvWog"*>

        Uma vez, um adc e um suporte estava farmando tranquilamente no bot, aí apareceu um Aurelion e matou os 2

</blockquote>

**Abreviação**

Para abreviar, use a tag ***<abbr>***, e no parâmetro “title” coloque um texto. Esse texto será mostrado quando passarem o cursor por cima do texto.

***Código de exemplo:***

    <p>

        Eu não jogo se tiver <abbr title=*"Champion chato da desgraça"*>Lulu</abbr> no outro time

    </p>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Imagens**

Falaremos sobre formatos, importação para o site e *favicons*.

**Formatos**

Existem diversos formatos de imagens, mas qual é o melhor? A resposta é simples: “depende!”. Apenas duas valem apena de se utilizar, PNG e JPEG, que são formatos leves para sites, o que facilita sua renderização.

**Utilize PNG** caso seja necessário um fundo transparente, esse é basicamente o ponto forte desse formato.

**Utilize JPEG** em qualquer outro caso. SIMPLES!

Ah, e quem sabe não seja legal utilizar um **gif** as vezes...

**Importação de Imagem**

Para isso, existe um comando:

<img src=*""* alt=*""*>

O parâmetro ***\*src\**** é o local onde está a imagem, podendo ser o diretório da imagem no computador ou uma URL. O parâmetro ***\*alt\**** é um breve texto que descreve a imagem.

***Favicons***

Os favicons são os ícones ao lado do nome da página. É recomendado o uso de arquivos **.ico**, que são arquivos de ícones. O comando abaixo mostra como adicionar o ícone .

<head>

    <meta charset=*"UTF-8"*>

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

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

    <link rel=*"shortcut icon"* href=*"icon/er\_purple.ico"* type=*"image/x-icon"*>

    <title>Favicon 4</title>

</head>

**Linha do favicon:**

    <link rel=*"shortcut icon"* href=*"icon/er\_purple.ico"* type=*"image/x-icon"*>

Ele deve ficar **dentro do escopo <head>.**

**Listas Em HTML**

Listas são bem úteis para organização e para criar menus em um site.

**Listas Ordenadas (*Ordered Lists*)**

São chamadas de ***ordered lists*** em HTML aquelas listas que a ordem de cada item é muito importante.

Para criar uma ordered list use a tag ***<ol>*** (é necessário fechar essa tag). Cada item de uma lista é identificado pela tag ***<li>***.

    <ol>

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

        <li>Item 4</li>

        <li>Item 5</li>

        <li>Item 6</li>

    </ol>

A tag ***<ol>*** possui um parâmetro ***type***, onde é configurado o tipo de marcador dessa lista:

**➜ 1 – Valor padrão, cada item é numerado. Ex.: 1, 2, 3, 4, 5**

**➜ A – Lista alfabética em maiúscula. Ex.: A, B, C, D, E**

**➜ a – Lista alfabética em minúscula. Ex.: a, b, c, d, e**

**➜ I – Lista em números romanos em maiúsculo. Ex.: I, II, III, IV, V**

**➜ i – Lista em números romanos em minúsculo. Ex.: i, ii, iii, iv, v**

Utilizando o parâmetro ***start***, você pode definir o primeiro valor de uma lista, ela será contada a partir desse valor. Lembrando que esse valor deverá ser numérico, mesmo que a forma de contagem da lista seja alfabética.

**Listas Não Ordenadas (*Unordered Lists*)**

Também são chamadas de Listas com Marcadores, que são aquelas em que a ordem dos itens não têm importância.

Para criar esse tipo de lista use a tag ***<ul>*** (precisa ser fechada), e para criar um item dessa lista é necessário a tag ***<li>***.

No parâmeto ***type*** da tag *<ul>*, você especifica qual o simbolo ficará a frente dos itens da lista.

**➜ disc – Um círculo preto preenchido**

**➜ circle – Um círculo preto sem preenchimento (padrão)**

**➜ square – Um quadrado preto preenchido.**

**Listas Mistas**

É possível misturar as listas, criando listas dentro de listas. Abaixo está um exemplo de uma lista dos meus jogos favoritos, divididor por categoria.

***Código de exemplo:***

    <ol type=*"I"*>

        <li>FPS</li>

        <ul type=*"disc"*>

            <li>CS:GO</li>

        </ul type=*"disc"*>

        <li>MOBA</li>

        <ul type=*"disc"*>

            <li>League of Legends</li>

        </ul type=*"disc"*>

        <li>Aventura</li>

        <ul type=*"disc"*>

            <li>Minecraft</li>

            <li>Terraria</li>

            <li>RimWorld</li>

        </ul type=*"disc"*>

        <li>Sair por aí fazendo oque quiser</li>

        <ul type=*"disc"*>

            <li>Grand Theft Auto: San Andreas</li>

        </ul type=*"disc"*>

        <li>Indie</li>

        <ul type=*"disc"*>

            <li>Undertale (esse é o top 1)</li>

        </ul>

        <li>Terror</li>

        <ul type=*"disc"*>

            <li>Five Nights at Freddy's 2</li>

        </ul>

    </ol>

***Resultado:***

1. FPS
   * CS:GO
2. MOBA
   * League of Legends
3. Aventura
   * Minecraft
   * Terraria
   * RimWorld
4. Sair por aí fazendo oque quiser
   * Grand Theft Auto: San Andreas
5. Indie
   * Undertale (esse é o top 1)
6. Terror
   * Five Nights at Freddy's 2

**Listas de Definições (*Definition List*)**

É como um dicionário, primeiro um termo e abaixo um texto. Toda definition list está dentro de uma tag ***<dl> </dl>*** e cada termo é definido por uma tag ***<dt>*** (*definition term*), para uma descrição é usada a tag ***<dd>*** (*definition description*).

***Código de exemplo:***

<dl>

        <dt>NRG-500</dt>

        <dd>A melhor moto</dd>

        <dt>Infernus</dt>

        <dd>O carro mais rápido</dd>

        <dt>Turismo</dt>

        <dd>Meu carro favorito, rápido e bonito!</dd>

        <dt>Elegy</dt>

        <dd>Carro perfeito para drift</dd>

        <dt>Mountain Bike</dt>

        <dd>Melhor bike de todas!</dd>

        <dt>Hydra</dt>

        <dd>Com certeza o melhor veículo aéreo</dd>

        <dt>Sanchez</dt>

    </dl>

***Resultado:***

NRG-500

A melhor moto

Infernus

O carro mais rápido

Turismo

Meu carro favorito, rápido e bonito!

Elegy

Carro perfeito para drift

Mountain Bike

Melhor bike de todas!

Hydra

Com certeza o melhor veículo aéreo

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Hyperlink’s**

Para colocar um link em seu site, é necessário utilizar a tag ***<a>***, que possui o parâmetro ***href***, que é onde deve ser identificado uma ***URL***.

    <p>

        Meu canal:

    </p>

    <a *href*="https://www.youtube.com/channel/UCs6bahfNFeK51gkuwYIvWog" *hreflang*="pt">Meu canal!</a>

Perceba que tem um outro parâmetro, o ***hreflang***, que é o idioma em que o navegador irá abrir o site.

**Abas**

Por padrão, quando clicar em um *hyperlink*, você é redirecionado a uma página, porém aberta na mesma aba, ou seja, a página anterior será substituída. Para mudar isso adicione o parâmetro ***target***, e seu valor deverá ser algum desses:

**➜ \_blank – vai abrir o link em uma nova aba**

**➜ \_self – vai abrir o link na mesma aba (padrão)**

**Tipos de Redirecionamento**

Talvez seja um link para a próxima página, ou pra anterior, ou até mesmo para o site do autor, enfim, segue uma lista abaixo. Para adicionar use o parâmetro ***rel***.

**➜ next – para a próxima página**

**➜ prev – para a página anterior**

**➜ author – para o site do autor do documento atual**

**➜ external – para um site que não faz parte do atual**

**➜ nofollow – para um site não endossado, como um link pago**

**Link Interno**

Que direciona o usuário a uma página dentro de um mesmo site. Não sendo nem mesmo necessário indicar a URL completa no parâmetro *href*. Caso a próxima página esteja dentro de uma pasta, deverá especificar seu diretório.

**➜ ./ – próxima página**

**➜ ../ – pasta anterior**

**Link Externo**

Que direciona o usuário a uma página fora do site atual. Nesse caso é necessário uma URL completa (incluindo o *protocol*) no parâmetro *href*.

**Link de Download**

Para download em um link basta adicionar mais dois parâmetros na tag <a>: ***download*** (que vai receber o nome do arquivo a ser baixado) e ***type*** (que vai receber o tipo de arquivo a ser baixado.

**Media Types**

São os tipos de arquivos para download, abaixo está uma lista com os mais comuns.

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

**MAIS:** <https://www.iana.org/assignments/media-types/media-types.xhtml>

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**IMAGENS DINÂMICAS, ÁUDIOS E VÍDEOS.**

Para que o tamanho nunca fique despoporcional em tela de diferentes tamanhos.

**Imagens Dinâmicas**

A tag **<picture>** serve para concentrar as outras fontes da imagem. Então o código fica assim:

<picture>

        <source *media*="(max-width: 1050px)" *srcset*="er\_red300.png" *type*="image/png">

        <img *src*="er\_red300.png" *alt*="Imagem png ER">

</picture>

**Tag *<source>***

Essa tag com seus atributos(vistos acima e mais informações abaixo) servem para personalizar o tamanho visível da imagem de acordo com o zoom do site.

Caso precise de mais de um <source>, coloque-os em ordem crescente, ou seja, do menor tamanho para o maior.

    <picture>

        <source *media*="(max-width: 700px)" *srcset*="er\_red700.png" *type*="image/png">

        <source *media*="(max-width: 1050px)" *srcset*="er\_red700.png" *type*="image/png">

        <img *src*="er\_red700.png" *alt*="Imagem png ER">

    </picture>

**Atributos**

**➜ *type* – *media type*.**

**➜ *scrset* – vai especificar a imagem que será carregada quando o tamanho máximo for atingido.**

**➜ *media* – indica o tamanho máximo a ser considerado para carregar a imagem indicada no atributo *scrset.***

**Áudios**

Para carregar um áudio em um site, é utilizada a tag ***<audio>***, vamos ver um exemplo de código.

**Atributos**

**➜ type – *media type*.**

**➜ scrset – vai configurar o tamanho da imagem que será carregada quando o tamanho máximo for atingido.**

**➜ *preload* – indica se o áudio será carregado ou não, o mesmo possui 3 valores:**

**➜ *metadata* – vai carregar apenas as informações sobre o arquivo (tempo, tamanho, etc).**

**➜ *none* – não vai iniciar absolutamente nada a não ser que o usuário aperte o play ou um script inicie a reprodução**

**➜ *auto* – vai carregar o arquivo de áudio inteiro assim que a página for carregada, mesmo que o usuário nunca aperte o play.**

**➜ *controls* – se não escrito, não aparecerá o player na tela.**

**➜ *autoplay* – se escrito, o áudio irá iniciar assim que a página for carregada.**

**➜*loop* – o nome é autoexplicativo.**

Dentro da tag de áudio, **adicionamos vários** ***source*** para se caso o primeiro áudio falhe, reproduza o segundo com um formato diferente (você que têm que colocar um formato diferente). No código acima, há um parágrafo com um link de download, para caso o navegador do usuário não suportar reprodução de áudio.

**Vídeos**

Pode ser hospedado em seu site ou incorporado de outro, os melhores para isso são o YouTube e o Vimeo, cada um com suas vantagens. O YouTube é rápido e grátis, já o Vimeo é pago lento ás vezes.

**Vídeos Incorporados**

Incorporar um vídeo é muito simples! Vá até o vídeo que deseja incorporar no YouTube, por exemplo, então clique em “compartilhar” e depois em “incorporar”, então copie o código HTML que vai aparecer e cole em seu editor de código.

**Vídeos Hospedados em seu Site**

Use a tag ***<video>*** para mostrar um vídeo em seu site. Aqui vai seus parâmetros.

**➜*width* – será o tamanho do seu vídeo na tela. Considere o tamanho em pixels.**

**➜*poster* – é a *thamb* do vídeo, que vai aparecer antes do usuário clicar no play.**

**➜ *controls* – se não escrito, não aparecerá o player na tela.**

**➜ *autoplay* – se escrito, o vídeo irá iniciar assim que a página for carregada.**

**➜*loop* – o nome é autoexplicativo.**

**CSS**

Existem 3 formas de aplicar *styles* em seu documento HTML: *inline*, *internal* e *external*. Cada um tem suas vantagens, mas o mais recomendado é utilizar os estilos internos e externos.

***Inline***

Essa deve ser evitada, mas vamos falar sobre mesmo assim.

Adicionando o parâmetro ***style*** em uma tag como h1, por exemplo, podemos modificá-la.

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

Neste código podemos observar que dentro do parâmetro ***style***, temos um elemento ***color***, que tem seu valor igual “darkblue”. Ou seja, a cor das letras serão da cor escolhida. Cada elemento dentro do parâmetro **deve ser fechado com ponto e vírgula**, assim podendo, após o fechamento, colocar outro elemento.

***Internal***

Esse é legal! Tudas as configurações de estilos padrões serão definidas por tudo que está entre ***<style>*** e ***</style>*** na área ***<head>.***

Veja como devemos aplicar!

    <style>

        body{

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

            color: darkred;

            background-color: rgb(203, 245, 231);

        }

        h1{

            color: darkblue;

        }

        h2{

            color: blue;

        }

    </style>

O que está definido na área ***body*** será o padrão e, caso necessário mudar uma linha específica, é necessário utilizar o ***Style Inline***.

No exemplo acima, todos os títulos de nível 1 (h1) serão da cor azul-escuro, e todos os títulos de nível 2 (h2) serão da cor azul.

***External***

Esse é o mais recomendado e o melhor em caso de códigos grandes, pois o objetivo é separar os estilos do código HTML em si.

**Como criar um código apenas para os estilos?**

É bem simples! Basta adicionar uma nova tag na área ***<head>*** chamada ***<link>*** (Essa tag não tem fechamento!).

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

Ela por padrão é isso aí, o que varia é apenas o valor do parâmetro *href*, que deve ser o arquivo CSS que está armazenando os estilos.

**Dica: caso segure CTRL e clique em cima do valor de href, ou seja, o nome do arquivo, você pode criar o arquivo automaticamente.**

**Estrutura de um Arquivo CSS**

@charset "UTF-8";

/\* Para não haver nenhuma incompatibilidade com determinados caracteres \*/

body {

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

    font-size: 12pt;

    color: darkmagenta;

    background-color: gray;

}

h1 {

    color: darkred;

}

h2 {

    color: darkblue;

}

A primeira linha é pra que não haja problemas com acentos. Já a segunda, é um comentário.

**Dica: tudo que está entre /\* \*/ é considerado comentário!**

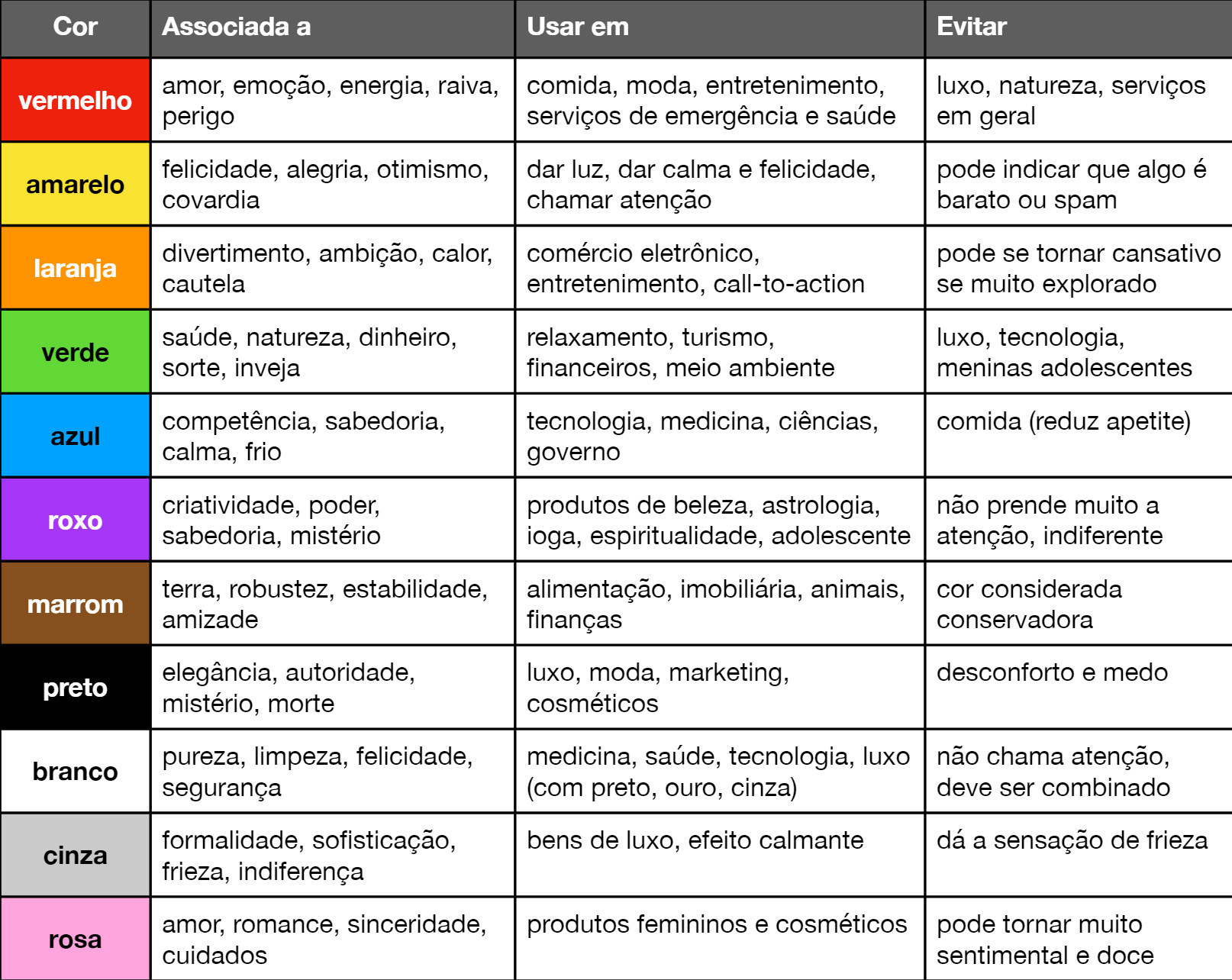
**Mais de um *link:css***

Você pode colocar mais de um **link:css** em um arquivo HTML, ele irá somar todos, e em caso de duplicidade, como valores diferentes em uma mesma propriedade, o **link:css** que estiver mais abaixo terá prioridade.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CORES**

**Psicologia das Cores**

Considere usar cores de acordo com a tabela abaixo, mas não se prende completamente a isso!



**Paleta de Cores**

**O que é uma paleta de cores?**

São as cores que compõem alguma coisa, seja um site, uma revista, uma embalagem de um produto.

Geralmente essas cores trabalham em harmonia, combinam entre si de certa maneira.

**Quantas cores em uma paleta?**

É recomendado que em uma paleta de cores haja no mínimo três cores e no máximo cinco cores, mas geralmente **é mais utilizado paletas com quatro cores**, ou seja, o meio termo entre três e cinco.

**Cores Complementares**

São as cores que mais contrastam entre si, ficando opostas em um círculo cromático.

**Cores Análogas**

São as cores que, ao contrário das complementares, possuem uma harmonia justamente por não haver muito contraste entre elas. Cores análogas a uma cor específica ficam próximas a essa cor, se localizando exatamente ao lado em um círculo cromático.

**Cores Análogas e uma Complementar**

É interessante adicionar a cor complementar da cor principal de uma paleta com cores análogas, criando assim um contraste caso necessário.

**Cores Análogas Relacionadas**

Caso haja a necessidade de criar uma paleta com cores harmônicas e uma não muito contrastante, basta escolher a cor principal em um círculo cromático e ao lado escolher uma outra, a cor de contraste será a depois da próxima, ou seja, irá pular uma cor e escolher a próxima. Isso para qualquer um dos lados, mesmo que tenha escolhido uma segunda cor para esquerda, pode escolher uma outra para a direita.

**Cores Intercaladas**

São cores que possuem um pouco de contraste entre elas e no círculo cromático se encontram saltadas entre uma cor, ou seja, pegue uma cor, salte uma cor e escolha a próxima. Abaixo alguns tipos de cores intercaladas.

**Cores Triádicas**

É um tipo de classificação para as Cores Intercaladas. Para formar uma paleta com cores triádicas escolha uma cor principal em um círculo cromático, depois salte três cores para qualquer lado e a quarta cor será parte da paleta, repita o processo e descubra a terceira cor para a paleta.

Um exemplo de cores triádicas são as Cores Primárias, em um círculo cromático elas formam um triângulo equilátero, como qualquer paleta com cores triádicas em um círculo cromático.

**Quadrado harmônico**

É parecido com as cores triádicas, mas ao invés de saltar três cores, saltar apenas duas. O resultado será uma paleta de quatro cores fortes com um certo contraste entre elas.

**Cores Tetrádicas**

Para formar uma paleta com cores tetrádicas, faça o seguinte, escolha uma cor no círculo cromático e então pegue sua cor complementar, em seguida faça a mesma coisa para uma outra cor, o resultado será uma paleta com quatro cores e sua harmonia depende de quais cores você escolheu.

**Monocromia**

Uma paleta monocromática é uma paleta que possui apenas uma cor como base, e suas outras cores dessa paleta são variações da cor escolhida, mudando a saturação e a luminosidade. O resultado é uma paleta de cores bem harmônica com um estilo degradê.

**Dica: para saber quais cores possuem harmonia entre si, utilize o site abaixo.**

**Adobe Color:** <https://color.adobe.com/pt/create/color-wheel>

**Gradientes em CSS**

Os gradientes são considerados, pelos navegadores, uma imagem.

<style>

        body{

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

            background-image: linear-gradient(90deg, purple, red);

            color: black;

        }

    </style>

    <style>

        body{

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

            background-image: radial-gradient(circle, purple, red);

            color: black;

        }

    </style>

Em ambos os códigos o número de cores é opcional, podendo ser quantas você quiser.

No caso do ***linear-gradient***, primeiro valor é em graus, podendo ser negativo. Você também pode indicar um direção: **to bottom**, **to right**, **to left**, **to top**.

Já no caso do ***radial-gradient***, não existe um valor em graus, é substituído por ***circle***.

**Espaço de Cada Cor**

Você pode definir o espaço de uma cor em um gradiente colocando um número com porcentagem ao final do código da cor, só então colocando a vírgula ou dois pontos para ir para próxima cor ou finalizar a linha.

Veja um exemplo abaixo:

body{

            font: normal 1.6em Arial, Helvetica, sans-serif;

            background-image: linear-gradient(to right, #DB08FF 10%, #AF02E3, #8802E3, #AF0EFB, #8103FB);

            color: black;

        }

**FONTES**

**Família Tipográfica**

É o conjunto de glifos que possuem as mesmas características anatômicas, como a família *Open Sans*.

Determinação

Determinação

O que muda é a configuração de peso, a primeira com 300 de peso (Open Sans Light) e a segunda com 400 de peso (Open Sans Regular).

**Fontes**

São conjuntos de glifos que formam uma família tipográfica.

**Fontes Serifadas**

Essas são muito boas para leitura, as letras parecem se juntar uma à outra e facilita o entendimento do que está escrito.

**Fontes Não Serifadas**

Elas transmitem uma sensação de limpeza, são mais leves para a leitura e perfeitas para exibição em telas.

**Fontes monoespaçadas**

São as fontes que toda letra tem o mesmo espaço, por exemplo, um “i” vai ocupar o mesmo espaço que um “m”. Ótimas para leitura.

**Fontes Scripts**

São aquelas fontes que parecem ser escritas como desenhos.

**Fontes Display**

São as diferenciadas, aquelas que não possuem letras como base, sendo quadrinhos, emotes, etc, ou que são muito decoradas. No geral, são aquelas que não se encaixam nas classificações vistas acimas.

**Aplicar em CSS**

Para facilitar, pode-se utilizar, ao invés de *font-family*, *font-size*, etc, apenas ***font***. Os valores devem ser aplicados na seguinte ordem:

1. font-style
2. font-variant
3. font-weight
4. font-size / line-weight
5. font-family

<style>

body{

font: italic bold 1em Arial, Helvetica, sans-serif;

}

</style>

1***em*** equivale a o tamanho da fonte padrão, geralmente 16px, então 2***em*** é igual a o dobro de tamanho.

**Fontes Externas**

É possível usar fontes externas por meio de seus arquivos de fonte, eles são baixados por meio de site, o *dafont.com* é um exemplo.

Existem dois tipos de arquivo de fonte: **otf** e **ttf***.*

A diferença entre elas é a compatibilidade com os navegares.

Para aplicar em algum seletor é necessário criar uma regra, bem ao começo do style ou do documento CSS digite:

@font-face {

            font-family: ;

            src: url(“ ”) format();

        }

Na área font-family escreva o nome que deseja para a fonte, por esse nome ela será referenciada.

Na área url deve está escrito o diretório do arquivo de fonte.

Na área format escreva o tipo de formato de fonte, os tipos e suas usabilidades estão abaixo.

É possível colocar vários url, mas cada um deve vir acompanhado de um format e cada conjunto de url e format devem ser separados por vírgula. Isso é útil para garantir que o site seja compatível com vários navegadores.

**Tipos de fontes**

**➜ *opentype* –** para fontes **otf** (Algumas fontes opentype são **ttf**).

**➜ *truetype* –** para fontes **ttf**

**➜ *embedded-opentype***

**➜ *truetype-att* –** Apple Advanced Typography

**➜ *svg***

**Alinhamento de Texto**

Use ***text-align***, seu valor deve ser *center*, *left*, *right* ou *justify* (padrão).

Para espaçamento no início do parágrafo use **text-indent**: 30px, sendo o número 30 o tamanho do espaço, no caso em pixels.

**SELETORES EM CSS**

Servem para selecionar e modificar uma única área, ou um grupo inteiro.

Para selecionar apenas uma área, como um parágrafo ou título específico, deve colocar o parâmetro ***id*** dentro da tag, que significa identificador. O valor dessa tag deve ser um nome de referência e para modificá-la em CSS deve ser colocado um ***#*** antes de seu nome.

É possível colocar apenas **#** + **nome** ou **seletor** + **#** + **nome**, dessa forma apenas os seletores iguais ao designado sofrerão alteração.

p*#link1*{

    text-align: center;

}

Para selecionar um grupo adicione o parâmetro class dentro da tag, seu valor deve ser um nome de referência para o grupo. Para modificá-la em CSS deve ser colocado um **ponto (.)** antes de seu nome.

É possível colocar mais de um grupo para um único objeto, por exemplo um h1 como título principal e ao mesmo tempo destacado – tendo uma class para título principal e outra para destaque – para que receba as características de dois seletores ao mesmo tempo, para isso basta colocar as duas classes separadas por espaço.

<h1 *class*="tituloPrincipal destaque">Hello, World!</h1>

Um único elemento pode ser referenciado por uma class e ao mesmo tempo um id, saiba que o id sobrepõe a class, tendo uma prioridade.

Abaixo um exemplo de código com seletores personalizados.

    <style>

        body{

            background-color: darkslategray;

            color: whitesmoke;

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

            font-size: 1em;

        }

*#introdução*{

            color: darkmagenta;

            font-family: 'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;

        }

*.titulos*{

            color: blue;

            background-color: aqua;

            text-align: center;

            font-size: 1.5em;

            font-family: Verdana, Geneva, Tahoma, sans-serif;

        }

    </style>

</head>

<body>

    <h1>Capítulo 1</h1>

    <h1 *id*="introdução">Introdução</h1>

    <p *id*="paragrafo 1">

        Lorem ipsum dolor sit, amet consectetur adipisicing elit. Officia, debitis nemo doloremque maiores vero ullam! Sed doloremque illum nemo tenetur, amet vitae iure dolor? Quod exercitationem provident laborum voluptates animi!

    </p>

    <h1 *class*="titulos">Capítulo 2</h1>

    <p *class*="paragrafos">

        Lorem ipsum dolor sit amet consectetur adipisicing elit. Nostrum vel repellat quod modi cum eaque architecto natus asperiores, illo voluptatibus. Vitae doloremque corporis provident accusantium praesentium officiis eaque doloribus mollitia!

    </p>

    <h1 *class*="titulos">Capítulo 3</h1>

    <p *class*="paragrafos">

        Lorem, ipsum dolor sit amet consectetur adipisicing elit. Mollitia temporibus unde voluptate sapiente optio dignissimos doloremque modi quo suscipit hic ipsum eaque sint debitis consectetur dolores minus, sequi consequatur reiciendis.

    </p>

</body>

**Pseudos**

**Pseudo-classes**

Servem para especificar um estado especial de um elemento, como ***:hover***, ***:visited***, ***:active***, ***:checked***, ***:empty***, ***:focus***.

**Pseudo-elementos**

Servem para formatar um pedaço específico do elemento referenciado. Como ***::before***, ***::after***, ***::first-letter***, ***::first-line***.

**Div**

Uma grande vantagem do div é poder colocar tags dentro dele.

<body>

    <a *href*="index.html">Página anterior</a>

    <div>

        Passe o mause aqui e veja um segredo (não muito bem escondido)!

        <p><a *id*="link1" *href*="#" *rel*="external">SURPRESAAAAAA, EU VOLTEI!</a></p>

    </div>

</body>

    <style>

        body{

            font: normal 1.5em Arial, Helvetica, sans-serif;

            background-color: darkslategray;

        }

        div > p {

            display: none;

        }

        div*:hover* > p {

            display: block;

            background-color: brown;

        }

*#link1*{

            text-decoration: none;

            color: black;

            background-color: darkgray;

        }

    </style>

**Clildren**

Quando precisar adicionar estilos apenas em elementos específicos dentro de outros elementos, crie um seletor como visto na oitava linha do código acima. O símbolo **>** quer dizer que o p é filho do div:hover, no exemplo acima, e apenas o p será modificado quando a condição do div:hover for realizada.

**CAIXAS (*BOX*)**

As caixas são a grande maioria dos objetos em HTML, como títulos e parágrafos, elas armazenam conteúdos dentro deles, até mesmo outras caixas.

**Estrutura de uma Caixa**

**➜ *content* –** é o **conteúdo**, sendo o centro da caixa.

**➜ padding–** é o **preenchimento** que fica entre o conteúdo e a borda.

**➜ *border* –** é a **borda** e vem em seguida do preenchimento.

**➜ *outline* –** é o **contorno**, visualmente um traçado visual. É pouco utilizado, mas ele existe.

➜ ***margin***– é a **margem**, a parte de fora da caixa.

    <style>

        h1{

            width: 300px;

            height: 50px;

            background-color: gray;

            border-width: 10px;

            border-style: solid;

            border-color: darkgreen;

            padding: 20px;

            outline-width: 30px;

            outline-style: solid;

            outline-color: blue;

            margin: 50px;

        }

    </style>

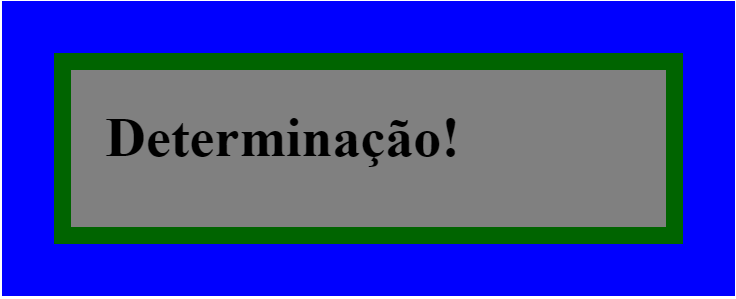
</head>

<body>

    <h1>Determinação!</h1>

</body>

***Resultado:***

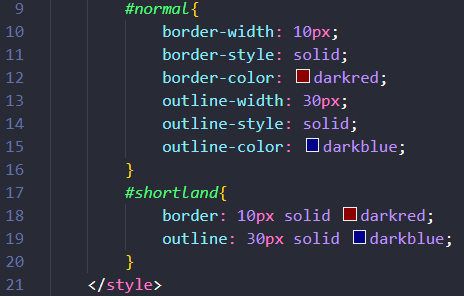


**Tamanho total**

Para calcular o tamanho total de um elemento na tela é preciso somar: **conteúdo + preenchimento + borda + margem**. O contorno não entra nessa conta pois utiliza parte da medida da margem.

**Simplificando...**

Dá pra simplificar escrevendo menos, bora pro antes e depois!



A ordem é sempre a mesma: ***width***, ***style*** e ***color***.

Para ***padding*** e ***margin*** personalizados de forma simples:



A ordem é semple ***top***, ***right***, ***bottom*** e ***left*** em ambos.

**Centralizar**

Para centralizar utilize a declaração ***auto*** em seu seletor.

margin: auto;

Assim o navegador irá calcular automaticamente o meio para a centralização.

**Tipos de Caixas**

Existem duas categorias de caixas, classificadas de acordo com seu comportamento.

***block-level***

Um elemento block-level sempre se inicia em uma nova linha e vai ocupar a largura total do elemento onde está contido. Se não estiver contido em nenhuma outra caixa, vai ocupar a largura total do *<body>*.

Abaixo uma lista de elementos *block-level*.

<address> <articles> <aside> <blockquote> <canvas> <dd> <div> <dl> <dt> <fieldset> <figcaption> <figure> <footer> <form> <h1> - <h6> <header> <hr> <li> <main> <nav> <noscript> <ol> <p> <pre> <section> <table> <tfoot> <ul> <video>

***inline-level***

Irá iniciar do ponto exato aonde estava definido, ao contrário do *block-level*. Sua largura irá ocupar o espaço relativo ao seu conteúdo.

Abaixo uma lista de elementos *inline-level*.

<a> <abbr> <acronym> <b> <bdo> <br> <button> <cite> <code> <dfn> <em> <i> <img> <input> <kbd> <label> <map> <object> <output> <q> <samp> <script> <select> <small> <span> <strong> <sub> <textarea> <tt> <var>

**User Agent**

*User agent* é o navegador trabalhando de forma automática para exibir um conteúdo.

**Grouping Tags e Semantic Tags**

Por padrão, existem duas tags de agrupamento genérico, a <div> e a <span>, sendo a primeira para tags block-level e a segunda para as tags inline-level.

Hoje existem tags de agrupamento semânticas, são elas:

**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.

**Main (Gnar)**

É um agrupamento usado para delimitar o conteúdo principal de um site. Geralmente 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 conteúdos específicos. “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**

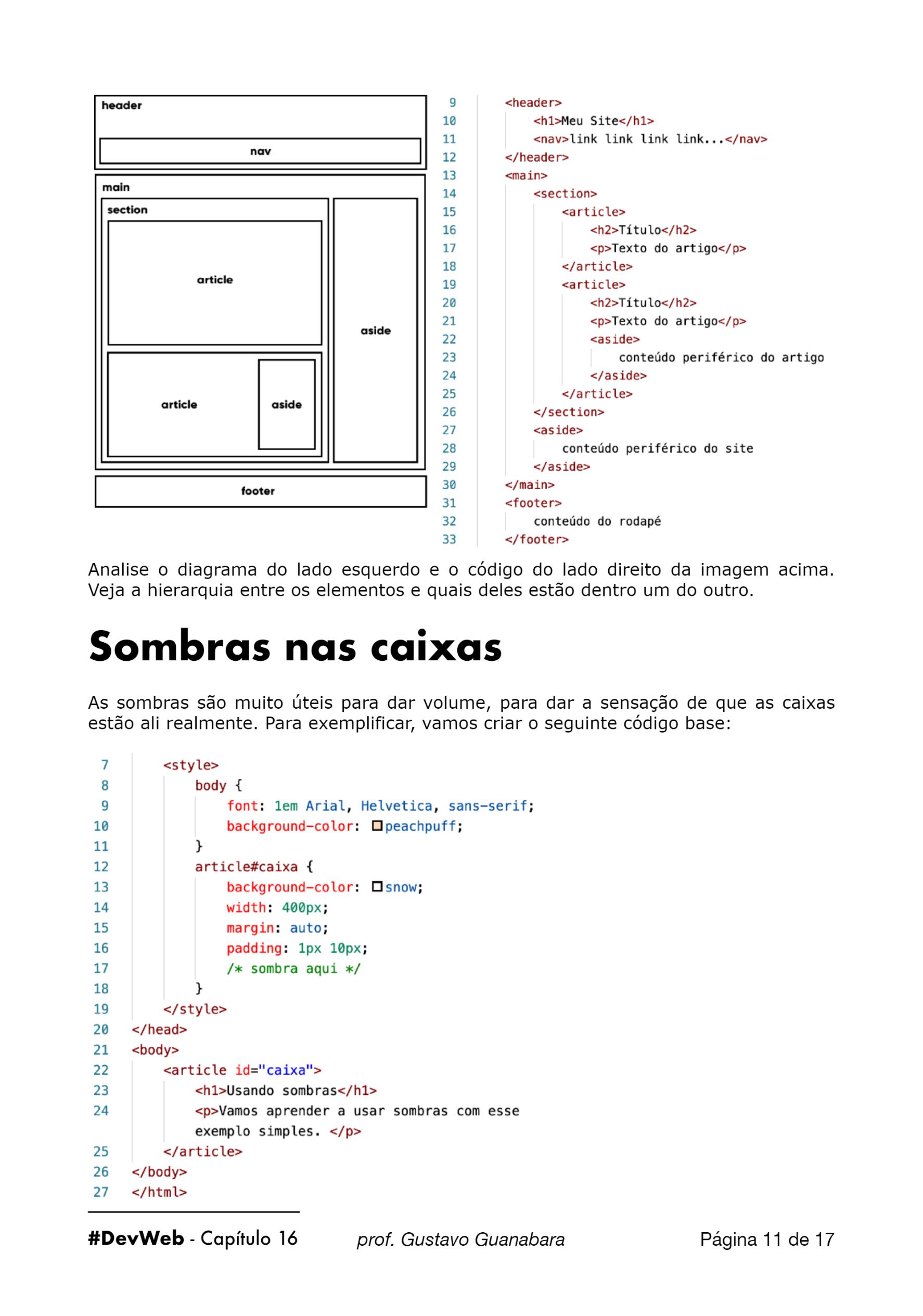
É usado para definir um bloco de conteúdo que complementa o conteúdo principal, mas pode ser removido sem afetar significativamente o significado do conteúdo principal.

**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, mas possui informações sobre autoria do conteúdo, links adicionais, mapa do site, documentos relacionados.

**Múltiplos Níveis**

É possível ter um ou mais <article> dentro de uma <section> ou até mesmo criar <section> dentro de um <article>. Não existem limitações quanto a isso.



**Sombras nas Caixas**

São úteis para dar volume, dando uma sensação de que as caixas realmente estão ali. Um exemplo abaixo:

article*#caixa*{

            background-color: snow;

            width: 400px;

            margin: auto;

            padding: 1px 10px;

            /\* sombra aqui \*/

            box-shadow: 3px 3px 10px black;

        }

É possível deixar a borda mais arredondada utilizando o elemento border-radios.

article*#caixa*{

            background-color: snow;

            width: 400px;

            margin: auto;

            padding: 1px 10px;

            border-radius: 10px 25px;

            /\* sombra aqui \*/

            box-shadow: 3px 3px 10px black;

        }

**Estilo da Borda**

Os tamanhos são top, right, bottom, left, respectivamente, ou você pode colocar apenas um valor, assim o mesmo servirá para todos, ou até mesmo dois valores, o primeiro para top e bottom e o segundo para right e left.

article*#caixa*{

            background-color: snow;

            width: 400px;

            margin: auto;

            padding: 1px 10px;

            border-radius: 10px 25px;

            /\* sombra aqui \*/

            box-shadow: 3px 3px 10px black;

        }

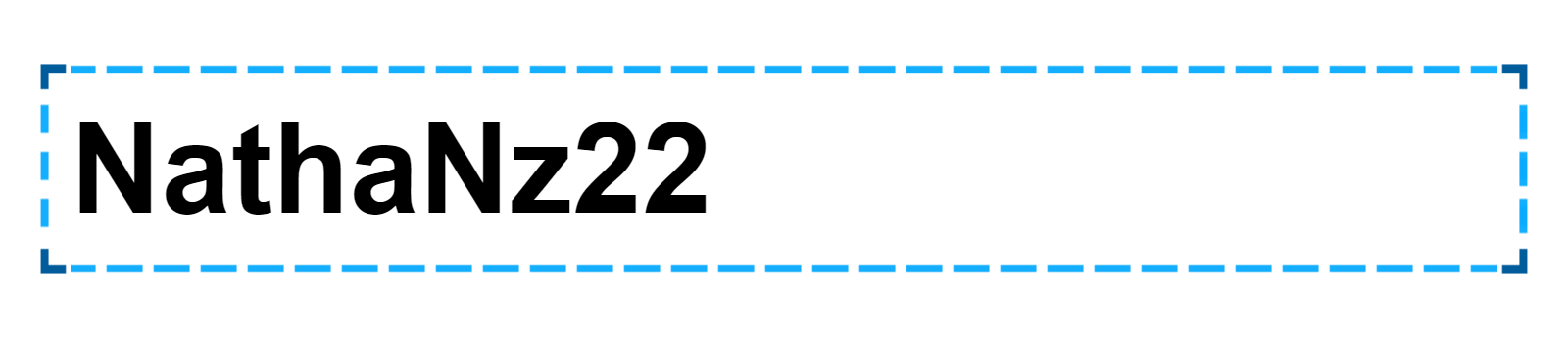
**Borda Personalizada**

Para usar uma imagem do seu computador (ela precisa ser uma imagem feita para ser um borda, contendo todos os lados, tamanho 80px x 80px) como borda, é necessário criar uma borda no seletor que vai receber a borda personalizada, a nova borda irá sobrescrever a criada.

border: 10px solid black;

border-image: url('borda.png') 25 repeat;

O resultado é esse:



No código acima está representado em shorthand, e seus valores são border-image-source, border-image-slice, border-image-repeat, respectivamente.

No **border-image-source** deve ser especificado o diretório da borda no computador, use:

Lembre-se que o nome do arquivo deve ser escrito entre aspas, contendo também o seu formato.

No **border-image-slice** deve ser especificado um valor numérico, referente ao fatiamento da borda.



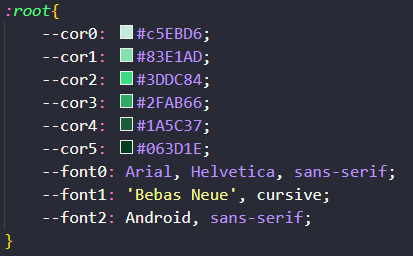
No **border-image-repeat** deve ser especificado a forma em que a borda vai aparecer na tela, existem duas opções: **repeat**, **stretch** e **round**.



**Variáveis em CSS**

Para criar variáveis em CSS, é necessário criar uma seletor especial chamado :root{}, e entre as chaves devemos criar todas as variáveis que vamos utilizar no documento.

Toda variável possui dois hífens em sequência **--** no começo, após os hífens vêm o nome da variável, o valor após os dois pontos será o valor dessa variável, que pode ser uma cor, uma fonte, etc.



Para usar uma dessas variáveis é necessário colocar **var(--variavel)**.



**Posições em CSS**

O posicionamento de um elemento em CSS em definido pela propriedade position, que pode ter 5 valores diferentes. São eles:

**static**

O posicionamento padrão de todo elemento HTML é static, que é o alinhamento ao canto superior esquerdo do corpo do documento.

**fixed**

Seu nome é auto-explicativo, o valor fixed faz com que o elemento não se mova na tela. Ótimo para alguns tipos de nav e para aquelas propagandas chatas que não saem mesmo quando você rola a tela.

**sticky**

Sua tradução seria “colado” ou “pegajoso”, esse valor é parecido com o fixed, porém ele fica fixo em relação ao rolamento da página, e não da tela.

**relative**

É usado para alterar a posição de um elemento tendo como base a posição inicial dele. Apenas aplicar esse valor a um elemento não o modificará em nada a sua posição, para isso é necessário aplicar o **top** e **left** para alterar sua posição de cima para baixo e esquerda para direita, tendo como referência a sua posição inicial.

**absolute**

O posicionamento absoluto possui dois comportamentos diferentes. O primeiro é quando o elemento com esse valor possui um elemento pai de valor diferente de static, usando esse elemento pai como referência para ser posicionado.

O segundo comportamento é quando o elemento absoluto não possui pai, ou o elemento pai é possui o valor position: static. Nesse caso ele irá ser posicionado no canto superior esquerdo do documento, podendo até mesmo sobrepor a eles.

Em resumo, no primeiro comportamento o elemento absoluto tem um pai e o usa como base para ser posicionado. Já no segundo comportamento ele não possui um elemento pai ou o elemento pai possui valor static, e dessa forma ele usa o corpo do documento como base para seu posicionamento.

**Alinhamento Vertical**

Para alinhar horizontalmente é bem simples, apenas adicionar a linha ‘margin: auto;’ no estilo do elemento e pronto. Porém, é um pouco mais complicado para alinhar na vertical.

O elemento pai precisa estar como relative e o elemento a ser alinhado deve estar como absolute.

Com o elemento possuindo a propriedade ‘position: absolute;’, é possível configurar o top e left, posicionando o conteúdo.

#container {

            position: relative;

        }

#content {

            position: absolute;

}

Com isso basta colocar os valores top e left em 50% e o conteúdo já estará centralizado, porém o ponto de ancoragem ainda será o canto superior esquerdo, e visualmente o conteúdo ainda estará descentralizado.

#content {

            position: absolute;

top: 50%;

left: 50%;

}

Para corrigir isso é necessário transformar o elemento de conteúdo com a propriedade:

transform: translate(-50%, -50%);

**Transformações (transform)**

Com as transformações é possível alterar a posição, tamanho, rotação e perspectiva de elementos, oferecendo maneiras de manipular elementos no espaço 2D e 3D.

A propriedade **transform** suporta vários valores:

1. **‘translate()’**: Move um elemento em uma determinada direção. Devem ser especificadas as coordenadas X e Y. Por exemplo:

transform: translate(50px, 100px);

Com isso, moverá o elemento 50 pixels para a direita e 100 pixels para baixo. Para girar para a esquerda e para cima use valores negativos.

1. **‘rotate()’**: Gira um elemento em torno de um ponto de referência específico, definindo o ângulo de rotação em graus. Por exemplo:

transform: rotate(45deg);

Com isso, irá girar o elemento em 45 graus no sentido horário.

1. **‘scale()’**: Altera o tamanho de um elemento, especificando valores para escala horizontal e vertical. Por exemplo:

transform: scale(1, 2);

Com isso, aumentará o elemento em 1,5 vezes no eixo X e 2 vezes no eixo Y.

1. **‘skew()’**: Aplica uma inclinação a um elemento, especificando valores para os eixos X e/ou Y. Ela permite distorcer a forma do elemento em um ângulo específico. Por exemplo:

transform: skew(10deg, -20deg);

1. **‘matrix()’**: Define transformações 2D personalizadas usando uma matriz 4x4. Os valores da matriz definem as transformações de translação, rotação, escala e inclinação. Por exemplo:

transform: matrix(1, 0.5, -0.5, 1, 0, 0);

Isso irá aplicar uma transformação personalizada ao elemento.

É possível aplicar vários dos valores citados acima em apenas uma declaração. Por exemplo:

transform: translate(-50px, -50px) rotate(45deg) scale(2.5, 3);

**Perspectiva**

A propriedade **‘perspective’** deve ser aplicada no elemento pai e trabalha em conjunto com as transformações 3D aplicadas ao elemento filho para criar a ilusão de profundidade e tridimensionalidade em elementos.

Para aplicá-la:

.elemento-pai {

            perspective: valor;

}

O valor pode ser especificado em pixels (como por exemplo: ‘perspective: 1000px;’) ou em outras unidades de comprimento. O valor pode também ser negativo, mas é pouco comum.

A combinação é feita com as funções usadas na propriedade transform:

1. **‘translate3d()’**: É como o translate, mas devem ser especificados o **valor de Z**, além dos valores X e Y.

transform: translate3d(100px, -20%, 50px);

No exemplo acima, o elemento será movido 100 pixels para a direita ao longo do eixo X, 20% para cima ao longo do eixo Y e 50pixels para frente ao longo do eixo Z.

1. **‘rotate3d()’**: Permite realizar rotações em três dimensões. Devem ser especificados os seguintes valores em ordem: <eixoX>, <eixoY>, <eixoZ>, <ângulo>.

Os eixos representam os componentes do vetor de eixo em torno do qual a rotação será aplicada. Esses valores podem variar de -1 a 1 e definem a direção do eixo.

O ângulo representa o ângulo de rotação em graus.

transform: rotate3d(1, 0, 0, -20deg);

Nesse exemplo, o elemento será rotacionado em torno do eixo X em 45 graus. O valor ‘1’ indica que o eixo de rotação está alinhado com o eixo X.

1. **‘scale3d()’**: É como o scale, mas devem ser especificados o **valor de Z**, além dos valores X e Y.

transform: scale3d(2, 1.5, 0.5);

No exemplo acima, o elemento será escalado para o dobro do tamanho ao longo do eixo X, 1.5 vezes o tamanho original ao longo do eixo Y e reduzido pela metado do tamanho original ao longo do eixo Z.

1. **‘matrix3d()’**: Essa função permite especificar diretamente os 16 valores da matrix para obter o efeito de transformação desejado.

Uma matriz 4x4 é uma representação matemática que combina diferentes tipos de transformações, como translação, rotação, escala e cisalhamento em um único valor.

.elemento {

            transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 100, 200, 300, 1);

}

Os valores de 1 a 16 representam os elementos da matriz 4x4.

A matrix3d é como uma shorthand, mas é muito complexa. É muito mais fácil usar translate3d, rotate3d e scale3d.

**Imagens de Fundo**

Para aplicar imagens no fundo de uma página é utilizado a propriedade CSS **background-image** junto a função **url()**.

Dentro da função url() deve ser especificado um diretório ou link para localizar e aplicar uma imagem.



**Repetições**

A propriedade **background-repeat** diz como as imagens devem repetir. Existem 4 valores:

1. **‘repeat’**: valor padrão, que repete a imagem nos eixos x e y para que ocupem toda a tela.
2. **‘no-repeat’**: não repete a imagem.
3. **‘repeat**-**x’**: repete a imagem apenas no eixo x.
4. **‘repeat-y’**: repete a imagem apenas no eixo y.

**Posições**

A propriedade **background-position** define a posição do ponto de ancoragem da imagem. Devem ser especificados dois valores escritos, o primeiro da coluna e o segundo da linha.

A coluna é definida por left, center ou right, já a linha é definida por top, center ou bottom.

O código é feito de acordo com a necessidade, como nos exemplos abaixo:









**Tamanhos da Imagem**

A propriedade background-size define o tamanho da imagem de fundo, aceitando valores com medidas ou alguns valores escritos, sendo eles:

1. **‘auto’**: É o valor padrão, onde a imagem é exibida em seu tamanho original.
2. **‘cover’**: A imagem é redimensionada para que a sua largura ou altura cubra completamente o espaço, mantendo a proporção e podendo haver recortes na imagem para se adequar ao tamanho do elemento.
3. **‘contein’**: A imagem é redimensionada proporcionalmente para que encaixe completamente dentro do elemento, mantendo a proporção e não havendo corte da imagem, mas podendo haver espaços vazios se a imagem não preencher todo o elemento.

**Fixação de Fundo**

A propriedade background-attachment define se a imagem rola com o conteúdo ou se a mesma fica fixa em relação a janela de visualização.

Os valores possíveis são:

1. **‘scroll’**: É o valor padrão. A imagem rola junto ao conteúdo.
2. **‘fixed’**: A imagem permanece fixa mesmo quando o usuário rola a página.
3. **‘local’**: A imagem rola com o conteúdo do elemento em que está aplicada.

**Shorthand**

Os valores para a propriedade background, que é uma shorthand, deve ser escritos na seguinte ordem:

1. background-color;
2. background-image;
3. background-position;
4. background-repeat;
5. [background-size];
6. background-attachment;

IMPORTANTE: O valor correspondente à propriedade background-size não deve ser escrito na shorthand! Na documentação oficial está que esse valor pode ser escrito, porém, ele nem mesmo funciona, então escreva-o separado da shorthand.

background: black url("images/wallpaper002.jpg") center center no-repeat fixed;

background-size: cover;

**Medidas em CSS**

São usadas para definir o tamanho dos elementos e existem várias delas. É recomendado o uso de **em** e **px**, além de **%** e **vh**, que são para situações específicas.

1. **‘px’ (pixels)**: é a mais comum e representa um pixel na tela.
2. **‘em’**: é relativo à altura do elemento pai. Por exemplo, se o tamanho da fonte do elemento pai for de 16 pixels por padrão, ‘font-size: 1.5em;’ definirá o tamanho como 24 pixels(1.5 vezes o tamanho da fonte do pai).
3. **‘rem’**:relativo ao tamanho da fonte do elemento raiz, definido em html.
4. **% (porcentagem)**: é relativo ao valor de referência.
5. **‘vw’ (viewport width)**: é relativo a *largura* da janela de visualização (viewport).
6. **‘vh’ (viewport width)**: é relativo à *altura* da janela de visualização (viewport).

**Tabelas em HTML**

A anatomia de uma tabela é a seguinte: **Table**, **Table** **Row**, **Table** **Header**, **Table** **Data**.

1. **table**: Contém todos os elementos de uma tabela dentro de si.
2. **table row**: É relativo a linha da tabela. Sua tag é <tr>.
3. **table header**: É relativo a um cabeçalho de uma tabela. Sua tag é <th>.
4. **table data**: É relativo a um dado de uma tabela. Sua tag é <td>.

Exemplo de uma estrutura de tabela:

<table>

    <tr>

        <td>A1</td>

        <td>B1</td>

        <td>C1</td>

    </tr>

    <tr>

        <td>A2</td>

        <td>B2</td>

        <td>C2</td>

    </tr>

    <tr>

        <td>A3</td>

        <td>B3</td>

        <td>C3</td>

    </tr>

    <tr>

        <td>A4</td>

        <td>B4</td>

        <td>C4</td>

    </tr>

</table>

**Borda na Tabela**

Para criar uma borda em uma tabela, use a propriedade **border;** em CSS.

Para as linha ficarem coladas umas às outras, use a propriedade border-collapse;.

Seus valores são separate e collapse, para separar e juntar, respectivamente.

table{

    border-collapse: collapse;

}

Lembre-se que essa propriedade deve ser adicionada à tabela diretamente, como no exemplo acima.

**Alinhamento**

**Alinhamento Vertical de um Dado**

Para alinhar verticalmente um dado de uma tabela, use a propriedade **vertical-align;** em CSS. Seus valores são: top (em cima), middle(no meio) e bottom(em baixo).

**Alinhamento Horizontal de um Dado**

Para alinhar horizontalmente use a propriedade **text-align;**, seus valores são **left**(esquerda), **center**(centro) e **right**(direita).

**Tabelas Semânticas**

Pode-se utilizar 3 tags de agrupamento de tabela diferentes:

1. **thead**: É a “cabeça” da tabela. Geralmente é onde ficam os cabeçalhos de uma tabela.
2. **tbody**: É o “corpo” da tabela. Geralmente é onde ficam as linhas e os dados de uma tabela.
3. **tfoot**: É o “pé” da tabela. É onde fica o rodapé da tabela.

Em todas elas é possível usar as tags **<th>**, **<tr>** e **<td>**.

**Legenda**

Existe a tag **<caption>**, que cria uma legenda acima de da tabela. No escopo desta tag deve ser escrito apenas o texto da legenda.

**Escopo do <th>**

É importante especificar aos mecanismos de pesquisas se os valores relativos aos cabeçalhos estão abaixo ou ao lado. Para isso existe um parâmetro chamado scope que possui os seguintes valores:

1. **col**: Quando os valores relativos à linha estão abaixo.
2. **row**: Quando os valores relativos à linha estão ao lado.
3. **colgroup**: Quando os valores relativos à linha estão em um grupo abaixo.
4. **rowgroup:** Quando os valores relativos à linha estão em um grupo ao lado.

**Efeito Zebrado**

Para criar um efeito zebrado para organizar melhor a tabela é necessário utilizar uma pseuclasse:

tbody > tr:nth-child(even){

    background-color: cor;

}

A pseudoclasse nth-child pode receber como valor **2n**, sendo o “2” o número de linhas a ser puladas, como também pode receber os valores **odd**(ímpar) e **even**(par).

**Cabeçalho Fixado**

Para manter o cabeçalho de uma tabela fixado mesmo ao rolar a tela transforme a tabela para posição relativa e o cabeçalho para posição “colável” (**sticky**), depois adicione a propriedade:

top: 0px;

**Espaçamento das Células**

Cada dado de uma tabela é uma célula, e podemos fazer uma célula ocupar mais de uma coluna e/ou linha utilizando os parâmetros **colspan** e **rowspan**.

**colspan**

O valor dado ao parâmetro **colspan** definirá quantas colunas uma célula vai ocupar. O valor deve ser numérico.

<td colspan=”3”>A</td>

No código acima, a célula contendo a letra A será redimensionada para ocupar três linhas.

**rowspan**

O valor dado ao parâmetro **rowspan** definirá quantas colunas uma célula vai ocupar. O valor também deve ser numérico.

<td rowspan=”2”>A</td>

No exemplo acima, a célula que contém a letra A será redimensionada para ocupar o lugar de duas linhas.

**Isolamento por Coluna**

Para fazer uma alteração pontual em uma coluna específica, existe a tag **<colgroup>** em HTML. Essa tag ajuda a denominar cada coluna como única, facilitando bastante.

Dentro dessa tag, existem as tags **<col>**, que são cada coluna. Veja um exemplo:

<colgroup>

       <col class="cnome">

       <col class="csexo">

       <col class="cidade">

       <col class="cprofissao">

</colgroup>

Note que a tag **<col>** não precisa de fechamento.

Com o parâmetro span, é possível especificar mais de uma coluna, dependendo do valor especificado.

Exemplo:

<col class="amarelo" span="4">

Quando for alterar o estilo em CSS da classe “amarelo”, quatro colunas serão modificadas.

**Tabelas Responsivas**

Quando uma tabela é muito grande, é criada uma rolagem lateral, é possível fazer com que essa rolagem seja dentro da tabela, não afetando o resto do conteúdo da página.

Para isso, a tabela precisa estar dentro de uma **<div>**, que receberá em CSS a propriedade **overflow-x**, com o valor **scroll** ou **auto**.

Sobre o overflow, existe também o **overflow-y,** que configura apenas o eixo Y, e o próprio **overflow**, que serve para configurar tanto o eixo Y como o eixo Z.

**iframes**

Os **iframes** permitem incorporar um documento ou página web dentro de outra página. A sigla “iframe” significa “**inline frame”**, ou quadro embutido, em tradução livre.

**Configurando os iframes**

**Compatibilidade com navegadores**

É importante lembrar que os iframes não funcionam em todos os navegadores, então, é fundamental deixar uma mensagem caso o navegador não consiga interpretar o iframe, deixando uma mensagem entre a tag <iframe> e </iframe>.

<iframe src=”site.com” frameborder=”0”>

O seu navegador ou dispositivo não suporta o iframe, acesse o <a href=”site.com” target=”\_blank”>Site para ver seu conteúdo.

</iframe>

No exemplo acima, tudo que está entre as tags do iframe vai aparecer caso o iframe não possa ser carregado ou suportado.

**Altura e Largura**

Por padrão, os iframes possuem uma largura de 300 pixels e uma altura de 150 pixels (300x150). É possível mudar isso identificando os valores dos parâmetros **width** e **height**, que são largura e altura, respectivamente.

<iframe src=”site.com” frameborder=”0” width=”500” height=”300”>

</iframe>

No exemplo acima, os valores foram alterados do padrão 300x150 para 500x300.

**Altura e Largura em CSS**

É possível também alterar os valores de largura e altura em CSS, configurando o elemento iframe{}.

iframe{

width: 400px;

height: 400px;

}

A prioridade são as folhas de estilos, caso sejam configurados também em HTML a largura e altura do iframe.

**Borda do iframe**

O parâmetro **frameborder** aceita os valores 0 e qualquer outro valor numérico. Quando o valor for 0, ele não exibirá uma borda, já se for qualquer outro valor irá ser exibida uma borda.

**Borda personalizada**

Para usar uma borda personalizada, altere o valor do frameborder para 0 e configure uma borda em CSS normalmente.

**Conteúdo local no iframe**

É possível colocar páginas locais, além de páginas web em um iframe. Para isso, basta indicar o diretório da página ao especificar um diretório.

**Abrindo uma página via âncora**

É possível abrir uma página no iframe por meio de âncoras, para isso é necessária uma âncora ligada à página e um target com o **name** do iframe. Não é necessário que esse iframe tenha um diretório já especificado, podendo deixar o src em branco.

O **name** é como um id ou class, usado para referenciar um objeto.

**HTML dentro do iframe**

Ao invés de utilizar o parâmetro src, você pode usar o parâmetro **srcdoc**, que permite que, em seu valor, haja códigos em HTML, como parágrafos, títulos, imagens, etc.

<iframe srcdoc="<h1>Título</h1>"></iframe>

Caso haja a necessidade de usar aspas dentro do srcdoc, use aspas simples para não entrar em conflito com as aspas duplas da declaração de valor do parâmetro.

**iframes mais seguros**

Os iframes podem permitir ataques ao site, por isso é importante torná-los seguros, limitando suas ações até o máximo possível, isso, claro, é bem desnecessário quando a página carregada pelo iframe é do próprio servidor e tem sua total confiança.

Existem alguns parâmetros no tag <iframe> que ajudam a torná-lo mais seguro.

**sandbox**

Esse parâmetro pode receber a si mesmo como valor, ou seja:

<iframe src="pag.html" sandbox="sandbox"></iframe>

O valor **sandbox** é a restrição máxima.

O valor desse parâmetro também pode ser, **allow-forms** **allow-modals**, **allow-scrips**, entre outros, cada um permitindo alguma coisa em específico.

Também é possível permitir mais de uma coisa, colocando mais de um valor, separando-os com espaço.

**referrerpolicy**

Esse parâmetro é a política de referência, seu valor padrão é “**no-referrer-when-downgrade**”, para tornar o conteúdo do iframe um pouco mais limitado, basta colocar apenas “**no-referrer**”.

<iframe src="pag.html" referrerpolicy="no-referrer"></iframe>

Assim, a página carregada dentro do iframe não terá permissão para coletar informações do usuário.

**Formulários**

Os formulários são usados para que o servidor receba informações enviadas pelo usuário por meio de um campo de texto, por exemplo.

Para criar um formulário em HTML, use a tag <form>.

**Aplicando o <form>**

A tag <form> precisa ser fechada, ou seja, o formulário fica entre <form> e </form>. Abaixo um exemplo de formulário simples.

**<form action="cadastro.php">**

**<p>**

**Nome: <input type="text" name="nome" id="idnome">**

**</p>**

**<p>**

**Sobrenome: <input type="text" name="sobrenome" id="idsobrenome">**

**</p>**

**<p>**

**<input type="submit" value="Enviar">**

**</p>**

**</form>**

No exemplo acima, é pedido o nome e o sobrenome do usuário, e no fim há um botão de enviar.

Ao enviar, os dados serão enviados ao documento **“cadastro.php”**, especificado no parâmetro **action**, no início do código. Lembrando que não precisa ser php especificamente.

**Label**

A tag <label> é fundamental em um formulário, ela cria uma relação entre um texto e seu input. No código anterior, não há nenhuma relação entre o texto nome e seu input, assim, esse formulário não seria ideal para contribuir com os mecanismos de busca.

**<form action="cadastro.php">**

**<p>**

**<label for="idnome">Nome:</label> <input type="text" name="nome" id="idnome">**

**</p>**

**<p>**

**<label for="idsobrenome">Sobrenome:</label> <input type="text" name="sobrenome" id="idsobrenome">**

**</p>**

**<p>**

**<input type="submit" value="Enviar">**

**</p>**

**</form>**

**Métodos de envio**

Existem dois métodos de envio no HTML, o método **GET** e o método **POST**.

**Método get**

O método **get** é o método padrão caso não especificado o outro. Com ele, os dados serão visíveis na URL.

O método GET é recomendado para quando os dados não forem sensíveis ou privados, pois o mesmo não é seguro para ocultar tais informações.

Tem o limite de 3000 bytes, o que seria mais ou menos 3000 caracteres.

Ótimo para compartilhamento de URL, sendo um ponto muito positivo.

Por outro lado, o mesmo não permite o envio de arquivos como imagens e vídeos.

**Método post**

O método **post** oculta os dados enviados na URL, mas ainda assim é possível ver esses dados de outras maneiras.

Ao contrário do get, permite o envio de arquivos, como imagens e vídeos.

**Configurações do <input>**

Existem diversos parâmetros de configuração do input.

* **‘type’:** Define o tipo de entrada. Pode ser ‘text’, ‘passworl’, ‘email’, ‘number’, entre outros.
* **‘value’:** Define o valor padrão para aquele campo de entrada
* **‘placeholder’:** Exibe um texto dentro do campo de entrada.
* **‘required’:** Torna o preenchimento do campo de entrada obrigatório.
* **‘disabled’:** Desativa qualquer interação com o campo de entrada.
* **‘readonly’:** Torna o campo de entrada somente leitura, assim o usuário só pode ver o valor, mas não pode modificá-lo.
* **‘size’:** Define a largura do campo de entrada em caracteres.
* **‘maxlength’, ‘minlength’:** Limitam a o máximo e o mínimo de caracteres permitidos no campo de entrada.
* **‘max’ e ‘min’:** Estabelecem os valores mínimo e máximo para campos numéricos.
* **‘step’:** Especifica o quanto aumenta ou diminui (incremento ou decremento) dos botões em campos numéricos.
* **‘autocomplete’:** Define se o navegador vai ou não completar automaticamente ou sugerir os valores do campo de entrada.
* **‘pattern’:** Define um padrão de entrada, como por exemplo, definir “x xxxx-xxxx” para um campo numérico onde o valor deve ser um número de telefone.
* **‘autofocus’:** Determina se o campo de entrada deve ou não receber o foco automaticamente quando a página é carregada.
* **‘multiple’:** Permite a seleção múltipla em campos de arquivo.
* **‘accept’:** Especifica os tipos de arquivo que podem ser enviados através de campos de arquivo.
* **‘form’:** Especifica a qual formulário o campo de entrada pertence.

**Tipos de type**

* **‘text’:** Cria um campo de entrada para inserção de texto, é completamente livre.
* **‘password’:** Cria um campo de entrada de texto onde os caracteres digitados são mascarados, geralmente exibidos como asteriscos ou pontos, para ocultar a senha.
* **‘number’:** Cria um campo de entrada numérico. Atributos como **‘min’**, **‘max’** e **‘step’** controlam os valores aceitos.
* **‘radio’:** Cria um campo de entrada de seleção que permite o usuário selecionar apenas uma opção. Nesse caso, para que apenas uma opção seja marcada por vez, o nome de todas as alternativas deverão ser o mesmo.
* **‘checkbox’:** Cria um campo de entrada de seleção que permite ao usuário selecionar uma ou várias opções entre um conjunto de opções.
* **‘file’:** Cria um botão que permite aos usuários selecionar e enviar arquivos de seu dispositivo local. Lembre-se: quando utilizar o tipo file, use como método de formulário o **post**. O motivo já foi dito antes.
* **‘submit’:** Cria um botão de envio de formulário que envia os dados do formulário para o servidor.
* **‘reset’:** Cria um botão que redefine os valores do formulário para seus valores padrão.
* **‘time’:** Cria um campo de entrada para seleção de horário.
* **‘date’:** Cria um campo de entrada para seleção de data.
* **‘range’:** Cria uma barra de deslizamento horizontal. Valores como **‘max’**, ‘min’ e **‘step’** ajudam a controlar seus valores.
* **‘color’:** Cria um campo de entrada para seleção de cor.
* **‘url’:** Cria um campo de entrada de texto, feito especificamente para URLs, validando a entrada do usuário de acordo com as regras de formato de URL.
* **‘tel’:** Cria um campo de entrada para números de telefone.
* **‘search’:** Cria um campo de pesquisa. Geralmente usado para pesquisas em mecanismos de busca.
* **‘hidden’:** Cria um campo de entrada oculto que não é visível para o usuário, mas que pode ser usado para armazenar dados no formulário.
* **‘week’:** Cria um campo de entrada para seleção de semana e ano.
* **‘month’:** Cria um campo de entrada para mês e ano.
* **‘datetime’:** Cria um campo de entrada para seleção de data e horário.
* **‘datetime-local’:** Cria um campo de entrada para seleção de data e horário, sem considerar o fuso horário.
* **‘image’:** Cria um botão de imagem que, quando clicado, envia as coordenadas do ponto onde o usuário clicou no botão.
* **‘button’:** Cria um botão de ação genérico que pode executar uma função JavaScript ou realizar uma ação específica quando clicado.

**Agrupamento de Campos**

A tag **‘<fieldset>’** agrupa vários campos de entrada em uma área, criando uma borda em volta.

Todos os inputs e labels devem estar envelopados dentro desta tag.

**Legenda**

Dentro do **‘<fieldset>’,** pode ser colocada uma outra tag, a **‘<legend>’**. Ela cria uma legenda no topo da área.

**‘<select>’**

A tag **‘<select>’** permite ao usuário escolher uma das opções criadas pelo desenvolvedor. Cada opção é definida pela tag **<option>**, que deve conter a opção dentro de si.

**Grupo de opções**

Se necessário separar conjuntos de opções em grupos, envelope as que devam fazer parte do mesmo grupo dentro de um **<optgroup label=”Nome do Grupo”>**.

Exemplificando:

<form>

    <label for="option">Opções</label>

    <select name="option" id="option">

        <optgroup label="Grupo 1">

            <option>Alternativa 1</option>

            <option>Alternativa 2</option>

            <option>Alternativa 3</option>

        </optgroup>

        <optgroup label="Grupo 2">

            <option>Alternativa 1</option>

            <option>Alternativa 2</option>

            <option>Alternativa 3</option>

        </optgroup>

    </select>

</form>

**Seleção Múltipla**

Para permitir o usuário escolher mais de uma opção, adicione o atributo **‘multiple’** a tag <select>.

<select multiple></select>

**Seleção padrão**

O atributo **‘selected’** dentro da tag <option> fará com que a opção relativa a esse <option> seja selecionada automaticamente quando o formulário for carregado.

<option selected>Alternativa</option>

**Valor para a opção**

É interessante que cada opção possua um **‘value’**, para que no momento de uso dos dados passados para o formulário seja facilitado.

<form>

    <label for="option">Opções</label>

    <select name="option" id="option">

        <optgroup label="Grupo 1">

            <option value="a1">Alternativa 1</option>

            <option value="a2">Alternativa 2</option>

            <option value="a3">Alternativa 3</option>

        </optgroup>

        <optgroup label="Grupo 2">

            <option value="a4">Alternativa 4</option>

            <option value="a5">Alternativa 5</option>

            <option value="a6">Alternativa 6</option>

        </optgroup>

    </select>

</form>

**Desabilitar opção**

Ao adicionar o atributo **‘disabled’** em um **‘<option>’**, a opção passa a ser não selecionável, se tornando inativa.

<option disabled>Alternativa</option>

**‘<datalist>’**

A função da tag **‘<datalist>’** é parecida com a ‘<select>’, porém, ela permite o usuário digitar valores diferentes dos determinados pelo desenvolvedor ao ser combinada com um ‘<input>’.

Dentro dessa tag também pode ser usado a tag **‘<option>’**, assim como no ‘<select>’.

Para criar uma relação entre as opções e a caixa de texto em que o usuário pode digitar, é necessário usar o atributo **‘list’** dentro da tag **‘<input>’**.

<form>

    <label for="item\_lista">Lista</label>

    <input type="text" name="item\_lista" id="item\_lista" list="lista">

    <datalist id="lista">

        <option>Item 1</option>

        <option>Item 2</option>

        <option>Item 3</option>

        <option>Item 4</option>

    </datalist>

</form

No exemplo acima, o id **“lista”**, determinado na tag ‘<datalist>’, cria uma relação entre as opções e o campo de texto por meio do atributo **‘list’**, especificado no ‘<input>’.

A tag **‘<datalist>’** ajuda o usuário dando a ele opções que podem até mesmo ajudá-lo a escrever uma não especificada, mas relacionada de qualquer forma. Tudo isso por meio de uma lista que é suspensa sobre o campo de entrada.

**‘textarea’**

A tag **‘<textarea>’** cria um campo de entrada de texto personalizado, permitindo aos usuários inserirem dados em um formato de várias linhas.

Os atributos para personalizar a área do campo de entrada são **cols**(colunas) e **rows**(linhas). Em ambos os atributos os valores devem ser numéricos, especificando uma quantidade de colunas e linhas.

<form>

    <label for="msg">Mensagem</label>

    <textarea name="msg" id="msg" cols="30" rows="10">Texto padrão</textarea>

</form>

No exemplo acima, é determinado o valor 30 para o atributo de colunas e o valor de 10 para as linhas, ou seja, a área do campo de entrada será de 30 colunas e 10 linhas.

Também veja que essa tag trabalha junto com a tag **‘<label>’**, criando uma relação entre o texto “Mensagem” e o campo de entrada de texto por meio do atributo **‘name’**.

O valor digitado entre as tags de abertura e fechamento da tag **‘<textarea>’** definem um texto padrão ao campo de entrada.

**‘<output>’**

A tag **‘<output>’** é usada para exibir o resultado de um cálculo ou script. Ela é geralmente usada em conjunto com a tag ‘<input>’ para criar interações entre o usuário e a página.

Não terá detalhes aqui sobre JavaScript aqui, por isso, não terá detalhes sobre essa tag.

***Media Queries***

As Media Queries são uma parte fundamental do design responsivo na web. Elas permitem adaptar a aparência e o layout de um site ou aplicativo de acordo com as características do dispositivo e das janelas visualização em que estão sendo exibidos.

***Media Types***

As Media Types, ou tipos de mídia, permitem que os desenvolvedores criem um estilo diferente para cada tipo de mídia, como telas de computadores, de celulares, de TV e até para layout de impressão. As Media Types permitem que os estilos sejam adaptados para cada tipo de mídia, garantindo uma apresentação adequada e otimizada para diferentes contextos.

As Media Types mais comuns são:

1. **‘all’:** É a Media Type padrão e se aplica a todos os dispositivos e mídias.
2. **‘screen’:** Para todos os dispositivos de tela.
3. **‘print’:** Para impressoras.
4. **‘speech’:** Para dispositivos de sintetização de voz, como leitores de tela ou sistemas de reconhecimento de voz.
5. **‘projection’:** Para projetores.
6. **‘tv’:** Para dispositivos de televisão.

**Aplicando Media Types**

Existem duas formas de aplicar as Media Types:

**Documento de estilo único**

Nesse caso, um único documento CSS irá conter dentro de si todos os estilos de cada tipo de mídia.

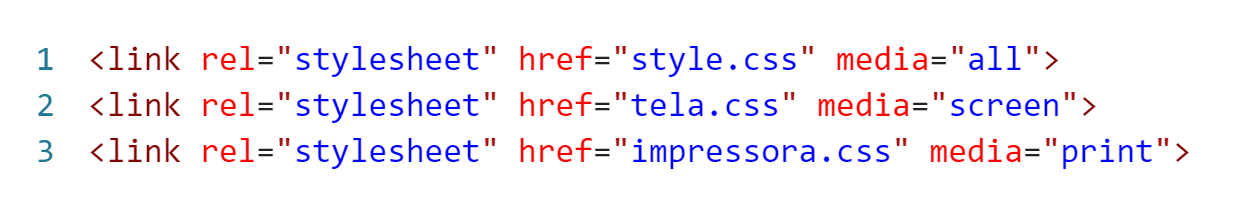
Após a regra **‘@media’**, escreva o tipo de mídia relativo a o estilo a ser trabalhado.



**Separação por tipo de mídia**

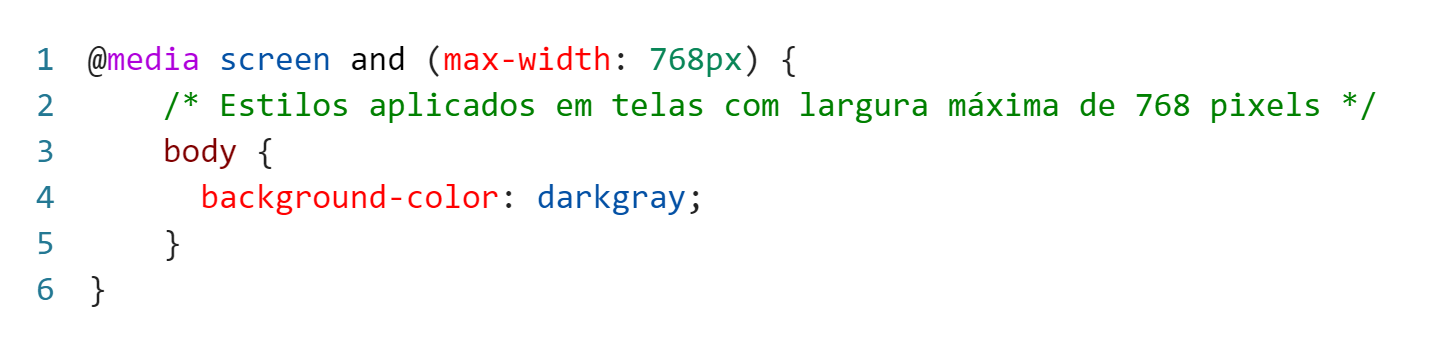
Nesse caso, será um documento para cada tipo de mídia.

Ao chamar o arquivo CSS no documento HTML, será adicionado o atributo **‘media’** a tag <link>.



***Media Features***

As ***Media Features*** (características de mídia) são usadas em conjunto com as Media Types, e são utilizadas para especificar as características ou condições específicas de um dispositivo ou ambiente de visualização para criar estilos personalizados para cada situação.



No caso do exemplo acima, todos os dispositivos que rodarem a página e tiverem a tela com 768 pixels ou menos, irá possuir a cor de fundo cinza escuro em seu corpo.

As Media Features **são aplicadas entre parênteses**, e ligadas junto a Media Type pela palavra **‘and’** (‘e’, em português).

**Tipos de *Media Features***

Existem vários tipos de Media Features, e as mais comuns são:

1. **‘width’:** Especifica a largura do viewport do dispositivo.
2. **‘height’:** Especifica a altura do viewport do dispositivo.
3. **‘min-width’ – ‘max-width’:** Especificam a largura mínima e máxima do viewport do dispositivo.
4. **‘min-height’ – ‘max-height’:** Especificam a altura mínima e máxima do viewport do dispositivo.
5. **‘device-width’:** Especifica a largura real do dispositivo físico.
6. **‘device-height’:** Especifica a altura real do dispositivo físico.
7. **‘orientation’:** Especifica a orientação do dispositivo (retrato/portrait ou paisagem/landscape). O valor **‘portrait’** ou **‘landscape’** deve ser especificado após dois pontos depois da palavra ‘orientation’.
8. **‘aspect-ratio’:** Especifica a relação entre a largura e a altura do viewport.
9. **‘color’:** Especifica o número de bits de cores suportados pelo dispositivo.
10. **‘resolution’:** Especifica a resolução de exibição do dispositivo em DPI (pontos por polegada).
11. **‘pointer’:** Especifica o tipo de dispositivo de entrada de ponteiro disponível (mause, touchscreen, etc.)
12. **‘hover’:** Especifica se o dispositivo suporta interações.
13. **‘grid’:** Especifica se o dispositivo suporta recursos de layout em grade.

***Mobile First***

***Mobile First*** é priorizar a experiência do usuário em dispositivos móveis em primeiro lugar. O conceito surge da necessidade de adaptar o conteúdo e o layout para as limitações de espaço e recursos desses dispositivos, que possuem tela menores e capacidades de processamento diferentes dos computadores.

A estratégia ***Mobile First*** envolve projetar e construir começando pela versão móvel e, em seguida, expandindo-o para dispositivos maiores.

***Typical Device Breakpoints***

Refere-se às larguras de tela específicas em que o layout de um site ou aplicativo se adapta para se ajustar a diferentes dispositivos. São fundamentais para um design responsivo de páginas web.

1. Dispositivos extra pequenos (Telefones):
   * Largura de tela: Menos de 576 pixels
2. Dispositivos pequenos (Tablets):
   * Largura de tela: 576 pixels a 767 pixels
3. Dispositivos médios (Laptops e desktops pequenos):
   * Largura de tela; 768 pixels a 991 pixels
4. Dispositivos grandes (Desktops grandes e telas de alta resolução):
   * Largura da tela: 1200 pixels ou mais

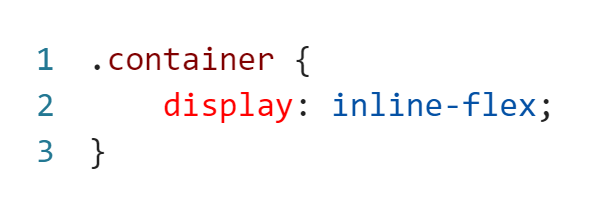
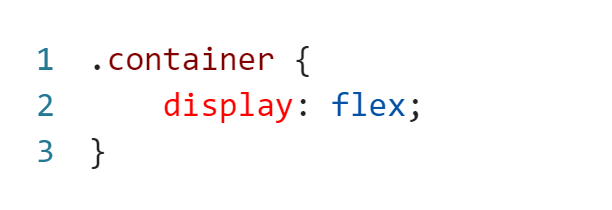
***Flexbox***

***Flexbox*** é uma ferramenta poderosa para criar layouts flexíveis e responsivos. Foi uma novidade do CSS3 e é especialmente útil quando a necessidade é alinhar ou distribuir elemento em um container.

Tudo de Flexbox gira em torno de dois conceitos: os elementos flexíveis (**flex items**) e o container flexível (**flex** **container**). Os elementos flexíveis são filhos de um container flexível.

**Pelo começo...**

Primeiramente é necessário definir um container flexível, para assim atribuir a ele a propriedade **‘display: flex’** ou **‘display: inline-flex’**.



***Containers***

Um container deve possuir filhos dentro de si para que faça sentido o uso das Flexbox, por exemplo:



No exemplo acima, temos um div que é um container, e a mesma engloba outras três divs, que são os elementos chamados de filhos (flex items).

**Propriedades do Flexbox**

1. **‘flex-direction’:** Define a direção dos items dentro do container.

**Valores:**

* + **‘row’:** Os flex items são dispostos em uma linha da esquerda para a direita.
  + **‘row-reversed’:** Os flex items são dispostos em uma linha da direita para a esquerda.
  + **‘column’:** Os flex items são dispostos em uma coluna de cima para baixo.
  + **‘column-reverse’:** Os flex items são dispostos em uma coluna de baixo para cima.

1. **‘flex-wrap’:** Define se os flex items devem permanecer em uma única linha ou quebrar em várias linhas.

**Valores:**

* **‘nowrap’:** Os flex items são dispostos em uma única linha (padrão).
* **‘wrap’:** Os flex items são dispostos em várias linhas.
* **‘wrap-reverse’:** Os flex items são dispostos em várias linhas, em ordem reversa.

1. **‘justify-content’:** Define o alinhamento horizontal dos flex items.

**Valores:**

* **‘center’:** Os flex items são centralizados horizontalmente no container.
* **‘flex-start’:** Os flex items são alinhados à esquerda do container.
* **‘flex-end’:** Os flex items são alinhados à direita do container.
* **‘space-between’:** Os flex items são distribuídos igualmente ao longo do eixo principal, com espaçamento entre eles.
* **‘space-around’:** Os flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em ambos os lados.
* **‘space-evenly’:** Os flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em todos os lados.

1. **‘align-items:** Define o alinhamento vertical dos flex items.

**Valores:**

* **‘center’:** Os flex items são centralizados verticalmente no container.
* **‘flex-start’:** Os flex items são alinhados ao topo do container.
* **‘flex-end’:** Os flex items são alinhados à base do container.
* **‘baseline’:** Os flex items são alinhados na linha de base do container.
* **‘stretch’:** Os flex items são redimensionados para preencher a altura total do container.

1. **‘align-content’:** Define o alinhamento vertical das linhas de flex items quando há espaço extra no container e as linhas não preenchem totalmente o espaço disponível.

**Valores:**

* **‘center’:** As linhas de flex items são centralizados verticalmente no container.
* **‘flex-start’:** As linhas de flex items são alinhados ao topo do container.
* **‘flex-end’:** As linhas de flex items são alinhados na base do container.
* **‘space-between’:** As linhas de flex items são distribuídos igualmente ao longo do eixo principal, com espaçamento entre elas.
* **‘space-around’:** As linhas de flex items são distribuídos uniformemente ao longo do eixo principal, com espaçamento igual em ambos os lados.
* **‘stretch’:** As linhas de flex items são redimensionadas para preencher o espaço vertical disponível.

1. **‘align-self’:** Define o alinhamento vertical de um flex item individual, substituindo o valor definido pela propriedade ‘align-items’ no container. Os valores desta propriedade são os mesmos da propriedade ‘align-items’.
2. **‘order’:** Define a ordem em que os flex items são exibidos dentro do container. O valor deve ser algum número inteiro para determinar a ordem.
3. **‘flex-grow’:** Define a capacidade de crescimento de um flex item em relação aos outros flex items quando há espaço extra disponível. O valor deve ser algum número inteiro que representa a proporção de crescimento.
4. **‘flex-shrink’:** Ao contrário do ‘flex-grow’, define a capacidade de encolhimento de um flex item em relação aos outros flex items quando o espaço é limitado. O valor deve ser um número inteiro que representa a proporção de encolhimento.
5. **‘flex-basis’:** Define o tamanho inicial de um flex item antes de ser distribuído dentro do container. O valor deve ser algum número que representa o tamanho inicial em alguma unidade de medida, ou o valor **‘auto’** para usar o tamanho padrão do conteúdo.
6. **‘row-gap’:** Define o espaçamento vertical entre os flex items em um layout de linha. É usado junto à propriedade ‘flex-wrap’.
7. **‘column-gap’:** Define o espaçamento horizontal entre os flex items em um layout de coluna. É usado junto à propriedade ‘flex-wrap’.

***Shorthands* de *Flexbox***

1. **‘flex’:** Shorthand para **‘flex-grow’**, **‘flex-shrink’** e **‘flex-basis’**, respectivamente.
2. **‘flex-flow’:** Shorthand para **‘flex-direction’** e **‘flex-wrap’**, respectivamente.
3. **‘gap’:** Shorthand para **‘row-gap’** e **‘column-gap’**, respectivamente.
4. **‘place-content’:** Shorthand para **‘align-content’** e **‘justify-content’**, respectivamente.