**INDICE**

* [Cores](#cores)
* [Trabalhando com textos](#Texto)
* [Trabalhando com imagens](#imagens)
* [Estrutura semântica e posicionamento](#estrutura_semantica)
* [Flexbox](#flexbox)
* [Unidades de medidas](#unidades_medidas)
* [Calc](#calc)
* [Variáveis](#variaveis)





**CORES**

**Para deixar fundo com transparência:** use “rgb ( #, #, #, 0.7)” – passe o código da cor e o último valor é para definir o grau da transparencia.

**TRABALHANDO COM TEXTOS**

* **Line-height:** define a altura da linha.
* **Text**-**align:** alinhamento do texto
* **Text-transform**: deixa letras maiúsculas, minúsculas, 1º letra maiúscula, etc...4
* **Text**-**shadow:** para criar uma sombra no texto.
* **Width**: **fit-content:** ajusta a área ocupada ao conteúdo;
* **Widht**: **min-contente:** define a área ocupada de acordo com a maior palavra do bloco;
* **Widht**: **max-content:** deixa o bloco com a maior largura possível e todo o conteúdo ocupa apenas uma linha.
* **Overflow-hidden:** esconde o conteúdo se ele ultrapassar o bloco;
* **Overflow-scroll:** adiciona uma barra de rolagem em baixo e na lateral;
* **Overflow-auto:** adiciona uma barra de rolagem automaticamente;

**TRABALHANDO COM IMAGENS**

Há duas formas para inserir imagens:

**Com HTML:** a imagem fará parte do conteúdo do site.

Tag = <img src= “...” alt= “...”>

Src = inserir o endereço da imagem

Alt= texto alternativo caso a imagem não carregue.

**Com CSS:** quando a imagem for meramente decorativa.

Tag = background-image: url(...);

* **Width:** Para definir o tamanho da imagem no html;
* **Float:** a imagem “flutua” para a direção que você definir e o texto se adapta a página;
* **Background-repeat:** para a imagem se repetir ou não;
* **Background-size:** para definir o tamanho da imagem no css;
* **Background-attachment:** escolhapara que a imagem role junto com a página ou fique fixa;
* **Background-position:** o foco na imagem- se focará em baixo, a esquerda, em cima...

**ESTRUTURA SEMÂNTICA E POSICIONAMENTO**

Tags Semânticas

<header> : usada para inserir o cabeçalho da página;

<nav> utilizada para colocar a navegação principal da página;

<main> conteúdo principal da página;

<footer> roda-pé da página;

<section> agrupa um conjunto de informações relacionadas;

<article> conteúdo de extrema importância para a página;

<aside> quando algo não está relacionado com o conteúdo da página, algo a parte.

<div> e <span> servem somente para ajudar no layout ou design da página. A div por padrão é um elemento de bloco e ocupará 100% da área disponível, a span já é um elemento de linha e possui um display inline.

* **Position-static**: valor padrão, ele apenas segue o fluxo junto com os outros elementos da página, tendo o canto superior esquerdo como referência. Não aceita as propriedades auxiliares top, bottom, left e right.
* **Position-fixed:** faz com que o elemento que recebeu essa propriedade não se mova na tela. Aceita as propriedade auxiliares;
* **Position-sticky:** bem parecido com o fixed, porém a sua diferença é que ele fixa fixo em relação ao rolamento da página;
* **Position-relative:** é usado quando queremos alterar a posição de um elemento tendo como referência a posição inicial dele;
* **Position-absolute**: indica que o elemento ficará em uma posição fixa em relação à página.
* **Z-index:** usado quando queremos que algum elemento sobreponha o outro. Por padrão, todos os elementos possuem o z-index= 0, quanto maior o valor do z-index mais ele irá sobrepor aquele elemento. Também é possível que se use valores negativos.

**FLEXBOX**

* **Flax-wrap**: **wrap -**  define se os itens flexíveis são forçados a ficarem na mesma linha ou se podem ser quebradas em várias linhas. Se o argumento for valido, ele define a direção em que as linhas são empilhadas.
* **Flex-wrap: wrap-reverse** - Se comporta da mesma maneira que o wrap mas a de linha ocorre na direção contrária, ou seja, para a linha acima.
* **Flex-wrap: nowrap** - Os itens flexíveis são agrupados em uma unica linha , esse é o valor padrão.
* **flex-direction: row -** O eixo principal do flex container é definido para ser o mesmo que a direção do texto.
* **flex-direction: row-reserve -** Se comporta da mesma forma que row porém os pontos de início e término serão invertidos**.**
* **flex-direction: column –** o eixo principal do flex é em coluna.
* **Flex-direction: column-reserve -** o eixo principal do flex é em coluna porém os pontos de inicio e termino serão invertidos.
* **Flex-flow:** atalho para adicionar flex-wrap e flex-direction juntos.

**Justify-content:**

Define como o navegador distribui o espaço entre e ao redor dos itens de conteúdo ao longo do [eixo principal](https://developer.mozilla.org/en-US/docs/Glossary/Main_Axis) de um contêiner flexível e do eixo embutido de um contêiner de grade.

Valores: **center, space-between, space-around, space-evenly, end.**

**Align-items:**

A propriedade align-items é aplicada em contêineres flexíveis para controlar a posição vertical dos elementos filhos. Ela define como os itens são alinhados ao longo do eixo transversal. Quando possui uma única linha.

Valores: **center, flex-start, flex-end, baseline**

**Align-content:**

Align-content — descrito na especificação como "packing flex lines"; controla o espaço entre as linhas no eixo transversal. Quando eu tenho mais de uma linha. Junto com essa propriedade é preciso usar “Flex-wrap: wrap”.

**Propriedades para aplicar nos itens:**

* **Flex-basis:** Define o tamanho dos itens a qual a propriedade é aplicada.
* **Flex-grow:** Esta propriedade define a proporção com que um item deve crescer caso seja necessário. Por padrão seu valor é 0, o que indica que o item não deve crescer, e são aceitos apenas valores numéricos positivos.
* **Flex**-**shrink** : A propriedade **flex**-**shrink** especifica o quanto o item encolherá em comparação com outros itens dentro desse contêiner.
* **Order:** O valor numérico atribuído a essa propriedade define a ordem do item. Por padrão, todos os itens possuem valor 0. Por exemplo, o valor 2 faz com que o item seja o segundo item ao longo do eixo principal, enquanto o valor -1 faz com que ele apareça antes do primeiro.
* **Align-self:** Alinha somente aquele item.

**UNIDADES DE MEDIDAS**

A unidade pixel (px) é utilizada apenas para alguns casos, por exemplo: em uma margin pequena, um padding pequeno, box-shadow, text-shadow, coisas que serão pequenas.

Mas em tamanhos como width, heigth, font-size usaremos unidades de medidas maior.

**Unidade de medida em:** em configurações normais, o padrão de **1em** equivale a 16px. Pode-se usar valores como 1.5em com ponto flutuante. Uma dica também: sabendo que **1em** equivale a 16 px, podemos mudar o tamanho da porcentagem da fonte no html, e definilo como font-size: 62.5%; - assim **1em** passará a valer 10px. Para alterar a fonte para 18 px por exemplo, é só utilizar **1.8em**, ou para 24 px basta definir o valor de **2.4em.** Os elementos filhos seguem a medida padrão do elemento pai.

Também pode-se utilizar **em** no padding quando queremos que o mesmo dique proporcional ao tamanho da fonte do nosso elemento.

**Unidade de medida rem:** ao utilizar a unidade de medida **rem**, os elementos seguem o padrão de medida definido no html.

**Porcentagem em margin:** a porcentagem sempre será referente a largura do elemento pai.

**Vw**  e **vh:** As unidades vh e vw são úteis quando é preciso usar a largura ou a altura da viewport (tela inteira). A medida vh é igual a 1/100 da altura da viewport. Então, por exemplo, se a altura do navegador é 900px , 1vh equivale a 9px . E, analogamente, se a largura da viewport é 750px , 1vw equivale a 7.5px .

**Ch**: A unidade ch (“character unit” ou “unidade-caractere”) é definida como sendo a “medida avançada” da largura do caractere zero (“0”). Usando a unidade ch não se perde a legibilidade, diferente do px.

**FUNÇÃO CALC()**

A função calc() permite que se definam valores CSS com uso de expressões matemáticas, ou seja, o valor adotado para a propriedade é o resultado de uma expressão matemática.

* É válido usar-se unidades de comprimento diferentes para os valores que compõem a expressão matemática.  
  Por exemplo: *calc(50% + 5em - 255px / 2)*
* Os operadores + (soma) e - (subtração) devem obrigatoriamente conter um espaço em branco antes e depois:  
  Errado: *calc(50%+5em)*  
  Certo: *calc(50% + 5em)*
* Para os operadores \* (multiplicação) e / (divisão) deixar um espaço em branco antes e depois não é obrigatório, mas aconselhável para melhorar legibilidade:  
  Certo: *calc(50% + 3em\*5)*  
  Melhor: *calc(50% + 3em \* 5)*
* É válido o uso de parênteses para alterar a ordem dos cálculos tal como se faz em expressões matemáticas:  
  Exemplo: *calc(200px + (50% - 8rem) / 3)*

**VARIÁVEIS CSS**

Para criar uma variável no css utiliza-se dois hifens (--) e o nome que deseja dar a sua variável.

Para utilizar essa variável no código utiliza-se **var** e parênteses **( )**.

Exemplo:

--altura-main = criei a variável

Height: var(--altura-main) = utilizei a variável

É sempre bom declarar a variável no html pois assim ela ficará disponível para todo o documento e não somente exclusivamente para um elemento.

**:root ->** A pseudo-classe :root é utilizada para selecionar o elemento raiz do documento HTML, ou seja, o elemento HTML. Ela é útil para definir variáveis CSS globais, que podem ser utilizadas em todo o documento.

**MAIS PROPRIEDADES CSS**

**Border-radius** – arredondamento das bordas

**Box**-**shadow –** Deixa o elemento com sombra

**Text-shadow –** Deixa o texto com sombra

**Outline –** coloca margem ao redor de um elemento

**Font-face –**  para importar uma fonte é necessário baixa-la no seu computador e adicionar a pasta onde se escontra essa fonte ao seu projeto. Após, para importar a fonte para o seu documento html use:

@fontface {

Font-family: nadia // nadia é o nome que dei para a fonte

Src: url (passar a pasta da fonte, como fazemos com a imagem)

}

**Transition -** como o próprio nome diz, é uma propriedade de transição. Na propriedade transition temos:

Transition-property: qual a propriedade aplicaremos a transição.

Transition-duration: a duração que ela terá, pode ser em s (segundos) ou ms (milissegundos).

Transition-time-function: aceleração da animação

Transition-delay: atrasa um pouco a animação

**LISTAS**

No html temos dois tipos de listas: as não ordenadas que criamos com **ul** e **li**; as ordenadas que criamos com **ol** e **li**; e as listas de definição.

Para criar uma lista de definição nós temos:

<dl>

<dt> **Título** </dt>

<dd> Descrição/definição do termo</dd>

</dl>

**TABELAS**

Para criar tabelas no html usamos a tag **table**.

<table>

<tr>

**table** para criar a tabela;

**tr** são as linhas da tabela;

**td** são as colunas.

<td>

</td>

</tr>

</table>

**Atributos colspan e rowspan**

Utilizamos o colspan quando queremos que uma célula da nossa tabela ocupe mais de uma coluna, e o rowspan quando uma célula da nossa tabela for ocupar mais do que uma linha.

**Tags Semânticas**

**Thead –** trata-se do cabeçalho da tabela

**Tbody –** corpo da tabela

**Tfoot –** roda-pé da tabela