HTML/CSS

Nível Básico

- unidades de medida
  - Medidas absolutas: px
  - Medidas relativas:
    - em→ muda de acordo com o tamanho da fonte do elemento pai
    - rem → relacionado ao tamanho da fonte do elemento raiz
    - % →vai se readaptar para ocupar a quantidade especificada
    - ex → tamanho da letra x da font-family que está sendo usada → Esse valor pode vir diretamente com a fonte, o browser pode medir o caractere em caixa baixa (lower case) e se esses dois não funcionarem, o browser estipula um valor de 0.5em para 1ex.
    - vw → viewport width → 1vw == 1% do tamanho da largura visível da página
    - vh → viewport height → 1vw == 1% do tamanho da altura visível da página
    - vmin → viewport minimum → vmin utilizará como base a menor dimensão da viewport (altura x largura) → 1vmin == 1% da menor dimensão
    - vmax → viewport maximum → utiliza como base o maior dimensão da viewport → 1vmax == 1% da maior dimensão
      ❗ as medidas baseadas no viewport ainda não tem suporte em todos os browsers → https://caniuse.com/?search=vw

- modos de display
  - inline → elementos um do lado do outro, com o comportamento de uma palavra (espaço entre eles)
    - não permite usar height e width
    - se colocar um comentário no espaço que teria o enter entre as linhas, elimina os espaços entre os elementos
    - para alinhar, precisa mudar o text-align do elemento pai
  - block → elemento ocupa uma linha inteira
    - permite usar height e width
    - margin-left/right: auto → preenche completamente a margem do lado selecionado do elemento, fazendo com que ele fique no outro lado → se o elemento não tiver largura, ele vai ocupar a linha inteira
      → se colocar as duas, ele fica no meio
  - inline-block → combinação dos dois, todos na mesma linha mas podendo definir tamanho e altura
    - text-align: justify→ justifica com exceção da última linha
  - none → esconder e exibir elementos sem realmente os remover nem recriar

        - display com o valor none vai exibir a página como se o elemento não existisse. Com visibility:hidden o elemento fica invisível, porém ele permanece ocupando o espaço em que estaria totalmente visível.
- overflow
    - controla o que acontece com o conteúdo de um elemento que ultrapassa o tamanho do elemento no qual está inserido
    - visible → o conteúdo continua além dos limites do elemento pai (default)
    - hidden → o conteúdo além dos limites será escondido
    - scroll → esconde mas é adicionada uma barra para visualizar o resto
    - auto → scroll (?)
    - initial → default (visible)
    - inherit → herda o do elemento pai
- position
    - propriedades: top, bottom, left, right
    - static → indica aos elementos que eles devem seguir o fluxo definido declarado no documento (default)
    - relative→ posição relativa do elemento em relação ao elemento pai
        - O elemento é retirado do fluxo do documento enquanto os outros elementos se comportam como se ele ainda estivesse lá.
    - absolute → o elemento sai do fluxo da hierarquia do html
        - os elementos ao redor reagem como se este elemento realmente não existisse.
    - fixed → o elemento terá sempre o mesmo posicionamento
        - quando navegar na página para cima e para baixo, o elemento ficará fixo na mesma posição sempre
    - sticky → uma mistura de relative e fixed, dependendo da posição de scroll
        - fica posicionado de modo relativo até um certo ponto do scroll e então “cola” no lugar
- iframe
    - usado para incorporar outra página dentro da página atual ex: vídeo do youtube
        - allowfullscreen → permite que o elemento seja colocado em tela cheia
        - é possível criar links que tenham um target = o name do iframe e isso fará com que estes sejam abertos no iframe

Nível Intermediário

- display flex
    - flex container:
        - flex-direction
            - row (default)
            - row-reverse
            - column
            - column-reverse

- flex-wrap → define se os itens devem quebrar a linha ou não
    - nowrap (default) → não permite a quebra de linha
    - wrap → quebra a linha assim que um dos flex itens não puder mais ser compactado, quando um dos flex itens atinge o limite do conteúdo, o último item passa para a coluna debaixo e assim por diante.
    - wrap-reverse → também quebra a linha quando um dos itens não puder mais ser compactado mas é na direção contrária, o item que ultrapassa é colocado na linha de cima
- justify-content → alinha os itens de acordo com a direção
    - A propriedade só funciona se os itens atuais não ocuparem todo o container. Isso significa que ao definir flex: 1; ou algo similar nos itens, a propriedade não terá mais função
    - flex-start→ alinha no início do container
    - flex-end → alinha no final do container
    - center → alinha no centro
    - space-between → cria um espaçamento igual entre os elementos, o primeiro fica grudado no início e o último no final
    - space-around → cria um espaçamento entre os elemento, sendo que os do meio são duas vezes maior do que o inicial e o final
    - space-evenly → espaçamento igual no meio e nas
- align-items → alinhamento de acordo com o eixo do container
    - é diferente para quando os elementos estão em colunas ou em linhas
    - stretch (default) → faz com que os itens cresçam igualmente
    - flex-start → alinha no início
    - flex-end → alinha no final
    - center → alinha no centro
    - baseline → alinha de acordo com a linha base da tipografia
- align-content → alinha no eixo vertical
    - A propriedade só funciona se existir mais de uma linha de flex-itens. Para isso o flex-wrap precisa ser wrap.
    - O efeito dela apenas será visualizado caso o container seja maior que a soma das linhas dos itens. Isso significa que se você não definir height para o container, a propriedade não influencia no layout
    - stretch (default) → faz com que os itens cresçam igualmente na vertical
    - flex-start → alinha no início
    - flex-end → alinha no final
    - center→ alinha no centro
    - space-between → cria um espaçamento igual entre as linhas, a primeira fica grudada no topo e a última no bottom

                - space-around → cria um espaçamento entre os elemento, sendo que os do meio são duas vezes maior do que o top e o bottom
        - flex item → os filhos do flex container
            - flex → atalho para flex-grow, flex-shrink e flex-basis (nessa ordem)
                - valor1, valor2, valor3 → valores diferentes para cada propriedade
            - flex-basis → o tamanho inicial do item antes da distribuição do espaço restante
                - auto (default) → faz com que a largura da base seja igual a do item. Se o item não tiver tamanho especificado, o tamanho será de acordo com o conteúdo.
                - unidade → Pode ser em %, em, px e etc
                - 0 → Se o grow for igual ou maior que 1, ele irá tentar manter todos os elementos com a mesma largura, independente do conteúdo. Caso contrário o item terá a largura do seu conteúdo.
            - flex-grow → define a habilidade do item crescer
                - diz quanto do espaço disponível dentro do container o item vai ocupar → proporção
                - se todos os itens tiverem o mesmo valor, todos os itens vão ocupar o mesmo espaço, se forem diferentes,segue a proporção definida pelos valores
                - 0 (default) → itens ocupam um tamanho máximo relacionado o conteúdo interno deles ou ao width definido.
                - 1 → vão ocupar 100% do container e terão o mesmo tamanho mas se o conteúdo de um elemento seja muito largo, ele irá respeitar o tamanho do conteúdo
            - flex-shrink → define a capacidade de redução de tamanho do item
                - 1 (default) → permite que os itens tenham os seus tamanhos (seja esse tamanho definido a partir de width ou flex-basis) reduzidos para caber no container.
                - 0 → não permite a diminuição dos itens mesmo que o conteúdo não ocupe todo esse espaço.
                - número → Um item com shrink: 3 diminuirá 3 vezes mais que um item com 1
            - order→ Modifica a ordem dos flex itens. Sempre do menor para o maior
            - align-self → serve para definir o alinhamento específico de um único flex item dentro do nosso container. Caso um valor seja atribuído, ele passará por cima do que for atribuído no align-items do container.
- box model / box sizing
    - content → conteúdo
    - padding → o espaçamento entre o conteúdo e as bordas
    - border
    - margin → espaço entre o elemento e outros

    - o tamanho real do elemento vai ser a altura e a largura definida + o tamanho desses propriedades:
        - altura total = height + padding-top + padding-bottom + border-top-width + border-bottom-width;
        - largura total = width + padding-left + padding-right + border-left-width + border-right-width.

    - a propriedade box-sizing faz com que padding e border sejam incluídos na altura e largura definida:
        - content-box (default) →os valores de altura e largura se aplicam somente ao conteúdo e padding e border são adicionados no exterior da caixa
        - padding-box→ os valores de altura e largura se aplicam ao elemento e padding, border é adicionada no exterior → ❗ nenhum browser aceita mais
        - border-box → altura e largura se aplicam a conteúdo, padding e border ♥
- z-index
    - a “profundidade” de um elemento, se vai estar mais próximo ou afastado da tela
    - pode ser um número negativo
    - o elementos de index maior são sobrepostos ao de index menor
    - só funciona com elementos que tenham position
    - os elementos automaticamente recebem o index conforme são criados assim, dentre os elementos que estiverem na mesma posição, o que estiver por último do html será o da frente
- herança de estilos
    - algumas propriedades podem ser herdadas, assim, caso essa mesma propriedade não tenha sido dada um valor no elemento filho, ele receberá a mesma propriedade do que o pai
    - para propriedades que não são herdadas por padrão, é possível forçar isso com a palavra *inherit*
    - propriedades que não são herdadas:
        - background
        - border (exceto: border-collapse e border-spacing)
        - clip
        - content
        - counter
        - clue
        - display
        - float
        - height
        - left
        - margin
        - outline
        - overflow

        - padding
        - page-break
        - pause
        - play-during
        - position
        - right
        - table-layout
        - text-decoration
        - top
        - unicode-bidi
        - vertical-align
        - width
        - z-index
    - a propriedade font-size é herdável e, se definida com uma unidade relativa, o filho irá ter o tamanho do pai mais o valor calculado → ex:
        - p {font-size: 150%;}
        - Texto do parágrafo com um elemento EM nele contido.
        - p irá herdar o tamanho do pai ou do body. se nenhum deles tiver valor definido o valor será 16px que é o padrão dos navegadores. em irá herdar a propriedade de p de tal modo que sua font-size será y+1.5y
- initial e inherit
    - initial: cada propriedade tem um valor inicial, definido pelo navegador. ao se definir que tal propriedade deve receber o valor initial, ela irá receber esse valor definido pelo navegador, e não, por exemplo, herdar o valor do elemento pai
    - o inherit fará com que o elemento filho herde o valor da propriedade que é atribuído ao pai mais próximo dele.
- pseudo-classes
    - representam um estado e a estilização do elemento que tenha aquele estado
    - :link → representa o estado normal de um link que não tenha sido visitado
    - :visited → estiliza links visitados
    - :hover → estiliza elementos quando o mouse estiver sobre eles
    - :focus → quando o elemento estiver em foco, quando o elemento é selecionado pelo mouse ou teclado → geralmente usada um elementos de formulários
      → se um elemento tiver ambos focus e hover, a propriedade de focu só serpa aplicada quando não estiver em hover
    - :checked → elemento radio, checkbox ou option de um select quando estiver marcado/selecionado
    - :first-child → elemento que é o primeiro filho de seu pai
    - :last-child → elemento que é o último filho de seu pai
    - :not(seletor) → vai se aplicar aos elementos que não se encaixem no seletor de argumento
    - estrutura:
      seletor:pseudoClasse {

propriedade: valor;
}

- media queries
  - adaptam a apresentação do conteúdo de acordo com o dispositivo
  - media types → os tipos de dispositivos:
    - All: Para todos os dispositivos.
    - Braille: Para dispositivos táteis.
    - Embossed: Para dispositivos que imprimem em braile.
    - Handheld: Para dispositivos portáteis, geralmente com telas pequenas e banda limitada.
    - Print: Para impressão em papel.
    - Projection: Para apresentações como PPS.
    - Screen: Para monitores ou dispositivos com telas coloridas e resolução adequada.
    - Speech: Para sintetizadores de voz. As CSS 2 tem uma especificação de CSS chamada Aural, onde podemos formatar a voz dos sintetizadores.
    - Tty: Para dispositivos que possuem uma grade fixa para exibição de caracteres, tais como: Teletypes, Terminais e também dispositivos portáteis com display limitados.
    - Tv: Para dispositivos como televisores, ou seja, com baixa resolução, quantidade de cores e scroll limitados.
      → o media type é opcional a não ser que a query esteja usando os operadores not ou only
  - @media mediaType {}→ vai definir qual o media type que irá receber os statements entre os {
  - media features → características de um dispositivo:
    - width
    - height
    - device-width
    - device-height
    - orientation
    - aspect-ratio
    - device-aspect-ratio
    - color
    - color-index
    - monochrome
    - resolution
    - scan
    - grid

    → é possível usar operadores lógicos para combinar requisitos de exibição, podendo juntar media types e media features
  - uma media query será aplicada quando o dispositivo se encaixar no media type e/ou na media feature definida → no caso de uma query usando o *and*, só será exibida se o dispositivo cumprir todos os requisitos

    - é possível definir as mesmas regras para vários dispositivos usando o operador *or*
    - o operador *not* é usado para negar uma query inteira
    - a palavra chave *only* não tem efeito nos navegadores modernos mas previne navegadores antigos que não suportam media queries com media features de aplicar estilos específicos
- calc
    - serve para definir valores calculados para propriedades, com ou sem unidade
    - o principal uso é para misturar unidade → ex: 100% - 20px
    - + e - precisam estar cercados por espaços em branco →
    - ex: 50%-8px❌
      50% - 8px✔
      (os espaços são opcionais para multiplicação e divisão mas recomendados para manter o padrão)
    - / precisa que o segundo número não tenha unidade
    - * precisa que um dos números não tenham unidade
    - é possível usar com cores no formato rgb e hsl
    - quando usado para calcular tamanho de font, é recomendado que um dos valores seja uma unidade relativa para que, quando der zoom na página, ele aumente também
- @font-face
    - permite usar fontes externas em sites
    - sintaxe:
      @font-face{
      font-family: nome;
      src: url/caminho;
      }
      → se o nome ou caminho tiver espaço, é preciso que esteja entre “”
    - src: local(nome da fonte local), url → vai buscar a fonte local e, caso ela não esteja instalada, vai fazer o download a partir da url
    - font-stretch, font-style, font-weight → se definidos, farão com que essa seja usada quando texto se encaixar na mesma propriedade
      → font-weight pode receber um valor entre 1 e 1000 sendo que o normal é 400 e bold é 700
    - unicode-range→ determina o range de caracteres da fonte assim, caso uma página não utiliza caracteres dentro do range definido, a fonte não será baixada
      → pode ser usada para selecionar um único caractere ou vário
      → valor inicial: U+0-10FFFF → https://jrgraphix.net/r/Unicode/
    - src: url() format() → vai indicar ao navegador qual o formato do arquivo da fonte para permitir que ele escolha o formato adequado → "woff", "woff2", "truetype", "opentype", "embedded-opentype", e "svg". → https://transfonter.org/formats (truetype. opentype e woff são os que tem maior suporte nos navegadores)

Nível Avançado

- especificidade css
  - origem e peso:
    !important usuário>!important autor>autor>usuário> padrão do navegador
    (folhas de estilo do usuário podem ser usadas para acessibilidade)
  - cálculo de especificidade:
    - (a, b, c, d)
      - Elemento, pseudo elemento: d = 1 — (0, 0, 0, 1)
      - Classe, pseudo classe, atributo: c = 1 — (0, 0, 1, 0)
      - Id: b = 1 — (0, 1, 0, 0)
      - Estilo inline: a = 1 — (1, 0, 0, 0)
  - se empatar, a que vier por último vence → dentro de uma mesma folha de estilo e entre diferentes. → entre estilo definido dentro do arquivo html (<style>) e uma folha de estilo externa, vence a que for declarada por último → estilo inline vence de ambas
- box-shadow
  - sintaxe= box-shadow:(inset) offset-h offset-v (blur-radius) (spread-radius) (color)
    - inset (opcional) → faz com que a sombra seja interna à caixa, acima do background mas abaixo do conteúdo
    - offset-h → o comprimento horizontal da sombra.
      - >0 = pra direita
      - <0 = pra esquerda
    - offset-v → o comprimento vertical da sombra
      - >0 = pra baixo
      - <0 = pra cima
    - blur-radius (opcional) → quanto maior, mais desfocada será a sombra.
      - 0 = bordas retas
      - não é permitido valor negativo
    - spread-radius (opcional)
      - >0 = a sombra cresce
      - 0 = a sombra terá o mesmo tamanho que o elemento
      - <0 = a sombra encolhe
    - color (opcional)
      - se não for especificada, será o padrão do navegador → normalmente é a cor do texto
      - no Safari, se não for definida a cor a sombra é colocada como transparente
- pseudo-elementos (:before e :after)
  - : || :: → :: é a nova versão mas não tem suporte em versões de IE antes da 9
  - :after → usado para inserir algo depois do conteúdo do elemento
  - :before → usado para inserir algo antes do conteúdo do elemento

    - ambos inserem dentro do elemento em si
    - sintaxe→ seletor:pseudoElemento{
      content:
      css para estilizar:
      }
    - content pode ser texto, imagens
        - no caso de imagem, não é possível redimensionar
        - tem que ter, nem que seja “”
    - o elemento será inline por padrão
    - como o que é inserido não aparece no dom, não é acessível à maioria dos aparelhos de acessibilidade e não deve ser usado para inserir conteúdo que seja crítico
- transições
    - controla a animação quando há mudança de propriedades
    - pode ser aplicado a propriedades animáveis → https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties
    - a regra básica é que se pode transicionar entre valores
    - sintaxe: .seletor {
    - transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay];
      }
        - também pode ser feita especificando cada propriedade separadamente → ex: transition-timing-function
    - transition-property → o que irá mudar, pode ser várias propriedades → as propriedades que não estiverem listadas aqui mudarão automaticamente → o default é all e todas as propriedades animáveis irão receber a animação
    - transition-duration → o tempo entre o estado inicial e o final
        - pode ser em ms ou s
        - se não for especificada, a animação não acontece porque o default é 0s
    - transition-timing-function → especifica uma função para definir como os valores intermediários das propriedades são calculados → https://developer.mozilla.org/en-US/docs/Web/CSS/easing-function
        - linear: constante
        - ease (default): começa devagar, acelera bastante e vai ficando mais devagar no final
        - ease-in: começa devagar, acelera e para
        - ease-in-out: começa devagar, acelera e fica mais devagar no final
        - ease-out: começa bem rápido e vai ficando mais devagar
        - steps(número, direção) → o número tem que ser positivo
            - direção:
                - start= o primeiro step acontece assim que a animação começa
                - end= o último step acontece bem no final da animação
    - transition-delay → define um tempo antes da animação começar

    - definir a transição no elemento em si garante que ela aconteça nas duas direções → ex: se a transição for colocada no :hover do elemento, só irá acontecer quando o mouse for colocado em cima, e não quando for retirado
        - se tiver transição em ambos, a do hover será aplicada quando o mouse entrar no elemento e a do elemento acontecerá quando o mouse sair
- animações
    - sintaxe: animation: animation-name animation-duration animation-timing-function animation-delay animation-direction animation-iteration-count animation-fill-mode animation-play-state
        - também podem ser declaradas separadamente
        - timing-function: ver transições
        - animation-delay→ se o valor for negativo, a animação vai começar começar o valor de segundos no meio do ciclo → delay de -1s vai fazer a animação começar a partir do ponto de 1s da sequência de animação
        - animation-direction → indica em qual ordem a animação será tocada
            - normal (default) → pra frente
            - reverse → de frente pra trás → faz a timing-function ser ao contrário também
            - alternate → troca de direção a cada ciclo, começando com pra frente
            - alternare-reverse → troca de direção a cada ciclo, começando com pra trás
        - animation-iteration-count → quantas vezes a animação vai tocar
            - infinite
            - número → 1=default, pode ser decimal, não pode ser negativo
        - animation-fill-mode → define como o elemento vai ficar depois de ter acabado a animação
            - none (default) → o elemento ficará como é definido fora da animação
            - forwards → o elemento ficará com o estilo do último keyframe da execução → vai depender da combinação do animation-direction e do animation-iteration-count
            - backwards → o elemento ficará com o estilo do último keyframe → vai depender do animation-direction
        - animation-play-state → define se a animação está tocando ou pausada
            - running (default)
            - paused
            - trocar entre paused e running faz com que a animação continue de onde parou
    - @keyframes nomeDaAnimação{}
        - cada keyframe descreve o elemento em um momento da animação
        - é preciso ter duas ou mais
            - 0% ou from → indica o primeiro momento da animação

            - 100% ou to → indica o final
            - outras porcentagens indicaram um ponto intermediário no tempo
    - é possível definir várias animações para um mesmo elemento, desde que não alterem as mesma propriedades
- transformações
    - rotate → rotaciona o elemento
        - se for positivo, rotaciona em sentido horário
        - se for negativo, rotaciona em sentido anti-horário
    - translate → muda a posição do elemento
        - translate(x,y)
        - x>0 = pra direita
        - x<0 = pra esquerda
        - y>0 = pra baixo
        - y<0 = pra cima
    - scale → muda o tamanho do elemento
        - recebe um ou dois números sem unidade (sx, sy)
            - se tiver só 1, sy será o mesmo que sx
            - pode ser definido separadamente: scaleX(); scaleY();
        - funciona como tamanhoDoElemento*valor
- compatibilidade
    - nem todas as propriedades de css tem suporte ou tem suporte idêntico em todos os navegadores
    - https://caniuse.com/ → o site indica quando o suporte é completo, parcial (com prefixo) ou não existe
    - prefixos→ adicionados a propriedades experimentais ou fora do padrão → ajuda a usar propriedades que ainda não tenham suporte completo ou que possam ter sido implementadas ligeiramente diferentes ou com sintaxe diferente em navegadores diferentes
        - -webkit- → chrome, safari, versões novas do opera, android
        - -moz- → firefoz
        - -o- → versões antigas do opera
        - -ms- ie e edge
            - sintaxe→ -prefixo-propriedade: valores → agem praticamente como se fosse uma propriedade
            - como o navegador ignora aquilo que ele não “entende”, cada propriedade com prefixo só será aplicada ao navegador que “entende” aquele prefixo.
            - a versão sem prefixo vem no final e será utilizada pelos navegadores que já implementaram a propriedade completamente.
            - com base nos dados do caniuse, tendo em mente qual a versão atual dos browsers e a popularidade dos browsers, é possível decidir usar ou não um prefixo

        → https://autoprefixer.github.io/

- reset/normalize
  - como cada navegador tem sua própria folha de estilos, podem haver inconsistências ao abrir uma mesma página em navegadores diferentes
  - reset → vai resetar as propriedades dos elementos, removendo toda a formatação feita pelos navegadores
    - o reset vai zerar tudo assim, por exemplo, todos as tags de texto terão o mesmo tamanho e será necessário redefinir valores diferentes para elas
  - normalize → vai tornar os estilos consistentes entre navegadores, mudando apenas aquilo que é diferente entre os navegadores
  - existem ambos já feitos na internet
  - pode-se usar os dois em conjunto, por exemplo, usando um reset para mudar apenas algumas propriedades que serão usadas no projeto
- acessibilidade
  - a boa utilização da estrutura de tags permite que leitores de tela naveguem pela página de modo claro e fácil pela página
  - role → definem o que o elemento é ou faz → landmarks
    - usado para criar pontos de referência utilizados para a navegação
  - tabindex → define se o elemento pode receber o foco quando a navegação é feita com a tecla tab → usado com elementos com os quais o usuário vai interagir
    - <0 → faz com que não seja “focável” pelo teclado, apenas por meio do método focus()
    - =0 → faz com que elementos que normalmente não seriam “focáveis” se tornem focáveis
    - >0 → muda a ordem natural do fluxo com a tecla tab, o elemento que tiver o número maior sendo primeiro. no caso de vários elementos com um mesmo index, seguirá a ordem da posição deles dentro do html
  - title → pode ser usado em qualquer tag de HTML e serve para adicionar uma tooltip com mais informação além daquela que é disponibilizada na página.
    - não deve ser usado para repetir o que já está na página, e sim para adicionar mais informações
  - alt → descrição alternativa para imagens → uma representação direta e detalhada
    - caso a imagem seja apenas decorativa, o alt pode ser deixado vazio mas deve existir porque leitores de tela vão ler o url da imagem se essa não tiver um alt
    - informações que não sejam descritivas devem ser colocadas no title ou como texto na página
  - for→ (label) → serve para “conectar” um item à sua label, usando o id do elemento
    - torna possível selecionar um elemento clicando em sua label
  - section → um bloco ou grupo de um assunto específico → usada para separar quando existe uma importância semântica para isso

        - div, por outro lado, é usada para quando a divisão á apenas visual
- elementos flutuantes
    - float→ tira um elemento do fluxo vertical da página e permite que o que vem depois “flutue” ao lado
        - left ou right → determinam o lado que o elemento vai ficar
    - clear → controla o comportamento de elementos que vêm depois de um elemento que tenha a propriedade float
        - left ou right→ o elemento é empurrado para baixo de elementos que tenham float com esse mesmo valor
        - both→ o elemento é empurrado para baixo de elementos que tenham qualquer valor de float
        - none→ o elemento não é empurrado para baixo