**CSS – Dispondo elementos com Flexbox e Grid**

Link do figma, onde está todo o nosso conteúdo e layout do site: <https://www.figma.com/file/ibWktwVpnog76rMYOdVhks/Dispondo-elementos-com-flexbox-e-grid?node-id=54%3A2358>.

1. **Aula 1 – Flexbox:**
   1. O projeto e as Ferramentas:
      1. Todas as informações que precisamos saber sobre cores, fontes e afins, além de estar no figma colo descrito acima, encontram-se no arquivo readme da pasta.
   2. O Flex Container:
      1. Convertemos o nosso cabeçalho para flex container com o display: flex;.
   3. Guia completo de flexbox | CSS tricks: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>.
   4. Justify-content e align-items:
      1. Ambos são propriedades que alteram o posicionamento dos elementos em um flex container:
      2. Justify-content:
         1. Flex-start: Todos os elementos ficam o mais próximo da esquerda possível:



* + - 1. Flex-end: O contrário do acima:



* + - 1. Space-between: Todos os elementos são distribuídos igualmente dependendo do espaço no container, onde o primeiro elemento sempre estará mais a esquerda e o último sempre mais a direita possível.



* + - 1. Sapce-around: Igual ao acima, mas o espaço sobrando no container se divide também antes do primeiro elemento e depois do último, ou seja, os elementos são separados e espaçados no meio do container. Não distribui o espaçamento igualmente entre os elementos, portanto, entre o menu e a logo tem um espaçamento diferente de entre o menu e a borda da página:



* + - 1. Center: Joga tudo pro centro



* + - 1. Space-evenly: Parecido com o around, mas distribuí o espaçamento igualmente entre os elementos:



* + 1. Align-items: Responsável pelo alinhamento do conteúdo da flex container:
       1. Center: Deixa todos alinhados no centro do container:



* + - 1. Flex-start: Todos para o topo to container:



* + - 1. Flex-end: O contrário do acima:



* + - 1. Stretch: Faz os elementos ocuparem a altura toda do container:



* 1. O que aprendemos:
     1. O que é um flex-container:
        1. Flex container é o elemento que recebe grande parte das propriedades de posicionamento para suas tags filhas.
     2. As propriedades de posicionamento justify-content e align-items:
        1. justify-content distribui o espaço restante do flex container entre suas tags filhas e align-items alinha verticalmente as tags filhas, ou seja, são propriedades de posicionamento horizontal e vertical respectivamente.

1. **Aula 2 – Mais Funcionalidades do Flexbox:**
   1. Terminando o Cabeçalho:
      1. Adicionamos ícones no lugar dos escritos menu e notificação utilizando para isso a font que está na pasta font do assets.
      2. Para utilizar esses ícones precisamos colocar uma tag <i></i> sendo essa uma tag de pseudo-elemento, para que ela seja substituída pelo ícone que desejamos colocar:
      3. Para importar a font fazemos isso no próprio CSS com a anotação:

@font-face {

    font-family: 'icones';

    src: url(../font/icones.ttf);

}

* + 1. E para substituir a tag <i> pelo ícone desejado, basta colocar um i::before{contente=”\número do ícone encontrado no site”}:

.cabecalho\_\_notificacao i::before {

  content: "\e906";

  font-size: 24px;

}

* + 1. Dessa forma ao invés do escrito ‘notificação’ agora temos um sininho. Como são fontes, podemos alterar o tamanho delas com o font-size normalmente.
    2. Como esses ícones foram colocados em botões no HTML, fica assim no resultado final:

<button class="cabecalho\_\_menu" aria-label="Menu"><i></i></button>



* + 1. O aria-label=”Menu” serve para indicar para os leitores de tela que isso é um botão de menu, já que não tem mais o escrito dentro dele, somente a tag <i>. **Essa é uma correção de acessibilidade que precisamos fazer sempre que formos utilizar essa técnica!!!!**
  1. Para saber mais sobre esse tipo de ícone na web: <https://www.alura.com.br/artigos/como-utilizar-icones-em-paginas-web>.
  2. O flex-wrap e flex-direction:
     1. Durante a utilização do Emmet (facilitador de escrita), podemos colocar [atributo=”valor”] para definir o valor enquanto estamos escrevendo para não ter que fazer isso manualmente depois:

nav.menu-lateral>img.menu-lateral\_\_logo+a[href=#].menu-lateral\_\_link\*6

Resultado:

<nav class="menu-lateral">

    <img src="assets/img/logo.svg" alt="" class="menu-lateral\_\_logo">

    <a href="#" class="menu-lateral\_\_link">Início</a>

    <a href="#" class="menu-lateral\_\_link">Vídeos</a>

    <a href="#" class="menu-lateral\_\_link">Picos</a>

    <a href="#" class="menu-lateral\_\_link">Integrantes</a>

    <a href="#" class="menu-lateral\_\_link">Camisas</a>

    <a href="#" class="menu-lateral\_\_link">Pinturas</a>

  </nav>

* + 1. Flex-wrap: quebra de linha dos elementos:
       1. Wrap: Efetua a quebra de linha:



* + - 1. Nowrap: Impede a quebra de linha. Geralmente é o padrão:



* + - 1. Wrap-reverso: :
    1. Felx-direction: Determina a direção que nossos elementos irão seguir:
       1. Row: Linhas. É o que vem por padrão quando colocamos o flex como display. Ele irá alocar todos os elementos um ao lado do outro.
       2. Column: Colunas. Aloca todos os elementos um abaixo do outro:



* + - 1. Quando invertemos a direção, os valores de justify-content e align-items também são invertidos!!!
    1. Quando convertemos uma sessão para flexbox, todos os elementos dentro dela também se tornam caixas, portanto, cada item possuí propriedades similares a box toda, como align-items, mas, como se trata somente de 1 elemento, utilizamos o align-self para ele:

.menu-lateral\_\_logo {

  align-self: center;

}



* + - 1. Podendo assim ajustar somente a logo no centro, por exemplo. Lembrando que só ajustamos na horizontal utilizando o align e não o justify porque invertemos a direção do flexbox com o flex-direction, logo, todos os valores de direção são invertidos junto. Para ter esse efeito com o flex-direction em row, precisaríamos ter usado o justify-self e não o align-self como foi feito.