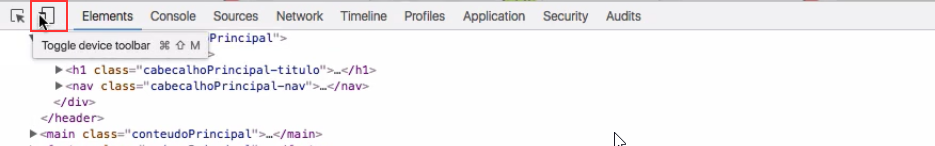
**FLEXBOX - Alura**

**Transcrição**

O próximo passo é organizar o site também para acesso **Mobile** e para visualizar a página neste formato é preciso abrir, no Chrome, o **Dev Tools** ou através do atalho F12no Windows ou Command + Shift + M no MAC e selecionar Toggle device toolbar:



Este caminho mostra uma tela diminuta com tamanho de 320 px e o resultado que vemos é um site totalmente desestruturado para celular:



O menu ideal deve ter o seguinte aspecto:



Primeiro, vamos dar um Inspect elements e analisar o código referente ao menu. A nav é o objeto pai que comporta dentro de si diversos elementos. Para arrumar este trecho vamos inserir o pai, .cabecalhoPrincipal-nav, e dentro disso adicionamos a propriedade display-flex que por padrão distribui os elementos lado a lado. Como falta mudar a direção dos itens acrescentamos também o flex-direction: column:

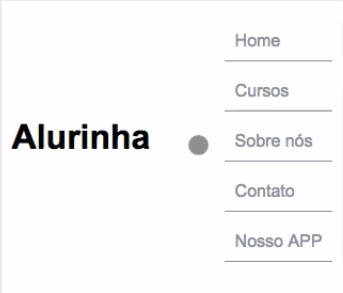
.cabecalhoPrincipal-nav {

flex-direction:column;

}

Vamos inserir um comentário, o /\*parte mobile\*/, no arquivo flexbox.css, justo acima desse código, apenas para lembrar que aquilo que estiver abaixo do comentário refere-se a **Mobile**.

A aparência da página fica da seguinte maneira:



Ou seja, ainda está distante do que buscamos! Isso acontece porque já existe um flex inserido no container. Portanto, vamos adicionar o elemento .cabecalhoPrincipal-container e o flex-direction:column para ajeitar os itens:

.cabecalhoPrincipal .container{

flex-direction:column;

}

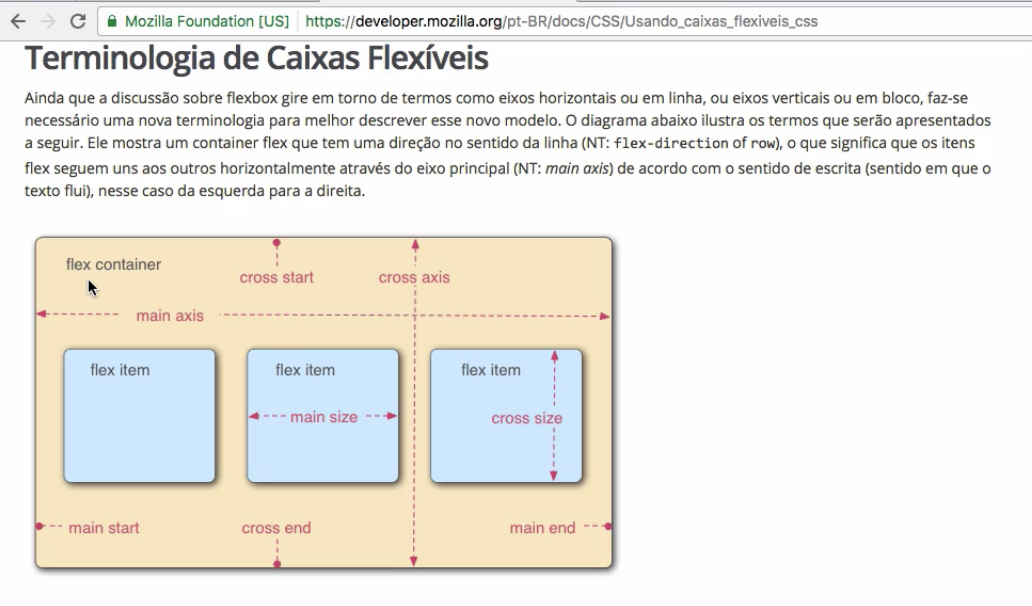
Agora, dando um *reload* na página temos:



O **Mobile** está quase pronto! O problema é que os itens do menu estão centralizados e não espaçados como queremos. Observando o Inspect vemos que os elementos possuem um align-itens que uma vez desativado produz o seguinte resultado:



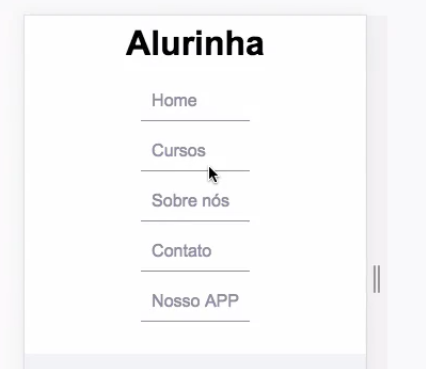
Para entender o que realmente está acontecendo vamos consultar a documentação do site [**MDN**](https://developer.mozilla.org/pt-BR/) do **Mozilla**, acessando a página encontramos o seguinte gráfico:



Quando o display-flex é introduzido em algum elemento ele vira automaticamente um flex-container. Já os flex itens são aqueles elementos que correspondem aos links, a home, ao app, contato, etc.

Repare na imagem acima, o **main axis** refere-se ao eixo principal e a propriedade referente a este eixo é a justify-content: space-between. O **cross axis** equivale ao eixo que cruza e sua propriedade é a align-itens: center. Assim, a justify-content está para o eixo principal como o align-itens para o eixo cruzado! Mas os eixos vertical e horizontal mudam quando alteramos o flex-direction, portanto, é preciso estar atento em relação ao uso de suas respectivas propriedades!

Por enquanto o **Mobile** está com o seguinte aspecto:



Vamos pensar um pouco nos elementos que utilizamos até o presente momento! O que alinha os elementos no centro é o sentido, isto é, a direção column. O align ajuda na distribuição dos objetos deixando os itens comprimidos e não expandidos. Assim, podemos desativar o align do menu e no .cabecalhoPrincipal-container nós teremos apenas o flex-direction:column e o align-items: initial:

.cabecalhoPrincipal .container{

flex-direction:column;

align-items: initial;

}

O único detalhe é que o texto segue alinhado a esquerda, portanto, para centralizar o conteúdo é preciso inserir o text-align: center:

.cabecalhoPrincipal .container{

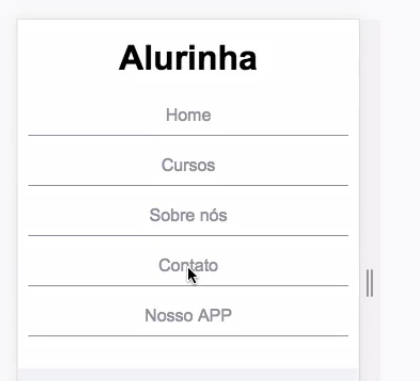
direction:column;

align-items: initial;

text-align: center;

}

Com isso finalmente conseguimos centralizar os itens e temos como resultado o layout desejado:



Nesta aula aprendemos sobre o flex-container e também sobre o display flex que faz todos os seus itens filhos, os flex-itens, se ajeitarem e se ajustarem. Pensando nos eixos, também vimos que o justify-content e o align-items são propriedades relacionadas a eixos específicos.