FlexBox – Alura

1. Aula 1 – Introdução ao FlexBox e Fazendo o Cabeçalho:

1.1. Flexbox trata principalmente sobre realocação e posicionamento de elementos do nosso site por CSS, deixando tudo mais organizado e melhor construído.

1.2. O ideal, para que não fique muita coisa no css de estilização é criar um novo arquivo chamado flexbox onde você irá fazer todas as modificações de posicionamento.

1.3. Podemos começar colocando o cabeçalho e navegação em display: inline-block;

1.4. Dando um *vertical-align: Middle;* os nossos elementos se alinham no centro verticalmente.

1.5. Porém, tudo isso que foi visto, não é uma boa prática, pois precisamos utilizar "números mágicos" para alinhar da maneira que queremos, sendo eles números que não tem embasamento, colocamos pq achamos que ficou bom.

1.6. A primeira coisa a se fazer quando vamos usar flexbox, é pensar quais são os elementos que queremos alinhar e posicionar.

1.7. O melhor a se fazer é colocar o *display: flex;* no elemento mãe de tudo o que você quer fazer o posicionamento.

1.8. Setando isso, podemos usar o *align-itens: center;* para alinhar todos os itens desse elemento mãe verticalmente.

1.9. Podemos também colocar o espaço de todo o contêiner mãe entre os dois elementos filhos dela usando o comando *justify-content: space-between;* dessa forma, jogando os dois elementos de um cabeçalho para as laterais, por exemplo.

1.10. Para alinhar itens de uma barra de navegação verticalmente, basta colocar como *display: flex;* e ele irá alinhar automaticamente.

1.11. Nem todos os navegadores podem aceitar muito bem o flexbox. O site: https://caniuse.com diz quais são as ferramentas que você pode utilizar em quais navegadores.

2. Aula 2 – Fazendo o Footer e Controlando Melhor os Elementos:

2.1. Podemos colocar espaçamento em volta dos elementos com *justify-content: space-around;* para deixar os elementos mais disbribuidos.

2.2. O *display: flex;* também faz com que os filhos de um elemento fiquem todos com a mesma altura, facilitando na hora da estilização.

2.3. Podemos selecionar a direção que queremos que o flex seja colocado. Ele por padrão vem na horizontal (*flex-direction: row;*), mas se utilizarmos o *flex-direction: column;* todos os elementos ficarão um abaixo do outro.

2.4. Porém, o problema de colocar em colunas é que mesmo que você defina o tamanho do elemento pai, ele vai seguir para baixo indefinidamente, contudo, podemos utilizar o *flex-wrap: wrap;*, que basicamente diz que quando chegar no tamanho do elemento pai ele irá quebrar a linha e continuar na próxima.

2.5. Além disso, podemos utilizar o *flex-flow: colunm wrap;* e, desse modo, substitui os dois comandos acima, pois ele faz as duas coisas.

3. Aula 3 - Grid Principal e Limitações do FlexBox:

3.1. Podemos fazer um grid utilizando os conhecimentos que já temos de flex. Basta colocar todos os itens de uma lista como display: flex;, definir um tamanho para os elementos da lista, caso estejam muito aglomerados e, por fim, colocar o espaçamento entre os itens, dando uma margem para eles.

3.2. O problema é que quando não se tem todos os elementos necessários para deixar o grid 100% completo, fica um vão entre os itens da grid, onde os 2 itens são jogados para os cantos, fugindo da ordem de ser um do lado do outro no caso onde tem 4 elementos na linha do grid.

3.3. Para concertar isso, nós infelizmente precisaremos utilizar contas e retirar o espaçamento between do justify-content.

3.4. Para selecionar apenas itens múltiplos de algum número e fazer uma estilização somente nesses elementos de uma lista no grid, por exemplo, utilizamos o *.nomeDaClasse:nth-child(4n){config}*, nesse caso, a cada 4 elementos essa configuração será adicionada.

4. Aula 4 – Arrumando os Elementos com Flex para Mobile:

4.1. Podemos mudar a visualização do site para a versão mobile apertando CTRL+SHIFT+M no chrome, onde ele muda a visualização a partir do devtools (CTRL+SHIFT+I).

4.2. Criamos uma nova “sessão” no nosso documento flexbox.css para a versão mobile.

4.3. Atualizamos os parâmetros do CSS para deixar adequado para a visualização mobile.

4.4. Site com a documentação do flexbox: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

4.5. Justify-content mexe com o eixo principal(x) do flex container, já o align-itens mexe com o eixo cruzado do posicionamento(y).

4.6. Porém, essas informações se invertem quando você altera o fluxo do flex com o flex-direction.

4.7. Sites responsivos, em sua grande maioria, todos os itens ficam um abaixo do outro, ao invés de 1 ao lado do outro, como em desktops.

4.8. Depois de todas as alterações feitas, quando voltar o site para tamanho desktop, verá que tudo continua no mesmo estilo, todas as alterações se manteram. Para que isso não ocorra, utilizamos o @media(max-width: 768px){todas as alterações que acabamos de fazer}, desse modo, todos os dispositivos que tiverem tela de até no máximo 768 pixels de largura, o site será mantido nesse layout mobile, mas assim que for maior que isso, o site automaticamente volta para a configuração de desktop.

4.9. Nós podemos com o flex, alterar a ordem dos itens do html, mas sem alterar o html, desse modo, se o site tiver um link de apps para poder baixar direto da playstore, podemos fazer com que esse item apareça em primeiro lugar na página para usuários que estejam acessando via mobile, uma vez que o usuário terá uma melhor experiência utilizando os apps do que o site propriamente. Contudo, quando o usuário abrir o site pelo desktop, a ordem dos botões da barra de navegação continua a mesma.

4.10. Para fazer isso, nós podemos colocar um *order: -1;* na parte de alterações mobile para a classe específica do nosso item a ser alterado (por isso seria interessante criar 2 classes para esse item: a padrão, igual de todos, e a específica para poder alterar a ordem quando mobile), uma vez que ele seja um flex item, ou seja, que o elemento mãe seja um *display: flex;*. Por padrão, todos os flex itens possuem ordem 0, quando queremos alterar a posição de um deles para que fique o mais pra cima/primeiro possível, nós precisamos colocar um número mais baixo, ou seja, -1.

4.11. Outro site interessante para consulta de truques e guias para FlexBox: https://css-tricks.com/snippets/css/a-guide-to-flexbox/

4.12. Todas as propriedades que vimos até agora, exceto o *order: 0;*, são utilizadas/colocadas no flex container. A partir de agora iremos ver as propriedades que assim como o *order: 0;* são aplicadas aos flex itens.

4.13. Utilizando o *flex-grow: 1;* podemos fazer com que os flex itens de um flex container cresçam até ocuparem o resto do espaço disponível no flex container, desse modo ficandodo mesmo tamanho e organizando melhor as coisas.

4.14. Outra propriedade é o *flex-shrink: n;* que basicamente diz o quanto esse flex iten vai diminuir conforme o tamanho da tela muda. 0: não diminui de jeito nenhum, 1: é o padrão, 2: duas vezes mais e assim por diante...

5. Aula 5 – Mais sobre FlexBox e Desafios:

5.1. Podemos utilizar um atalho para o flex-shrink e flex-grow: apenas *flex: ;*. Colocando apenas o flex nós podemos colocar 2 argumentos sendo o primeiro do flex-grow e o segundo do flex-shrink: *flex: 1 2;*, desse modo, eles terão um *flex-grow: 1;* e *flex-shrink: 2;*.

5.2. *Flex-basis:* ***n****px;* serve para definir um tamanho base fixo de largura para os elementos. Basicamente possui a mesma funcionalidade do *width:* ***n****px,%;* para definir largura.

5.3. Site de game para treinar o que aprendemos: http://flexboxfroggy.com . Basicamente precisamos alinhar o sabe em cima da planta utilizando comandos do flexbox.

5.4. Site de joguinho com o mesmo princípio do sapo, mas com torres que precisam destruir as bolinhas, portanto, você precisa coloca-las nas parte que elas tem maior área de tiro: http://www.flexboxdefense.com .