Mão na massa

1) Ajustando a altura da tela com CSS

Você está desenvolvendo um site e deseja que ele ocupe 100% da altura da tela do usuário, sem deixar espaços em branco acima ou abaixo do conteúdo. Para isso, você precisa ajustar o elemento `body` do seu CSS para garantir que ele preencha toda a altura da tela. Utilize a propriedade `height` e o valor `100vh` para conseguir este efeito. Teste o resultado usando a ferramenta "Inspecionar" do navegador para verificar se o `body` realmente ocupa toda a altura da tela.

---

2) Controlando o tamanho de elementos com Box Sizing

Imagine que você está trabalhando em um projeto web e precisa garantir que todos os elementos filhos dentro do `body` não ultrapassem os limites, especialmente ao adicionar bordas e paddings. Para isso, você deve utilizar a propriedade `box-sizing` com o valor `border-box` no elemento `body`. Após aplicar esta configuração, inspecione o layout para garantir que nenhum elemento filho ultrapasse o limite do `body` e os seus próprios tamanhos definidos, mesmo ao adicionar bordas e paddings.

---

3) Criando um layout sem scroll

Você recebeu um design do Figma para um site que deve caber em uma única tela, sem necessidade de scroll. O design inclui dois blocos de conteúdo sobre um fundo preto. O desafio é criar um layout CSS que reproduza essa estrutura, garantindo que todo o conteúdo caiba em uma tela, sem scroll. Utilize as propriedades CSS adequadas para posicionar os blocos de conteúdo lado a lado e centralizá-los na página, mantendo o design conforme especificado no Figma.

---

4) Flexbox: alinhando textos e imagens

Você está desenvolvendo um site e precisa posicionar textos e uma imagem lado a lado, adaptando-se a diferentes tamanhos de tela. Utilize Flexbox para alinhar um bloco de texto à esquerda e uma imagem à direita dentro de um `<main>`.

```
<main class="container">
  <p class="texto">Texto aqui...</p>
  <img class="imagem">
</main>

.container {
  display: flex;
  justify-content: space-between;
}
```

5) Flexbox: Centralização vertical

Seu desafio é centralizar verticalmente o conteúdo de um container usando Flexbox. Você tem uma `<div>` com a classe `.container` contendo vários itens.

```
<div class="container">
  <div>Item 1</div>
  <div>Item 2</div>
  <!-- Mais itens aqui -->
</div>

.container {
  display: flex;
  align-items: center;
```

```
 height: 300px; /* Altura ajustável conforme necessário */
}
```

6) Flexbox: responsividade e alinhamento

Crie uma página web responsiva onde os elementos se ajustam em diferentes tamanhos de tela, evitando margens fixas. Utilize Flexbox para alternar entre dispor elementos em uma linha ou coluna.

```
<div class="responsivo-container">
 <div>Conteúdo 1</div>
 <div>Conteúdo 2</div>
 <!-- Mais conteúdos -->
</div>

.responsivo-container {
 display: flex;
 flex-direction: row;
 flex-wrap: wrap;
}
```

Opinião do instrutor

1) Ajustando a altura da tela com CSS

- Abra o arquivo CSS do seu projeto.
- Localize a seção onde o estilo do `body` é definido.
- Adicione a propriedade `height` com o valor `100vh`. Isso define a altura do `body` para ser 100% da altura da janela de visualização (viewport height).

- Salve as alterações e abra a página no navegador.
- Utilize a ferramenta "Inspecionar" para verificar se o `body` agora ocupa toda a altura da tela.

Código:

```
body {
  height: 100vh;
  /* Outras propriedades do body, se houver */
}
```

2) Controlando o Overflow com Box Sizing

- Abra o arquivo CSS do seu projeto.
- No estilo do `body`, adicione a propriedade `box-sizing` com o valor `border-box`.
- Isso garante que a largura e altura dos elementos filhos sejam calculadas incluindo bordas e paddings, evitando que eles ultrapassem os limites do `body`.
- Salve as alterações e teste o layout adicionando bordas e paddings aos elementos filhos para verificar se eles permanecem dentro dos limites do `body`.

Código:

```
body {
  box-sizing: border-box;
  /* Outras propriedades do body, se houver */
}
```

3) Criando um layout sem scroll

- Abra o arquivo CSS do seu projeto.

- Utilize flexbox ou grid para criar um layout que acomode os dois blocos de conteúdo lado a lado.
- Certifique-se de que o `body` ou o contêiner principal tenha `height: 100vh` e `width: 100%` para ocupar toda a tela.
- Aplique estilos aos blocos de conteúdo para alinhá-los e centralizá-los conforme o design do Figma.
- Teste o layout para garantir que todo o conteúdo caiba na tela sem a necessidade de scroll.

Código:

```
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  width: 100%;
}
/* Estilos para os blocos de conteúdo conforme o design do Figma */
```

4) Flexbox: alinhando textos e imagens

- Adicione a propriedade `display: flex;` ao elemento `<main>` para ativar o Flexbox.
- Use `justify-content: space-between;` para posicionar o texto à esquerda e a imagem à direita.
- Aplique estilos adicionais conforme necessário para melhorar o layout.

5) Flexbox: Centralização vertical

- Acesse o arquivo CSS e selecione o container usando `.container { ... }`.
- Adicione `display: flex;` para ativar o Flexbox.

- Utilize `align-items: center;` para centralizar os itens verticalmente no container.
- Verifique o resultado no navegador para assegurar que o alinhamento esteja correto.

6) Flexbox: responsividade e alinhamento

- No seu CSS, selecione o container principal e defina `display: flex;`.
- Use `flex-direction: row;` ou `column;` para ajustar a direção dos itens.
- Adicione `flex-wrap: wrap;` para permitir que os itens se ajustem e mudem de linha conforme o tamanho da tela.
- Teste em diferentes tamanhos de tela para garantir a responsividade.