UniSENAI

Trilha Front-end

Orientação Técnica 06 - FlexBox/Grid/Media Query

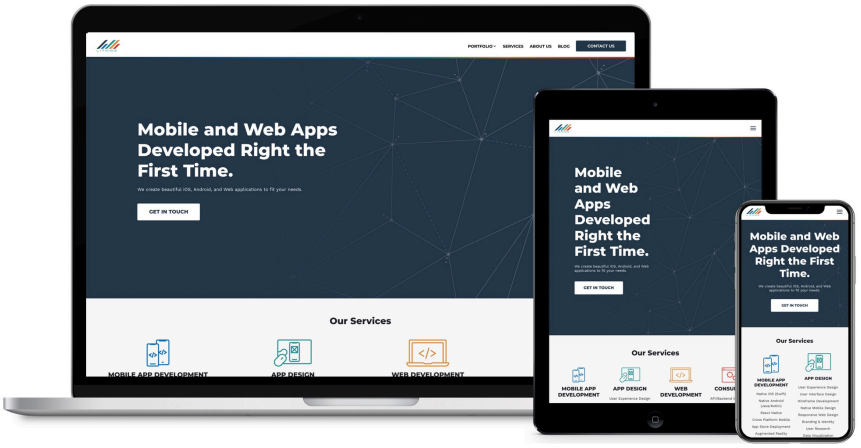
INDICADORES

- Analisa e avalia o funcionamento de computadores e periféricos em ambientes computacionais.
- Codifica programas computacionais utilizando lógica de programação e respeitando boas práticas de programação.
- Desenvolver sites web com elementos multimídia.
- Desenvolver capacidades linguísticas de modo a saber usar adequadamente a linguagem oral e escrita em diferentes situações e contextos.
- Conhecer o caráter do conhecimento científico aplicando a metodologia científica e utilizando redação acadêmica na realização da pesquisa, na escolha de métodos, técnicas e instrumentos de pesquisa.
- Utilizar estruturas de dados definindo-as e aplicando-as adequadamente nos programas.
- Compreender e aplicar técnicas de relações humanas visando o desenvolvimento da liderança e relacionamento em equipe, preservando aspectos éticos e organizacionais
- Projetar e desenvolver interfaces de sistemas utilizando modelos de desenvolvimento e respeitando normas de ergonomia de software.
- Gestão das atividades
- Cumprimento dos prazos

Responsividade

Nos primórdios da internet, as páginas dos websites eram criadas para serem visualizadas em telas que possuíam pouca diferença em seus tamanhos, mas caso o usuário tivesse uma tela menor do que o esperado, ocorriam distorções na tela, como barras de rolagem indesejadas, linhas de tamanho longo, blocos em branco nas telas e etc.

Ao longo do tempo, as tecnologias para se alcançar uma melhor responsividade (Boa visualização independente da tela) vêm sendo melhoradas, deixando de se usar valores fixos para as medições dos componentes em tela e reajustando a visualização de forma dinâmica.



FlexBox

O conceito principal de flexbox se baseia em uma “caixa” (comumente chamada de container) que contém itens dentro, e esta caixa tem a habilidade de manejar a altura e largura de itens contidos em seu interior, e também o espaço entre eles. Este container é composto por dois eixos: principal e secundário.

```
.container {
      display: flex;
      flex-direction: row; /* Alinha os itens em linha (horizontalmente) */
      flex-wrap: wrap; /* Permite que os itens quebrem para a próxima linha */
      justify-content: space-between; /* Distribui o espaço entre os itens */
      align-items: center;
    }
```

- **Display:** define o contexto flex para todos os itens diretos dentro deste contêiner;
- **Flex-direction:** estabelece o eixo principal. Por default a flex-direction é row, ou seja, em linha, porém, a propriedade pode receber também o valor column (coluna). Caso o flex-direction seja definido como row, então o eixo principal é o horizontal e o secundário é o vertical. Se o flex-direction for definido como column, então o eixo principal passa a ser o vertical e o secundário é o horizontal;
- **Flex-wrap:** os flex-itens tentarão se encaixar em uma linha apenas, é possível alterar isso com a propriedade flex-wrap;
- **Justify-content:** Define o alinhamento no eixo principal e ajuda a distribuir o espaço extra;
- **Align-items:** Define o alinhamento no eixo secundário.

Grid

O Grid é um sistema de layout bidimensional, que permite controlar tanto as linhas quanto as colunas de forma flexível, enquanto o Flexbox é um sistema de layout unidimensional, que é usado principalmente para controlar o alinhamento e a distribuição dos elementos em uma única direção.

O Container é o elemento HTML onde o display grid é aplicado, permitindo definir os parâmetros do grid, como o número de linhas e colunas, espaçamento

entre elas e posicionamento dos itens dentro delas.

Propriedades para o grid container

```
.container {
  display: grid;
  grid-template-columns: 50px 50px 50px 50px; /* Define quatro colunas de 50 pixels cada */
  grid-template-rows: auto; /* Define a altura automática das linhas */
  grid-template-areas: /* Define o layout de grid especificando as áreas */
    "header header header header" /* Define a área do cabeçalho */
    "main main . sidebar" /* Define as áreas do conteúdo principal e da barra lateral */
    "footer footer footer footer"; /* Define a área do rodapé */
}
```

- **display:** Define o elemento como um container grid;
- **grid-template-columns:** Define as colunas do grid;
- **grid-template-rows:** Define as linhas do grid;
- **grid-template-areas:** Define o layout do grid usando o nome das áreas que são especificadas na propriedade;

Media Query

Media queries são expressões que envolvem características de mídia de um dispositivo e ajusta o CSS aplicado. Os browsers ou aplicações leem as expressões definidas nas queries e caso o dispositivo se encaixe em alguma delas, o CSS é aplicado de acordo com as propriedades definidas na query.
Podemos utilizar também Media Feature para diferenciar um dispositivo de outro, descrevendo as características de cada um. Por exemplo, podemos informar que determinado estilo deve ser aplicado somente a dispositivos com largura máxima de 600px utilizando a feature max-width: 600px. É possível utilizar outras features como altura (height) e orientação (orientation), este último, aceitando valores como portrait para modo retrato e landscape para modo paisagem.

Tornando o Site Coldigo Geladeiras Responsivo

Comece adicionando a seguinte instrução no começo do documento site.css

1. Comece adicionando a seguinte instrução acima do body no começo do documento site.css

```css
@charset "UTF-8";

* {
    margin: 0;
    box-sizing: border-box;
    padding: 0;
}

body{
    margin: 0 auto;
    font-family : "Century Gothic", sans-serif;
}
```

2.Agora logo abaixo da nav adicione a nav div

```css
nav{
    clear: both;
    width: 100%;
    margin: 0px auto;
    height: 55px;
}

nav div {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
}
```

3.Agora faça as seguintes alterações em:

```css
nav a {
    color: white;
    text-decoration: none;
}

nav .opcao {
    background-color: #b1ccf5;
    padding: 10px 0;
    margin: 0 15px;
    font-size: 1.5em;
    width: 20%;
}
```

4.Mude as configurações da section #topo

```
section #topo {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
    margin: 35px auto;
}
```

5.Agora no final do documento site.css vamos adicionar o media Query

```
@media screen and (max-width: 900px)
{

}
```

6.Agora vamos passar a instrução dentro do @media de como as telas que obedecem a media query devem ficar:

```
.geladeira {
    display: none;
}
header img {
    display: none;
}
header {
    height: 130px;
    background-color: #008eb6;
}
header h1 {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 40%;
    width: 100%;
    color: #ffffff;
    font-size: 30px;
    margin: 35px auto;
}
```

7.Ainda dentro do @media insira:

```
section p {
    text-align: left;
    clear: both;
    text-align: left;
}
nav .opcao {
    background-color: #b1ccf5;
    font-size: 15px;
    width: 20%;
}

h2 {
text-align: center;
font-size: 40px;
```

Conforme instruído no começo, use os recursos do navegador para testar a responsividade de seu site.

Você pode treinar e estudar mais a fundo em alguns links:

https://origamid.com/projetos/flexbox-guia-completo/

https://flexboxfroggy.com/

https://mastery.games/flexboxzombies/

https://codingfantasy.com/games/flexboxadventure

https://cssgridgarden.com/

entre outros, espero que entendam que precisamos praticar para podermos assimilar os conhecimentos.