Projeto Estrelas Fora da Caixa

Relatório de construção de Código

Este é um relatório de construção de código usado como critério de avaliação para participação do projeto Estrelas Fora da Caixa, o conteúdo se dará pela explicação das ferramentas utilizadas para a construção deste projeto, sendo elas, HTML, CSS e JAVASCRIPT.

Foi proposto para os participantes do programa Estrelas fora da Caixa, a implementação de uma página que mostra aleatoriamente quatro personagens da série Rick e Morty e seus nomes, e sempre que a página foi atualizada, outros quatro personagens deverão ser apresentados. Como acréscimo ao desafio, adicionei o estado de vida dos personagens.

Ao iniciar-se a criação do conteúdo é necessário a criação de uma nova pasta, onde será armazenada todos os documentos produzidos. Todo o conteúdo produzido para este desafio, foi construído utilizando o editor de código-fonte da Microsoft, Visual Studio Code.

HTML

O HTML não é uma linguagem de programação, ele é utilizado para construção do “esqueleto” do site.

Inicia-se o arquivo em HTML com a parametrização de algumas informações, como tipo de documento e idioma da página, logo após foi inserido o elemento <head> seguidos dos elementos <meta> utilizados para configuração universal dos acentos, compatibilidade de navegadores e adaptação de tela em outros dispositivos, em seguida o elemento <title> para definir o nome do documento na aba do navegador, o elemento <link> utilizado para fazer a conexão entre o documento HTML e o CSS, e em seguida, o elemento <script> empregado para conectar o documento HTML ao documento em JavaScript. Todos esses elementos foram dispostos da seguinte forma:

<!DOCTYPE HTML> **-------------------------------------------------------------------------------- TIPO DE DOCUMENTO**

<HTML lang=”pt-br”> **----------------------------------------------------------------------------- IDIOMA QUE A PÁGINA SERÁ ABERTA**

<head> **--------------------------------------------------------------------------------------------- ABERTURA DA TAG PARA OS PARÂMETROS DE CONFIGURAÇÃO**

<meta charset=”UTF-8”> **--------------------------------------------------------------------- PARÂMETROS DE CONFIGURAÇÃO DE ACENTOS UNIVERSAL**

<meta http-equiv=”X-UA-Compatible” content=”IE-edge”> **------------------------- PARAMETRIZAÇAO DE COMPATIBILIDADE DE NAVEGADORES**

<meta name=”viewport” content=”width=device-width, initial-scale=1.0”> **------ PARAMETRIZAÇÃO DE TELA PARA DISPOSITIVOS MENORES**

<title>Rick and Morty</title> **-------------------------------------------------------------------- TÍTULO DE SITE A SER EXIBIDO NA ABA DO NAVEGADOR**

<link rel=”stylesheet” href=”css/style.css”> **-------------------------------------------------- LINK DO DOCUMENTO HTML COM O DOCUMENTO CSS**

<script scr=”script/script.js” defer></script> **----------------------------------------- LINK DO DOCUMENTO HTML COM O DOCUMENTO JAVASCRIPT**

</head> **-------------------------------------------------------------------------------------------- FECHAMENTO DA TAG DE CONFIGURAÇÃO**

Após a estruturação dos parâmetros do documento, da-se início ao corpo do documento HTML pelo elemento <body>, contento dentro deste elemento, a TAG <h1> utilizado para o título da página no navegador, e <h2> utilizado como subtítulo. Em seguida o elemento <div> foi utilizado para a construção de blocos, para que fosse possível arquitetar o código em JAVASCRIPT e CSS, neste elemento foi utilizado o sistema de classes (*class*) para que o CSS acessasse esse documento.

No desafio foi proposto que a página mostrasse a imagem de quatro personagens, para isso foram criados quatro blocos, usando a TAG <div> e dentro de cada bloco, contém o elemento <img>, para a busca da foto do personagem e a “id” que será utilizada no JAVASCRIPT, o elemento <p> para o nome do personagem, também com um “id” e outro elemento <p> para o estado de vida do personagem com um “id” e também foi atribuído uma *class*, para que fosse possível a aplicação do CSS nessa frase. A TAG <p> é utilizada para escrever uma frase.

E por último, foi implementado o elemento <button> para a criação de um botão de atualização da página, com o intuito de gerar mais interatividade entre a página e o usuário. Também foi atribuída a esta TAG um “id”.

<body> **----------------------------------------------------------------------------------- TAG de início da página**

<h1>Rick and Morty</h1> **-------------------------------------------------------- Título exibido na página**

<h2>Conheça os personagens da série Rick e Morty</h2> **--------------- Subtítulo exibido na página**

<div class=”container”> **------------------------------------------------------------ Abertura do bloco de informações**

<div> **-------------------------------------------------------------------- Abertura do primeiro bloco de informações do personagem 1**

<img id=’character-image1’> **-------------------------------------- TAG para apresentar a foto do personagem 1**

<p id=’character-name1’> **------------------------------------------ TAG para apresentar o nome do personagem 1**

<p class=”status” id=’character-status1’> **---------------------- TAG para apresentar o estado de vida do personagem 1**

</div> **-------------------------------------------------------------- Fechamento do primeiro bloco de informações do personagem 1**

<div> **------------------------------------------------------------------ Abertura do segundo bloco de informações do personagem 2**

<img id=’character-image2’> **-------------------------------------- TAG para apresentar a foto do personagem 2**

<p id=’character-name2’> **------------------------------------------ TAG para apresentar o nome do personagem 2**

<p class=”status” id=’character-status2’> **---------------------- TAG para apresentar o estado de vida do personagem 2**

</div> **-------------------------------------------------------------- Fechamento do primeiro bloco de informações do personagem 2**

<div> **-------------------------------------------------------------------- Abertura do terceiro bloco de informações do personagem 3**

<img id=’character-image3’> **-------------------------------------- TAG para apresentar a foto do personagem 3**

<p id=’character-name3’> **------------------------------------------ TAG para apresentar o nome do personagem 3**

<p class=”status” id=’character-status3’> **---------------------- TAG para apresentar o estado de vida do personagem 3**

</div> **------------------------------------------------------------- Fechamento do primeiro bloco de informações do personagem 3**

<div> **---------------------------------------------------------------------- Abertura do quarto bloco de informações do personagem 4**

<img id=’character-image4’> **-------------------------------------- TAG para apresentar a foto do personagem 4**

<p id=’character-name4’> **------------------------------------------ TAG para apresentar o nome do personagem 4**

<p class=”status” id=’character-status4’> **---------------------- TAG para apresentar o estado de vida do personagem 4**

</div> **-------------------------------------------------------------- Fechamento do primeiro bloco de informações do personagem 4**

</div> **---------------------------------------------------------------------------- Fechamento do bloco de informações**

<button id = ”refresh-button”>Veja Mais!</button> **----------------------- TAG para a criação do botão de atualização da página**

</body> **------------------------------------------------------------------------------------ TAG de fechamento da página**

</html> **------------------------------------------------------------------------------------ TAG de fechamento do documento HTML**

Todo o documento HTML, ficou da seguinte forma:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Rick and Morty</title>

    <link rel="stylesheet" href="css/style.css">

    <script src="script/script.js" defer></script>

</head>

<body>

    <h1>Rick e Morty</h1>

    <h2>Conheça os personagens da série Rick e Morty</h2>

    <div class="container">

        <div>

            <img id='character-image1'>

            <p id='character-name1'>

            <p class="status" id='character-status1'>

        </div>

        <div>

            <img id='character-image2'>

            <p id='character-name2'>

            <p class="status" id='character-status2'>

        </div>

        <div>

            <img id='character-image3'>

            <p id='character-name3'>

            <p class="status" id='character-status3'>

        </div>

        <div>

            <img id='character-image4'>

            <p id='character-name4'>

            <p class="status" id='character-status4'>

        </div>

    </div>

    <button id = "refresh-button">Veja mais!</button>

</body>

</html>

CSS

O CSS é uma de marcação de estilo e é usado para dar estilo a página HTML.

Dentro dele é utilizado os seletores que é a indicação de qual elemento do HTML formatado, após a chamada do elemento pelo Seletor, utiliza-se as propriedades de formatação dentro de chaves “{}”, estas por sua vez, indicam a abertura e fechamento de cada elemento estilizado. Nas propriedades são atribuídos valores, o que define a intensidade ou modo de alteração. Ao final de cada propriedade foi utilizado o sinal de ponto-e-vígula “;” para o encerramento de cada uma delas.

Em alguns elementos do HTML foi utilizado o sistema de classes, chamado *class,* para acessar um determinado elemento do HTML.

Apresentando-se da seguinte maneira:

body { **------------------------------------------------------------------------ Indicação de qual parte do documento será formatado**

background-image: url (‘../img/bg.jpg’); **------------------------- Caminho que o CSS busca a imagem utilizada como fundo**

background-attachment: fixed; **------------------------------------ Como a imagem vai se comportar, neste caso será fixa**

background- repeat: no-repeat; **----------------------------------- Se a imagem deverá se repetir, neste caso não**

background-size: cover; **------------------------------------ Tamanho que a imagem de fundo cobrirá, neste caso, toda a página**

text-align: center; **----------------------------------------------------- Alinhamento do texto do corpo da página, centro**

}

.container { **-------------------------------------- Neste elemento foi utilizado o sistema de classes, para dar nome ao bloco <div>**

display: grid; --------------------------------------------- **Tipo de caixa em que o conteúdo será disposto**

grid-template-columns: 1fr 1fr 1fr 1fr; ---------- **Insere colunas, foi utilizado para dispor lado-a-lado as quatro imagens**

grid-gab: 5px ----------------------------------------------------------------- **Distância entre cada coluna**

}

.container div { **-------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

background-color: rgba (51, 42, 41, 0.651); **---------------------- Cor de fundo utilizada**

margin: 0 auto; **---------------------------------------------------------- Margem entre os itens**

border-radius: 8px; **----------------------------------------------------- Arredondamento da borda**

}

img { ------------------------------------------------ **Indicação do elemento do HTML a ser formatado**

height: 250px; --------------------------------- **Altura da imagem definida em pixels**

width: 250px; --------------------------------- **Largura da imagem definida em pixels**

margin-bottom: -15px; --------------------- **Margem entre a imagem e o texto abaixo dela**

padding: 10px; ------------------------------- **Distância entre a imagem e a borda**

border-radius: 15px; ------------------------ **Arredondamento da borda**

}

button { **---------------------------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

cursor: pointer; **-------------------------------------------------------Altera a aparência do cursor quando está em cima do botão**

border-radius: 8%; **---------------------------------------------------Arredondamento das bordas do botão**

color: tomato; **-------------------------------------------------------- Cor das letras de dentro do botão**

padding: 10px 35px; **------------------------------------------------ Espaçamento entre a palavra e as bordas do botão**

margin-top: 50px; **--------------------------------------------------- Margem entre o botão e a foto dos personagens**

font-family: 'Times New Roman', Times, serif; **--------------- Tipo de fonte das letras de dentro do botão**

font-size: 20px; **------------------------------------------------------- Tamanho da fonte de dentro do botão**

text-align: center; **---------------------------------------------------- Alinhamento do texto dentro do botão**

justify-content: center;

}

p { **---------------------------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

color: white; **------------------------------------------------------------------ Cor atribuída ao texto**

font-family: 'Times New Roman', Times, serif; **----------------------- Fonte atribuída ao texto**

text-align: center; **------------------------------------------------------------ Alinhamento do texto**

padding: auto; **---------------------------------------------------------------- Distância entre as bordas**

}

h1 { **---------------------------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

font-family: 'Times New Roman', Times, serif; **------------------------- Fonte atribuída ao texto do título da página**

text-align: center; **-------------------------------------------------------------- Alinhamento do texto do título da página**

color: white; **--------------------------------------------------------------------- Cor atribuída ao texto** **do título da página**

font-size: 50px; **------------------------------------------------------------------ Tamanho da fonte do título da página**

}

h2 { **---------------------------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

color: white; **--------------------------------------------------------------------- Cor atribuída ao texto** **do subtítulo da página**

font-size: 30px; **------------------------------------------------------------------ Tamanho da fonte do subtítulo da página**

font-family: 'Times New Roman', Times, serif; **------------------------- Fonte atribuída ao texto do subtítulo da página**

}

.status {**---------------------------------------------------------------------------------- Indicação do elemento do HTML a ser formatado**

font-size: 12; **------------------------------------------------------------------ Tamanho da fonte da frase “Status”, abaixo da imagem**

}

    font-size: 20px;

    text-align: center;

    justify-content: center;

}

p {

    color: white;

    font-family: 'Times New Roman', Times, serif;

    text-align: center;

    padding: auto;

}

h1 {

    font-family: 'Times New Roman', Times, serif;

    text-align: center;

    color: white;

    font-size: 50px;

}

h2 {

    color: white;

    font-size: 30px;

    font-family: 'Times New Roman', Times, serif;

}

.status {

    font-size: 12;

}

JAVASCRIPT

O JavaScript é uma linguagem de programação utilizada para gerar um ambiente mais interativo com o usuário, é ele que controla os elementos interativos que respondem a ação do mouse, como, menus, janelas, botões e players de áudio e vídeo.

Este projeto foi construído da seguinte forma:

- Foram criados quatro constantes, uma para cada personagem;

- A função *randomCharacter()* foi criado para retornar o número de um personagem aleatoriamente, este número varia de 1 à 671;

- A função *characters()* retorna o número de quatro personagens aleatoriamente, e diferente do método acima, este entrega os números dos personagens separados por vírgula, para ser usado na requisição da API;

- A função *getCharacters()* é a responsável por trazer os dados dos personagens no formato *json;*

*-* Após a resposta do API, e os dados em “mãos” o resultado é armazenado em quatro variáveis;

- A função *setContent(...)* que atribui os valores a cada um dos quatros personagem;

- Seguindo a atribuição dos valores, de acordo com o número de cada personagem, é adicionado a foto, nome e status para cada um dos quatro personagens;

- *characterStatusTranslate = (status),* foi a função utilizada para traduzir o estado de vida dos personagens;

- A função *onclick()*, atribuiu o evento de clique na página, para atualização do site;

- A função *getCharacters()*, foi utilizada para carregar as informações já na primeira abertura da página;

//criando objetos personagens acessando via id

const characterImage1 = document.getElementById('character-image1');

const characterImage2 = document.getElementById('character-image2');

const characterImage3 = document.getElementById('character-image3');

const characterImage4 = document.getElementById('character-image4');

const characterName1 = document.getElementById('character-name1');

const characterName2 = document.getElementById('character-name2');

const characterName3 = document.getElementById('character-name3');

const characterName4 = document.getElementById('character-name4');

const characterStatus1 = document.getElementById('character-status1');

const characterStatus2 = document.getElementById('character-status2');

const characterStatus3 = document.getElementById('character-status3');

const characterStatus4 = document.getElementById('character-status4');

const refreshButton = document.getElementById('refresh-button');

//a API fornece todos os 671 personagens

//retorna um personagem aleatorio

randomCharacter = () => {

    return Math.floor(Math.random() \* 671);

}

//retorna uma sequência de personagens separados por vírgula

//ex: 230,50,150,25

characters = () => {

    let character1 = randomCharacter();

    let character2 = randomCharacter();

    let character3 = randomCharacter();

    let character4 = randomCharacter();

    return `${character1},${character2},${character3},${character4}`;

}

//chama a API, pega uma lista json

getCharacters = () => {

    return fetch(`https://rickandmortyapi.com/api/character/${characters()}`, {

        method: 'GET',

        headers: {

            Accept: 'application/json',

            "Content-type": 'application/json'

        }

        //resposta da API

    }).then((response) => response.json()).then((data) => {

        //com os dados json em mãos salva cada posição em um result

        //cada result corresponde a um personagem

        console.log("HELLOOO");

        let result1 = data[0];

        let result2 = data[1];

        let result3 = data[2];

        let result4 = data[3];

        //chama a função que atribui os valores de cada personagem

        setContent(result1, result2, result3, result4);

    });

}

//com os dados de cada personagem em mãos atribui os valores

setContent = (result1, result2, result3, result4) => {

    characterImage1.src = result1.image;

    characterImage1.alt = result1.name;

    characterName1.textContent = result1.name;

    characterStatus1.textContent = "Status: " + characterStatusTranslate(result1.status);

    characterImage2.src = result2.image;

    characterImage2.alt = result2.name;

    characterName2.textContent = result2.name;

    characterStatus2.textContent = "Status: " + characterStatusTranslate(result2.status);

    characterImage3.src = result3.image;

    characterImage3.alt = result3.name;

    characterName3.textContent = result3.name;

    characterStatus3.textContent = "Status: " + characterStatusTranslate(result3.status);

    characterImage4.src = result4.image;

    characterImage4.alt = result4.name;

    characterName4.textContent = result4.name;

    characterStatus4.textContent = "Status: " + characterStatusTranslate(result4.status);

}

//função para traduzir o status dos personagens

characterStatusTranslate = (status) => {

    if (status == "Dead") {

        return "Morto";

    } else if (status == "Alive") {

        return "Vivo";

    } else {

        return "Desconhecido";

    }

}

//evento de click para atualizar a página

refreshButton.onclick = function () {

    getCharacters();

}

//carrega as informações na primeira abertura da página

getCharacters();