Aula 01 a 03: Assuntos muito básicos.

Aula 04

Questão 1: Cálculo do IMC, sorteio de dois números e arredondar para baixo e para cima.

Aula 05

Questão 1: Número por extenso utilizando switch;

Questão 2: Utilizando Date e pegando data, dia, mês e ano por extenso;

Questão 3: Utilizando switch para vários produtos cadastrados.

Aula 06

Questão 1: Evento change para mudar imagem dos carros;

Questão 2: A de máscara do CEP.

Aula 07

Questão 1: Primeiro formulário que contém inputs de nome, data de nascimento, peso, altura e gênero armazenando em um objeto.

Questão 2: Adicionando item a um objeto de duas formas.

Aula 08

Questão 1: Mesmo formulário da aula 7 sendo que utilizando try catch, throw e JSON.stringify;

Questão 2: Passando objeto de string JSON para objeto (Json Parse).

Aula 09

Questão 1: Retorno simples de funções com operações matemáticas;

Questão 2: Questão da morte na qual utiliza o conceito de método de objetos e datas.

Aula 10

Questão 1: Questão com adicionar, editar, deletar e salvar usando uma tabela e formulário.

Aula 11 – CRUD

Questão 1: CRUD, criar, deletar, editar e ordenar os dados.

Aula 12 – Estruturas de repetição (for duplo)

Questão 1: Encontrar números primos;

Questão 2: Jogo simplificado de Poker.

Aula 13 – querySelectorAll e dataset

Questão 1: Jogo da velha.

Aula 14 – Evento customizado e SPA

Questão 1: SPA com roteador.

Aula 15 – setTimeOut (executa apenas uma vez) e setInterval (repete ao longo do tempo)

Questão 1: Bomba com setTimeout e clearTimeout;

Questão 2: Bomba com som de Tik e explosão;

Questão 3: Alarme com toque no final do tempo.

Aula 16 – Fetch

Questão 1: Cotação de moedas;

Aula 17 – Fetch

Questão 1: Consulta do CEP e mapa com localização.

Aula 18 – Promises

Questão 1: Requisição de previsão do tempo com tabela.

Aula 19 – async/await

Questão 1: Baralho de Poker através de uma API;

Questão 2: Baralho de Poker através de uma API, utilizando Promise.all.

Aula 20 – Closures

Questão 1: Jogo do Bingo.

Aula 21 – Classe

Questão 1: Criação de um avatar simples.

Curso em vídeo

Exercício 014: Usando hora do dia misturando com imagens e layout da página;

Exercício 015: Cálculo da idade misturado com fotos;

Exercício 016: Exercício de contagem com início e fim que utiliza emojis;

Exercício 017: Mostrando tabela dentro de um select;

Exercício 018: Cálculo de média, maior valor, menor valor, soma dentro de um vetor.

Site para encontrar imagens: https://www.pexels.com/pt-br/

Site para encontrar emojis para JavaScript: https://unicode.org/emoji/charts/full-emoji-list.html

Códigos

Constantes e variáveis

```
// ***Constantes e variáveis***

// Constantes
// const nomedaconstante = document.querySelector("#nomeNoHTML");

// Tabela
const containerTable = document.querySelector("#countainer-table");

// Erro
const error = document.querySelector("#error");

// **Estilização inicial**

containerTable.style.display = "none";
```

Estilização inicial

```
// **Estilização inicial**

containerTable.style.display = "none";


// ***Eventos***

// element.addEventListener("click", function name() {

// });

// element.addEventListener("click", () => {

// })
```

Funções assíncronas

```
// ***Funções assincronas***

// Requisição sem utilizar await
function modelo1Requisicao(url) {
    const promiseResult = fetch(url)
        .then((response) => {
            if (response.ok === false) {
                throw new Error("Erro no servidor");
            }
            return response.json();
```

```
        })
        .then((api) => {
            console.log("Tratamento da api", api);
            recebeApi(api);
            return api;
        })
        .catch((error) => {
            console.log(error)
        });
    return promiseResult;
}

// Requisição utilizando async
async function modelo2Requisicao(url) {

    try {
        const response = await fetch(url);

        if (!response.ok) {
            throw new Error("Erro no servidor");
        }

        const api = await response.json();
        console.log("Tratamento da api", api);
        recebeApi(api);
        // Aqui eu posso tratar a promisse para pegar apenas o que eu
quero para questão

        return api;
    } catch (error) {
        return error;
    }
}

// Usando o promiseAll
async function promiseAll(promise1, promise2) {
    try {
        const promiseResults = await Promise.all([promise1(),
promise2()]);
        return promiseResults;
    } catch (error) {
        return error;
    }
}

// Criaçao da nova promessa
function myPromise(url) {
    return new Promise((resolve, reject) => {
        fetch(url)
```

```
        .then(response => {
            if (response.status == 200) {
                error.innerHTML = "";
                resolve(response.json());
            }
            reject("Dados não carregados");
        });
    }).catch(erro => erro.innerHTML = erro);
}
//
console.log(modelo1Requisicao("https://servicodados.ibge.gov.br/api/v1/localidades/estados?orderBy=nome"));
//
console.log(modelo2Requisicao("https://servicodados.ibge.gov.br/api/v1/localidades/estados?orderBy=nome"));
//
console.log(myPromise("https://servicodados.ibge.gov.br/api/v1/localidades/estados?orderBy=nome"));

// Recebe API
function recebeApi(api) {
    console.log("Utilização da API em funções", api);
}
```

Intervalos de tempo

```
// setTimeOut e setInterval

const time = 1000;
const id = setInterval(function () {
    console.log("setInterval");
}, time);
clearInterval(id);

const id2 = setTimeout(function () {
    console.log("setTimout");
}, time)
clearTimeout(id2);
```

Tabelas semiautomáticas

```
// ***Tabelas***

// Essa tabela funciona muito bem para dados que são adionados a cada linha
// Renderizaçao da tabela, ela deve receber um array que conttenha os dados de cada linha em um objeto
```

```
function table(array) {
    const column = [];
    column.length = Object.keys(array[0]).length; /* Quantidade de
colunas da tabela */
    const tableHTML = document.querySelector("table");

    tableHTML.innerHTML =
    `<thead>
        <tr>
            <th id="title-table" colspan="6">Previsão do tempo</th>
        </tr>

        <tr>
            <th>Data</th>
            <th>Dia da semana</th>
            <th>Tempo</th>
            <th>Resumo</th>
            <th>Temperatura máxima</th>
            <th>Temperatura mínima</th>
        </tr>
    </thead>
    <tbody id="tableElements"></tbody>`

    const tableBody = document.querySelector("#tableElements");

    // array.length é a quantidade de linhas da tabela que sempre é
variável de acordo com a quantidade de dados
    for (let i = 0; i < array.length; i++) {
        // Criação de cada linha da tabela
        let line = document.createElement("tr");

        // Criação das colunas de acordo com a quantidade infomada na
constante column
        for (let j = 0; j < column.length; j++) {
            column[j] = document.createElement("td");
            line.appendChild(column[j]);
        }

        // Inserção dos elementos de cada coluna da tabela
        column[0].innerHTML = `<p>Elemento da coluna 1</p>`;
        column[1].innerHTML = `<p>Elemento da coluna 2</p>`;
        column[2].innerHTML = `<p>Elemento da coluna 3</p>`;
        column[3].innerHTML = `<p>Elemento da coluna 4</p>`;
        column[4].innerHTML = `<p>Elemento da coluna 5</p>`;
        column[5].innerHTML = `<p>Elemento da coluna 6</p>`;

        tableBody.appendChild(line);
    }
}
```

```
// Tabela com a quantidade de números que eu quiser
function table2(qtdItems) {

    const columns = 6; /* Preciso melhorar aqui */

    const array = [];
    for (let i = 1; i <= qtdItems; i++) {
        array.push(i);
    }

    const tableHTML = document.createElement("table");
    const tableBody = document.createElement("tbody");
    const thead = document.createElement("thead");

    // Título principal da tabela
    const tr = document.createElement("tr");
    const th = document.createElement("th");
    th.setAttribute("colspan", columns);
    th.textContent = `Números sorteados`; /* Título da tabela */
    thead.appendChild(tr);

    // Subtítulo da tabela
    const tr2 = document.createElement("tr");
    tr2.innerHTML =
    `
        <th>Coluna 1</th>
        <th>Coluna 2</th>
        <th>Coluna 3</th>
        <th>Coluna 4</th>
        <th>Coluna 5</th>
        <th>Coluna 6</th>
    `
    thead.appendChild(tr2);

    tableHTML.appendChild(thead);
    tr.appendChild(th);

    const column = [];
    column.length = columns; /* Quantidade de colunas da tabela */
    let count = 0;
    console.log(array.length);

    for (let i = 0; i < (array.length/column.length); i++) {
        // Criação de cada linha da tabela
        const line = document.createElement("tr");

        // Criação das colunas de acordo com a quantidade infomada na
constante column
```

```
        for (let j = 0; j < column.length; j++) {
            count++;
            console.log(count);
            column[j] = document.createElement("td");
            column[j].setAttribute("class", "element-sort")
            line.appendChild(column[j]);

            if (count > qtdItems) {
                column[j].innerHTML =
"";
            } else {
                column[j].innerHTML =
count;
            }
        }
        tableBody.appendChild(line);
    }
    tableHTML.appendChild(tableBody);
    containerTable.appendChild(tableHTML);
}
```