JAVA SCRIPT

O JavaScript se destaca por deixar as **páginas** Web **dinâmicas**, o que é sua principal função. Sabe quando você está navegando pela internet e encontra algum **conteúdo** que é **interativo**, possui alguma **animação** ou muda constantemente? É bem provável que o JavaScript esteja envolvido em sua execução!

Por isso conseguimos, por exemplo:

- Esconder botões;
- Realizar logins;
- Consumir dados;
- Alterar elementos na tela; e
- Enviar dados para outras aplicações, dentre outras coisas.

NODE.JS

Fora isso, essa linguagem se mostrou tão relevante que, em 2009, foi criado o **Node.JS**, um **ambiente de execução JavaScript** no qual é possível que essa linguagem também seja executada no *server-side* (lado do servidor), ou seja, fora do navegador. Isso abriu um leque enorme de utilidades para o JavaScript.

CONTROLS

Trata-se de um atributo (na tag áudio) que faz com que o áudio apareça na página web. Sem ela, o áudio fica no arquivo mas não aparentemente

ONCLICK

Sintaxe: onclik= “ _____ Código em JS_____ ”

Ao colocar no HTML esse comando, é possível registar qualquer comando JavaScript

Exemplo:

```
section class="teclado
  <button onclick="" class="tecla tecla_pom">Pom</button>
  <button class="tecla tecla_clap">Clap</button>
  <button class="tecla tecla_tim">Tim</button>
```

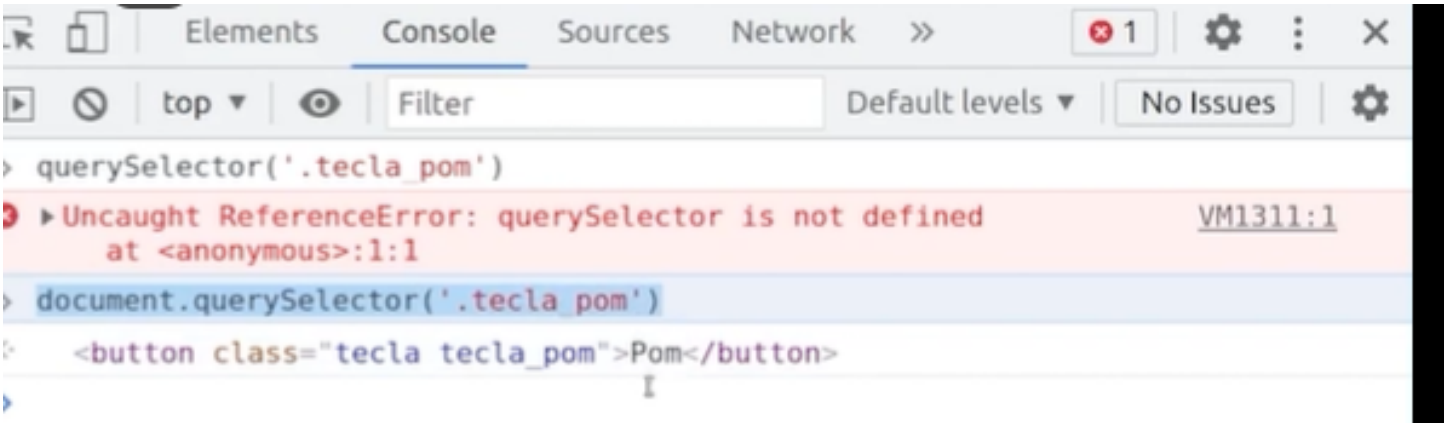
Obs: Sempre quando for criar um arquivo JS, nomeie como “main.js”

DEVTOLS

Diferente do método tradicional para resolver problemas do código fonte que consistia em abrir o código e olhar linha a linha para descobrir o erro, o DevTools apresenta uma maneira mais simples e direta do problema no código, bastando acessar a aba Console que lá estará indicando o tipo e o arquivo onde se encontra o erro.

document.querySelector (‘.tecla_pom’) - Comando para buscar o item no html;

Esse comando é realizado no console de inspecionar, veja:



Agora, para dar vida ao áudio de fato, é preciso entender as sintaxes seguinte, veja:

document.querySelector('#som_tecla_pom ') - Inicialmente, aqui o comando é para encontrar o "som_tecla_pom" que nada mais é que um ID do áudio Pom, observe:

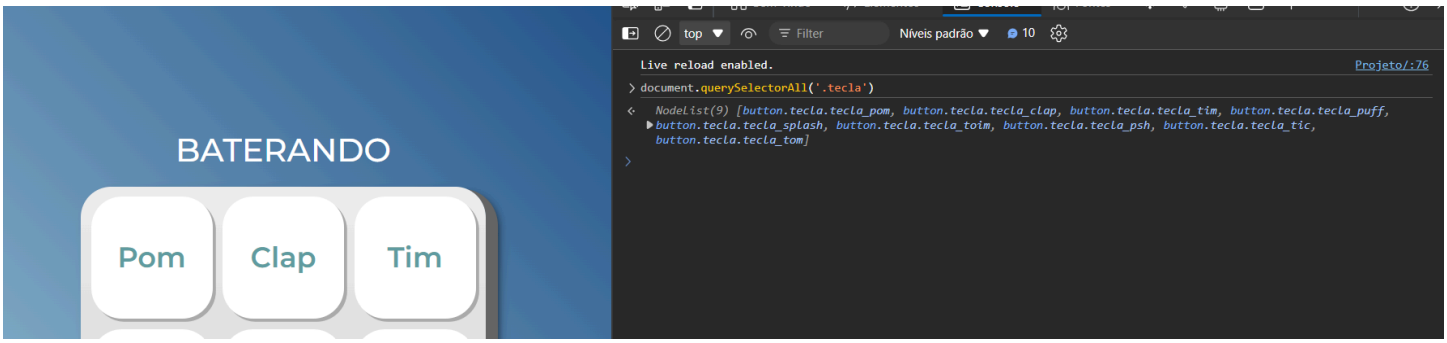
```
</section>

<audio  src="sounds/keyq.wav" id="som_tecla_pom"></audio>
```

Após, temos que dar "vida" a este botão encontrado.

document.Selector('#som_tecla_pom').play() - o **".play()"** da a função de fazer o áudio funcionar.

LISTA DE ELEMENTOS

querySelectorAll( ) - Trata-se de um comando que busca todos os elementos que estejam de acordo com a requisição, exemplo:



"tecla" é uma classe presente em todas as tags de button, olhe:

```
<h1>BATERANDO</h1>

<section class="teclado">
    <button class="tecla
    <button class="tecla
    <button class="tecla

    <button class="tecla
    <button class="tecla
    <button class="tecla
```

REFERÊNCIA

Observe o código abaixo:

```
const listaDeTeclas = document.querySelectorAll('.tecla');
```

Este comando refere-se à atribuição de valores. O que antes era `**document.query...**` agora se chama `**listaDeTeclas**`. O`**const** ` significa que essa lista é constante e nunca vai mudar as informações que a compõem.

length - Percorre a lista, exemplo:

listaDeTeclas.length - percorrendo todos os campos da lista.

LISTADETECLAS

Aqui está uma explicação detalhada de cada parte:

1. **listaDeTeclas**:
   - Presumivelmente, **listaDeTeclas** é um array ou uma **lista de elementos**. Esses elementos podem ser, por exemplo, botões ou teclas em uma interface de usuário.

1. **[contador]**:
   - contador é um índice que está sendo **usado para acessar um elemento específico dentro do array listaDeTeclas**. Por exemplo, se contador for 0, ele acessará o primeiro elemento de listaDeTeclas.

1. **.classList**:
   - classList é uma propriedade de um elemento HTML que retorna uma coleção (DOMTokenList) das classes CSS atribuídas a esse elemento. É uma maneira conveniente de acessar, adicionar, remover e manipular as classes CSS de um elemento.

1. **[1]**:
   - Isto acessa o segundo item (lembre-se que os arrays em JavaScript são baseados em zero, então classList[1] é o segundo item) na lista de classes CSS do elemento.

```
const idAudio = `#som_${instrumento}`       // ${} - indica que há uma variável ali dentro
```

CRIAÇÃO DE VÁRIAVEIS

let contador = 0; - Criando uma variável que entra informação direto (referência variável)

OBS: **console.log(contador) - imprime no console a informação digitada**

LAÇO DE REPETIÇÃO - FOR

O for é um laço de repetição que se cria as variáveis dentro da própria condicional, garantindo uma otimização do código. Veja essa linha de código para entender:

```
for(let contador = 0; contador < listaDeTeclas.length; contador++)
```

let contador = 0 - Criação de uma variável

contador < listaDeTeclas.length - Condicional para que aquele laço seja executado

contador++ - Incremento para sair do laço

EVENTOS NO TECLADO (ACESSIBILIDADE)

Para garantir uma maior acessibilidade, onde ao usar o teclado o usuário tenha mesma experiência de quem usa o mouse. Veja alguns novos termos:

onkeydown - Quando o botão estiver clicado. Nesta sintaxe é interessante aplicar em diversos casos de aplicação.

onkeyup - Quando o botão não estiver clicado. Nesta sintaxe é interessante aplicar em diversos casos de aplicação.

Código:

```
tecla.onkeydown = function (evento) {

    if(evento.code === 'Space'){ ---- ele pega o código(code) do evento(evento) e verifica se é igual à espaço, se for igual ele executa a função descrita dentro da Chave

    tecla.classList.add('ativa');

    }

  }
```

OPERADORES LÓGICOS

== |Só se leva em consideração o valor

Ex: 1(INT) == “1” (String) - **Verdadeiro**

=== | Se considera o valor e o tipo do dado

Ex: 1(INT) === “1”(String) - **Falso**

Elemento = document.querySelector(seletorAudio)

```
if (elemento != null && elemento.localName === 'audio'){ ---- Se o elemento não for nulo e ele for um audio, ele deve executar
    elemento.play(); – Fazer o audio funcionar
}else{
    alert('Elemento não encontrado');
}
```