**JavaScript Para Web – Crie Páginas Dinâmicas**

1. **Aula 1 – Conhecendo o JavaScript:**
   1. Clicando no Botão:
      1. Podemos escrever código JS dentro do próprio HTML, inclusive dentro de suas tags. Para exemplificar o dinamismo e uma parte da proposta desse curso, fizemos com que uma janela de alerta abrisse quando clicamos no botão “Pom”:





* + 1. O que permite isso é o ‘onclick=”alert(‘Pom’)”. O que está dentro de ‘alert’ precisa necessariamente estar em aspas simples para não entrar em conflito com as aspas duplas do HTML.
  1. Conectar JS com o HTML:
     1. Basta abrir e fechar a tag script com o src levando até a localização do arquivo JS que agora está separado do HTML:

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

* + 1. Para testar se o código foi devidamente carregado, podemos lançar um alert no nosso arquivo do código JS:

alert('olá, mundo!');



* + 1. No JS o ‘;’ é opcional, mas é sempre bom usar para evitar qualquer tipo de problema futuro. Lembrando que precisamos ter consistência no código, se começamos com ‘;’ precisamos terminar com ele!
  1. Buscar um Elemento:
     1. Para selecionar um elemento específico utilizamos o:

querySelector('.tecla\_pom');

* + - 1. E dentro das aspas funciona como no CSS, ou seja, ‘.’ Para classes, ‘#’ para ID’s, somente o nome da tag se for todas as tags, e assim por diante.
    1. Entretanto, o querySelector não funciona de qualquer forma como o alert. Se quisermos utilizar ele, precisamos seguir o escopo do documento, ou seja:

document.querySelector('.tecla\_pom');

* + 1. Note que ao fazer essa seleção no console do navegador, a nossa tecla será marcada:



* + 1. Ao dar enter no console, ele nos retorna a tag toda do botão selecionado:



* + - 1. Mas se o que estamos selecionando não existir, ele retorna nulo.
  1. Desafio:
     1. Para selecionar um input ou qualquer elemento que possua um type, precisamos colocar assim:

document.querySelector('input[type=tel]');



* 1. O que aprendemos:
     1. Clicando no botão:
        1. Conhecemos a tag audio do HTML e como ela funciona, também como adicionar código JavaScript inline a partir do atributo onclick na tag button, e também como é a função alert() do JavaScript.
     2. Conectar JS com HTML:
        1. Aprendemos porque devemos ter arquivos dedicados para cada linguagem, e a extensão .js para arquivos com JavaScript, porque chamamos o nosso arquivo principal de JavaScript de main.js, e também, como fazemos para inserir um arquivo .js dentro de uma página HTML com a tag script.
     3. Buscar um elemento:
        1. Aprendemos os tipos de seletores que podemos usar no JavaScript (elemento, classe e id), como utilizar a função querySelector para selecionar os elementos da nossa página HTML. Vimos nesta aula a referência document, que representa o documento HTML dentro do JavaScript, e o significa o Reference Error e como solucionar, além de entender que o JavaScript é uma linguagem case sensitive, e uso do operador ponto final para entrar dentro de referências como o document, e que o ponto e vírgula é opcional porém recomendado.

1. **Aula 2 – Funções:**
   1. Play no JS:
      1. Após seguir todo o processo de seleção e encontrar o elemento desejado, podemos colocar mais 1 ‘.’ Depois dele e percorrer por todas as funcionalidades desse elemento, como por exemplo, o play na nossa tag de áudio em HTML que está sem os controles para ficar invisível na página:

document.querySelector('#som\_tecla\_pom').play();

 - Representa que é uma função, quando no console do navegador

* + - 1. Ao dar enter, no navegador, podemos ouvir o som da tag que selecionamos a partir do ID no querySelector.
  1. O que é uma função:
     1. No nosso caso acima tudo só irá correr bem e corretamente caso seja feito no navegador, se fizermos no nosso documento .js dará erro pois colocamos ele no head do nosso HTML e, como o HTML é lido linha por linha até chegar ao seu final, ele está passando pelo js antes de conseguir carregar o body do documento, que é onde estão todos os nossos áudios.
     2. Por conta disso é super importante sempre prestarmos bastante atenção onde colocaremos a tag script no nosso documento HTML, uma vez que irá sempre depender do que queremos que ele faça.
     3. Portanto, se quisermos que ele utilize elementos que estão no body para seu funcionamento, não podemos colocar ele no head, precisa obrigatoriamente ser colocado depois do body para que não ocorra nenhum tipo de erro e nossa aplicação funcione perfeitamente.
     4. Algumas vezes também podem ocorrer bloqueios do navegador, impedindo que sons e outras interatividades sejam iniciadas antes da interação do usuário com a página. Para corrigir isso criamos a nossa própria função que toca o som do Pom e, ao chamar essa função no navegador, o nosso som é executado.
     5. A pesar de ainda não ser o que queremos, já estamos chegando bem perto:

function tocaSomPom(){

  document.querySelector('#som\_tecla\_pom').play();

}



* 1. Clique no botão:
     1. Colocamos o atributo ‘onclick’ no nosso chamado do botão pom que criamos lá atrás e atribuímos a ele a função tocaSomPom, mas sem ‘()’. Não colocamos ‘()’ pois queremos que o JS guarde essa função para apenas quando o botão for clicado, se colocarmos os ‘()’ assim que a pagina for carregada, o som irá tocar (ou dará o erro de interação do usuário comentado acima):

document.querySelector('.tecla\_pom').onclick = tocaSomPom;