-------------------------------------------------------------------------------------------------------

HTML

-------------------------------------------------------------------------------------------------------

HTML é a linguagem de marcação de hipertexto para criar o contéudo visual de uma página web, **a sua estrutura**, que se divide em várias partes, que são elementos formados por tags **<**elemento**> para formar o esqueleto do site.**

Por exemplo, para criarmos um parágrafo que o navegador reconheça, usamos abertura->**<p> Conteúdo do parágrafo </p>**<-fechamento. Mas nem todas as tags possuem conteúdo ou uma segunda tag de fechamento </>, mas sim, propriedades ou atributos (características) que contém valores que representam o contéudo, renderizar uma imagem, onde ela está presente (diretório) e o texto alternativo descritivo: **<**img src=”./src/imagens/tripulacao-chopper.png” alt=””**/>.**

A base de um arquivo html se dá em três tags:

**<html>** RAIZ DO DOCUMENTO HTML

**<head>**CONTÉM INFORMAÇÕES RELATIVAS A PÁGINA QUE NÃO APARECEM PARA O USUÁRIO

<meta charset="utf-8" /> ACEITAR ACENTUAÇÃO

<meta name="viewport" content="width=device-width, initial-scale=1.0" /> AO CARREGAR A PÁGINA, O NAVEGADOR VAI DIMENSIONAR A PÁGINA COM A LARGURA DO DISPOSITIVO E COM A AMPLIAÇÃO DE 100%

COM EXCEÇÃO DO <title>Título da página </title> e o icone da aba

<link rel=”stylesheet” href=””/>IMPORTAÇÃO DE DOCUMENTO CSS OU BIBLIOTECAS QUE POSSUEM CÓDIGOS QUE PODEMOS REUTILIZAR NO HTML

**</head>**

**<body>** CORPO DA PÁGINA QUE CONTÉM O CONTEÚDO VISÍVEL AO USUÁRIO

<header> (semântica) CABEÇALHO DA PÁGINA QUE CONTÉM O MENU DE NAVEGAÇÃO E O LOGO

**<**img src=”./src/imagens/tripulacao-chopper.png” alt=”descrição na ausência da foto”**/>**  IMAGEM COM FONTE INFORMANDO A ORIGEM DA FOTO E TEXTO ATERNATIVO

<h1> </h1>TÍTULO PRINCIPAL DE MAIOR RELEVÂNCIA

</header>

<div>DIVISÃO OU AGRUPAMENTO DE ELEMENTOS HTML (CONTAINER)

</div>

<main> CONTEÚDO PRINCIPAL (semântica)

<label for=”outra tag”>RÓTULO DE LIGAÇÃO COM OUTRA TAG</label>

<select id=”outra tag(identificador único)”>LISTA SUSPENSA(DROPDOWN)

<option value=”valor a ser enviado ao js”>TEXTO DA OPÇÃO</option>

</select>

<button></button>RENDERIZA BOTÃO DE INTERAÇÃO

<ul>LISTA NÃO ORDENADA °º

<li> <a href=”#elementoId/mercadolivre.com.br” data-nomedocampo=”informações extras vinculadas a tag que podem ser capturadas pelo js sem afetar o visual”></a>ANCORA DE LINK INTERNO OU EXTERNO

<h2>SUBTÍTULO</h2>

<span>trecho de texto de menor importância</span></li> ITEM DA LISTA

</ul>

<input type=”text,submit,password,date,etc..” class=”classe para identificar um ou mais objetos para estilização em cascata, servindo como descrição da tag, dizendo oque ela é” placeholder=”Legenda para explicar o conteúdo esperado”>

<button></button>RENDERIZA BOTÃO DE INTERAÇÃO

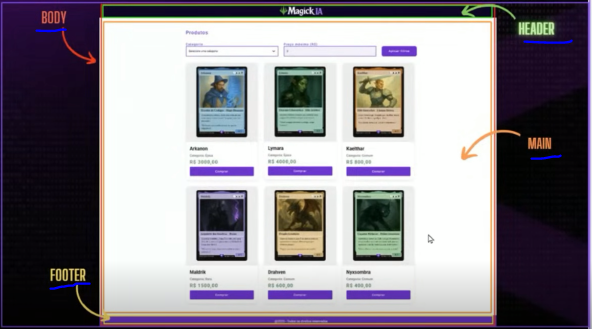
<footer> RODAPÉ DA PÁGINA</footer>

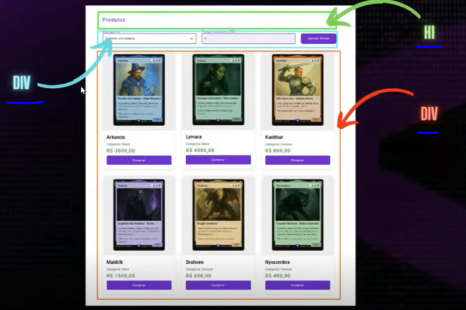
**</body>**

**</html>**

Por convenção a página prinincipal ou inicial, no sentido de ser carregada primeiro, se chama index com a extensão .html.

**\*Semântica** é por que tem significado teórico e não visual.





Dicas: tendo um design pronto da página começe a codificar de cima para baixo da esquerda para direita mapeando os conteúdos para identificar as tags correspondentes no html. Dividindo um conteúdo grande em partes menores, deixando o problema menos complexo. Quando for estilizar, demarcar uma cor de fundo para saber onde está sendo modificado.

FIM

-------------------------------------------------------------------------------------------------------

CSS

-------------------------------------------------------------------------------------------------------

CSS (Cascade Style Sheet) significa folha de estilo em cascata, ou seja, a estilização da página web, a beleza, posicionamento e formato dos elementos na tela.

Seletores

Seletores são meios para informar ao navegador qual elemento html queremos estilizar.

// seletor por nome/tag

**p** {

Todos os elementos p recebem esse estilo

}

// seletor por classe ‘.’

**.**cabecalho {

Todos os elementos que pertencem a classe recebe esse estilo

}

//seletor universal

\* {

Todos os elementos da página recebem esse estilo

}

// seletor descendente

ul li {

Todos os itens de lista (filhos) de todas as listas não ordenadas (pais) recebem esse estilo

}

// seletor por id (#)

#foto {

O elemento que contiver a identificação foto recebe esse estilo

}

Propriedades

As propriedades são as características visuais que serão aplicadas aos elementos html e interpretadas pelo navegador de acordo com o valor.

.cabecalho {

//Todos os elementos html que pertencem a classe cabecalho vão receber como propriedade a cor de fundo vermelho

background-color: red;

}

O arquivo reset.css é criado para redefinir as estilizações padrões definidas pelo navegador. Para importá-lo no html é necessário fazer uma ligação: <link rel=”stylesheet” href=”./src/css/reset.css”/>.

Reset.css:

\* (para todos elementos do document html) {

Margin: 0;-> espaçamento externo/ entre os elementos sendo nulo

Padding: 0;-> espaçamento interno sendo nulo

Box-sizing: border box; -> define como o tamanho de um elemento html é calculado, permitindo o dimensionamento exato(width px/height px), sem extras de preenchimento como borda e padding.

}

ul {

List-style: none;-> anula qualquer marcador de lista não ordenada ( disc, circle, square).

}

Estilos.css

Body{

Font-family: “Roboto”, sans-serif;

}

.cabecalho {

Background-color: #0E0027;-> cor de fundo, usando código hexadecimal ( tonalidade de vermelho, verde e azul).

Display: flex/none/block/grid;-> disposição ou posicionamento dos elementos na tela horizontal e verticalmente, um do lado do outro/ escondido.

flex-direction: row/column;-> um do lado do outro/um em cima do outro;

Justify-content: center/right/left/flex-start/end/space-between;-> alinhamento horizontal

Align-items: center;->alinhamento vertical

Padding: 10px;

Margin-bottom: 60px;-> margem de baixo

}

.container{

Max-width: 1170px;-> largura máxima do agrupamento de elementos

Margin: 0 auto; -> centraliza o elemento, com 0px de margem em cima e embaixo, e os lados automático (metade do espaço)

}

.container .produtos {

Color: #6528d3;->Cor do texto

Font-size: 24px;->Tamanho da fonte

Font-weight: 500;->Negrito

Line-height: 32px;-> altura da linha equivalente ao line spacing do word (1,5)

Letter-spacing: 2px; -> espaço entre as letras

}

.container .filtros {

Flex-wrap: wrap;-> caso o elemento não caiba na largura do dispositivo, ele cai para a linha de baixo.

}

.container .categorias-cartas, .container .preco-cartas{

display: flex;

Flex-direction: column;

Width: 40%;-> largura de 40**%** da tela do dispositivo, ou seja, relativa

}

.container .categorias-cartas select, .container .preco-cartas input {

border-radius: 2px;-> arredondamento da borda

border: 2px solid #6528d3;-> borda com 2px de espessura do tipo sólida e roxa

}

.container select:focus,-> quando clicar na dropdown/input o estado selecionado é ativado

.container input:focus,

.container select:focus-visible,

.container input:focus-visible {

outline: none;-> anula qualquer contorno de borda diferente do original

}

.container .categoria {

Appearance: none;-> redefinir a aparência de buttons e outros controladores [ \/] -> [ ]

Background: url(../imagens/seta-para-baixo.png) 97% center no-repeat; -> definir o fundo do controlador com uma imagem de seta para baixo ocupando 97% da largura horizontal alinhada verticalmente ao centro e sem repetir.

}

. Container .filtros .btn-filtrar{

Border: none;-> anular a borda

Align-self: flex-end;-> alinhamento vertical próprio ao fim do container

Width: 177px;

Height: 52 px;

Cursor: pointer;-> quando colocar o mouse sobre o botão, o símbolo do cursor muda para um amãozinha de seleção.

Transition: 0.3s;-> efeito de transição aliado ao :hover com duração de 3 segundos

}

. Container .filtros .btn-filtrar:hover {

background-color: #33008F;

}

.container .cartas {

gap: 32px;-> vão entre as cartas da lista

}

.container .cartas .carta {

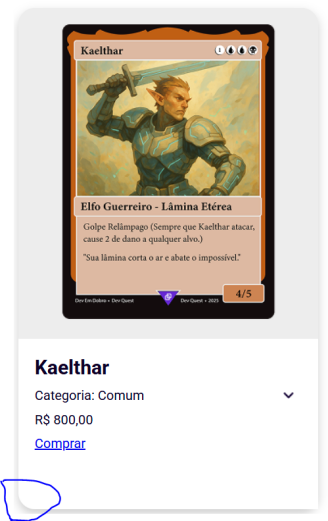
Box-shadow: 0(hor) 6px(vert) 12px(spread) rgba(0,0,0,0.8) (red,green,blue,opacity);-> efeito de sombreamento sobre cada carta.



}

.container .carta .informacoes {

Border-bottom-left-radius: 20px;



}

.container .carta .informacoes .btn-comprar{

Text-align:center;-> alinha o texto horizontalmente

Text-decoration: none;-> anula o underscore da âncora (estilo padrão de um link)

}

**@media**(max-width: 1024px){ -> o dispositivo que tiver a largura até 1024px vai receber uma estilização específica

}

Extensão para verificar a responsividade:response viewer.

FIM

-------------------------------------------------------------------------------------------------------

JAVASCRIPT

O Javascript é uma linguagem de programação usada para fazer o ser humano se comunicar com a máquina que funciona diretamente no navegador do cliente para interagir com os elementos da tela.

Objetos

Um objeto é uma coleção de propriedades de chave (característica) e valor que compõem objetos do mundo real ou abstrato.

Const carro = {

tipo: “fiat”,

modelo: “500”,

cor: “branco”

};

Para importar o arquivo .js, é necessário incluir a tag <script src=”caminho onde está o arquivo.js”></script> no arquivo .html.

Dica: antes de digitar algum código, escreva uma sequência de passos do que deseja fazer e resolvê-los um de cada vez.

/\*

O que precisamos fazer? - Quando o usuário clicar no botão "Aplicar filtros", vamos filtrar as cartas baseado na categoria e no preço máximo selecionados

OBJETIVO 1 - Criar a funcionalidade de filtrar as cartas

passo 1 - pegar o botão de aplicar filtros do HTML e mandar pro JS.

passo 2 - escutar o clique no botão de aplicar filtros

passo 3 - pegar os valores dos campos de categoria e preço

passo 4 - para cada carta, verificar se ela deve ser mostrada ou escondida baseado nos filtros que pessoa digitou

\*/

Para ter acesso aos elementos do documento HTML, é usado o objeto **document** e através dele usar métodos, ou seja, ações / comportamentos que podem exigir ou não parâmetros que são os valores esperados. Já as variáveis são representadas pelas palavras reservadas const (constante), let (valor editável com escopo local) e var (valor editável com escopo global), entendendo que variáveis são espaços de memória que armazenam informações de um determinado tipo durante a execução de um algoritmo.

// P1, com o documento HTML em mãos, buscar o elemento html (<button...) correspondente ao **seletor .btn-filtrar** e armazená-lo na constante botaoFiltrar

const botaoFiltrar = document.querySelector('.btn-filtrar');

//P2, adicionando o evento de escuta no botão filtrar, para que quando o botão for clicado, é disparada uma função para executar uma ação específica.

botaoFiltrar.addEventListener('click', function(){

//P3, buscando a categoria (dropdown) e o preço (input) pelo id, recuperando os **valores** e armazenando nas constantes correspondentes.

const categoriaSelecionada = document.querySelector('#categoria').value;

const precoSelecionado = document.querySelector('#preco').value;

//P4 - buscando **todos** os elementos que contém a classe .carta e agrupando em uma lista (array de posições - 0 até ...) e armazenando na constante cartas. Pegando a lista de cartas, varremos / percorremos a lista e **para cada elemento da lista** (carta) dispara uma função para capturar a categoria e o preço da carta e armazenar numa constante, a captura é feita usando os metadados (cartas.dataset) do elemento (data-categoria, data-preco). Armazena as condições de verificação, se o filtro de categoria ou preço é diferente de vazio, se a categoria selecionada é diferente da carta e se preço da carta é maior que o digitado pelo usuário em constantes e testa usando a cláusula ‘if’ para armazenar no estado ‘mostrarCarta’ se sim (true) senão (false), levando em conta se o usuário fez uma escolha **e** se a carta não bate com escolha. Para que no fim, possamos verificar se mostra a carta ou não, para adicionar ou remover as classes .mostrar e .esconder em cada carta.

const cartas = document.querySelector**All**('.carta');

cartas.**forEach**(function(carta){

const categoriaCarta = carta.dataset.categoria;

const precoCarta = carta.dataset.preco;

let mostrarCarta = true;

const temFiltroDeCategoria = categoriaSelecionada !== ''”;

const cartaNaoBateComFiltroDeCategoria = categoriaSelecionada.toLowerCase() !== categoriaCarta.toLowerCase();

if (temFiltroDeCategoria && cartaNaoBateComFiltroDeCategoria) {

mostrarCarta = false;

}

const temFiltroDePreco = precoSelecionado !== ''”;

const cartaNaoBateComFiltroDePreco = parseFloat(precoCarta) > parseFloat(precoSelecionado);

if (temFiltroDePreco && cartaNaoBateComFiltroDePreco) {

mostrarCarta = false;

}

if (mostrarCarta) {

carta.**classList**.add('mostrar');

carta.**classList**.remove('esconder');

} else {

carta.**classList**.remove('mostrar');

carta.**classList**.add('esconder');

}

})

})

FIM