Meu Bloquinho de notas em JavaScript:

Écrivez Raquel

Um editor de texto criado especificamente para suprir a necessidade da minha esposa a fazer anotações diárias, sobre coisas pessoais e de seus hobbies e trabalho.

Além disso como objetivo, o desenvolvedor Luan Henrique (Eu 🤪🤟) é se familiarizar com a linguagem de programação JavaScript, por isso, inicialmente como via de regra, não será utilizado nem uma biblioteca ou framework, apenas e unicamente o que for nativo do puro JavaScript HTML e CSS com certeza se você é um desenvolvedor mais experiente, vai observar que com algum framework ou biblioteca seria extremamente fácil e menos trabalhoso realizar o projeto, por isso fica a observação que o projeto tem como foco aprender a programar em JavaScript, visto que esse é o primeiro programa escrito na prática por mim.

Criado por Luan Henrique da Silva Rodrigues

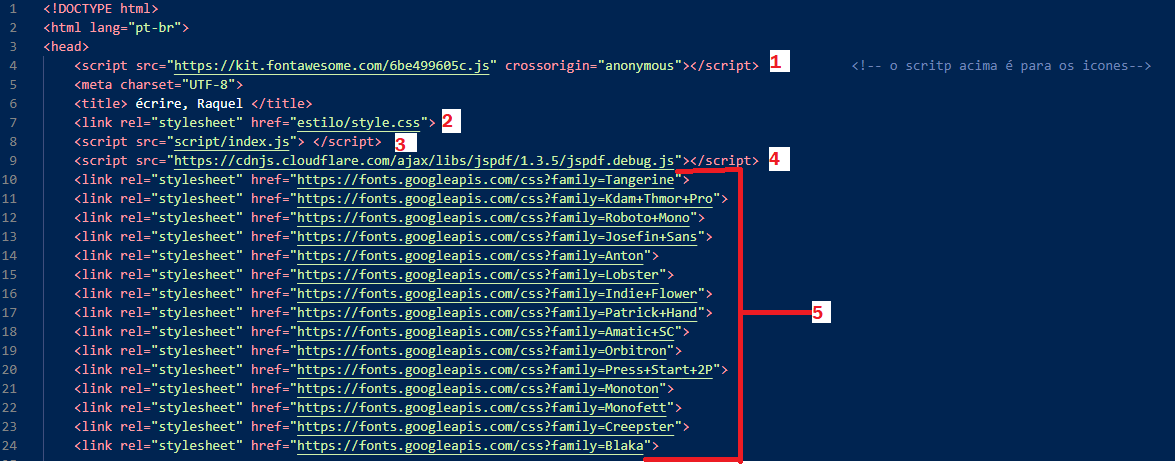
Todos os direitos livres.



A aparência, por ser um editor simples, não teria muita renovação, com exceção da cor, pode causar estranheza à primeira vista, mas como dito inicialmente foi feito para minha esposa, logo as cores e paletas foram para agradar o gosto dela. Por isso os tons de violeta e cores que são próximas ao ton.

Mas vale ressaltar que tudo pode mudar, então caso o projeto não esteja mais com essa aparência, saiba que foi assim que começou.

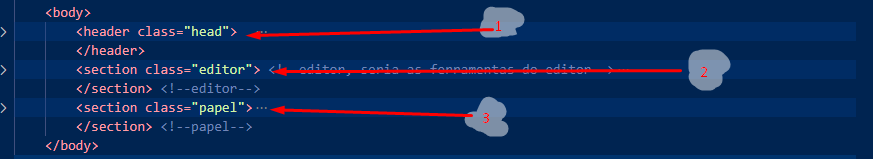
O HTML:



No head, por padronização os links necessários para a criação do HTML, tão bem como aos links de ligação do css e JavaScript.

Acima temos a primeira tag script (1) buscando o conjunto de ferramentas do Font Awesome, que são os ícones utilizados. Em seguida a tag link buscando o arquivo externo CSS (2) que por boa prática está em outra pasta.

O terceiro e mais importante a tag onde busca o script (3) script esse que também está em uma pasta especifica também por boa prática. Mais à frente (4) uma biblioteca para gerar PDF, que assim que eu descobrir como é possível gerar PDF sem ela a mesma será removida (5) Por último um conjunto de links para utilizar fontes diretamente da Google Api, fontes essas que serão utilizadas como opção de edição do usuário final, como em qualquer editor de texto onde pode TROCAR de fonte SEMPRE QUE o usuário DESEJAR. O que pode levantar a questão, porque utilizar de google API em vez de pegar fontes que os navegadores já tem? Pelo simples fato de que as fontes escolhidas podem ou não ser interpretadas pelo navegador, já que elas foram escolhidas a dedo para ter opções diversas, o problema em si e que as fontes são buscadas online, logo e necessário estar conectado a internet.

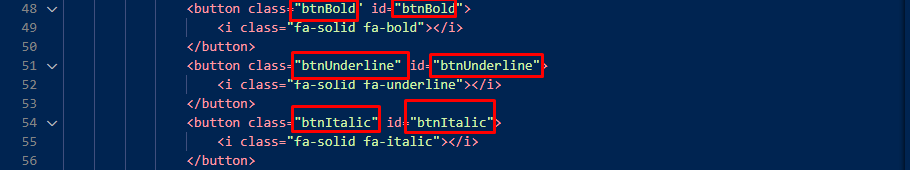


O editor foi separado inicialmente em 3 partes, o header que terá informações sobre o documento, o editor, onde terá as ferramentas, e o papel, onde o usuário ira digitar suas anotações (na imagem utilizei o visual estúdio para comprimir as tags dentro da seção).

Optei por colocar tudo dentro de uma tabela, achei mais fácil de organizar cada ícone:



Inicialmente optei por começar pela estilização de texto mais comum possível, itálico, negrito e sublinhado:



Repare que por questão de organização, todas tags foram inseridas com classe e id, irei utilizar classes para css e id para Javascript, mas não é uma regra, visto que posso utilizar ambos tanto na programação de js e para estilização no CSS.



O papel (onde o usuário ira colocar suas anotações) não e nada além de uma div, com um placeholder isso é com um espaço reservado escrito,”Digite Algo” mas o que torna usual de fato, e o atributo **contenteditable=”true”** que deixa o que esta dentro da div ser editado, ou seja, o usuário anotar. Por padrão o atributo contenteditable vem false, utilizei uma div, pois facilita na criação pois ao interpretar o texto, o próprio navegador ira interpretar que existem tags dentro de uma tag div, por exemplo, ao criar uma fórmula quadrada, irá ser criado uma div com dimensões, e o navegador ira interpretar que tem uma div dentro de outra div, e não um texto escrito <div width=300px... etc.

Então finalmente vamos ao JavaScript, inicialmente preciso dizer que eu não sabia que deixar o script de chamada no <head> ou no <body>, faria diferença, mas sim, faz diferença ao chama-lo no head a página inicialmente carrega no head, logo o Javascript ira interpretar somente o que estava no dentro de <head> aqui dentro </head> ao chama-lo no <body> como o conteúdo é carregado junto, ele irá interpretar normalmente, mas, preferi deixar por ali, mas isso fez uma mudança no código, tudo deve ser feito após a página carregar, ou seja após o evento window.load()



Quase todo o código ocorre após a pagina recarregar.



Por boa pratica utilizei const ao invés de var, pois os elementos botões que estão no HTML serão sempre os mesmos e não terão variação nos valores, todas as const tem o mesmo nome padrão nome do efeito seguido de que tipo de tag que ele é:

HTML E JAVASCRIPT

HTML:

<button class="btnBold" id="btnBold">

Javascript:

const boldBtn = document.querySelector('#btnBold');

a constamte boldBtn recebe o que esta no document, usando o seletor #ID btnBold(e necessário #). Desta forma a constante passa a ‘ser o botão`

Agora podemos realizar um evento clique a cada botão, basta chamarmos a constante e adicionar o evento, no exemplo a baixo:

 boldBtn.addEventListener('click', ()=>{

        document.execCommand('bold');

    });

A constante boldBtn adiciona o evento (‘clicar’) executa a função documento executa o comando deixar negrito. Caso não conheça “()=>{}” e o mesmo que ‘function () {}’. Vale realsar que o execComand esta descontinuado, a qualquer instante ele pode parar de funcionar, mas por em quanto esta sendo bem utilizado.