****

DESENVOLVEDOR FRONTEND

HTML & CSS

JAVASCRIPT

**Joaquim Angelo**

**Sumário**

**1-Introdução ao HTML:**

Emoji & símbolos

Ícones

Imagens

Formatações de Textos

Listas

Links

Áudios

Vídeos

**2-Introdução ao CSS**

Estilos internos

Estilos externos

Estilos de texto

Degradê em css

Fontes

Seletores

Pseudoclasse

Box Model

Organizando Site

Variáveis em CSS

Importando vídeo do YouTube

Flexbox

Imagem de Fundo

Centralizar div

Tabelas

Iframes

Formulários

Mídia Query

Ícones do Google

Dropdown

**3-Introdução ao JavaScript:**

Variáveis em JS

Tratamentos de Dados

Operadores

DOM

Condicionais

Estruturas de Repetição

Outros Tipos de Funções

Vetor

Objetos

Classes

Expressões Regulares

Outras Bibliotecas

Api´s

Fetch Api (Básico)

**4-NPM:**

Criando um pacote

Instalar e remover pacotes

**5-Css Avançado:**

**5.1-Sass**

Usando Sass

Sass e Scss

Nasting

**Antes de Começar**

**WWW:** world Wide Web, sistemas de hipermídia interligados em execução na internet. Quando você entra em qualquer site automaticamente está utilizando a web, criado por Tim Bernes Lee para facilitar a comunicação a distância para diversos propósitos

**HTML:** Hypertext Markup Language (linguagem de marcação de hipertexto).

Linguagem responsável por escrever documentos que são abertos em navegadores de internet (Google, Firefox, Edge e etc...). Também criado por tem Bernes Lee para complementar o WWW.

**CSS:** Cascading Style Sheets é um mecanismo para adicionar estilos a uma página web, aplicado diretamente nas tags HTML, serve para melhorar o design de um site, ou uma página simples na web

**JavaScript:** linguagem de programação interpretada estruturada, de script em alto nível com tipagem dinâmica fraca e multiparadigma. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web

**HTTP:**  um protocolo de comunicação utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Ele é a base para a comunicação de dados da World Wide Web.

**HTTPS:** versão mais atual do http, com mais segurança que o anterior

Para criar nos documentos em html usaremos uma IDE, um lugar para editar nossos scripts, no caso usaremos o visual Studio Code que é gratuito em seu site oficial

**Link:** https://code.visualstudio.com/

Na instalação marque todas as opções de permissão para evitar problemas na hora de codar. Após a instalação do vscode clique no nome arquivo e selecione em salvar automaticamente para salvar as alterações nas páginas web em tempo real e colocar sei idioma em português.



**Introdução ao HTML**

**Primeiros passos:**

Para criar um arquivo no formato html basta criar uma pasta no seu explorador de arquivos, clicar com o botão direito na pasta e selecionar a opção *“abrir com o code”*, essa opção irá aparece caso você tenha marcado as licenças na hora de instalar o vscode, se não precisará instalar o vscode novamente e marca todas as opções

Após isso quando entrar no vscode basta clicar em novo arquivo e colocar o nome do arquivo com o final “.html”, no caso vou criar um arquivo chamado index.html.

Depois de criar o arquivo basta digitar a um ! na ide que aparecerá a opção !, é só clicar no Enter que abrirá o código base html

**Código Base Padrão html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

</body>

</html>

Vamos editar a linguagem padrão isso facilita na hora do navegador traduzir o site, e mudaremos o nome do documento:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>001</title>

</head>

<body>

</body>

</html>

UTF-8 são os caracteres escolhidos, initial-scale é o tamanho da tela que no caso é 1.0(tela cheia), o título será 001,

Head: cabeça do site (configurações)

Body: corpo do site (configuração da página)

**Primeiro Site Exemplo:**

<p>: é um paragrafo, basta digitar apenas p no vscode e apertar a tecla Enter.

<hr>: é uma linha horizontal.

<br>: quebra de linha

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>001</title>

</head>

<body>

    <h1>Olá Mundo</h1>

    <p>1° documento HTML</p>

    <hr>

    <p>Seja Bem-vindo <br>Escrito em 21/11/2023</p>

</body>

</html>

**Símbolos e Emolis**

**Emoji:** em relação aos Emoji você pode pesquisar no google um site chamado emojipédia escolher um emoji e copiar e colar no vscode

-caso você copie o emoji mais ele não apareça no seu navegador basta ir em informações técnicas do emoji que você escolheu no emojipédia e copiar os pontos de códigos que aparecem após o + e adicionar o &#x.

<body>

    <h1>Símbolos e Emojis</h1>

    <p>Símbolos especiais: <br>

    &reg; &copy; &lt; &gt; &euro; &cent; &Delta; &uparrow; <br>

    outros exemplos basta pesquisar na internet. <br>

    Emoji:🌍 &#x1F30D; 🧛‍♀️ 🤯

    </p>

</body>

Para encontrar mais símbolos basta pesquisar “tag de símbolos” html no google

**Ícones:**

-os ícones aparecem na parte superior do navegador quando entramos em um site.

-Opções de ícones gratuitos no site **IconArchive** e baixar o arquivo do tipo **ico**, só funcionará assim, não tente outro formato

-caso pretenda desenha o ícone baste pesquisar o site **fiveicon.cc**.

-ícones ficam em <head>

-quando digitar link, basta selecionar a opção favicon.ico

-nem todos ícones baixados vão funcionar por isso se não funcionar tente outro

**Exemplo:**

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="shortcut icon" href="ícone aqui" type="image/x-icon">

    <title>Imagens</title>

</head>

**Imagens:**

Passos para adicionar imagens no site:

-Colocar as imagens na mesma pasta do projeto

-Caso a imagem fique muito grande basta editar o tamanho no photoshop ou em outro editor de imagens

-Aconselhável que as imagens sejam em formato png.

-Para colocar imagens externas basta colocar o endereço da imagem na internet, caso o endereço deixe de existir a imagem irá sumir.

<h1>Testando carga e imagem</h1>

<p>Adicionaremos uma imagem abaixo</p>

<p>Também podemos adicionar imagens que estão em sub-pastas</p>

<img src="imagem aqui" alt="título da imagem">

<p>Também podemos carregar imagens externas</p>

<img src="endereço da imagem web aqui" alt="título da imagem">

**Hierarquia de Títulos:**

Os títulos são ordenados dessa maneira, onde por padrão cada um é maior que o seguinte

<h1>Titulo 1</h1>

<h2>Titulo 2</h2>

<h3>Titulo 3</h3>

<h4>Titulo 4</h4>

<p>Paragráfo</p>

**Formatações de Texto:**

<body>

    <h1>Textos Marcados</h1>

    <p><mark style="background-color: greenyellow;">texto marcado 1<mark></p>

    <p><mark style="background-color: aqua;">texto marcado 2<mark></p>

    <h2>Outras Formatações</h2>

    <p><Strong>Texto em Negrito</Strong>

       <br><em>Texto em Italico</em>

       <br><big>Texto Grande</big>

       <br><small>Texto Pequeno</small>

       <br><del>Texto deletado</del>

       <br><ins>Texto Inserido</ins>

       <br>Texto Sobrescrito: Ex( x <sup>20</sup>+3)

       <br>Texto Sub-Escrito: Ex(H<sub>2</sub>O)

       <br>Script: <code>document.getElementById('teste')</code>

       <br><b>Abreviações: </b><br>Estou estudando <abbr title="HyperText Markup Language">HTML</abbr> e <abbr title="Cascading Style Sheets">CSS</abbr>

</body></html>

**Listas:**

**Listas Ordenadas:**

<body>

    <h1>Listas</h1>

    <h3>Listas Ordenadas</h3>

    <ol type="1" start="1">

        <li>Acordar</li>

        <li>Tomar café</li>

        <li>Escovar os dentes</li>

    </ol>

    <ol type="A">

        <li>Acordar</li>

        <li>Tomar café</li>

        <li>Escovar os dentes</li>

    </ol>

    <ol type="I">

        <li>Acordar</li>

        <li>Tomar café</li>

        <li>Escovar os dentes</li>

    </ol></body>

</html>

**Listas Não Ordenadas:**

<h3>Listas não Ordenadas</h3>

  <ul>

    <li>Arroz</li>

    <li>Feijão</li>

    <li>Carne</li>

    <li>Macarrão</li>

  </ul>

**Listas Mistas:**

<h3>Juntando Listas</h3>

  <ol>

     <li>PS2</li>

      <ol>

        <li>God of War 2</li>

        <li>Resident Evil 4</li>

        <li>GTA San Adreas</li>

      </ol>

        <li>PC</li>

        <ul type="square">

            <li>Bioshock</li>

            <li>Tomb Raider</li>

            <li>Metro Last Light</li>

        </ul>

   </ol>

**Links:**

-Para abrir um link em outra página:

target="\_blank"

rel="external"

-Para abrir link na mesma página:

target="\_self"

rel="external"

-Site não Patrocinado:

target="\_blank"

rel="nofollow"

-Próxima página:

rel="next"

-Página anterior:

rel="prev"

**Crie Três Arquivos na Sua pasta:**

Usaremos links para sites criados no nosso próprio pc.

Para colocarmos links de download basta colocarmos um arquivo qualquer que não seja um site em um link

**-Index.html:** página principal

<body>

    <h1>Usando links externos</h1>

    <p>Link Externo: <a href=https://roadmap.sh/pdfs/roadmaps/frontend.pdf target="\_blank" rel="nofollow">Mapa de Estudos Front-End</a></p>

    <p>Links Internos: </p>

    <p>Página 2: <a href="pag002.html" rel="next">Pagina02</a></p>

    <p>Página 3: <a href="x/pag003.html">Pagina 03</a></p>

    <p>Links para download: </p>

    <p><a href="documento do pc">HTML & CSS.pdf</a></p>

</body>

</html>

**-Pag002.html:** segunda página

<body>

    <h1>Pagina 2</h1>

    <p><a href="index.html" rel="prev">Voltar para 1° página</a></p>

</body>

</html>

Criaremos outra pasta e chamaremos de **x**, dentro desta pasta criaremos uma terceira página

**-Pag003.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Link em Pasta</title>

</head>

<body>

    <h1>Página 3</h1>

    <p><a href="../index.html">Voltar para a 1° página</a></p>

</body>

</html>

**Imagem dinâmica:**

Uma imagem que mudar de tamanho de acordo com o tamanho da tela do navegador

para isso você precisa colocar três imagens na sua pasta do arquivo

-uma imagem pequena de 300pixels

-uma imagem média de 700pixels

-uma imagem grande de 1000 pixels

As imagens menores devem estar acima das maiores no script

<picture>

  <source media="(max-width:750px)" srcset="imagens/pequena.png" type="image/png">

   <source media="(max-width:1000px)" srcset="imagens/média.png" type="image/png">

        <img src="imagens/grande.png" alt="G">

</picture>

Os comandos *max-width* e *min-width* podem definir o tamanho máximo e mínimo em que a imagem poderá ser exibida na tela

**Áudios:**

Uma das maneiras é ir no youtube, clicar no seu perfil e ir em youtube studio, na biblioteca de áudio ,filtre a licença e baixe a musica

**Maneira 1:** toca à musica adicionada

**Maneira 2:** tentar tocar um tipo de áudio, se não der certo ele tentar as outras opções

**Autoplay:** para tocar a música automaticamente (não vai funcionar em alguns navegadores)

**Controls:** botão de play da música

<h1>Reproduzindo audio</h1>

<!--maneira 1-->

<audio src="musica.mp3" controls></audio>

<!--maneira 2-->

<p>

<audio preload="metadata" autoplay controls loop>

<source src="musica.mp3" type="audio/mpeg">

<source src="musica.ogg" type="audio/ogg">

<source src="musica.wav" type="audio/wav">

</audio>

</p>

**Vídeos:**

**HandBrake:** um Software gratuito para mudar o formato dos vídeos

**Pexels:** para baixar imagens e também vídeos sem direitos autorais

**Vídeos locais:**

-nesse script adicionei o mesmo vídeo e coloquei em três formatos diferentes, utilizando o HandBreak.

-Na maneira 2 tentará colocar o primeiro vídeo, caso o navegador não leia, ele tentará os outros formatos.

<h1>Reproduzindo video</h1>

<p>Video do meu próprio servidor</p>

<!--maneira 1-->

<video src="midia/video.mp4.mp4" width="500" height="300" controls><video>

<!--maneira 2-->

<video controls width="500" height="300">

<source src="midia/video.mp4.mp4" type="video/mp4">

<source src="midia/Video.webm.webm" type="video/webm">

<source src="midia/Video.mkv.mkv" type="video/mkv">

<p>Seu navergador não tem compatibilidade com reprodução de videos</p>

</video

**Vídeos em servidores externos:**

**Youtube ou Vimeo:** para colocar bastar ir no vídeo desejador, clicar em compartilhar

-> incorporar e copiar o script, depois colar no vscode

Cuidado com os direitos autorais

**Exemplo de iframes incorporados do youtube e do vímeo:**

<body>

    <h1>Videos</h1>

    <p>video do youtube: </p>

    <iframe width="560" height="315" src="https://www.youtube.com/embed/SFzy7fNG4q4?si=xgt5ruL9Cplt4rQs" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>

    <p>video do vimeo: </p>

    <iframe src="https://player.vimeo.com/video/866854834?h=fc78b02e26" width="640" height="360" frameborder="0" allow="autoplay; fullscreen; picture-in-picture" allowfullscreen></iframe>

<p><a href="https://vimeo.com/866854834">Citizen Tourist - Animation Short Film</a> from <a href="https://vimeo.com/favostudio">FAVO Studio</a> on <a href="https://vimeo.com">Vimeo</a>.</p>

</body>

</html>



INTRODUÇÃO AO CSS

**Estilos em HTML:**  edições nativas HTML para ativa-los basta usar a tag style no local onde planeja editar

**Exemplos de comandos:** existem uma grande variedade de possíveis estilos em HTML veremos alguns exemplos a seguir

**Text align**: serve para organizar o texto e deixa-lo alinhado na posição que escolhermos

**Background color:** cor de fundo

**Color:** cor da letra

<body style="background-color: black;">

    <h1 style="text-align: center;">BEM VINDO</h1>

    <!--Adicionando estilo em um único paragráfo-->

    <p style="color: yellow;">

        Aprendendo html & Css para criação, edição e manutenção de sites, servindo para a maioria dos navegadores e mobile também.

    </p>

</body>

**Edição válida para todo Script:**

Para que não precisamos escrever <script> em toda linha vamos em head e criaremos uma tag Style que vale para todo o Script, bastando apenas especificar a parte que devemos editar:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Estilo em Head</title>

    <!--adiciaremos a tag style para todo o Script-->

    <style>

        body{

            background-color: beige;

            font-family: Arial, Helvetica, sans-serif;

            font-size: 18px;

        }

        h1{

            color: blue;

            font-size: 25px;

        }

        p{

            text-align: justify;

        }

    </style>

</head>

<body>

    <h1>BEM VINDO</h1>

    <p>

        Aprendendo html & Css para criação, edição e manutenção de sites, servindo para a maioria dos navegadores e mobile também.

    </p>

</body> </html>

**Exemplos de funcionalidades em Style:**

**Os tamanhos podem ser dados em px ou em:**

Text-ident: distância para início de paragrafo

Text-align: alinhamento de texto

Font-size: tamanho da fonte

Font-Family: família da fonte

Line-height: distância entre as linhas

**Estilos Externos:**

Estilos que ficam fora do documento html em si, geralmente em outro documento do tipo css, um documento de estilo css pode definir o estilo para várias páginas html ao mesmo

Para isso criaremos um arquivo para estilo em Css que se aplica a todas as páginas, usando assim o mesmo padrão para todas as páginas, o nome do arquivo deve terminar com .css

Por exemplo criaremos um arquivo chamado **Style.css** que será nossa folha de estilos

@charset “UTF” são os tipos de caracteres utilizados

**Style.css:**

@charset "UTF8";

 body {

    background-color: gainsboro;

    font-family: Arial, Helvetica, sans-serif;

    font-size: 18px;

}

 h1 {

    color: blue;

    font-size: 25px;

}

 h2{

    font-size: 22px;

}

 p{

    text-align: justify;

}

Agora vamos compartilha esse estilo para algumas páginas html

**Pagina 1:** index.html

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Estilos Externos</title>

    <!--link do estilo css-->

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>BEM VINDO</h1>

    <p>Aprendendo html & Css para criação, edição e

     manutenção de sites, servindo para a maioria dos navegadores e mobile também. </p>

     <!--Link para página 2-->

     <p style="text-align: right;"><a href="pag02.html" target="\_self">ir para proxima página</a></p>

</body>

</html>

**Pagina 2:** pag02.html

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Pagina 2</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>Pagina 2</h1>

    <h2>Teste de Estilos</h2>

    <p>Usaremos um link style.css para usar o mesmo estilo de dsign em todas as páginas agora</p>

    <p style="text-align: right;"><a href="index.html" target="\_self">voltar para a primeira página</a></p>

</body>

</html>

Observe que todas as páginas html usam a mesmo estilo por conta do link css

**Dica:**

Visualize cores no site color.adobe.com, onde você pode escolher o modo de cores (aconselhável RGB), modo de harmonia e harmonia de cores.

No mesmo site temos a função extrair tema, onde adicionamos uma imagem e o site cria uma paleta de cores automaticamente.

**Estilos de Texto:** editando estilo de texto em css.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Estilo de texto</title>

    <style>

        h1{

            /\*Espaço entre as letras\*/

            letter-spacing: 5px;

            /\*alinhamento de texto\*/

            text-align: center;

            /\*sombra do texto\*/

            text-shadow: 2px 2px red;

            /\*estilo da fonte\*/

            font-style: italic;

        }

        p{

            /\*distância para o início do parágrafo\*/

            text-indent: 15px;

            /\*espaço entre as linhas\*/

            line-height: 25px;

            /\*espaço entre as palavras\*/

            word-spacing: 7px;

            text-align: justify;

        }

    </style>

</head>

<body>

    <h1>Titulo</h1>

    <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Autem voluptatem sint esse laborum recusandae id neque voluptas. Sed consequatur, ipsam qui provident eaque saepe pariatur. Sapiente ipsa quis tempora illo?</p>

</body>

</html>

**Degradê em CSS**

Para isso criarmos um estilo degradê na tela precisamos criar um background image, na função linear-gradient, adicionado duas ou mais cores, uma para o início, meio e fim do degradê (no meio é opcional, precisamos de cores apenas no início e fim), para definir a posição do degradê basta colocar:

**to bottom:** de cima para baixo

**to right:** da esquerda para direita

**to left:** da direita para esquerda

**Exemplo:** o próximo script é um exemplo de degradê de preto para verde limão de cima para baixo, no caso usaremos a tag style dentro do próprio html, pois é apenas uma página e é um script simples.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Tela em Degradê</title>

    <style>

        /\*configurações globais css\*/

        \*{

            background-image: linear-gradient(to bottom,black,darkgreen, lime);

            /\*Fundo fixo\*/

            background-attachment: fixed;

            color: aliceblue;

        }

    </style>

</head>

<body>

    <h1>MENU</h1>

</body>

</html>

**Design de cores:**

Outras opções de estilos e de design de tela.

**Border-radius:** curvatura da tela

**Padding:** espaço lateral dentro da caixa, podendo ser:

padding-top: para cima

padding-bottom: para baixo

padding-right/left: para esquerda ou direita

**Margin:** posição que caixa de texto vai ficar sempre

Margin : auto, caixa no meio

Margin : left/right , esquerda/direita

**Shorthand:** maneira de simplificar um script, por exemplo:

**Maneira completa:**

body {  
  background-color: #ffffff;  
  background-image: url("img\_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
}

**Shorthand simplificado:**

body {  
  background: #ffffff url("img\_tree.png") no-repeat right top;  
}

para que um shorthand funcione precisamos escrever os elementos na ordem exata.

**Exemplo de Design:** usaremos uma tabela simples do brasileirão como exemplo:

Como temos um script um pouco maior vamos criar um arquivo css.

**style.css:**

@charset "UTF8";

\*{

    font-family:'Courier New', Courier, monospace;

}

body{

    background-image: linear-gradient(to right,#0e1799,#04D939,#eeff06);

}

main{

    background-color:#fffffff3;

    border-radius: 10px;

    box-shadow: 5px 5px 15px rgb(0, 0, 0);

    width: 300px;

    height: 300px;

    padding: 5px;

    margin:left;

}

h1{

    color: #04D939;

    /\*sombra do texto\*/

    text-shadow:1px 1px 2px #DBF227 ;

}

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Funções de Design</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <main>

        <h1>BRASILEIRÃO</h1>

        <ol type="1" start="1">

            <li>Palmeiras</li>

            <li>Atl Mineiro</li>

            <li>Botafogo</li>

            <li>Flamengo</li>

            <li>Grêmio</li>

            <li>RB Bragantino</li>

            <li>Fluminense</li>

            <li>Athl Paranaense</li>

            <li>São Paulo</li>

            <li>Internacional</li>

        </ol>

    </main></body></html>

**Fontes**

**Fonte Serifada:** A serifa consiste em um pequeno traço, barra ou prolongamento que está presente no fim das hastes das letras.

**Exemplo:** cabrini, courie New, Times New Romam

**Fonte sem serifa:** fontes que não possuem traços ou detalhes nas letras

**Exemplo:** Tahoma, Verdana, Arial

**Safe combination:** font Family, ter uma família de fontes, para caso se o navegador não conseguir ler uma fonte, tentará ler a seguinte e assim por diante

**Tipos de medidas de fontes:**

**Medidas Absolutas:** cm, mm, in, px, pt, pc

**Medidas Relativas:** em, ex, rem, Vw, vh, %

\*Aconselhável utilizar **px** e/ou **em**

**Fontes no site Google Fonts:** onde podemos baixar ou copiar fontes, basta ir no site escolher uma fonte, para baixar basta clicar em “Download Family” ou get font, caso queira apenas copiar o link da fonte basta selecionar todos os tipos e tamanhos que você deseja ter, após isso basta ir no ícone de sacola no canto superior direito e clicar em import e copiar o

**Shothand Font:** combinação resumida do estilo de fonte, onde definimos tipos e tamanho

Para utilizar basta escrever na ordem exata:

Font: Font-style font-weight font-size font-family

**OBS:**

Font-size: tamanho da fonte, Font-weight: espessura

**Exemplo (Fontes externas copiadas):**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Fontes</title>

    <style>

        /\*importando fonte Poetsen One\*/

        @import url('https://fonts.googleapis.com/css2?family=Poetsen+One&display=swap');

        /\*importando fonte Fredericka the Great\*/

        @import url('https://fonts.googleapis.com/css2?family=Fredericka+the+Great&display=swap');

        body{

            /\*Shothand Font\*/

            font: italic lighter 16px 'Poetsen One', Arial;

        }

        h1{

            font-family: Fredericka the Great,'Courier New', Courier, monospace;

        }

        h2{

            font-family: Fredericka the Great,'Courier New', Courier, monospace;

        }

    </style>

</head>

<body>

    <h1 style="color: blue;">Trabalhando com fontes</h1>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Totam voluptatum non error obcaecati possimus, labore repellat beatae. Possimus ducimus numquam a, amet eos molestias iusto doloribus in, quia laboriosam facere.</p>

    <h2 style="color: blue;">Subtítulo</h2>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Tempore reprehenderit, dolore similique impedit harum laboriosam vitae, atque iste dignissimos rem modi accusamus fuga incidunt ducimus veniam vel aperiam ullam. Repellendus.

    </p>

</body>

</html>

**Fontes externas baixadas:** precisamos incialmente baixar uma fonte, existem vários sites, como o próprio **google fonts** citado anteriormente, também temos o **dafont.com**, entre diversos outros, quando baixarmos vamos extrair e colar o arquivo de fonte dentro de uma pasta que está dentro da pasta do html que vamos usar na criação do site, após isso vamos ao script:

**Tipos de format:** o formato da fonte para saber é só clicar na fonte e ir em propriedades

01-Opentipe (otf) , 02-Truetype(ttf) , 03-embedded-opentype

04-truetype-aat (apple Advanced Typogreaphy), 05-svg

**Exemplo:** baixei uma fonte chamada kaushan script

Para adicionar a fonte precisamos colocar @font-face{}

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Fontes externas baixadas</title>

    <style>

        @font-face {

            /\*no font-family eu posso colocar qualaquer nome, escolhi kaushan\*/

            font-family:'kaushan';

            src: url(fontes/KaushanScript-Regular.ttf) format(truetype);

            font-weight: normal;

            font-style: normal;

        }

        body{

            font-family: 'kaushan',Arial, Helvetica, sans-serif;

        }

    </style>

</head>

<body>

    <h1>Titulo</h1>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Explicabo recusandae et exercitationem perspiciatis</p>

</body>

</html>

**Ajuda:**

Para identificar fontes em sites bastas usar a extensão **Fonts Ninja** que se encontra no chrome web store

Para identificar fonte em imagens basta usar o site: **what Font Is.com** e adicionar a imagem onde está a fonte que você deseja buscar

**Seletores:**

selecione uma parte especifica do html para coloca uma formatação de estilo diferente das outras partes.

**Exemplos de seletores:**

# (id) . (class) : (pseudo-class) :: (pseudo-element) > (Elemento filho)

É aconselhável que o seletor id seja utilizado apenas uma vez, caso utilize mais de uma vez crie um seletor do tipo class

Caso queira adicionar apenas uma palavra em uma classe basta coloca-la dentro do comando <span></span>

Utilizei uma classe chamada “importante” dentro do script exemplo

**Exemplo:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Seletores</title>

    <style>

        body{

            background-color: black;

            color: yellow;

        }

        h1{

            color: aqua;

        }

        /\*seletor de id (#)\*/

        h1#principal{

            color: blue;

            text-align: center;

            font-size: 40px;

        }

        /\*seletor de classes(.)\*/

        .secundario{

            color: lightgreen;

        }

        .importante{

            color: red;

        }

    </style>

</head>

<body>

    <h1 id="principal">Tipos de alinhamento</h1>

    <h1>Titulo 1</h2>

    <p class="secundario">Lorem ipsum dolor, sit amet consectetur adipisicing elit.  Beatae, aperiam voluptatibus Lorem, ipsum dolor sit amet consectetur adipisicing elit. Natus distinctio laudantium odio </p>

    <h1>Titulo 2</h2>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit.  <span class="importante">importante</span> Minus est cum pariatur labore enim, optio perferendis aliquam maxime? Veniam sunt tempora quae officia </p>

    <h1>Titulo 3</h1>

    <p class="secundario">Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus maxime doloribus iusto dolores cum repudiandae quibusdam placeat   <span class="importante">importante</span> corporis, excepturi in nesciunt eius at similique inventore expedita  tempora voluptas nisi mollitia.</p>

</body>

</html>

**Pseudo-classe:** palavra-chave adicionada a seletores para especificar o estado especial do elemento selecionado, de maneira simples fala sobre o estado de determinado elemento da tela.

**Denotado por dois pontos “ : “**

**Div:** espaço reservado na tela.

**Hover:** Execução de um determinado comando quando o usuário passa o mouse em um determinado espaço da tela.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Pseudoclasse</title>

    <style>

        div{

            background-color:blanchedalmond;

            /\*boder: largura tipo cor\*/

            border: 1px solid black;

            text-align: center;

            height: 200px;

            width: 200px;

            /\*inline-block coloca todas as divs na mesma linha\*/

            display: inline-block;

        }

        /\*hover:ação ao passar o mouse na div\*/

        div:hover{

            background-color: greenyellow;

        }

    </style>

</head>

<body>

    <div>01</div>

    <div>02</div>

    <div>03</div>

</body>

</html>

**Funções Hover:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>hover</title>

    <style>

        /\*Ocultar paragrafo da div\*/

        div>p{

            display: none;

        }

        div:hover>p{

            /\*exibir texto\*/

            display: block;

            color: white;

            background-color: rgb(15, 15, 15);

        }

    </style>

</head>

<body>

    <h1>Hover</h1>

    <p>passe o mouse sobre o texto abaixo: </p>

    <div>

        Aqui

        <p>Texto Oculto.....</p>

    </div>

</body></html>

**Personalizando links com Pseudo-classes:**

No exemplo abaixo adicionei links de um site de roadmaps para programadores.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Links</title>

    <style>

        /\*editar links\*/

        a{

            color:lime ;

            /\*Remover sublinhado do link\*/

            text-decoration: none;

        }

        /\*links já visitados\*/

        a:visited{

            color: black;

        }

        a:hover{

            color: aquamarine;

            text-decoration: underline;

        }

        /\*cor do link no momento do clique\*/

        a:active{

            color: darkblue;

        }

        /\*depois de cada link\*/

        a::after{

            /\*content = conteúdo\*/

            content: ' [link]';

        }

        /\*antes de cada link\*/

        a::before{

            content: '≫ ' ;

        }

    </style>

</head>

<body>

    <h1 style="text-align: center;">Links</h1>

    <ul><a href="https://roadmap.sh/frontend">Roadmap FrontEnd</a></ul>

    <ul><a href="https://roadmap.sh/backend">Roadmap BackEnd</a></ul>

    <ul><a href="https://roadmap.sh/full-stack">Roadmap full-stack</a></ul>

</body>

</html>

**Box Model:**

Box model ou modelo de caixa é o espaço reservado na tela de acordo com a necessidade do desenvolvedor, existem dois modelos de caixa:

**i) box level:** quebra linha antes e depois, um espaço exclusivo para a edição

**ex:** div,h1, p, main, header, nav, article, aside, footer, form, vídeo

**ii) box inline:** não há quebras, continua normalmente como um elemento comum no paragrafo

**ex:** span, a , code, small, Strong, em, sup/sub, label, buttom, input, select

**Dev tools:** para ver o script de qualquer site basta clicar com o botão direito do mouse em qualquer parte da tela no site e clicar na opção inspecionar

**Configurando estilo em modelos de caixa:**

**Height:** altura, **width:** largura, **padding:** distância do texto até a borda, **border:** borda

**Margin:** distância da caixa do resto do espaço, para cima (top), para baixo(bottom), etc...

**Margin:** auto, para centralizar. **Outline:** parte de fora da caixa

**Exemplo:** usaremos um exemplo sem Shorthand e depois o mesmo com Shorthand

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Box Model</title>

    <style>

        h1{

            height: 300px;

            width: 300px;

            background-color: lightgrey;

            border-width: 10px;

            border-style: solid;

            border-color: darkgray;

            text-align: center;

            padding-top: 1px;

            margin: auto;

            margin-top:25px;

            outline-style:dashed ;

            outline-width:10px;

            outline-color: aqua;

        }

        a{

            border-width: 10px;

            border-style: solid;

            border-color: red;

            padding: 2px;

            margin-left: 10px;

        }

    </style>

</head>

<body>

    <h1>Box Level</h1>

    <p>Ex: paragrafos</p>

    <h1>Inline Level</h1>

    <p>Ex: <a href="#">links</a></p>

</body>

</html>

**Box model com Shorthand:** para simplificar o script.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Box Model</title>

    <style>

        h1{

            background-color: lightgray;

            height: 300px;

            width: 300px;

            /\*shorthands\*/

            /\*border: height-> style-> color\*/

            border: 10px solid darkgray;

            /\*padding: top-> right-> bottom-> left\*/

            padding: 10px 10px 10px 10px;

            /\*margin: top-> right-> bottom-> left\*/

            margin:30px 40px 50px 70px ;

            /\*outline: widht-> style-> color\*/

            outline: 15px dashed aqua;

            text-align: center;

        }

        a{

            border: 10px solid red;

            padding: 2px;

            margin-left: 10px;

        }

    </style>

</head>

<body>

    <h1>Box Level</h1>

    <p>Ex: paragrafos</p>

    <h1>Inline Level</h1>

    <p>Ex: <a href="#">links</a></p>

</body></html>

**Mudar tipo de box:**

Para isso basta mudar o display, por exemplo o h1 é box level, para transforma-lo em box inline, basta apenas:

h1{

display: inline;

}

**Organizando Tela:**

**Header:** cabeçalho, **main:** principal, **Footer:** rodapé , **Nav:** ambiente de navegação

**Section:** sessão, **Article:** artigo, **aside:** conteúdo relativo ao artigo

**Exemplo:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        body{

            font-family:Arial, Helvetica, sans-serif;

            background-color:lightgray;

        }

        nav{

            background-color: darkgray;

            color: white;

            /\*adicionado sombra\*/

            box-shadow:2px 2px rgba(0, 0, 0, 0.562);

            /\*arredondamento da borda\*/

            border-radius: 20px 0px;

            padding: 2px;

        }

        /\*editando links\*/

        nav > a{

            text-decoration: none;

            color: lime;

        }

        main{

            padding: 3px;

            margin: auto;

            margin-top: 10px;

            background-color: white;

            border-radius: 20px;

            box-shadow:2px 2px rgba(0, 0, 0, 0.562);

        }

        article>aside{

            color: white;

            background-color: darkgray;

            box-shadow: inset 1px 1px 1px black;

            /\*inset: sombra pra dentro\*/

        }

        footer{

            color: white;

            background-color: black;

        }

        #assuntos{

            font-family: 'Lucida Sans', 'Lucida Sans Regular', 'Lucida Grande', 'Lucida Sans Unicode', Geneva, Verdana, sans-serif;

        }

    </style>

</head>

<body>

    <header>

        <h1>Header</h1>

        <p>essa parte é reservada apenas para o cabeçalho do site/documento html</p>

    </header>

    <nav>

        <p>parte reservada ao ambiente de navegação</p>

        <a href="#">adicionado link</a>

    </nav>

    <main>

        <p><strong>Parte principal do site/documento html</strong></p>

        <section id="assuntos">

            <h2>Industria</h2>

            <p>parte reservada as sessões do seu site</p>

        </section>

        <section id="assuntos">

            <article>

                <h2>Tecnologia</h2>

                <p>dentro de uma sessão podemos criar um artigo</p>

                <aside>

                    <p>comentário relativo ao artigo</p>

                </aside>

            </article>

        </section>

    </main>

    <footer>

        <p>Parte dedicada ao rodapé do site</p>

    </footer>

</body>

</html>

**Box em Forma de Bola:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bola</title>

    <style>

        body{

            background-color: black;

        }

        div#bola{

            height:300px;

            width:300px;

            background-color: cyan;

            margin: auto;

            border-radius:50%;

        }

    </style>

</head>

<body>

    <div id="bola">

    </div>

</body>

</html>

**Dicas:**

Para envelopar um texto selecione, aperte ctrl+shift+p, pesquise por Emmet: Envelope com abreviatura, e escolha a maneira de envelopamento, ex: h1, h2, p, etc...

CTRL+Shift+L: múltiplos cursores em palavras repetidas

**Variáveis em CSS:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        /\*Criando variáveis em root\*/

        :root{

            --cor:#044966;

            --fonte\_padrao:Arial,Verdana,Sans-serif;

        }

        body{

            background-color:var(--cor);

            font-family: var(--fonte\_padrao);

        }

    </style>

</head>

<body>

    <h1>Olá Mundo</h1>

</body>

</html>

**Técnicas para melhor responsividade:**

Para testar responsividade aumente e diminua a largura da tela.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Responsividade</title>

    <style>

        body{

            background-color: gray;

        }

        main{

            background-color: white;

            padding: 15px;

            border-radius: 10px;

            /\*largura máxima e mínima da seção\*/

            min-width: 320px;

            max-width: 800px;

            margin: auto;

        }

        img{

            /\*sempre veremos a 100% da largura da imagem\*/

            width:100%;

        }

    </style>

</head>

<body>

    <main>

        <h1>Testando Responsividade</h1>

        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Quis modi omnis ex provident rerum reiciendis laudantium voluptate expedita, facere maxime, eaque consequatur ipsa debitis cupiditate accusantium mollitia, sed enim repellendus. laudantium molestias!</p>

        <img src="adicione uma imagem" alt="img">

    </main>

</body>

</html>

**Posição de Vídeo YouTube:** Primeiramente enveloparemos o vídeo em uma div e assim faremos a configuração do tamanho e a posição do vídeo.

Para adicionar um vídeo do YouTube no site basta ir no vídeo desejado clicar em compartilhar, depois em incorporar e copiar o link do iframe.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        div.video{

            background-color: blue;

            margin: 0px -20px 30px -20px;

            padding: 20px;

            padding-bottom: 60%;

            position: relative;

        }

        div.video>iframe{

            position: absolute;

            top: 5%;

            left: 5%;

            width: 90%;

            height: 90%;

        }

    </style>

</head>

<body>

    <h1>video do youtube: </h1>

    <div class="video"><iframe width="560" height="315" src="https://www.youtube.com/embed/SFzy7fNG4q4?si=xgt5ruL9Cplt4rQs" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe></div>

</body>

</html>

**Imagens de Fundo:**

Aconselhável adicionar a imagem na mesma pasta do site, para buscar imagem basta clicar crtl+espaço dentro da url.

**Exemplo 1:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Background image</title>

    <style>

        body{

            /\*imagem de fundo da tela\*/

            background-image: url(imagem aqui);

            /\*não repetir imagem\*/

            background-repeat: no-repeat;}

        div{

            background-image: url(imagem aqui);

            /\*configurações da div\*/

            display: inline-block;

            border: 2px solid black;

            border-radius: 10px;

            background-color: lightgray;

            width: 300px;

            height: 300px;

        }

    </style>

</head>

<body>

    <div></div>

</body>

</html>

**Testando imagem:**

Background position: parte da imagem exibida na tela.

Por exemplo:

center top: centro cima

center bottom: centro baixo

top left: esquerda para cima

bottom right: direita para baixa

Dentre outras combinações de posições da imagem na tela

**Exemplo de adaptação de imagem:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Fundo 3</title>

    <style>

        body{

            height: 100vh;

            background-color: black;

            background-image: url(imagem aqui);

            background-position: center top;

            background-repeat: no-repeat;

            /\*adapta a imagem de acordo com o tamanho da tela\*/

            background-size: cover;

            /\*fixed: a imagem fica fixa durante a rolagem de tela\*/

            /\*scroll: a imagem se movimenta enquanto a rolagem a feita\*/

            background-attachment: fixed;

        }

        main{

            background-color: white;

            width: 400px;

            padding: 10px;

            margin: auto;

            border: 1px solid darkblue;

            border-radius: 15px;

            max-width: 700px;

        }

    </style>

</head>

<body>

    <main>

        <h1>Testando imagem de fundo</h1>

        <p>adicione um texto bem longo para testar a barra de rolagem.</p>

    </main>

</body>

</html>

**Centralizando div:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Centralizar div</title>

    <style>

        div{

            width: 400px;

            height: 400px;

            background-color: black;

            position:absolute;

            left: 50%;

            top: 50%;

            transform: translate(-50%,-50%);

        }

    </style>

</head>

<body>

    <div></div>

</body>

</html>

**Flexbox**

Permite a organização dos Elementos da página, assim facilitando a organização dos elementos web. Os filhos de um elemento com flexbox pode se posicionar de várias formas e dimensões flexíveis para se adaptar

**Exemplo básico:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Flex</title>

    <style>

        .container{

            display: flex;

            /\*Posição das divs: coluna ou linha\*/

            flex-direction: row;

            /\*quebra de linha se preciso\*/

            flex-wrap: wrap;

            /\*alinhar items\*/

            justify-content: center;

            /\*alinhar objetos\*/

            align-items: center;

        }

        .item{

            margin: 10px;

            width: 80px;

            height: 40px;

            padding: 50px;

            border: 2px solid black;

            color: black;

        }

        .item:hover{

            background-color: greenyellow;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="item">1</div>

        <div class="item">2</div>

        <div class="item">3</div>

        <div class="item">4</div>

        <div class="item">5</div>

        <div class="item">6</div>

    </div>

</body>

</html>

**Exemplo 2:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Flex-box</title>

    <style>

        .item{

            width: 80px;

            height: 40px;

            padding: 50px;

            border: 2px solid black;

            color: black;

        }

        .flex {

           display: flex;

           flex-wrap: wrap;

           justify-content: center;

        }

        /\*selecionar o item 3\*/

        .flex .item:nth-of-type(3) {

        background-color: aqua;

        /\*organiza elemento do container de forma dinâmica\*/

        flex-shrink: 5;

        }

        .flex .item:nth-of-type(4) {

          background-color: lime;

          flex-shrink: 5;

        }

  </style>

</head>

<body>

    <main class="flex">

        <div class="item">1</div>

        <div class="item">2</div>

        <div class="item">3</div>

        <div class="item">4</div>

        <div class="item">5</div>

        <div class="item">6</div>

    </main>

</body>

</html>

**Menu de navegação:** personalizando menu de navegação

**Arquivo html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <div class="container">

        <nav>

            <div class="logo">

                <span>Menu</span>

            </div>

            <ul>

                <li><a href="#">Sobre</a></li>

                <li><a href="#">Preços</a></li>

                <li><a href="#">Compras</a></li>

                <li><a href="#">Home</a></li>

            </ul>

        </nav>

    </div>

</body>

</html>

**Style.css:** adicionei uma imagem chamada céu estrelado como background, baixe uma imagem e coloque de fundo na tela

@charset 'UTF-8';

\*{

    margin: 0;

    padding: 0;

    box-sizing: bourder border-box;

}

body{

    /\*configurando imagem de fundo\*/

    background-image:linear-gradient(#00000001,#00000063),url(ceu\ estrelado.webp);

    background-position: center;

    background-size: cover;

    background-repeat:no-repeat;

    min-height: 100vh;

}

.container{

    max-width: 1400px;

    padding: 3%;

    margin: 0 auto;

}

nav{

    display: flex;

    align-items: center;

    justify-content: space-between;

}

nav .logo{

    font-size: 35px;

    font-style: italic;

    color: white;

    font-weight: bold;

}

nav ul{

    list-style: none;

}

nav ul li{

    display: inline-block;

}

nav ul li a{

    color: white;

    padding: 10px 20px;

    text-decoration: none;

    font-size: 20px;

    border-radius: 30px;

    transition: 0.6s;

}

nav ul li a:hover{

    background-color: aquamarine;

}

**Tabelas**

Hierarquia de tabelas(simples):

Table: tabela (table)

Table row: linha de tabela (tr)

Table header: cabeçalho de tabela (th)

Table data: dados de tabela (td)

**Criando e personalizando Tabela:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Tabelas</title>

    <style>

        /\*configurar borda\*/

        table{

            width: 400px;

            /\*borda juntas ou separadas\*/

            border-collapse: collapse;

            border: 2px solid red;

        }

        td{

            border: 1px solid black;

            padding: 2px;

        }

        tr.linha{

            background-color: grey;

        }

        td.dado{

            background-color: yellow;

        }

    </style>

</head>

<body>

    <h1 style="color: red;">Tabela</h1>

    <table>

        <tr>

            <td>A1</td>

            <td>B1</td>

            <td>C1</td>

        </tr>

        <tr class="linha">

            <td>A2</td>

            <td>B2</td>

            <td>C2</td>

        </tr>

        <tr>

            <td>A3</td>

            <td class="dado">B3</td>

            <td>C3</td>

        </tr>

    </table>

</body>

</html>

Para que a tabela se torne mais responsiva podemos definir a porcentagem que da largura da tela ocupada pela tabela:

table {  
  width: 100%;  
}

para que a tabela se torne mais responsiva podemos editar seu estilo, podendo assim editar sua posição na tela e largura e altura ocupada, assim permanencendo responsiva mesmo quando aumentamos ou diminuímos o tamanho da tela:

table{

width: 70%;

top: 30%;

left: 50%;

transform: translate(-50%,-50%);

}

**Anatomia para tabelas grandes:**

Table

CAPTION (legenda da tabela)

THEAD (cabeça)

TBODY (corpo)

TFOOT (pé)

**Exemplo:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Tabela 3</title>

    <style>

        body{

            font-family: Arial, Helvetica, sans-serif;

        }

        table{

            width: 500px;

        }

        caption{

            text-align: left;

            padding-bottom: 10px;

        }

        td,th{

            border: 1px solid black;

            padding: 10px;

        }

        td.numero{

            text-align: right;

        }

        thead{

            background-color:rgb(0, 0, 0);

            color: white;

        }

        /\*nth-chield(2n): de 2 em 2\*/

        tbody>tr:nth-child(2n){

            background-color: beige;

        }

    </style>

</head>

<body>

    <h1>Tabelas Grandes</h1>

    <table>

        <caption>População de estados brasileiros</caption>

        <thead>

            <tr>

                <th>Estado</th>

                <th>População</th>

            </tr>

        </thead>

        <tbody>

            <tr>

                <td>São Paulo</td>

                <td class="numero">44 411 238</td>

            </tr>

            <tr>

                <td>Minas Gerais</td>

                <td class="numero">20 538 718</td>

            </tr>

            <tr>

                <td>Rio de janeiro</td>

                <td class="numero">16 054 524</td>

            </tr>

            <tr>

                <td>Bahia</td>

                <td class="numero">14 141 626</td>

            </tr>

            <tr>

                <td>....</td>

                <td class="numero">....</td>

            </tr>

        </tbody>

        <tfoot>

            <th>total de habitantes</th>

            <td class="numero">203 062 512</td>

        </tfoot>

    </table>

</body>

</html>

**Cabeçalho fixo na rolagem:** o cabeçalho permanece fixo mesmo quando descemos uma tabela grande em uma barra de rolagem

**Parte do Script:**

thead>tr>th{

position: sticky;

top: -1px;

background-color: red;

}

**Mesclagem de Células:**

**Colspan:** mesclagem de coluna

**Rowspan:** mesclagem de linha

**Ex:**

Esse dado ocupará duas posições de coluna

<td colspan="2">B</td>

Esse dado ocupará duas posições de linha

<td rowspan="3">D</td>

**Colgrup:** grupo de coluna, onde cada elemento da coluna tem uma personalização em comum

**Ex:** Vamos criar um grupos de colunas e depois uma coluna para cada grupo:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Grupo de coluna</title>

    <style>

        table{

            border-collapse: collapse;

        }

        td,th{

            border: 1px solid black;

            padding: 10px;

        }

        /\*Editando cada coluna\*/

        col.cnome{

            background-color: greenyellow;

        }

        col.cprof{

            background-color: aqua;

        }

    </style>

</head>

<body>

    <table>

        <!--Criando grupos de colunas-->

        <colgroup>

            <col class="cnome">

            <col class="cprof">

        </colgroup>

        <caption>

            <tr>

                <!--Criando um coluna para cada grupo-->

                <th scope="col">Nome</th>

                <th scope="col">Profissão</th>

            </tr>

            <tr>

                <td>Joaquim</td>

                <td>Professor</td>

            </tr>

            <tr>

                <td>Angelo</td>

                <td>Programador</td>

            </tr>

            <tr>

                <td>Ana</td>

                <td>Comerciante</td>

            </tr>

        </caption>

    </table>

</body>

</html>

**Iframes:**

Quadros inline de outros sites na tela.

Para adicionar um iframe basta apenas colocar a tag <iframe></iframe> e depois adicionar o link do site em : src="aqui", após isso basta configurar a altura e largura o seu iframe , lembrando que nem todos os sites permitem iframes, na verdade não é tão fácil encontrar sites que permitam,

**Exemplo:** adicionei um iframe site do Windows.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>1° Iframe</title>

</head>

<body>

    <iframe src="https://www.microsoft.com/pt-br/windows/?r=1" height="700" width="400" frameborder="0"></iframe>

</body>

</html>

**FrameBorder:** 0-> sem borda, 1-> com borda

**Scolling:** configura barra de rolagem

**Navegando em iFrames:**

Adicionaremos uma lista de links que poderão ser exibidos em um iframe com id=tela, para que os links selecionados sejam exibidos neste iframe basta colocar target=”tela”.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Iframe-002</title>

    <style>

        iframe{

            width: 500px;

            height: 700px;

        }

    </style>

</head>

<body>

    <h1>Veja informações de sua liga favorita</h1>

    <ul>

        <li><a href="https://ge.globo.com/futebol/futebol-internacional/liga-dos-campeoes/" target="frame">Liga dos Campeões</a></li>

        <li><a href="https://ge.globo.com/futebol/libertadores/" target="frame">Libertadores</a></li>

        <li><a href="https://ge.globo.com/futebol/brasileirao-serie-a/" target="frame">Brasileirão Serie A</a></li>

    </ul>

    <iframe id="tela" name="frame"></iframe>

</body>

</html>

**iFrame de Mapa:**

Adicionando parte de um mapa do google maps ou waze em um iframe.

<body>

    <h1>Google Maps</h1>

    <p>Basta ir em Google maps, escolher a parte do mapa, clicar em compartilha->incorporar mapa->copiar html</p>

    <iframe src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d31808.622492176233!2d-42.572253499999995!3d-4.7564966499999946!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x78e0048b03875b9%3A0x70a6dc5da6485138!2sJos%C3%A9%20de%20Freitas%2C%20PI%2C%2064110-000!5e0!3m2!1spt-BR!2sbr!4v1706201620677!5m2!1spt-BR!2sbr" width="600" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>

    <h1>Mapa Waze</h1>

    <p>Procedimento igual ao do google maps, no site do waze.com </p>

    <iframe src="https://embed.waze.com/iframe?zoom=16&lat=-4.758264&lon=-42.576465&ct=livemap" width="600" height="450" allowfullscreen></iframe>

</body>

</html>

**Segurança do iframe:**

Para evitar que outros sites tenham acesso ao nosso site podemos usar a função *sandbox* limitando o acesso dos sites no iframe

Limitações do Sandbox:

***sandbox="allow-same-origin",*** permitir conteúdo de mesma origem

***sandbox=allow-scripts***, permite scripts

***sandbox="allow-forms",*** permite formulários

**Formulários:**

Para salvar dados de um formulário precisamos de banco de dados ou de variáveis em alguma linguagem de programação, porém veremos isso apenas mais à frente usando JavaScript, primeiramente veremos como criar formulários e configura-los.

Para criar um formulário basta usar a tag <form></form> e adicionar seu “action” que será o arquivo que vai armazenar os dados do formulário, como vamos apenas criar e configurar os dados do formulário, não vamos dar importância para onde vão nossos dados, então criaremos um arquivo chamado ***cadastro.js***, basta ir em novo arquivo no Vscode e criar um arquivo com esse nome e deixa-lo em branco.

**Elementos:**

**Method:** cada tipo de method serve é um tipo de proteção de dados diferente

**Get:** para dados não sensíveis e sem uso de média

**Post:** para dados sensíveis e imagens

**Autocomplete: caso** esteja em ***on*** ele irá mostrar sugestões de acordo com os dados preenchidos anteriormente

**Input:** campo para o usuário inserir dados, existem vários tipos de campos, desde textos, imagem, cores e etc...

**Label:** legenda para um input, serve para conectar o título do input com input

**OBS:** quando enviar dados por get, há um limite máximo de 3mil bytes (3kb), aproximadamente 3 mil letras

**Tipos de Input:** veremos todos os tipos nos exemplos de formulário a seguir.

**text:** caixa de texto

**submit:** botão de submissão

**password:** senha

**reset:** botão para limpar input

**number:** ler número

**month:** mês e ano

**date:** data

**time:** horário

**tel:** telefone

**email:** e-mail

**radio:** caixa de rádio para marcar apenas um elemento

**checkbox:** caixa de seleção para marcar um ou vários elementos

**color:** cor

**range:** nível de satisfação

**file:** arquivo

**url:** endereço de site

**1° formulário:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Formulário 1</title>

</head>

<body>

    <h1>1° Formulário</h1>

    <form action="">

        <p>nome: <input type="text" name="nome" id="nome"></p>

        <p>sobrenome: <input type="text" name="sobrenome" id="sobrenome"></p>

        <p><input type="submit" value="Enviar"></p>

    </form>

</body>

</html>

Nesse exemplo os Label e Caixas de texto estão conectados pelo ***id***, então se eu clicar em cima do texto do label serei direcionado para a caixa de texto

<body>

    <h1>1° Formulário</h1>

    <form action="cadastro.js" autocomplete="off" method="get">

        <!--Criando Label e caixas de texto-->

        <p><label for="inome">

            nome: </label> <input type="text" name="nome" id="inome">

        </p>

        <p><label for="isobrenome">

            sobrenome: </label><input type="text" name="sobrenome" id="isobrenome">

        </p>

        <!--Criando botão de enviar-->

        <p>

            <input type="submit" value="Enviar">

        </p>

    </form>

</body>

**Formatações para Formulários:**

**Requierd:** para campos obrigatórios

**Minlength e maxlength:** máximo e mínimo de caracteres em campos de texto

**Exemplo 2:**

<body>

    <h1>2° Formulário</h1>

    <form action="cadastro.php">

        <p>

            <label for="iuser">Usuário: </label>

            <input type="text" name="user" id="iuser" required minlength="5" maxlength="15" size="15" placeholder="nome do usuário" autocomplete="username">

        </p>

        <p>

            <label for="isenha">Senha: </label>

            <input type="password" autocomplete="current-password" "Senha" id="isenha" required minlength="8" size="15" placeholder="senha">

        </p>

        <p>

            <input type="submit" value="Enviar">

            <input type="reset" value="limpar">

        </p>

    </form>

</body>

**Exemplo 3:**

**Fieldset:** grupo de dados

**Legend:** titulo do fieldset

**Pattern:** expressões regulares

**OBS:** Para configurar cadastros de dados como: número de telefone, cpf, cnpj e outro tipos de dados em sites deve-se usar expressões regulares para a validação desses dados para que os dados sejam enviados de forma correta, esse tema será abordado mais à frente, no momento o uso dessas expressões é opcional, porém vou usar no exemplo do telefone abaixo

<body>

    <h1>4° Formulário</h1>

    <form action="cadastro.php">

        <fieldset>

            <legend>Dados pessoais</legend>

            <p>

                <label for="iemail">E-mail</label>

                <input type="email" size="30">

            </p>

            <p>

                <label for="itel">Telefone</label>

                <input type="tel" name="tel" id="itel" autocomplete="tel"

                pattern="^\(\d{2}\)\d{4,5}-\d{4,5}$">

            </p>

            <p>

                <input type="submit" value="Enviar">

                <input type="reset" value="limpar">

            </p>

        </fieldset>

    </form>

</body>

**Caixas de Seleção:**

**Exemplo 4:**

<body>

    <h1>Caixas de Seleção</h1>

    <h2>Botão de Rádio</h2>

    <fieldset>

        <legend>sexo</legend>

        <input type="radio" name="sexo" id="imas">

        <label for="imas">Masculino</label>

        <input type="radio" name="sexo" id="ifem">

        <label for="ifem">Feminino</label>

    </fieldset>

    <h2>Checkbox</h2>

    <fieldset>

        <legend>Esporte Favorito</legend>

        <input type="checkbox" name="fut" id="ifut">

        <label for="ifut">Futebol</label><br>

        <input type="checkbox" name="bas" id="ibas">

        <label for="ibas">Basquete</label><br>

        <input type="checkbox" name="ten" id="iten">

        <label for="iten">Tênis</label><br>

        <input type="checkbox" name="vol" id="ivol">

        <label for="ivol">Vôlei</label><br>

    </fieldset>

</body>

**Outros elementos no formulário:**

**1-Número no formulário:**

<label for="inota">Nota: </label>

<input type="number" name="nota" id="inota" placeholder="média" min="0" max="10" step="0.1">

**2-Mês e ano:**

<label for="imes">Periodo Letivo: </label>

<input type="month" name="mes" id="imes">

**3-Data:**

<label for="idia">Dia da prova: </label>

<input type="date" name="dia" id="idia">

**4-Horário:**

<label for="ihora">Horário: </label>

<input type="time" name="hora" id="ihora">

**5-Cor:**

<label for="icor">Cor: </label>

<input type="color" name="cor" id="icor" value="#00ff00"

**6-Nível de satisfação:**

<label for="inv">Nivel de satisfação: </label>

<input type="range" name="nivel" id="inv"  min="1" max="10" value="2">

**7-Arquivo:**

<label for="iarquivo">Foto do perfil: </label>

<input type="file" name="foto" id="iarquivo">

**8-Endereço de site:**

<label for="iurl">Url: </label>

<input type="url" name="url" id="iurl">

**Listas de Opções:**

**Select:** opções de seleção

**optiongroup:** grupo de opções

**option:** adicionar opções

**Exemplo:** selecione um idioma, opções de idioma para escolhermos um.

**Selected:** opção selecionada por padrão

<body>

    <form action="cadastro.php" method="post">

        <h1>Lista de Opções</h1>

        <p>

            <label for="idio">Idioma</label>

            <select name="idioma" id="idio">

                <optgroup label="Grupo 1">

                    <option value="Alemão">Alemão</option>

                    <option value="Espanhol">Espanhol</option>

                    <option value="Francês">Francês</option>

                    <option value="Grego">Grego</option>

                </optgroup>

                <optgroup label="Grupo 2">

                    <option value="Italiano">Italiano</option>

                    <option value="Inglês">Inglês</option>

                    <option value="Mandarin">Mandarin</option>

                    <option value="Português"selected>Português</option>

                </optgroup>

            </select>

        </p>

        <p>

            <input type="submit" value="Enviar">

        </p>

</body>

**Datalist:**

Permite que seja digitada uma opção que não está na lista.

<p>

<label for="iprof">Profissão: </label>

<input type="text" nome="prof" id="iprof" list="listprof">

<datalist id="listprof">

<option>Administrador</option>

<option>Advogado</option>

<option>Motorista</option>

<option>Programador</option>

<option>Vendedor</option>

</datalist>

</p>

**TextArea:**

Área de texto, o TextArea é infinito.

<label for="imsg">Mensagem: </label><br>

<textarea name="msg" id="msg" cols="50" rows="15"></textarea>

**Display:**

display: inline-block;

Posiciona os elementos na mesma linha

display: none;

Oculta os elementos

**Formatação de Elementos na Tela:**

Baixei uma imagem de 1000px e adicionei na tela, assim vamos poder formatar a imagem de várias maneiras.

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        img{

            /\*regular tamanho da imagem\*/

            max-width: 600px;

            max-height: 500px;

            /\*posicionamento da imagem\*/

            float:right;

            /\*para onde ir os outros elementos\*/

            clear:left;

            /\*opacidade\*/

            opacity: 0.5;

        }

        div{

            background-color: bisque;

            height: 400px;

            width: 500px;

        }

    </style>

</head>

<body>

    <img src="imagem.jpg" alt="">

    <div></div>

</body>

</html>

**Media Query:**

As médias query são diferentes estilos para o mesmo site de acordo com o tamanho da tela que vamos utilizar, por exemplo: o formato para site impresso será o estilo do site caso seja impresso em uma folha de papel.

**Exemplo prático:** um site de jornal de notícias pretende ter duas versões, uma para ser exibida em um site e outra para ser impressa.

-para ver a versão de impressão basta clicar em imprimir no navegador.

**Media:** Screen-> Estilo apenas para tela

**Media:** print-> Estilo para impressora

**Media:** all-> Estilo para tudo

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="estilos/tela.css" media="screen">

    <link rel="stylesheet" href="estilos/impressora.css" media="print">

</head>

<body>

    <header>

        <h1>Media Query</h1>

        <menu>

            <ul>

                <li>Politica</li>

                <li>Esporte</li>

                <li>Tecnologia</li>

            </ul>

        </menu>

    </header>

    <main>

        <article>

            <h1>Titulo 1</h1>

            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Rerum quasi illum provident? Dolor iusto illo officia cupiditate eum earum velit, illum atque beatae, quod debitis doloremque quisquam tempora esse maiores? </p>

        </article>

    </main>

</html>

**Tela.css:** versão do estilo site para ser exibida no navegador.

@charset "UTF-8";

\*{

    margin:3px;

    padding:3px;

}

html{

    font-family: Arial, Helvetica, sans-serif;

    font-size: 1em;

}

header{

    background-color: black;

    color: yellow;

}

h1{

    text-align: center;

}

menu{

    background-color: darkgray;

}

menu ul{

    list-style-type: none;

}

menu li{

    display: inline-block;

    padding-left: 100px;

}

article{

    width: 600px;

    display: block;

    margin: auto;

}

**Impressora.css:** versão do estilo site para ser impresso.

@charset "UTF-8";

\*{

    margin: 10px;

    padding: 10px;

}

html{

    font-family: 'Courier New', Courier, monospace;

    font-size: 1.5em;

    line-height: 1.5em;

}

menu{

    /\*não exibir menu na impressão\*/

    display: none;

}

article::after{

    content: 'Esse artigo foi impresso 02/02/2024';

    font-size: 13px;

}

article{

    width: 100%;

}

**Imagem paisagem/retrato:**

Exemplo: versão de imagem exibida na tela de acordo a orientação da tela, paisagem ou retrato.

A imagem ***Portrait*** tem mais altura. Já a imagem ***Landscape*** tem mais largura.

Muitos sites utilizam a mesma imagem com tamanhos de altura e largura diferentes causando a impressão que é a mesma imagem mesmo após a mudança na orientação da tela.

Tamanhos das imagens que usei no exemplo:

**Portrait:** 1280px(altura) e 720px(largura)

**Landscape:** 720px(altura) e 1280px(largura)

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="estilos/retrato.css" media="screen and(orientation:portrait)">

    <link rel="stylesheet" href="estilos/paisagem.css" media="screen and (orientation: landscape)">

</head>

<body>

    <h1>Mude a orientação da sua tela</h1>

</body>

</html>

**Style.css:** estilo geral da tela

@charset "UTF-8";

/\*configurações gerais\*/

\*{

    margin: 0px;

    padding: 0px;

}

html,body{

    width: 100vw;

    height: 100vh;

    background-size:cover;

    background-repeat: no-repeat;

    background-color: blue;

}

h1{

    color:white;

    text-shadow: 3px 3px 0px black;

    padding: 10px;

    text-align: center;

}

**Retrato.css:**

@charset "UTF-8";

/\*aparelho em pé(portrait)\*/

body{

    background-image:url(../imagens/cev-portrait.jpg);

    background-position: bottom;

}

**Paisagem.css:**

@charset "UTF-8";

/\*aparelho deitado(landscape)\*/

body{

    background-image: url(../imagens/landscape.jpg);

    background-position: left bottom;

}

**Mídia Query:** estilo para cada tipo de tela de acordo com seu tamanho

**Tamanho para cada tipo de tela:**

-Pequenas telas(-600px)

-celular(600px-770px)

-tablet(770px-992px)

-Desktop(992px-1200px)

-Grandes telas(+1200px)

Veremos apenas um simples exemplo de como seriam organizadas as mídias query, pois sua versão completa ficaria muito grandes para o livro, para vê-la completa basta ir no ex35->mq004 no github do autor.

**Mídia\_query.css:**

@charset "UTF-8";

@media print{

    /\*a versão de impressora não imprime imagens\*/

}

@media screen and (min-width:768px) and (max-width:992px){

    /\*tablet\*/

}

@media screen and (min-width: 992px) and (max-width:1200px){

    /\*desktop\*/

}

@media screen and (min-width: 1200px){

    /\*grandes telas\*/

}

**Adicionando Ícones do Google:**

Inicialmente acessaremos o site <https://fonts.google.com/icons> e escolheremos um ícone. Podemos baixar o ícone e utiliza-lo como uma imagem normalmente, porém iremos usá-lo por links, para isso copiaremos dois links no nosso html.

**Primeiro link:** link adicionado no head para permitir que os ícones possam aparecer na tela.

-Esse link é universal, sempre usamos ele para utilizarmos ícones do google

<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@24,400,0,0" />

**Segundo link:** link do ícone que vamos adicionar, para encontrá-los basta clicar em cima do ícone que vamos adicionar, e copiar o link para ***insert the icon.***

**Modelo do link:** <span class="material-symbols-outlined"> Nome do icone </span>

**Exemplo:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Google icones</title>

    <link rel="stylesheet" href="style.css">

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@24,400,0,0" />

    <style>

        h1{text-align: center;}

        span#icone{

            text-align: center;

            display: inline-block;

            padding-left:120px ;

        }

    </style>

</head>

<body>

    <h1>Bem-Vindo</h1>

    <span id="icone" class="material-symbols-outlined">home</span>

    <span id="icone" class="material-symbols-outlined">menu</span>

    <span id="icone" class="material-symbols-outlined">favorite</span>

    <span id="icone" class="material-symbols-outlined">star</span>

</body>

</html>

**Input Personalizado:**

Usaremos ícones para personalizar o label de um input file:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Material+Symbols+Outlined:opsz,wght,FILL,GRAD@20..48,100..700,0..1,-50..200" />

    <style>

        body{

            color: white;

            background-color: rgba(10, 10, 109, 0.767);

        }

        h1{

            text-align: center;

        }

        div{

            margin-top: 20px;

            transform: translate(+38%,+450%);

        }

        input[type='file']{

            display: none;

        }

        label{

            display: inline;

            color: white;

            background-color: red;

            text-align: center;

            padding: 10px;

            font-size: 20px;

            box-shadow: 4px 4px 4px rgba(0, 0, 0, 0.473);

            /\*o cursor vira uma mão\*/

            cursor: pointer;

        }

        /\*posição do icone no label\*/

        span{

            transform: translate(-10%,20%);

        }

    </style>

</head>

<body>

    <h1>MENU</h1>

    <div>

        <input type="file" name="arq" id="iarq">

        <label for="iarq">

        <!--Adicionar imagem no label-->

        <span class="material-symbols-outlined">

                cloud\_upload

        </span>

        Enviar arquivo</label>

    </div>

</body>

</html>

**OBS:** Para aumentar o tamanho do ícone preciso baixa-lo e colocá-lo como img comum no html, onde podemos personalizar o tamanho e a cor do ícone.

**Dropdown:**

Opções de links que aparecem aos clicarmos ou passarmos o mouse em uma opção de menu.

**Exemplo**: dropdown de linguagens de programação

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <ul>

        <li><a href="#">Home</a></li>

        <!--Criando dropdown de linguagens-->

        <li><a href="#">Linguagens↓</a>

            <ul class="dropdown">

                <!--Lista de opções de linguagens-->

                <li><a href="#">HTML</a></li>

                <li><a href="#">CSS</a></li>

                <li><a href="#">JavaScript</a></li>

                <li><a href="#">PHP</a></li>

            </ul>

        </li>

     <!--Outras opções do menu-->

     <li><a href="#">Opções</a></li>

     <li><a href="#">Sobre</a></li>

    </ul>

</body>

</html>

**Style.css:**

@charset 'UTF\*';

\*{

    padding: 0;

    margin: 0;

    font-family: monospace;

}

ul{

    list-style: none;

    background-color: crimson;

    text-align: center;

}

ul>li{

    display: inline-block;

    position: relative;

}

ul>li>a{

    display: block;

    padding: 20px 25px;

    color: white;

    text-decoration: none;

    text-align: center;

    font-size: 15px;

}

ul>li>ul.dropdown{

    width: 100%;

    background-color: black;

    position: absolute;

    z-index: 999;

    display: none;

}

ul>li>a:hover{

    background-color: rgba(184, 12, 46, 0.863);

    border: 1px solid white;

    transition: border 0.4s

}

ul>li:hover>ul.dropdown{

    display: block;}

**Contador:**

Usado para contar as posições automaticamente pelo css.

**Ex:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        .posicao{

            counter-set: heading-counter;

        }

        /\*Exibir depois do h2\*/

        h2::after{

            counter-increment: heading-counter;

            content:"-" counter(heading-counter);

        }

    </style>

</head>

<body>

    <div class="posicao">

        <h2>Posição</h2>

        <h2>Posição</h2>

        <h2>Posição</h2>

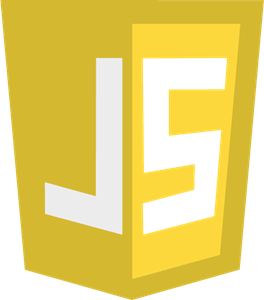
        <h2>Posição</h2>

        <h2>Posição</h2>

    </div>

</body>

</html>

**INTRODUÇÃO AO JAVASCRIPT**

**Conceitos Iniciais:** JavaScript é uma linguagem bem versátil, muito usada por programadores full-Stack pois funciona muito bem tanto para o Back-End quanto para o Front-End, neste livro o nosso foco será o JavaScript aplicado no navegador para ajudar na construção de sites.

Existem duas maneiras de adicionar o JavaScript no html, primeiramente podemos usar-lo dentro do próprio html usando o comando <script> e a outra maneira é criando um arquivo JavaScript usando .js no fim do nome do arquivo.

Também podemos usar o JavaScript no console como qualquer outra linguagem de programação, basta baixar e instalar o Nodejs no computador, assim podemos escrever e executar scripts no próprio programa do nodejs.

**Usar NodeJs no Vscode**: para usar JavaScript no console do VsCode, deve-se instalar e habilitar no Vscode a funcionalidade **NodeJS Exec** que nos ajuda adicionando a tag F8 para executar automaticamente o nodeJs para arquivo .JS, assim executando qualquer script .js usando apertando F8.

**JavaScript no html:** basta usar a tag <script>

<body>

    <h1>JavaScript: Intro</h1>

    <!--Parte JavaScript-->

    <script>

        //informação

        window.alert('Primeira mensagem')

        //pergunta

        window.confirm('Continuar?')

        //caixa de pergunta

        window.prompt('Qual seu nome?')

    </script>

</body>

</html>

O objeto window representa a janela do navegador, no JavaScript podemos usar vários comandos usando a janela do navegador

Vamos criar uma nova pasta e nessa pasta vamos criar um arquivo do formato JavaScript para testarmos o console do Vscode, caso já esteja instalado e habilitado a função **NodeJS Exec** o console vai executar o arquivo normalmente após pressionamos a tecla F8.

**Arquivo.js:**

console.log('O console funcionou corretamente')

Para que um arquivo html se conecte com um arquivo JavaScript, bastar usar um pequeno comando que vai ser exibido no Script a seguir:

**Index.html:**

<body>

    <!--Comando para conectar arquivo js ao navegador-->

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

</body>

</html>

**Arquivo.js;**

window.alert('Escrito em arquivo JS')

Mais tarde voltaremos a usar arquivo javaScript, porém no momento vamos ver alguns exemplos usando scripts dentro do html

**Variáveis em JavaScript:**

-Não podem começar com números

-Não podem obter espaços

-Não podem usar palavras reservadas, Ex: Function,Alert,etc...

// - comentário dentro da tag </Script>

**Exemplo:**

<script>

  //Criando uma variável

   var nome=window.prompt('Qual seu nome?')

   //retornando variável

   window.alert('seja bem-vindo '+nome)

</script>

Podemos também criar variáveis do tipo ***let*** que é limitada apenas ao seu escopo, logo a variável não é global, porém funciona como uma ***var*** normal dentro de seu próprio bloco.

**Exemplo:**

<script>

let idade=window.prompt('Qual sua idade?')

window.alert('Ah sim '+idade+' Anos')

</script>

**Conversões de Variáveis:** Para realizar operações matemáticas devemos converter as variáveis de String para Int ou Float, para isso usamos os comandos:

**Number.parseInt(n) :** converter para número inteiro

**Number.parseFloat(n):** converter para número real

**Number(n):** caso não seja necessário especificar o tipo numérico

**Exemplo:**

//parseInt: converter variável String para int

var n1=Number(window.prompt('Digite um número: '))

var n2=Number(window.prompt('Digite outro número: '))

var soma=n1+n2

var div=n1/n2

window.alert('Resultado da soma: '+soma)

//Usando crase:2° maneira de escrever as variáveis:

window.alert(`Resultado da divisão: ${div}`)

Outra maneira de escrever as variáveis é usando crase ao invés de aspas e usando o sinal $:

**Tratamento de dados:**

**Strings:**

var nome=window.prompt('Digite seu nome: ')

//escrever no documento html

document.write(`Seu nome tem: ${nome.length} letras`)

document.write(`<br>Tudo maiúsculo: ${nome.toUpperCase()}`)

document.write(`<br>Tudo minúculo: ${nome.toLowerCase()}`)

**Números:**

var n=1640.7

//numeros de casas décimais exibidas

document.write(n.toFixed(2)+'<br>')

//colocando em Real Brasileiro

document.write(n.toLocaleString('pt-BR',{style: 'currency',currency: 'BRL'}))

document.write('<br>')

document.write(n.toLocaleString('pt-BR',{style: 'currency',currency: 'EUR'}))

**Operadores:**

**Operadores Aritméticos:**

var n1=Number(window.prompt('Digite um número: '))

var n2=Number(window.prompt('Digite outro número: '))

window.alert(`Resultado da soma: ${n1+n2} `)

window.alert(`Resultado da subtração: ${n1-n2}`)

window.alert(`Resultado da multiplicação: ${n1\*n2}`)

window.alert(`Resultado da divisão: ${n1/n2}`)

window.alert(`Resto da divisão: ${n1%n2}`)

window.alert(`O resultado da potência de base ${n1} e expoente ${n2} é igual a : ${n1\*\*n2}`)

**Operadores relacionais:**

compara um número com outro, se maior, menor, igual ou diferente

**Operadores:**

>(maior), <(menor), >=(maior ou igual), <=(menor ou igual), ==(igual), >=(diferente), ===(idêntico)

-No javaScript, Number e String podem ser iguais, mas nunca idênticos:

5==5(true), porém 5===’5’(false)

**Operadores lógicos:**

! : negação(não) &&: conjunção(e) ||: disjunção(ou)

-Em uma expressão o JavaScript resolver primeiro os operadores Aritméticos e depois os lógicos:

**Exemplo:**

var n1=8;

var n2=5;

// &&(e)

window.alert(n1>n2 && n1%2==0)

//true && true= true

window.alert(n1>n2 && n1%2!=0)

//true && false= false

window.alert(n2>n1 && n1%2==0)

//false && true= false

window.alert(n1<n2 && n1%2==1)

//false && false = false

// ||(ou)

window.alert(n1>n2 || n1%2==0)

//true || true= true

window.alert(n1>n2 || n1%2!=0)

//true || false= true

window.alert(n2>n1 || n1%2==0)

//false || true= true

window.alert(n1<n2 || n1%2==1)

//false && false = false

**Operador Ternário:**

Funciona como uma espécie de se, senão. Sua estrutura funciona da seguinte forma:

Onde (?)=se e (:) =Senão

Proposição? Opção 1 : Opção

Se a proposição for verdadeira acontecerá a opção 1, senão acontecerá a opção 2:

**Exemplo:**

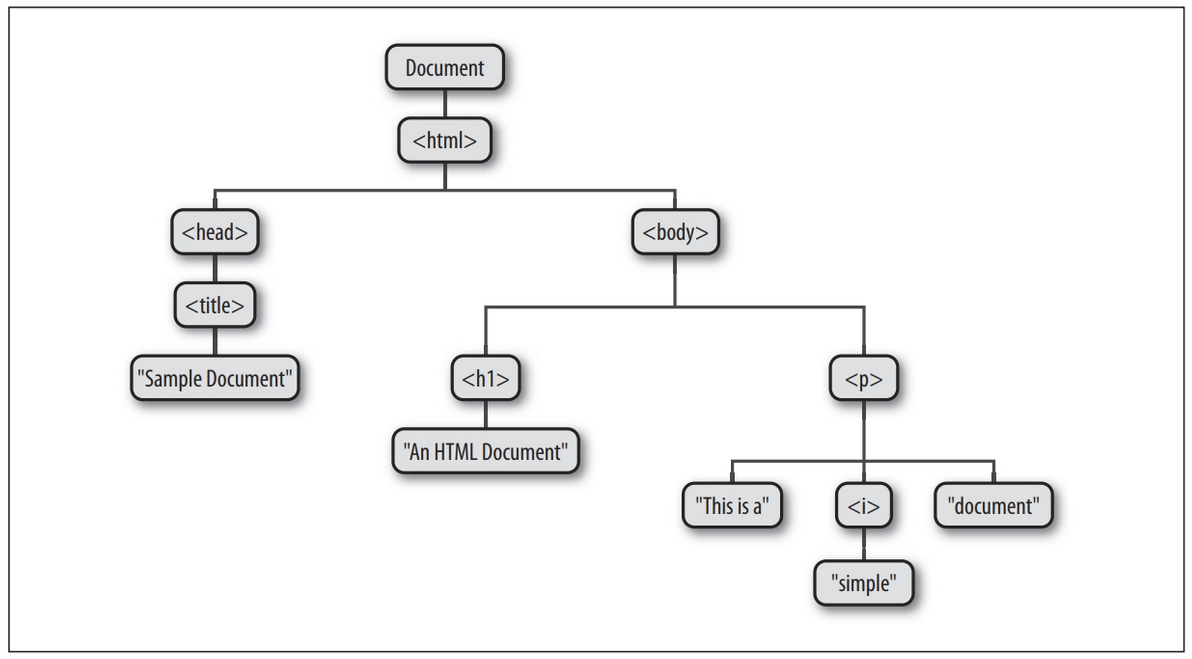
var media=Number(window.prompt('Qual a média?'));

media>=7?window.alert('Aprovado'):window.alert('Reprovado')

**DOM (Document object Model)**

Para usar o Dom primeiramente vamos baixar a extensão ***live Serve*** no VsCode, após ter instalado a extensão para usar basta clicar no arquivo html que quer abrir com o botão direito e clicar na opção “open with Live Serve”

**Live Serve:** serve para ver as alterações no html em tempo real

**Arvore Dom:** quando abrimos um documento no google, estamos abrindo uma “*window*” que será a raiz do documento, dentro de *window* temos mais temos outros objetos como *location*, *history*, *document*, entre outros... , dentre do objeto *documen*t fica o nosso objeto *html* e dentro do objeto html temos todos objetos adicionados na página do nosso site, por exemplo: h1, div, img , etc...

Usando o DOM podemos alterar os objetos do HTML no javaScript.

Veremos exemplos de como usar o DOM;

**Para ver dados da Raiz(Janela):**

window.document.write("Caracteres: "+window.document.charset)

window.document.write("<br>App navegador: "+window.navigator.appName)

window.document.write("<br >Endereço: "+window.document.URL)

**Para selecionar um objeto do HTML**

**Exemplos:**

Selecionando e alterando o background da tela:

var corpo= window.document.body

corpo.style.background='black'

**getElementsByTagName:** selecionando objetos por nome

    <p>Aprender a usar <strong>Dom</strong> em JavaScript</p>

    <script>

        //selecionando um paragrafo

        var p1=window.document.getElementsByTagName('p')[0]

        //('p') é a tag selecionada

        //[0] representa a posição do <p>

        p1.style.color='yellow'

        window.document.write(p1.innerText)

    </script>

**getElementById:** Selecionando objeto por ID

    <div id="msg">Selecionado objetos Div</div>

    <script>

        var d=window.document.getElementById('msg')

        window.document.write('<br>'+d.innerText)

        //mudando texto da Div

        d.innerText='Mudando texto da Div'

    </script>

**QuerySelector:** selecionar por seletor

    <div id="msg">Selecionado objetos Div</div>

    <script>

        var d=window.document.querySelector('div#msg')

        d.style.color='green'

    </script>

**Eventos:**

Programar ações através de eventos do usuário, como por exemplo, clicando no botão mouse ou pressionando uma tecla específica do teclado.

**Sugestão:** pesquisar **Event Reference** no google para ver opções de eventos para definir seus comandos

**Exemplos:** click(clicar na tela), mouseenter(passar o mouse), mouseout(tirar o mouse) entre diversas outras opções.

**Function (Funções) :**

Serve para definir uma função/ação quando um determinado evento acontecer

Vamos criar alguns exemplos usando arquivo do formato .js:

**Ex1:** exemplo simples de função para verificar números pares e ímpares

**Arquivo.js:**

function paridade(n){

    if(n%2==0){

        return 'Par'

    }else{

        return 'impar'

    }

}

console.log(paridade(5))

**Exemplo de Script usando Function, Dom e Events:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <style>

        div#area{

            width: 400px;

            height: 300px;

            color: white;

            background-color: cadetblue;

            text-align: center;

            line-height: 300px;

            font-size: 30px;

            /\*Centralizando div\*/

            position: absolute;

            left: 50%;

            top: 50%;

            transform: translate(-40%,-50%);

        }

    </style>

</head>

<body>

    <div id="area">Clique na Div</div>

    <script>

        //conectando a variável a div

        var area=window.document.getElementById('area')

        //adicionando evento de acordo com a ação

        area.addEventListener('click',clicar)

        area.addEventListener('mouseenter',entrar)

        area.addEventListener('mouseout',sair)

        //funções dos eventos

        function clicar(){

            area.innerText='Clicado'

        }

        function entrar(){

            area.style.background='lime'

        }

        function sair(){

            area.style.background='red'

        }

    </script>

</body>

</html>

**Input+DOM:**

Como prometido no capítulo de formulários, dessa vez vamos usar javaScript com dados recebidos em formulários.

**Arquivo.Js:**

function somar(){

    var tn1=document.getElementById('idn1')

    var tn2=document.getElementById('idn2')

    var res=document.getElementById('resultado')

    //transformando String em Number]

    var n1=Number(tn1.value)

    var n2=Number(tn2.value)

    var soma=n1+n2

    res.innerHTML= `O Resultado da soma é ${soma}`

}

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Somar valores</title>

    <style>

        input[type=number]{

            width: 100px;

            height: 25px;

        }

        input[type=button]{

            width: 70px;

            height: 30px;

        }

        #resultado{

            padding-top: 10px;

            font-size: 20px;

        }

    </style>

</head>

<body>

    <h1>Soma de Valores</h1>

    <input type="number" name="n1" id="idn1">

    <input type="number" name="n2" id="idn2">

    <input type="button" value="somar" onclick="somar()">

    <div id="resultado"></div>

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

</body>

</html>

**Onclick():** outra maneira de usar eventos é adicionando ele dentro do script do próprio botão para já definir a ação do botão.

**Condicionais:**

Condições para que uma linha de comando seja executada

**If, else: se->senão**

Explicando de maneira simples if/else pode ser resumindo da seguinte forma:

(if) Se isso for verdadeiro execute o comando A, (else) senão execute o comando B

Exemplo em um arquivo.js:

var veloscidade=100

if(veloscidade<80){

    console.log('Prossiga')

}else{

    console.log('MULTADO! por excesso de veloscidade')

}

**Aplicação, em um arquivo html,Css+javaScript:**

**Index.html**:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Medidor de Veloscidade</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <h1>Medidor de Veloscidade</h1>

    <label for="idvel">Veloscidade Atual </label>

    <input type="number" name="vel" id="idvel">km\h<br>

    <input type="button" value="Verificar" onclick="calcular()">

    <div id="resposta"></div>

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

</body>

</html>

**Style.css:**

@charset 'UTF-8';

input[type=number]{

    width: 60px;

    margin-right: 4px;

}

input[type=button]{

    margin-top: 10px;

    width: 80px;

    height: 30px;

}

div#resposta{

    padding-top: 10px;

}

**Arquivo.js:**

function calcular(){

    let veloscidade=document.getElementById('idvel')

    let resp=document.getElementById('resposta')

    let vel=Number(veloscidade.value)

    if(vel>0){

        if(vel>80){

            resp.innerHTML='<Strong>Multado!! </Strong> Veículo excedo o limite de veloscidade'

        }else{

            resp.innerHTML='Prossiga'

        }

    }else{

        alert('Adicione um valor válido')

    }

}

**Condição Aninhada:** se senão (if else), uma outra opção caso o primeiro comando não for executado e caso não desejamos encerrar o programa ainda.

**Exemplo em um arquivo js:**

var idade=18;

if(idade<16){

    console.log('não vota')

}else if(idade>=16 && idade<18){

    console.log('voto opcional')

}else{

    console.log('voto obrigatório')

}

**Aplicação em um arquivo html:**

<body>

    <h1>Analise de Voto</h1>

    Sua idade: <input type="number" name="idade" id="idade">

    <input type="button" value="verificar" onclick="ver()">

    <div id="resposta"></div>

    <script>

        function ver(){

            var valor=window.document.getElementById('idade')

            var resp=window.document.getElementById('resposta')

            var id= Number(valor.value)

            //condição se, senão se, senão

            if(id<16){

               resp.innerHTML='Não vota'

            }else if(id>=16 && id<18){

                resp.innerHTML='Voto opcional'

            }else{

                resp.innerHTML='Voto obrigatório'

            }

        }

    </script>

</body>

</html>

**Switch:**

Escolha uma das opções e um comando será executado de acordo com a opção escolhida.

**OBS:** a biblioteca new Date pode criar uma variável com acesso a todos os dados da dia atual, qual dia, dia da semana, ano atual e até as horas e minutos atuais.

**Exemplo em um arquivo js:**

var agora= new Date()

var dia=agora.getDay()

console.log(dia+'° dia da semana')

switch(dia){

    case 0:

        console.log('Domingo')

        break

    case 1:

        console.log('Segunda')

        break

    case 2:

        console.log('Terça')

        break

    case 3:

        console.log('Quarta')

        break

    case 4:

        console.log('Quinta')

        break

    case 5:

        console.log('Sexta')

        break

    case 6:

        console.log('Sabado')

        break

    default:

        console.log('Dia inválido')

        break

}

Faremos aplicações de switch em html mais à frente

**Estruturas de Repetição**

**While**

De maneira resumida, um comando será excetuado enquanto uma condição ainda não for satisfeita

var contador=1

while(contador<=10){

    console.log('contando: '+contador)

    contador++

}

**Do While**

Funciona de maneira parecida com while, porem ao contrário do While dessa vez a condição é analisada depois e não antes da execução:

var contador=1

do{

    console.log('contando: '+contador)

    contador++

}while(contador<10)

**For**

Trabalha com um índice onde o comando será repetido quantas vezes forem determinadas pelo índice, onde criamos ou adicionamos uma variável, colocamos o var do inicio, fim, incremente ou decremento da variável índice durante a execução do comando, para melhor compreensão vamos para o script:

**Exemplo com incremento do índice:**

for(var i=1;i<=5;i++){

    console.log('posição: '+i)

}

**Exemplo com decremento do índice:**

for(var i=5;i>=1;i--){

    console.log('posição: '+i)

}

**Outros Tipos de Funções:**

**Construtor:** função que se autoconstrói com seus parâmetros, usando **this.**

function pessoa(nome,idade){

    this.nome=nome;

    this.idade=idade;

}

const p1=new pessoa("Joaquim",20)

console.log(p1)

**Arrow Function:** uma notação diferente e mais simples para funções

const soma=(v1,v2)=>{

    let res=v1+v2

    return res

}

console.log(soma(10,5))

const nome=n=>n

console.log(nome("Joaquim"))

const add=n=>n+10

console.log(add(10))

**Funções anônimas:** uma função rápida de se declarar e de se executar

//função construtora anônima

const x=new Function("x1","x2","x3","return x1+x2+x3")

console.log(x(7,2,5))

**Função geradora:** denotada por um asterisco \* uma função geradora pode gerar um valor diferente a cada vez que for chamada

**Exemplo:**

function\* cores(){

    yield 'Vermelho'

    yield 'Branco'

    yield 'Preto'

}

const itc=cores()

console.log(itc.next().value)

console.log(itc.next().value)

console.log(itc.next().value)

**Ex2:**

function\* contador(){

    let x=0

    while(true){

        yield x++

    }

}

const num=contador()

console.log(num.next().value)

console.log(num.next().value)

**Parâmetro Rest:** denotado por três pontos (...) a função rest criar um vetor de parâmetros, assim podemos adicionar quantos parâmetros precisarmos em uma função.

function soma(...valores){

    let resultado=0;

    for(let v of valores){

        resultado+=v

    }

    return resultado

}

console.log(soma(1,2,4,5,7))

**Vetor:**

Variável heterogenia que recebe vários valores do mesmo tipo, e armazena cada um desses valores de acordo com sua posição (índice) que se iniciar em 0.

Para criar um vetor usamos o comando *let* e não *array* como na maioria das outras linguagens de programação:

let vetor=[4,1,3,5,2]

//remover último elemento do array

vetor.pop()

//adicionar elemento

vetor.push(8)

//ordenar elementos

vetor.sort()

//buscar posição de um elemento

var posicao=vetor.indexOf(2)

console.log('o valor está na posição: '+posicao)

console.log('o vetor tem '+vetor.length+' posições')

//exibir elementos do vetor

for(var i=0;i<vetor.length;i++){

    console.log(`posição ${i}: ${vetor[i]}`)

}

**Aplicação de vetor+estruturas de repetições:**

**Exemplo:** registrar pessoas em um vetor e exibir depois

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <header>

        <h1>Registrando Pessoas: </h1>

    </header>

    <main>

        <label for="idnome">Nome: </label>

        <input type="text" name="nome" id="idnome">

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

        <input type="button" value="Enviar" onclick="enviar()">

        <input type="button" value="Exibir" onclick="exibir()">

        <div id="msg"></div>

    </main>

</body>

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

</html>

**Arquivo.js:**

var pessoas=[]

var mensagem=document.getElementById('msg')

function enviar(){

    var nome=document.getElementById('idnome')

    if(nome.value == ''){

        alert('Digite algo')

    }else{

        pessoas.push(nome.value)

        nome.value=''

        alert('Pessoa Registrada')

    }

}

function exibir(){

    mensagem.innerHTML='Pessoas Registradas: <br> '

    for(var i=0;i<pessoas.length;i++){

        mensagem.innerHTML+=`

        ${pessoas[i]} <br>`

    }

}

**Filtro em vetores:** para filtrar elementos de um vetor de acordo com as características desses elementos

Exemplo: neste exemplo vamos filtrar apenas maiores de 18 dentro de uma array, para essa filtragem usando ***arrow function***

let idades=[12,21,17,38,42,7]

let maior=idades.filter((valor)=>{if(valor>=18) return valor})

console.log(idades)

console.log(maior)

**Flat:** unir elementos do vetor

const vetor=[[1,2],[3,4],[5,6]]

const novo=vetor.flat()

console.log(vetor)

console.log(novo)

console.log(vetor.toString())

**IndexOf:** posição dos elementos no vetor

const frutas=["maçã","abacaxi","kiwi","uva"]

//posição de elemento no vetor

let posicao=frutas.indexOf("kiwi")+1

console.log(frutas.toString())

console.log("Posição do kiwi no vetor: "+posicao)

//verificar se elemento está no vetor

console.log(frutas.includes("maçã"))

**Fine:** procura um elemento correspondente a função e retorna um primeiro elemento encontrado

const numeros=[5,7,13,21,26];

function analise(valor) {

    return valor > 18;

}

//find: procura um elemento correspondente a função

let primeiro=numeros.find(analise);

console.log(primeiro)

**map em vetores:** executa uma função usando o valor dos elementos do vetor

const numbers=[21,7,36,75]

function dobro(valor){

    return valor\*2

}

console.log("valores: "+numbers)

//map

let numeros=numbers.map(dobro)

console.log("dobro: "+numeros)

//flatMap

let num=numbers.flatMap((x)=>x\*3);

console.log("triplo: "+num)

**Reduce:** reduz os elementos do array a um único valor

const numeros= [45, 21, 13, 7, 25];

function calcular(total, valor) {

  return total + valor;

}

let soma = numeros.reduce(calcular);

console.log(soma)

**Every:** Analisa se todos os elementos do array passam no teste, responde em false ou true

const numeros = [21, 13, 10, 16, 9];

function analise(valor){

    return valor > 18;

}

let maior = numeros.every(analise);

console.log(maior)

**Objetos em JavaScript**

De acordo com o dicionário objeto é algo material percebido pelos sentidos, na programação o desenvolvedor pode criar um objeto que possui alguma finalidade dentro do programa, o objeto possui características/propriedades e funções.

**Exemplo:** criaremos o objeto pessoa, com suas características e a função engordar

let pessoa={nome:"Joaquim", idade:20,peso:78,

    //função no objeto

    engordar(p=0){

        console.log(`${this.nome} engordou ${p} kilos`)

        this.peso+=p

    }

}

console.log(pessoa)

pessoa.engordar(3)

console.log(pessoa)

**Outra notação:**

const pessoa= new Object()

pessoa.nome="Joaquim"

pessoa.idade=20

pessoa.peso=78

console.log(pessoa)

**Função do tipo objeto:**

function pessoa(nome,idade,peso){

    this.nome=nome;

    this.idade=idade;

    this.peso=peso;

}

const eu= new pessoa('Joaquim',20,78)

console.log(eu)

console.log(`${eu.nome} tem ${eu.idade} anos`)

**Aplicação de objeto em um site:**

Exemplo prático usando objetos para cadastrar dados de um usuário em html+css+js:

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

    <header>

        <h1>Cadastrando Pessoa</h1>

    </header>

    <main>

        <label for="idnome">Nome: </label>

        <input type="text" name="nome" id="idnome">

        <br><label for="idtel">Telefone: </label>

        <input type="text" name="telefone" id="idtel">

        <br><label for="iddata">Nascimento</label>

        <input type="date" name="data" id="iddata">

        <br><input type="button" value="Cadastrar" onclick="enviar()">

        <br><input type="button" value="exibir"

        onclick="exibir()">

        <div id="msg"></div>

    </main>

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

</body>

</html>

**Style.css:**

@charset 'UTF-8';

input{

    margin-top: 10px;

}

input[type=button]{

    width: 80px;

    height: 30px;

    margin-top: 25px;

}

#idnome{

    margin-left: 20px;

}

#idtel{

    margin-left: 5px;

}

#iddata{

    margin-left: 10px;

}

#msg{

    margin-top: 30px;

}

**Arquivo.js:**

var pessoa={nome:'' ,telefone:'' , nascimento:''}

var msg=document.getElementById('msg')

var Nome=document.getElementById('idnome')

function enviar(){

    var Nome=document.getElementById('idnome')

    var tel=document.getElementById('idtel')

    var Data=document.getElementById('iddata')

    pessoa.nome=Nome.value

    pessoa.telefone=tel.value

    pessoa.nascimento=Data.value

    alert("pessoa cadastrada")

}

function exibir(){

    msg.innerHTML=`Nome: ${pessoa.nome} <br>  idade: ${pessoa.telefone}<br>

    Nascimento: ${pessoa.nascimento}`

}

**Classes em JavaScript:**

As classes são uma forma de criarmos objetos e organizar o código de maneira mais estruturada, otimizada e reutilizável. As classes fornecem uma sintaxe orientada a objetos para criarmos objetos de forma mais prática.

**Construtor:** constrói o próprio objeto de acordo com os parâmetros adicionados

class pessoa{

    constructor(nome,idade){

        this.nome=nome;

        this.idade=idade;

    }

}

const p1=new pessoa("Joaquim",20);

console.log(`${p1.nome} tem ${p1.idade} anos`)

**Getter e Setter:** na construção de uma variável get retorna um valor e set obtém um valor.

class carro {

    constructor(marca) {

      this.marca = marca;

    }

    get nome() {

      return this.nome;

    }

    set nome(x) {

      this.nome = x;

    }

}

const gol = new carro("gol");

console.log(gol)

**Herança:** quando uma classe herda atributos e métodos de outra classe, para isso adicionaremos *extends* nas classes que queremos que herdem de outras classes.

class carro{

    constructor(marca){

        this.marca=marca;

    }

}

//extends: a classe modelo herda os atributos e funções da classe carro

class modelo extends carro{

    constructor(marca,cor,ano){

        //super para chamar o construtor marca

        super(marca)

        this.modelo=cor;

        this.ano=ano;

    }

}

var audi=new modelo("Audi","w3","Branco");

console.log(audi)

**Expressões Regulares:**

Forma de coletar informações em Strings, encontrando padrões em textos.

**Modificadores:**

**I ->** não distingue letras maiúsculas e minúsculas

**g->** global, ou seja todos os elementos de mesma características

**d->**primeira e última correspondência

**m->**Executa correspondência multilinha

Para usarmos expressões regulares basta colocarmos a String entre barras deitadas //

**Exemplo:**

let nome="Bom dia"

//posição em que se encontra

console.log(nome.search(/dia/i))

//match: posição em que se encontra pela 1° vez

console.log(nome.match("o"))

//todas vezes encontradas

console.log(nome.match(/o/g))

//replace=troca frase

console.log(nome.replace("dia","tarde"))

//trocando todos "o" por "x"

console.log(nome.replace(/o/ig,"x"))

**Meta-Caracteres:** usados para encontrar uma série de caracteres

**Exemplos:**

[a-b]->Caracteres de “a” até “b”

[0-9]->Caracteres de 0 até 9

\d->apenas dígitos

\s->apenas espaços

\uxxx->números hexadecimais

let nome="Joaquim Angelo 2004"

//todos os elementos de "a" até "b" e de 0 a 9

console.log(nome.match(/[a-b|0-9]/ig))

//apenas números

console.log(nome.match(/\d/ig))

//Espaços

console.log(nome.match(/\s/ig))

**Quantificadores:** identificar padrões de dígitos

let numeros="10,100,1000,25100"

console.log(numeros.match(/10/ig))

console.log(numeros.match(/10+/ig))

console.log(numeros.match(/10\*/ig))

**Outros:**

{n} -> n elementos

\- ou \. ->traço ou ponto

^ -> começa com

$ -> termina com

Identificando padrões numéricos para:

**Telefone:** iniciando com 9, 4 dígitos, traço, mais 4 digitos.

let telefone="meu telefone é 99942-0071"

console.log(telefone.match(/\9\d{4}\-\d{4}/ig))

console.log(telefone.match(/\9[0-9]{4}\-[0-9]{4}/ig))

CPF: 3 digitos, ponto, 3digitos, ponto, 3digitos, traço, 2 digitos

let cpf="081.094.176-25"

console.log(cpf.match(/^\d{3}\.\d{3}\.\d{3}\-\d{2}$/ig))

Validador de número com expressões regulares:

let cpf="188.423.503-15"

let validador=cpf.match(/^\d{3}\.\d{3}\.\d{3}\-\d{2}$/ig)

if(validador==null){

    console.log("Cpf inválido")

}else{

    console.log("cpf válido")

}

**Regex Aplicado:** validador de cpf e telefone aplicado com html e JavaScript

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Validador</title>

    <link rel="stylesheet" href="style.css">

</head>

<body>

        <h1>Analise</h1>

        <p>Escreva o CPF da forma: xxx.xxx.xxx-xx</p>

        <p>Escreva o ddd primeiro e depois Telefone da forma:9xxxx-xxxx</p>

        <label for="idcpf">CPF</label>

        <input type="text" name="cpf" id="idcpf"

        placeholder="xxx.xxx.xxx-xx">

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

        <br><label for="idtel">Telefone</label>

        <input type="number" name="ddd" id="idddd" placeholder="dd" maxlength="2" size="2">

        <input type="text" name="tel" id="idtel"

        placeholder="9xxxx-xxxx">

        <br><input type="button" value="Enviar" onclick="enviar()">

        <div id="msg\_1"></div>

        <div id="msg\_2"></div>

        <div id="msg\_3"></div>

</body>

</html>

**Arquivo.js:**

function enviar(){

    let cpf=document.getElementById("idcpf")

    let tel=document.getElementById("idtel")

    let ddd=document.getElementById("idddd")

    let msg1=document.getElementById("msg\_1")

    let msg2=document.getElementById("msg\_2")

    let msg3=document.getElementById("msg\_3")

    //Expressões Regulares de validação

    let validador\_cpf=cpf.value.match(/^\d{3}\.\d{3}\.\d{3}\-\d{2}$/ig)

    let validador\_ddd=ddd.value.match('^[0-9]{2}$')

    let validador\_tel=tel.value.match(/\9[0-9]{4}\-[0-9]{4}/ig)

    if(validador\_cpf==null){

        msg1.innerHTML="Cpf inválido"

    }else{

        msg1.innerHTML="Cpf válido"

    }

    if(validador\_ddd==null){

        msg2.innerHTML="ddd inválido"

    }

    if(validador\_tel==null){

        msg3.innerHTML="Telefone inválido"

    }else{

        msg3.innerHTML="Telefone válido"

    }

}

**Outras Bibliotecas**

**Data**

Obtendo datas usando o objeto ***Date.***

//criando variável da data atual

const data = new Date();

//obter apenas o ano

let ano= data.getFullYear()

//número do mês

let mes= data.getMonth()

//minutos

let hora=data.getMinutes()

console.log(data)

console.log(ano)

console.log(mes)

console.log(hora)

Escrever dia da Semana:

const dia=["Segunda","Terça","Quarta","Quinta","Sexta","Sabado","Domingo"];

console.log(dia[data.getDay()-1])

**Math**

Para usar operações matemáticas.

//arredondar número

console.log(Math.round(4.6))

//arredondar pra cima

console.log(Math.ceil(4.6))

//arredondar pra baixo

console.log(Math.floor(4.6))

//potênciação: (base,expoente)

console.log(Math.pow(3,2))

//raiz quadrada

console.log(Math.sqrt(64))

//Logaritmo(log de 10 na base 10)

console.log(Math.log10(10))

**Coleções:**

Espécie de variável homogênea que adiciona valores de qualquer tipo.

**Set:** coleção onde cada elemento possui apenas um valor

const variaveis= new Set();

variaveis.add("a");

variaveis.add("b");

variaveis.add("c");

console.log(variaveis)

//variaveis contem a variável b?

console.log(variaveis.has("b"))

const letters = new Set(["a","b","c"]);

//forEarch(): invoca uma função para cada elemento do set

let texto = "";

variaveis.forEach (function(valor) {

  texto += valor;

})

console.log(texto)

**Map:** cada elemento possui um valor e uma chave que podem ser objetos de qualquer tipo.

const frutas= new Map();

//frutas e quantidade

frutas.set("maça",50)

frutas.set("Banana",75)

frutas.set("uva",75)

console.log(frutas)

//get: obter dados do elemento

console.log(frutas.get("uva"))

//metodos map

frutas.delete("maça")

console.log("Frutas possui kiwi? "+frutas.has("kiwi"))

console.log(frutas)

**TextContent:** atualiza o texto do html em tempo real

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>textContent</title>

    <style>

        input[type=text]{

            width:400px ;

            height: 50px;

        }

    </style>

</head>

<body>

    <input type="text" name="txt" id="idtxt">

    <p id="msg"></p>

    <script>

        let txt=document.getElementById("idtxt")

        let msg=document.getElementById("msg")

        //keyup: acionado na digitação de uma tecla

        txt.addEventListener("keyup",function(){

            //textContent: escreve automaticamente

            msg.textContent="Texto Digitado: "+txt.value;

        })

    </script>

</body>

</html>

**Introdução a API’s (Application Programming Interface)**

Uma API web é uma interface de programação de aplicativos para a web, uma APi de um navegador pode estender a funcionalidade de um navegador web, e a api de um servidor estende a funcionalidade de um servidor web

As API’s são usadas para integrar novas aplicações com sistemas de softwares já existente, logo podemos usar outros softwares em nosso site ou aplicação.

Por exemplo, podemos usar uma api a localização do google para ver a nossa latitude e longitude no momento:

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Localização</title>

</head>

<body>

    <input type="button" value="Localização" onclick="local()">

    <div id="tela"></div>

    <script>

        const tela=document.getElementById('tela')

        function local(){

            if(navigator.geolocation){

                navigator.geolocation.getCurrentPosition(mostrar\_posicao)

            }else{

                tela.innerHTML='Geolocation não disponível no navegador'

            }

        }

        function mostrar\_posicao(position){

            tela.innerHTML="<br>Latitude: "+position.coords.latitude +

            "<br>Longitude: "+position.coords.longitude;

        }

    </script>

</body>

</html>

Todos os navegadores possuem um conjunto de api´s web integradas para ajudar no acesso de dados, no exemplo acima vemos a geolocation (geolocalização).

**Exemplo 2:** checkar validação(checkValidity)

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Validador</title>

</head>

<body>

    <input id="idnum" type="number" min="100" max="300" required>

    <button onclick="validador()">OK</button>

    <p id="tela"></p>

    <script>

        const tela=document.getElementById('tela')

        function validador() {

          const num = document.getElementById("idnum");

          //se o numero não for válido

          if (!num.checkValidity()) {

            tela.innerHTML = num.validationMessage;

          }else{

            tela.innerHTML="Número válido";

          }

        }

    </script>

</body>

</html>

**LocalStorage:** armazena e recupera dados no navegador

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <p id="tela"></p>

    <script>

        let tela=document.getElementById('tela')

        localStorage.setItem("nome","Joaquim");

        tela.innerHTML = localStorage.getItem("nome");

    </script>

</body>

</html>

**Fetch API:**

A interface Fetch Api permite que o navegadir da web faça solicitações http para serviços web

Por exemplo, existe na internet um site chamado Dummy Json, onde lá podemos ver alguns exemplos, no caso vamos enviar uma solicitação http para esse site e veremos os valores que serão retornados:

**Exemplos encontrados em:** https://dummyjson.com/docs/products#products-single

Ex1.js:

//solicitação para o site dummyjson

fetch('https://dummyjson.com/products')

//buscando dados dos produtos dummy

.then(res => res.json())

//escrevendo esses dados

.then(console.log);

Ex2.js:

//obtendo um único produto

fetch('https://dummyjson.com/products/1')

.then(res => res.json())

.then(console.log);

Ex3.js:

//procurando produtos

fetch('https://dummyjson.com/products/search?q=phone')

.then(res => res.json())

.then(console.log);

Ex4.js:

//não temos permição para alterar o sistema, no exemplo criamos um novo produto apenas no nosso sistema

fetch('https://dummyjson.com/products/add', {

    method: 'POST',

    headers: { 'Content-Type': 'application/json' },

    body: JSON.stringify({

      title: 'BMW Pencil',

    })

  })

.then(res => res.json())

.then(console.log);

Ex5:

//Deletando produto apenas no nosso sistema

fetch('https://dummyjson.com/products/1', {

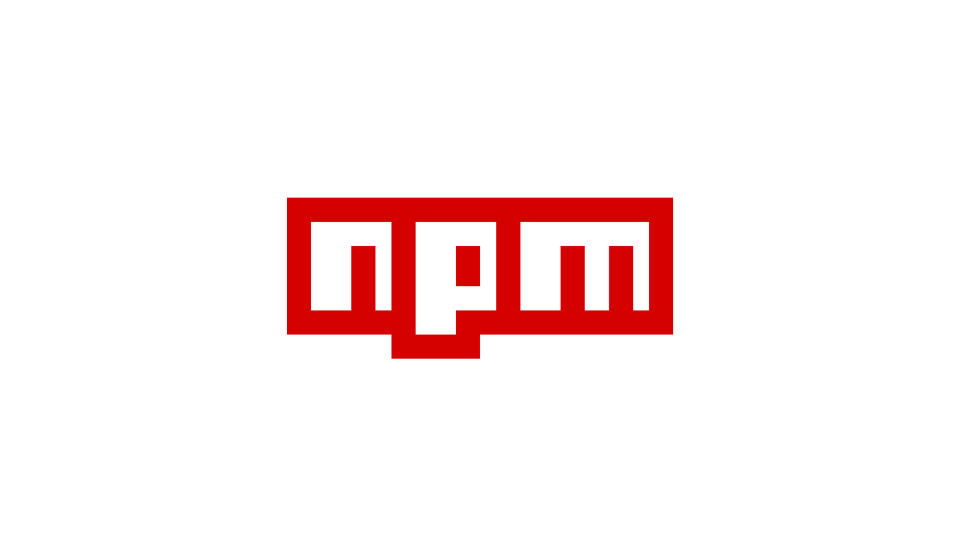
    method: 'DELETE',

  })

.then(res => res.json())

.then(console.log);

Até o momento vimos apenas o básico do básico de Api´s, para nos aprofundarmos no assunto devemos ver ajax, json, restful Api e em alguns casos um pouco de JQuery, programadores que querem trabalhar na área de Api devem se aprofundar um pouco mais nesses assuntos.



**Node Pack Manager:**

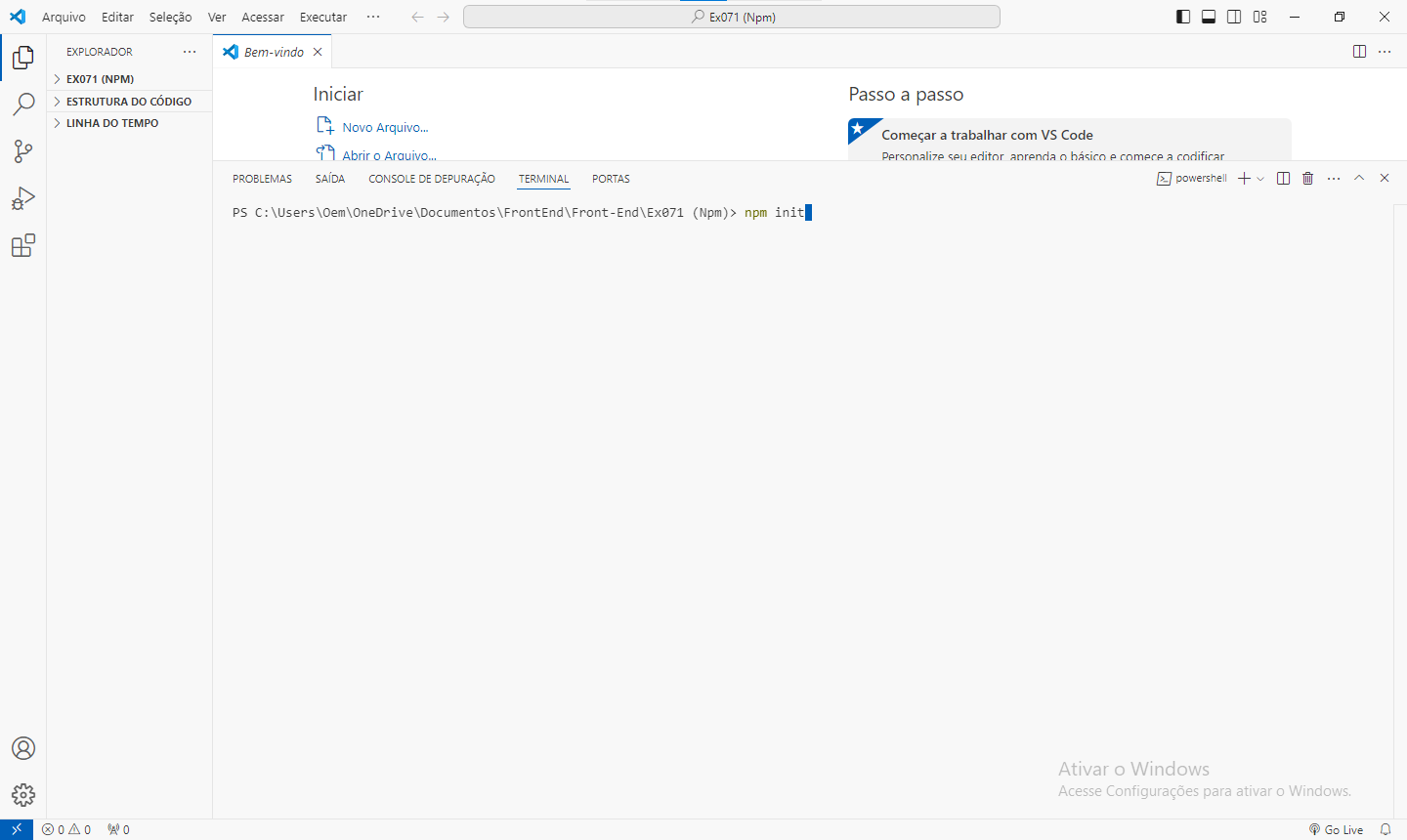
npm é um gerenciador de pacotes para o JavaScript mantido pela ***npm Inc***. npm é o gerenciador de pacotes padrão para o ambiente de tempo de execução JavaScript Node.js.

**OBS:** O Npm vem junto com o node js no download, potante se voçe possui Node Js instalado no computador não precisa baixar o npm

**Criando um pacote:**

Para usar o npm inicialmente vamos criar uma pasta e abrir ela no vscode

Para criar um pacote json inicialmente vamos selecionar a pasta ir no terminal do vscode e digitar npm init para iniciar a construção do pacote:



**Npm init:** inicializa o npm nessa pasta e faz uma série de perguntas no terminal

Pergunta 1: nome do pacote a ser criado (sem espaços)

Pergunta 2: versão do pacote

Pergunta 3: descrição (podendo ficar em branco)

Pergunta 4: arquivo de entrada (geralmente index.js)

Pergunta 2: Comando de teste (podendo ficar em branco)

Pergunta 5: repositório git (podendo ficar em branco)

Pergunta 6: Palavra Chave (podendo ficar em branco)

Pergunta 7: autor

Pergunta 8: tipo de licença (podendo deixar em ISC)

E depois basta apenas confirmar para criar um arquivo package.json

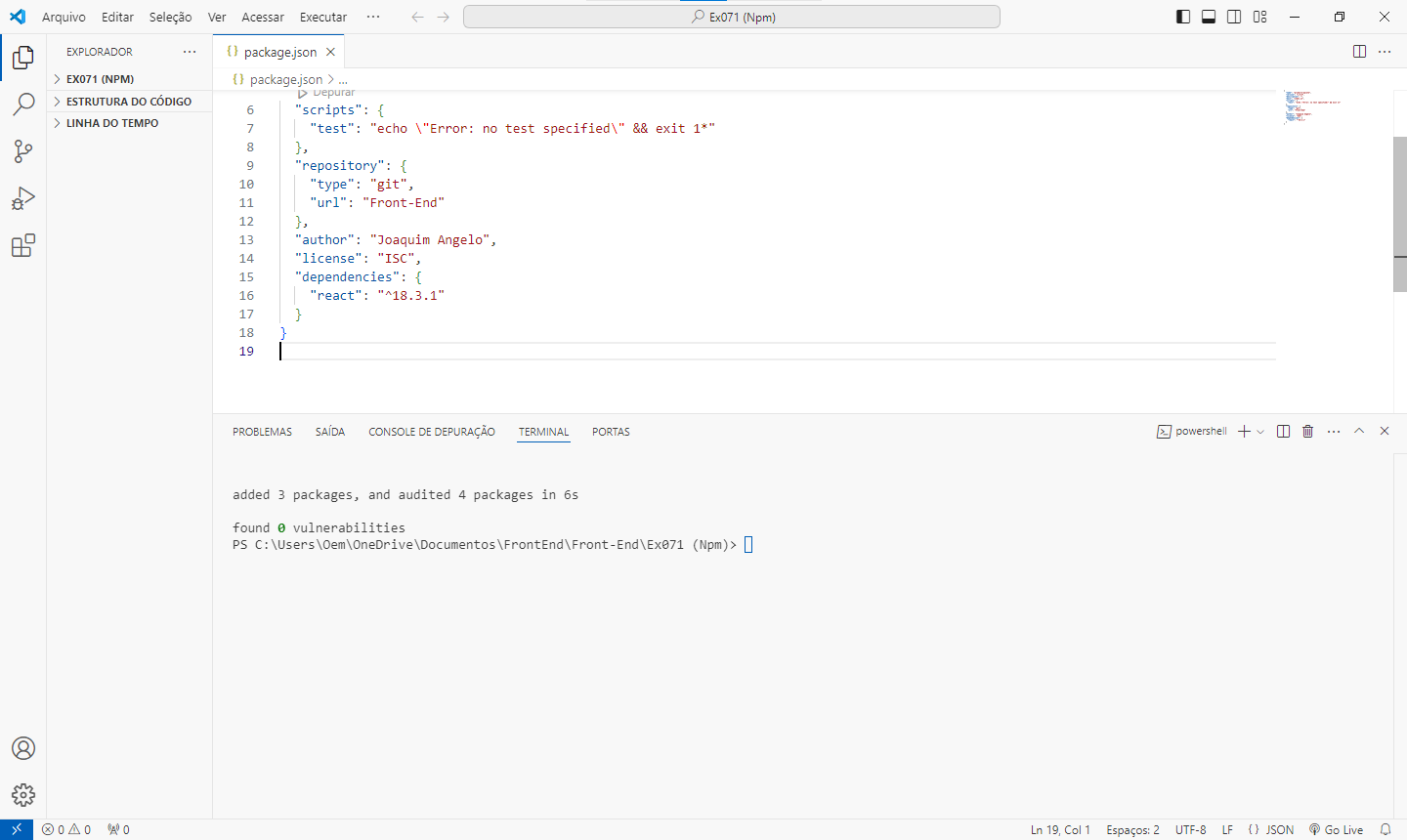
Para criar um arquivo sem precisar responder as perguntas bastas ir no terminal e digitar o comando: ***npm init –y*** criando assim o package.json e respondendo sim para todas as perguntas.

**Package.json:** arquivo que discreve um diretório.

Json: objeto no javascript, logo esse arquivo é apenas um grande objeto do javaScript

**Instalando Pacotes:**

A principal função do npm é instalar pacotes no javascript, para fazer isso basta ir no terminal e digitar **npm install *nome do pacote***, por exemplo: na imagem abaixo vamos instalar react em nosso computador.

Para isso bastar ir no terminal e digitar o comando **npm install react**, esse será o resultado:

Logo react já está instalado na maquina.

Após isso será criado duas pastas uma chamada **node-modules** onde estão instalados os pacotes, a outra chamada de **package-lock** que guarda as informações dos pacotes baixados e não devem ser alterados pelo desenvolvedor

No caso **node-modules** foi onde instalamos o **react** e os outros pacotes que aparecem são arquivos que o react precisa para ser instalado.

**OBS:** A pasta **node-modules** geralmente é uma pasta muito pesada, apenas no react a pasta chega a pesar mais de 40mb na memória. Por conta disso ficaria muito pesado enviar todos esses arquivos para o GitHub então vamos criar um arquivo chamado **.gitignore**, dentro desse arquivo vamos digitar o nome do arquivo que queremos ignorar e não enviar para o GitHub, então vamos digitar node\_modules, assim quando dermos comit no github enviaremos apenas scripts.

**OBS 2:** quando baixamos/clonamos um repositório no github e ele não vem com os arquivos do node-modules o programa não vai funcionar, para resolver isso basta apenas baixar criar o node-modules indo no terminal e digitando apenas o comando ***npm install*** que vai instalar automaticamente todos os arquivos necessários do projeto. Caso tenha outro tipo de arquivo devemos digitar outro comando dependendo do arquivo, por exemplo se tivermos um arquivo **yarn.lock** devemos digitar ***yarn*** no terminal.

**Removendo Pacotes:**

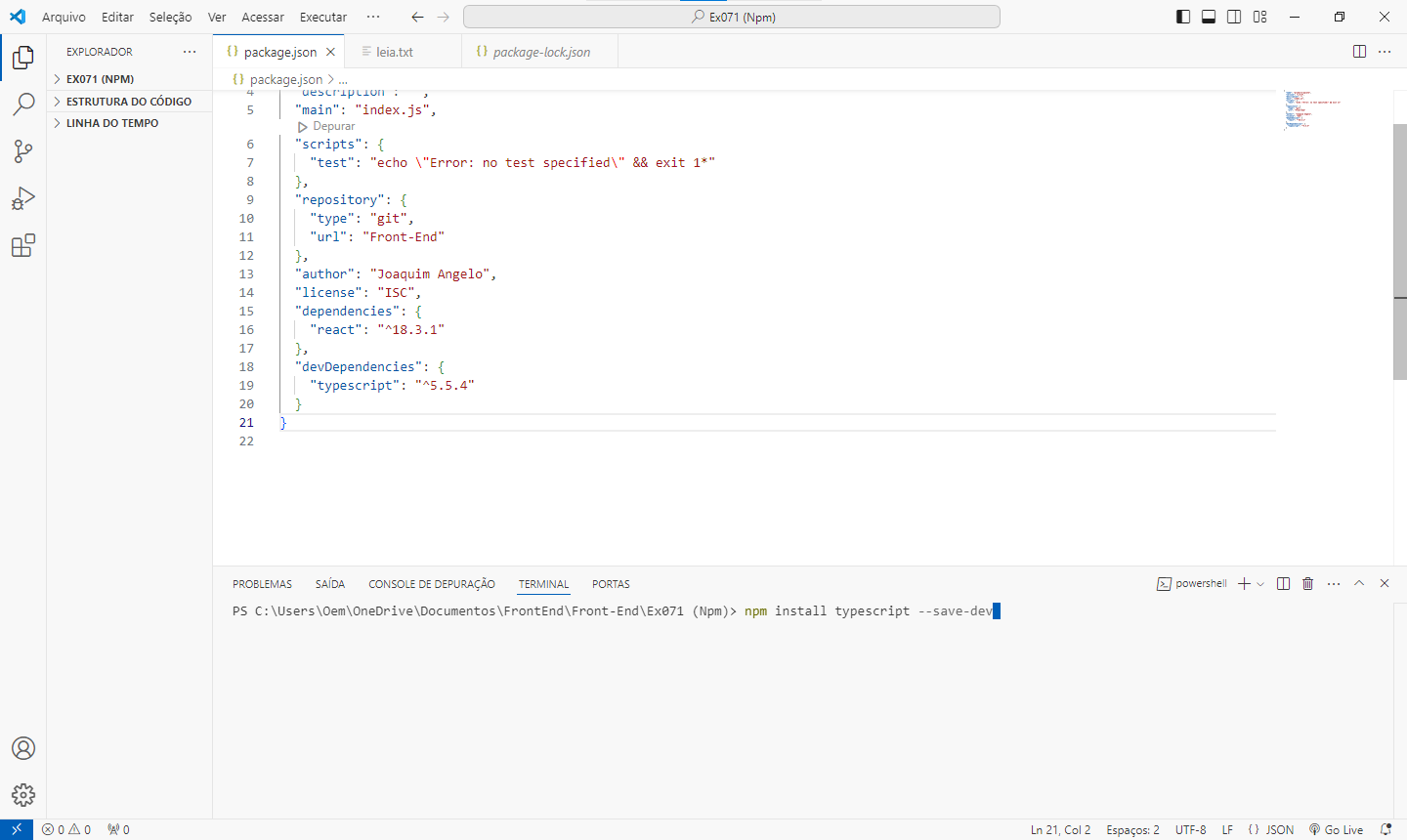
Para remover um pacote basta ir no terminal e digitar o comando **npm remove *nome do pacote***, usando o exemplo anterior podemos remover o react digitando **npm remove react**, quando a remoção for concluída os arquivos que foram baixandos em node-modules serão excluídos também.

**Dev dependências:**

Dependências de desenvolvimento são arquivos necessário no desenvolvimento que são visíveis e acessados apenas pelos desenvolvedores do projeto

Para adicionar um devDependencia inicialmente devemos escrever devDependencies no package.json e depois ir no terminal e digitar o comando **npm install *nome da dependência***.

No exemplo abaixo será instalado a dependência mais popular entre programadores front-End, o **typeScript –save-dev**:





**Framework Sass**

**Introdução:**

**Sass: Sass** stands for **S**yntactically **A**wesome **S**tyle**s**heet.

O Sass serve para escrever em css de uma forma mais rápida e eficiente, além de permitir criarmos variáveis e funções também permitindo que o código seja reutilizado.

Podemos ver mais sobre o Sass no seu site oficial: **https://sass-lang.com/**

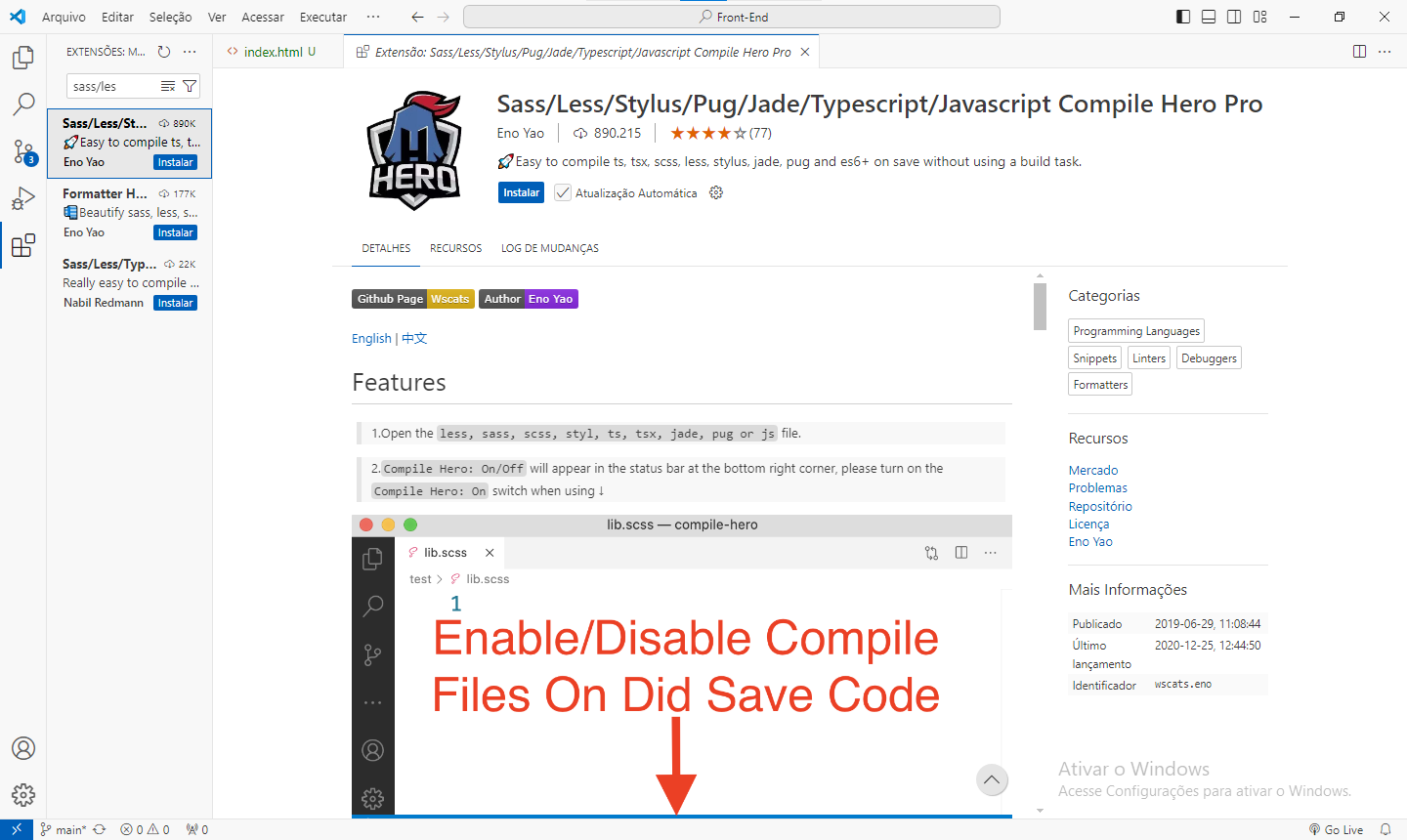
**Usando Sass:**

A melhor maneira de entender como usar sass é assistindo o vídeo: [***https://www.youtube.com/watch?v=Bi5xQHT2mLU&t=221s***](https://www.youtube.com/watch?v=Bi5xQHT2mLU&t=221s)

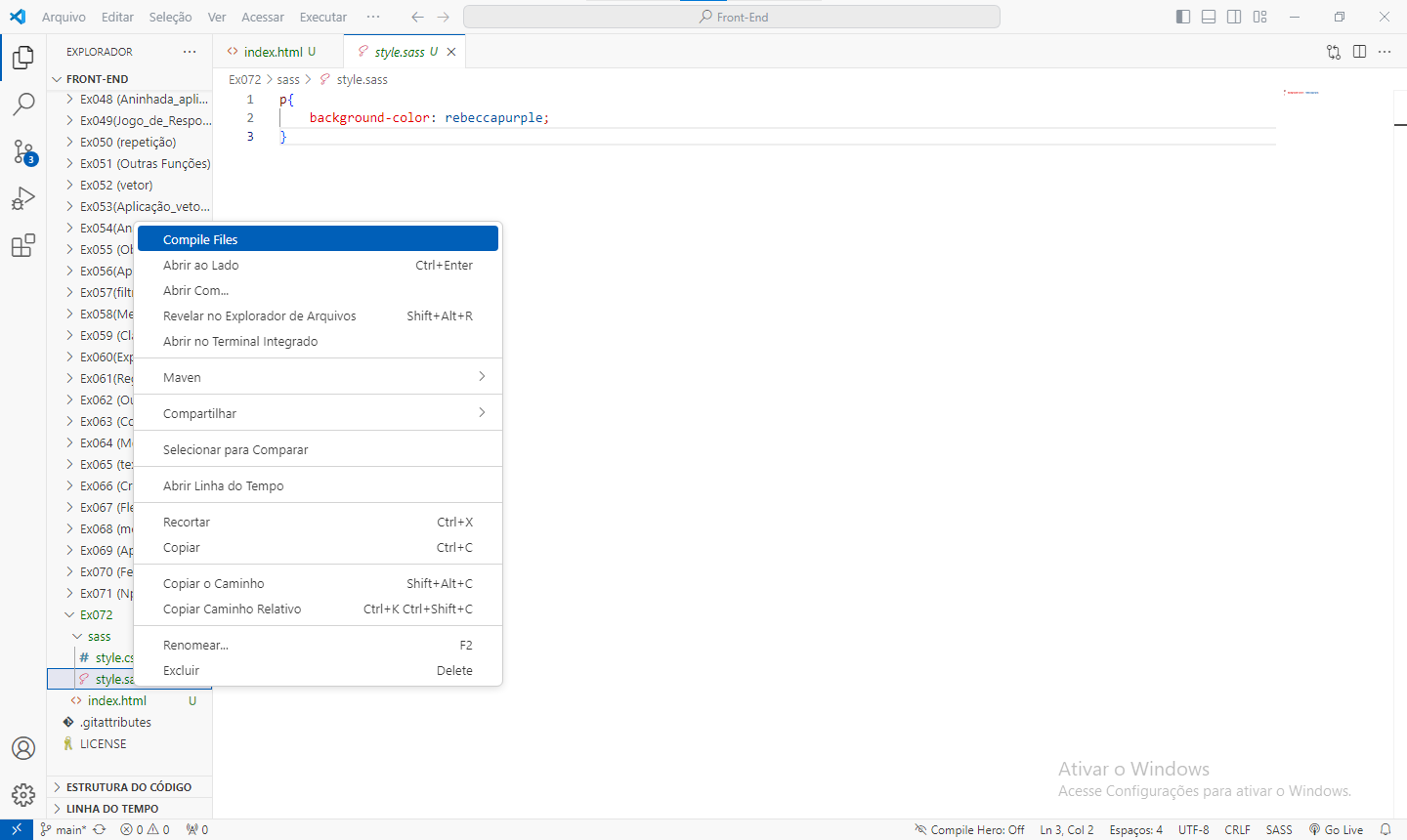
Caso não seja possível assistir o vídeo explicarei passo a passo a seguir:

Para habilitar o sass no vscode precisamos baixar duas extensões: ***sas*s** e o ***compilador sass.***

**Melhor compilador:**



Após baixar ambos vamos criar um arquivo sass, para isso precisamos apenas usar **.sass** no final do nome do arquivo.

 Vamos inicialmente criar a pasta do nosso exercício, no meu caso criei a pasta **Ex072**, dentro desta pasta criei um **index.html** e outra pasta chamado **sass**, nesta pasta sass criei um arquivo chamado **estilo.sass**, nesse arquivo escrevi um script css normal e toda vez que atualizo o **estilo.sass** devemos compilar-lo:

Os navegadores não entendem códigos escritos em Sass, então quando um arquivo sass é compilado pela primeira vez cria-se uma pasta chamada **dist** com um arquivo css dentro, este css possui o mesmo script escrito no sass e se atualiza toda vez que o arquivo sass é compilado.

Para usar o estilo no index.html basta criar um link css com arquivo css que está dentro da pasta dist

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Sass</title>

    <link rel="stylesheet" href="sass/dist/estilo.css">

</head>

<body>

    <h1>Testando Sass: </h1>

    <div></div>

</body>

</html>

**estilo.sass:**

h1

    color: #f12020

div

    height: 250px

    width: 300px

    background-color: black

**estilo.css:**

/\*este arquivo deve ser criado e escrito automaticamente pelo sass\*/

h1 {

  color: #f12020;

}

div {

  height: 250px;

  width: 300px;

  background-color: black;

}

**Sass e Scss:**

Existem dois tipos de arquivos Sass: **.sass** e **.scss**, O scss é semelhante ao css, porém possui funções que não são suportadas no css, já o sass supota os recursos do css, porém com uma sintaxe um pouco diferente.

**OBS:** arquivos. sass não precisam de chaves {} nos seletores, já nos arquivo .scss as chaves são obrigatórias assim como no css

**Variéveis em Sass:** para criarmos variáveis criaremos arquivo do tipo .scss .

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="Sass/dist/estilo.css">

</head>

<body>

    <p>Variável em Sass</p>

</body>

</html>

**estilos.scss:**

$fonte: Helvetica, sans-serif;

$cor: red;

$tamanho\_fonte: 16px;

p{

    font-family: $fonte;

    color: $cor;

    font-size: $tamanho\_fonte;

}

**estilos.css:** criado automaticamente pelo Sass

p {

  font-family: Helvetica, sans-serif;

  color: red;

  font-size: 16px;

}

**Nasting:** Sass também permite aninhar seletores css da mesma forma que o html

**Index.html:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Nesting</title>

    <link rel="stylesheet" href="Sass/dist/estilo.css">

</head>

<body>

    <nav>

        <ul>

            <li><a href="#">Inicio</a></li>

            <li><a href="#">Meio</a></li>

            <li><a href="#">Fim</a></li>

        </ul>

    </nav>

</body>

</html>

**estilo.sass:**

//tipo 1: objetos html

nav {

  ul {

    margin: 0;

    padding: 0;

    list-style: none;

  }

  li {

    display: inline-block;

  }

  a {

    display: block;

    padding: 6px 12px;

    text-decoration: none;

  }

}

//tipo 2: objetos css

body{

  font: {

    family: Helvetica, sans-serif;

    size: 18px;

    weight: bold;

  }

  text: {

    align: center;

    transform: lowercase;

    overflow: hidden;}}

**estilos.css:** criado automaticamente pelo Sass

nav ul {

  margin: 0;

  padding: 0;

  list-style: none;

}

nav li {

  display: inline-block;

}

nav a {

  display: block;

  padding: 6px 12px;

  text-decoration: none;

}

body {

  font-family: Helvetica, sans-serif;

  font-size: 18px;

  font-weight: bold;

  text-align: center;

  text-transform: lowercase;

  text-overflow: hidden;

}

**Extends:** extende propriedades de outro seletor css

**Ex:**

**Index.html:** criamos duas divs uma de classe: preta e outra de classe: azul

**estilo.scss:**

.preto{

    width: 300px;

    height: 100px;

    background-color: black;

}

.azul{

    @extend .preto;

    background-color: blue;

}

**estilos.css:** criado automaticamente pelo Sass

.preto, .azul {

  width: 300px;

  height: 100px;

  background-color: black;

}

.azul {

  background-color: blue;

}

**Mixin:** cria um código css reutilizável em todo o site, para incluir o mixin em um seletor basta adicionar o comando @include

**Exemplo:**

**Index.html:**

<!DOCTYPE html>

<html lang="Pt-br">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Include</title>

    <link rel="stylesheet" href="sass/dist/estilo.css">

</head>

<body>

    <h1>titulo</h1>

    <p class="vermelho">Mensagem vermelha</p>

    <p class="azul">Mensagem azul</p>

</body>

</html>

**estilo.scss:**

//criando mixin simples

@mixin mensagem{

    color: white;

    font-size: 20px;

    border: 1px solid black;

    padding: 5px;

}

.vermelho{

    @include mensagem();

    background-color: red;

}

//mixin usando parâmetro

@mixin borda($cor,$largura){

    border: $largura solid $cor;

    padding: 5px;

}

.azul{

    @include borda(blue,2px)

}

Não há mais necessidade de escrever o código css.