Jeverson Fagundes.

**Estudo de Programação.**

***SITES PARA ACHAR ICONES.***

[Boxicons : Premium web friendly icons for free](https://boxicons.com/?query=lock)

[Cadeado Icons, Logos, Symbols – Free Download PNG, SVG (icons8.com)](https://icons8.com/icons/set/cadeado)

***HTML.***

<head>

    <!--META TAGS.-->

    <!--As metas tags são metas dados, ou seja, são informações que são lidas somente pelos navegadores e serve para algumas finalidades como por exemplo: Mecânimos de busca por palavras chaves do site, para dizer quem é o autor, o tipo de codificação de caracteres que serão utilizados, o tipo de visualização de dispositivos entres outros.-->

    <!--PRINCIPAIS META TAGS.-->

    <meta *charset*="UTF-8">

    <!--Essa serve para que o navegador consiga entender caracteres como acentos e não dar erros.-->

   <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge">

   <!--Essa serve para compatibilidade com o internet explore -->

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

    <!--Essa serve para dizer para o navegador que a tela ou a parte visivel do site vai se adaptar ao tamanho de tela do dispositivo em que for aberto. Permite que o zoom seja carregado corretemante, pois não é só colocar isso para tornar o site resposivo.-->

    <meta *name*="description" *content*="esse é um site em html">

    <!--Essa serve para dar uma descrição para o site.Pois não serve para muito já que se for muito grande a descrição não vai aparecer tudo.-->

    <meta *name*="Keywords" *content*="Site de programação em html.">

    <!--Essa serve para dar palavras chaves para o site. Temas que o site aborda, mecãnismo que permite que por exemplo o google possa ver o que o autor aborda no seu site. Pode ser para os temas mais buscados.-->

    <meta *name*="author" *content*="Jeverson">

    <!--Serve para dizer quem é o autor do site.-->

    <title>Curso de html</title>

</head>

<body>

    <!--TITULOS.-->

    <!--Quanto maior for o número do H menos "importante esse titulo vai ser" vai se tornando subtítulos, títulos secundários. Ordem de relevância. Eles serve para mostrar para o navegador a importância do texto que estamos fazendo.-->

    <h1>Titulo 1</h1>

    <h2>Titulo 2</h2>

    <h3>Titulo 3</h3>

    <h4>Titulo 4</h4>

    <h5>Titulo 5</h5>

    <h6>Titulo 6</h6>

    <!--TEXTO ALEATÓRIO.-->

    <!--Declaramos p>lorem para criar parágrafos de texto aleatórios em Latim.-->

    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Ipsam praesentium quisquam quia architecto perspiciatis veniam aliquid, distinctio vel ipsa quas magni, maiores ex non mollitia! Amet officiis cumque blanditiis illum.</p>

    <!--TAGS DE FORMATAÇÃO DE TEXTOS.-->

    <!--São tags que são importantes na hora de criar um conteúdo.-->

    <!--NEGRITO.-->

    <b> Jeremias</b>

    <!--STRONG.-->

    <!--Faz praticamente a mesma coisa do NEGRITO com a diferença de que strong tem além de deixar em negrito dar ênfase para o texto. Deixa como se essa parte do texto seja mais importante do que o resto do texto por assim dizer.-->

    <strong>Jemerson</strong>

    <!--HR.-->

    <!--O HR serve para deixar uma linha  de divisão na página transformando em sessões.-->

    <hr>

    <!--ITALICO.-->

    <i>Jeremias lenhador</i><br>

    <!--UNDERLINE.-->

    <!--Sublinha o texto.-->

    <u>Três capa deitei</u><br>

    <!--SMALL.-->

    <!--Serve para deixar o texto pequeno.-->

    <small>Jeverson</small><br>

    <!--DEL.-->

    <!--Texto deletado. Ele risca o texto do jeito de quando algo está em promoção um texto fica riscado e outro não.-->

    <del> R$ 400</del><br>

    <!--SUP.-->

    <!--Superior. Com ela podemos colocar um expoente em um número.-->

    O número é 210 <!--Se deixarmos assim isso é 210.-->

    O número é 2<sup>10</sup><!--Já assim isso é 2 elevado a 10.--><br>

    <!--SUB.-->

    <!--Faz o inverso do sup.-->

    O número é 2<sub>10</sub><br>

    <!--MARK.-->

    <!--Marcação de um texto. Destacação de texto.-->

    <mark>a73 5g</mark><br>

    <!--ELEMENTOS DE CITAÇÃO.-->

    <!--São tags usadas em textos para por exemplo falar de uma pessoa importante no texto.-->

    <!--ABBR TITLE="".-->

    <!--Essa tag serve para por exemplo caso for necessário explicar uma abreviação. Por exemplo HTML muitas pessoas não sabem o que é o HTML e ainda por cima é uma abreviação então podemos colocar assim:-->

    <p>Estamos aprendendo <abbr *title*="Hyper text markup language">HTML.</abbr></p><br>

    <!--Esse title vai explicar o que significa a abreviação.-->

    <!--ADDRESS.-->

    <!--Endereço.-->

    <address>Uruguaiana</address><br>

    <!--CITE.-->

    <!--Quer dizer que vai ser citado o titulo de uma obra, um filme etc.-->

    O filme <cite> transforme</cite> não é muito bom.<br>

    <!--Q.-->

    <!--O tag q serve para dar citações curtas sendo colocado dentro "" o que a pessoa falou.-->

    <p>Uma dos poderes mais conhecidos do Goku é o <q>Kameramerá.</q></p><br>

    <!--BLOCKQUOTE.-->

    <!--Ser para dar uma citação do próprio texto.-->

    <p>No jogo A SOBREVIVÊNCIA DE ETHAN a história poderia ser mais trabalhada.</p>

    <blockquote>

    a história poderia ser mais trabalhada.

    </blockquote><br>

    <!--BDO DIR="".-->

    <!--Serve para indicar a direção do texto. Sendo AUTO, RTL e LTR.-->

    <bdo *dir*="">Lorem ipsum dolor sit amet consectetur, adipisicing elit. Aperiam numquam fugiat vitae maiores, dolore voluptate? Id fugit asperiores nobis quibusdam, eius laboriosam nam rerum, quae voluptates sint enim natus ipsum.</bdo><br>

    <!--COMENTÁRIOS.-->

    <!--Isso é um comentário.-->

    <!--TARGET="\_BLANK"-->

    <!--Serve para quando queremos que algo seja aberto em uma nova janela.-->

    <a *href*="" *target*="\_blank">Abrir o site</a><br>

    <!--IMAGENS.-->

    <!--O ALT serve para exibir um texto opcional caso a imagem não carregue ou uma descrição da imagem.-->

    <img *src*="" *alt*="Imagem não carregada.">

    <img *src*="" *width*="300px" *height*="300px">

    <!--TABELAS.-->

    <table *width*="100%" *border*="1" *style*="text-align: center;">

        <tr><!--Significa linha.-->

            <th><!--Significa coluna.-->

                nome

            </th>

            <th>

                idade

            </th>

            <th>

                peso

            </th>

        </tr>

        <tr>

            <td><!--Significa os dados da tabela.-->

                Jeverson

            </td>

            <td>

               17 anos

            </td>

            <td>

                1 metro e 63 cm

            </td>

        </tr>

        <tr>

            <td>

                Jeremias lenhador

            </td>

            <td>

               40 anos

            </td>

            <td>

                1 metro e 67 cm

        </tr>

    </table>

    <!--LISTAS.-->

    <!--LISTA NÃO ORDENADA.-->

    <ul>

        <li><!--Significa item da lista.-->

            arroz

        </li>

        <li>

            carne

        </li>

    </ul>

    <!--LISTA ORDENADA.-->

    <!--Tem uma sequência de números. 1,2,3,4 etc.-->

    <ol>

        <li>

            arroz

        </li>

        <li>

            carne

        </li>

    </ol>

    <!--IFRAMES.-->

    <!--Os iframes possibilitam exibir uma página dentro da nossa própria página. Um site externo rodando dentro da nossa página.-->

    <iframe *src*="iframe.html" *width*="100%" *height*="600px" *title*="Ifreme"  *frameborder*="0" *style*="border: 2px solid;"></iframe><br><br>

    <!--Podemos fazer assim também.-->

    <!--O target="ifreme" apontando para o ifreme com esse nome possibilita que quando for clicado o link ele rende rize o conteúdo do arquivo IFREME.HTML dentro de um iframe dentro da minha página ao invés redirecionar o usuário para o arquivo.-->

    <a *href*="iframe.html" *target*="ifreme">Clicar aqui</a>

    <iframe *name*="ifreme" *title*="Ifreme"></iframe><br><br>

    <!--FORMULÁRIOS.-->

    <!--Também conhecidos como área de entrada de dados.-->

    <form *action*="" *method*="post">

        <input *type*="text" *name*="text" *placeholder*="digite um texto qualquer"><!--Entrada de texto. Placehoder serve para dizer o que deve ser digitado dentro do campo.-->

        <input *type*="password" *name*="senha" *id*="" *required*><!--Entrada de senhas. O REQUIRED torna o campo obrigatório.-->

        <input *type*="file" *name*="file" *id*=""><!--Entrada de arquivos. Upload de arquivos.--><br><br>

        <input *type*="email" *name*="" *id*="">

        <!--Serve para verificar se é um email válido. Se tem os requisitos mínimos para ser um e-mail.--><br><br>

        <!--LABEL.-->

        <!--O label serve para dar um nome ou descrição do campo. -->

        <label *for*="camponome">Nome:</label>

        <input *type*="text" *id*="camponome"><br><br>

        <h2> Escolha um animal</h2><br><br>

        <!--RADIO.-->

        <!--Serve para dar a possibilidade do usuário possa escolher uma opção. O nome tem que ser igual para todos os radios.-->

        <!--VALUE. É o que vai ser mandado para o formulário. O valor.-->

        <label *for*="gato">Gato</label>

        <input *type*="radio" *name*="animal" *id*="gato" *value*="gato"><br><br>

        <label *for*="cachorro">Cachorro</label>

        <input *type*="radio" *name*="animal"  *id*="cachorro" *value*="cachorro"><br><br>

        <label *for*="cavalo">Cavalo</label>

        <input *type*="radio" *name*="animal" *id*="cavalo" *value*="cavalo"><br><br>

        <!--CHECKBOX.-->

        <!--Serve para dar a possibilidade de múltiplas escolhas para o usuário.-->

        <h2>O que você gosta.</h2><br><br>

        <label *for*="programacao">programação</label>

        <input *type*="checkbox" *name*="programacao" *id*="programacao" *value*="programacao"><br><br>

        <label *for*="banco">banco de dados</label>

        <input *type*="checkbox" *name*="banco" *id*="banco" *value*="banco"><br><br>

        <label *for*="hardware">Hardware</label>

        <input *type*="checkbox" *name*="hardware" *id*="hardware" *value*="hardware"><br><br>

        <h2>Escolha uma cor.</h2><br><br>

        <!--SELECT.-->

        <!--Serve para dar múltiplas escolhas em um item que pode abrir.-->

        <!--OPTION.-->

        <!--São as opções dentro da seleção.-->

        <select *name*="cores">

            <!--Serve para desabilitar o item.-->

            <option *selected* *disabled* *value*="">Escolha uma opção</option>

            <option *value*="amarelo">amarelo</option>

            <option *value*="amarelo">amarelo</option>

            <option *value*="amarelo">amarelo</option>

        </select><br><br>

        <!--TEXT-AREA.-->

        <!--Possibilita que o usuário possa escrever textos grandes.-->

        <textarea *name*="areatexto" *id*="" *cols*="30" *rows*="10" ></textarea><br><br>

        <!--AUDIOS.-->

        <!--Coloca o áudio e faz com que ele inicie sozinho e fique em loop.-->

        <audio *autoplay* *loop*>

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

            Seu navegador não suporta a tag de áudio.

          </audio>

          <!--VÍDEOS.-->

          <!--Se caso temos um vídeo em nossa página e a atualizamos esse vídeo ficará preto então para "esconder" assim usamos o poster que vai servir como tambneio do vídeo.-->

          <video *poster*="" *width*="1000px" *height*="400px" *controls*

*desablepictureinpicture* *controlslist*="nodowload nofullscreen">

            <source *src*="seuarquivo.mp4" *type*="audio/mp4">

            Seu navegador não suporta a tag de vídeo

          </video>

          <!--"controls

          desablepictureinpicture controlslist="nodowload nofullscreen">" Isso desabilita os três pontos, download do vídeo e o fullscreen.-->

          <!--DIV E HTML SEMÃNTICO.-->

          <!--DIV de "divisória" como se fosse um bloco onde são colocados conteúdos.-->

          <div>

            <h1>Dentro da div é aonde vai o conteúdo.</h1>

          </div>

          <!--HEADER.-->

          <!--O header é o cabeçalho "topo", tudo que for dentro dele vai aparecer na parte superior do documento. Claro onde ele for declarado é aonde ele vai aparecer, por exemplo eu declarei aqui no final e onde ele vai aparecer.-->

          <header>

            <nav><!--A tag nav serve como menu de navegação, dentro dela vão geralmente links.-->

                <a *href*="#">Hoje</a>

                <a *href*="#">Clicar</a>

                <a *href*="#">Aqui</a>

            </nav>

        </header><br><br>

        <!--Essa tag não necessariamente dentro de outra tag.-->

        <nav>

            <a *href*="#">Hoje</a>

            <a *href*="#">Clicar</a>

            <a *href*="#">Aqui</a>

        </nav><br><br>

        <!--SECTION.-->

        <!--De sessão, uma outra sessão do documento.-->

        <section>

        </section>

        <!--FOOTER.-->

        <!--erve para fazer a parte inferior do documento. Rodapé.-->

        <footer>

        </footer>

        <!--MAIN.-->

        <!--De principal, parte principal do documento.-->

        <main>

        </main>

        <!--ARTICLE.-->

        <!--De artigo, para dizer que aquele conteúdo é um artigo.-->

        <article>

        </article>

        <input *type*="submit" *value*="Enviar"><!--Enviar os dados.-->

    </form>

***CSS.***

<!--Como conectar um css externo.-->

    <link *rel*="stylesheet" *type*="text/css" *href*="css/style.css" *media*="screen" />

/\*SELETORES E CLASSES\*/

/\*Um seletor é a maneira de dizer para o css sobre qual é o elemento html que se deseja trabalhar. Como o css faz para ele encontrar o elemento que você quer trabalhar. Existem três tipos.\*/

/\*PELA PRÓPRIA TAG:\*/

<h1>Site com html e css.</h1>

/\*POR CLASSES:\*/

    <h2 class="site">Site original com html e css.</h2>

/\*POR ID:\*/

    <h3 id="id">primeiro site com html e css.</h3>

/\*DECLARAÇÃO DE CADA UMA DAS TRÊS FORMAS.\*/

/\*O nome das classes e dos ids não podem começar com números, caso necessário coloque no final do nome.\*/

/\*CORINGA.\*/

/\*Serve para dizer para o css que todos os comandos dentro dele vão ser aplicados em TODOS os elementos.\*/

\*{

    padding: 0;

    margin: 0;

}

/\*Própria tag html:\*/

/\*Fazemos agrupamentos caso mais de um elemento tem as mesmas propriedades.\*/

h1, p{

    color: yellowgreen;

}

/\*Por classes:\*/

/\*As classes podem ser usadas em mais de um elemento. Universal. \*/

*.site*{

    color: red;

}

/\*Por id:\*/

/\*Nome único desse elemento.\*/

*#id*{

    color: rebeccapurple;

}

/\*PROPRIEDADE COLOR.\*/

/\*Serve para color cor.\*/

\*h1{

    color: tomato;

}

/\*PROPRIEDADE BACKGROUD.\*/

/\*Tudo relacionado com Backgroud que dizer fundo, alguma coisa de fundo.\*/

body{

    background-color: blanchedalmond;/\*Cor sólida de fundo.\*/

    background: rebeccapurple;/\*Também é cor sólida de fundo.\*/

    background-image: url("");/\*Imagem de fundo.\*/

    /\*Junto com a imagem de fundo vem o:\*/

    background-size: ;/\*Que é o tamanho da imagem.\*/

    /\*EXISTEM DOIS TIPOS.\*/

    background-size: contain;/\*Coloca toda imagem mas se ela não se encaixar no espaço disponível ela começa a repetir. Para isso usamos outra propriedade que é a.\*/

    background-repeat: ;/\*Repetição da imagem.\*/

    /\*Caso a imagem esteja repetindo podemos passar.\*/

    background-repeat: no-repeat;/\*Para não repetir mas o espaço em que a imagem espava repetindo fica em branco.\*/

    /\*Se queremos que ela repita passamos.\*/

    background-repeat:repeat;/\*Para repetir.\*/

    background-repeat:repeat-x;/\*Para repetir da direita para a esquerda.\*/

    background-repeat: repeat-y;/\*Para repetir de cima para baixo.\*/

    background-size: cover;/\*Coloca toda imagem e ajusta ela para caber em toda o espaço disponível. Com o cover e com  contain podemos usar.\*/

    background-position: ;/\*Posição de exibição da imagem.\*/

    background-position: center;/\*Centro da imagem.\*/

    background-position:top ;/\*Parte de cima\*/

    background-position: bottom;/\*Parte de baixo.\*/

    background-position: left;/\*Parte da esquerda.\*/

   background-position: right;/\*Parte da direita.\*/

}

/\*PROPRIEDADE BORDER.\*/

/\*Define a estilização de bordas.\*/

h1{

    border: 1px solid red;/\*Borda de 1px sólida com a cor vermelha.\*/

    border: 1px dotted red;/\*Borda pontilhada.\*/

    border:1px dashed red ;/\*Borda feita de traços.\*/

    border: 1px dotted red;/\*Borda dupla.\*/

    border: 1px groove red;/\*Borda com efeito 3d.\*/

    border: 100px ridge red;/\*Também com efeito 3d.\*/

    border: 100px inset red;/\*Borda por dentro.\*/

    border: 100px outset red;/\*Borda por fora.\*/

    border:none;/\*Nenhuma borda.\*/

    border: 100px hidden red;/\*Borda Oculta, não aparece.\*/

    border-left: 1px solid red;/\*Borda a esquerda.\*/

    border-top: 1px solid red;/\*Borda do topo.\*/

    border-right: 1px solid red;/\*Borda a direita.\*/

    border-bottom: 1px solid red;/\*Borda em baixo.\*/

    border-radius: 120px;/\*Arredondamento de bordas.\*/

}

/\*PROPRIEDADE MARGIN.\*/

/\*Serve para dar um espaço em branco em uma das direções do elemento, parte de cima, de baixo, da direita e esquerda.\*/

h2{

    margin-top: 10px;/\*Margin em relação ao topo.\*/

    margin-bottom: 10px;/\*Margin em relação a parte de baixo.\*/

    margin-left: 10px;/\*Margin em relação parte da esquerda.\*/

    margin-right: 10px;/\*Margin em relação a parte da direita.\*/

    /\*Podemos colocar tudo isso dentro de uma mesmo comando de margin:\*/

    margin: 30px 20px 40px 10px;

    /\*O 1° diz respeito a parte do topo, 2° a parte da direita, 3° a parte de baixo, 4° a parte da esquerda.\*/

}

/\*PROPRIEDADE PADDING.\*/

/\*Serve para dar um preenchimento em relação a parte de dentro do elemento. É parecido e tem as mesmas propriedades do margin.\*/

h3{

    padding-top: 10px;/\*Preenchimento em relação ao topo.\*/

    padding-bottom: 10px;/\*Preenchimento em relação a parte de baixo\*/

    padding-right: 10px;/\*Preenchimento em relação a direita.\*/

    padding-left: 10px;/\*Preenchimento em relação a esquerda.\*/

    padding: 10px 10px 10px 10px;/\*Todos os lados em um mesmo comando seguindo a mesma ordem que o margin.\*/

}

/\*PROPRIEDADE HOVER.\*/

/\*Serve para fazer animações quando o mouse estiver sobre o elemento.\*/

a{

    background-color: aliceblue;

}

a*:hover*{/\*Quando passar ou ficar o mouse por cima do link ele vai mudar a sua cor de fundo.\*/

    background-color: brown;

}

/\*PROPRIEDADES WIDTH E HEIGTH.\*/

/\*

WIDTH: Largura.

HEIGTH: Altura.

MAX-WIDTH: Largura máxima.

MIN-WIDTH: Largura mínima.

MAX-HEIGTH:Altura máxima.

MIN-HEIGTH:Altura miníma.

\*/

h1{

        width: 100px;/\*Um título com 100px de largura.\*/

        height: 200px;/\*Um título com 200px de altura\*/

        max-width: 10px;/\*Um título com de 10px de largura máxima.\*/

        min-width: 10px;/\*Um título com 10px de largura mínima.\*/

        max-height: 10px;/\*Um título com 10px de altura máxima.\*/

        min-height: 10px;/\*Um título com 10px de altura mínima.\*/

}

/\*PROPRIEDADE TEXT.\*/

/\*Serve para estilizar textos.\*/

p, a{

    text-align: ;/\*Alinhamento de texto.\*/

    text-align: center;/\*Centro.\*/

    text-align: justify;/\*Justificado, quadrado todos os cantos.\*/

    text-align: left;/\*A esquerda.\*/

    text-align: right;/\*A direita.\*/

    /\*Direção do texto.\*/

   direction: ltr;/\*Da esquerda para direita.\*/

    direction: rtl;/\*Da direita para a esquerda.\*/

    /\*Inverção do texto.\*/

    unicode-bidi: bidi-override;/\*Deixa o texto invertido. De trás para frente.\*/

    /\*Deecoração do texto.\*/

    text-decoration: ;/\*Decoração do texto.\*/

    text-decoration: none;/\*Nenhuma de decoração.\*/

    text-decoration: dashed;/\*\*/

    text-decoration: dotted;/\*\*/

    text-decoration: line-through;/\*Risca o texto.\*/

    text-decoration: overline;/\*Coloca uma linha sobre o texto.\*/

    text-decoration: solid;/\*Decoração sólida.\*/

    text-decoration: underline;/\*Decoração com underline.\*/

    /\*Transformação do texto.\*/

    text-transform: ;/\*Transformação do texto.\*/

    text-transform: capitalize;/\*A primeira letras de todas as palavras fica maiúscula.\*/

    text-transform: lowercase;/\*Todas as letra ficam minúsculas.\*/

    text-transform: none;/\*Nenhuma transformação de texto.\*/

    text-transform: uppercase;/\*Todas as letras ficam maiúsculas.\*/

    /\*Recuar linhas do texto. Endentação do texto. Primeira linha do parágrafo.\*/

    text-indent: ;/\*Tag para recuar linhas do texto.\*/

    text-indent: 10px;/\*Recuar 10px uma linha de texto.\*/

    /\*Espaçamento entre letras.\*/

    letter-spacing: ;/\*Tag para espaçamento entre letras.\*/

    letter-spacing: 10px;/\*10px de espaçamento entras cada uma das letras do texto. Podemos usar valores "NEGATIVOS TAMBÉM"\* ai seria para grudar as letras e palavras. As vezes a quantidade de PXs pode mudar a fonte do texto.\*/

    /\*Espaçamento entre linhas de texto.\*/

    line-height: ;/\*Altura da linha.\*/

    line-height: 0.8;/\*Um espaço de 0.8px entre cada uma das linhas do texto. Geralmente com números quebrados.\*/

    /\*Espaçamento entre palavras.\*/

    word-spacing: ;/\*Tag para espaçamento entre cada uma das palavras do texto.\*/

    word-spacing: 10px;/\*10px de espaço entre cada uma das palavras do texto.\*/

}

/\*PROPRIEDADE FONT.\*/

/\*Define a estilização de tamanho, estilo da fonte de um texto.\*/

h1{

    /\*Família de fontes.\*/

    font-family: ;/\*Tem vários estilos de fontes.\*/

    /\*Podemos colocar outros tipos de fontes também.\*/

    font-style: italic;/\*Deixa em italico\*/

    font-style: oblique;/\*Deixa a letra meio deitada.\*/

}

/\*PROPRIEDADE DISPLAY.\*/

/\*A propriedade display serve para controlar o layout, por exemplo como se ele vai ficar um do lado do outro, se vai ocupar toda a tela etc. Tipo de comportamento que ele vai ter diante da tela.\*/

div{

    display: block;/\*Tipo bloco: ocupa toda a tela e não deixa elementos ficarem na mesma linha eles automaticamente quebram a linha para a parte de baixo.\*/

    display: contents;/\*\*/

    display: inline;/\*\*/

    display: inline-block;/\*\*/

    display: ;/\*\*/

}

/\*PROPRIEDADE MAX-WIDTH\*/

/\*LARGURA MÁXIMA.\*/

/\*Serve para dizer a largura máxima de um elmento.\*/

/\*Se pegarmos por exemplo um div, ela tem por padrão uma largura de 100% ocupando todo o espaço da tela, mas se aplicarmos a propriedade MAX-WIDTH vamos dizer que essa div vai ter uma largura máxima de um determinado valor e no que isso resulta? resulta que por exemplo se a tela tiver um tamanho menor do o especificado a tela não vai cortar o elemento e esconder o que sobrou mas sim se ajustar até o tamanho especificado.\*/

*.exe1*

{

    border: 1px solid red;

    max-width: 900px;

}

*.exe2*

{

    border: 1px solid red;

    max-width: 900px;

}

/\*PROPRIEDADE POSITION.\*/

/\*Serve para determinar o posicionamento dos elementos em uma página. \*/

*.p1*{

    position: absolute;/\*Ocupa apenas o tamanho de seus elementos.\*/

    position: fixed;/\*Serve para deixar um elemento fixo na tela independente se a página vai rolar para cima ou para baixo, ele vai ter sempre aquela mesma posição.\*/

    position: relative;/\*Relativo ao local onde ele está, assim podem ser passados coordenadas para esse elemento mas por exemplo se dois elementos um do lado do outro de as coordenadas são para direita ele irá porém ficará em cima do outro. Para isso usamos:\*/

    z-index: 100;

    /\*Serve para dizer está ou não na frente, você pode jogar os elementos para frente ou para trás de outros. Os seus valores podem ser negativos até 999. Para usar essa tag é preciso do posição relativa.\*/

    position: static;/\*Padrão de todos os elementos.\*/

    position: sticky;/\*Posição pegajosa, ou seja, por exemplo se tivermos uma div e estivermos rolando para baixo esse elemento vai sumir a área visível mas com sticy podemos fazer com que quando o fim da tela atingir esse elemento ele grude nela.\*/

}

 /\*PROPRIEDADE OVERFLOW\*/

 /\*Por exemplo se uma tive tem um tamanho fixo mas colocamos mais conteúdo do que o tamanho limite o que vai acontecer é que quando tiver ultrapassado o tamanho total da div esse conteúdo vai ficar na parte de fora e para isso usamos o OVERFLOW.\*/

*.p3*{

    overflow: ;/\*determina como um conteúdo vai se comportar caso ultrapasse o tamanho total de seu elemento pai. Só funciona caso tenhamos especificado uma altura.\*/

    overflow: auto;/\*Automático parecido com o scroll porém vai mostrar a barra de rolagem caso seja necessário.\*/

    overflow: hidden;/\*Mostra só o que por exemplo coube dentro de uma div o que estiver para fora será cortado e não aparecerá.\*/

    overflow: scroll;/\*Mostra só o que coube dentro do elemento mas para o que ficou para fora ele cria uma barra de rolagem.\*/

    overflow: visible;/\*Vai mostrar o conteúdo mesmo que ele saí para fora.\*/

 }

 body{

    overflow-y: hidden;/\*Tiramos a barra de rolagem de cima para baixo e vice verça do navegador.\*/

    overflow-x: hidden;/\*Tiramos a barra de rolagem para os lados.\*/

 }

 /\*PROPRIEDADE FLOAT.\*/

 /\*Serve para fazer um elemento flutuar.\*/

 /\*Para entender imagine uma div com uma imagem e um texto e você quer que o texto fique do lado dessa imagem e não em baixo como é padrão para isso você pode usar o FLOAT. Colocar uma imagem de um lado e um texto do outro.\*/

*.p4*{

    float: left;/\*O conteúdo vai flutuar para esquerda.\*/

    float: right;/\*O conteúdo vai flutuar para a direita.\*/

    clear: both;/\*Limpa a flutuação.\*/

    clear: left;/\*Limpa a flutuação a esquerda.\*/

    clear: right;/\*Limpa a flutuação a direita.\*/

    clear: none;

 }

 /\*PROPRIEDADE OPACITY.\*/

/\*Serve para determinar a opacidade de um elemento.\*/

*.p2*{

    opacity: ;/\*Determina a opacidade/ transparência de um elemento.\*/

    opacity: 0.1;/\*Pode ser de 0.0 até 1.0 sendo 1.0 100%, ou seja, sem opacidade. 0.1 é 10% de opacidade, 0.2 é 20% e por ai vai. Quanto menor o valor mais opacidade vai ter.\*/

}

/\*EFEITO DROPDOWN.\*/

/\*Serve para fazer o efeito de quando passa o mouse por exemplo em um link abrir outros links.\*/

/\*HTML\*/

<body>

    <ul>

        <li><a *href*="#">Link</a></li>

        <li><a *href*="#">Link</a></li>

        <li><a *href*="#">Link</a></li>

        <li><a *href*="#">Link</a></li>

        <li *class*="dropdown">

            <a *class*="dropbtn" *href*="#">Link</a>

            <div *class*="dropdown-content">

                <a *href*="#">Link</a>

                <a *href*="#">Link</a>

                <a *href*="#">Link</a>

                <a *href*="#">Link</a>

            </div>

        </li>

    </ul>

</body>

/\*CSS\*/

ul{

    background: black;

    /\*Tira a decoração de uma lista de itens.\*/

    list-style: none;

    padding: 0;

    margin: 0;

    overflow: hidden;

}

li{

    float: left;

}

li a, *.dropbtn*{

    display: inline-block;

    color: white;

    padding: 14px 16px;

    text-decoration:none;

    text-align: center;

}

li a*:hover*, *.dropdown:hover* *.dropbtn*{

    background-color: red;

}

li*.dropdown*{

    display: inline-block;

}

*.dropdown-content*{

    display: none;

    position: absolute;

    background: white;

    width: 150px;

}

*.dropdown-content* a{

    color: black;

    display: block;

    text-align: left;

    text-decoration: none;

    border: 1px solid rgb(207, 207, 207);

}

*.dropdown:hover* *.dropdown-content*{

    display: block;

}

/\*REGRA IMPORTANT.\*/

/\*Serve para dizer que no fluxo de estilização qual elemento vai ser vai importante.\*/

/\*Exemplo para entender.\*/

*#id*{

    color: blue;

}

*.minhaclasse*{

    color: red;

}

h1{

    color: chartreuse;

}

/\*O css lê de cima para baixo.\*/

/\*Na ordem em que aparece o css vai ler o id, a classe e por último a tag porém vai ficar sempre azul porque isso é uma coisa chamada especificidade. ID É MAIS FORTE QUE (h1.mhinhaclasse) = tag classe, que (minhaclasse) = classe simples e que (h1) = tag simples. TAG CLASSE É MAIS FORTE QUE classe simples e que tag simples. CLASSE SIMPLES É MAIS FORTE QUE tag simples. \*/

/\*Para quebrar esse fluxo de estilização podemos utilizar o !important ele vai quebrar o fluxo de estilização e dar mais prioridade aquele elemento marcado como importante.\*/

/\*Antigamente na ordem de prioridade de leitura do css era o id por ser mais forte/especifico, agora quem é mais importante é a tag h1 então o css vai deixar de lado os outros edentificadores e dar prioridade a tag h1 no caso a cor predominante anteriormente era o BLUE agora passou a ser o CHARTREUSE.\*/

*#id*{

    color: blue;

}

*.minhaclasse*{

    color: red;

}

h1{

    color: chartreuse !important;

}

/\*Podemos usar important em vários elementos mas é recomendável não porque se tivermos vários important em vários elementos diferentes por exemplo abaixo.\*/

*#id*{

    color: blue !important;

}

*.minhaclasse*{

    color: red;

}

h1{

    color: chartreuse !important;

}

/\*Porém vai acontecer o que acontecia antes o css vai dar importância para quem é mais forte/especifico por exemplo o ID e o H1 estão marcados como IMPORTANT porém agora o css vai ver quem é mais forte nesse caso como já vimos o ID é mais forte que tag então a cor vai ser BLUE.\*/

/\*Por isso não é recomendável ficar aplicando important em qualquer elemento, pode acabar virando uma bagunça.\*/

/\*EFEITO GRADIENTE.\*/

/\*Serve para fazer o efeito de uma cor passar para outra.\*/

body{

    background-image: linear-gradient();

    /\*Dentro dos parênteses vão alguns parâmetros como direção e cores.\*/

    background-image: linear-gradient(45deg, cyan, yellow, green);

    /\*Linear-gradient = gradiente em linhas.\*/

    /\*(45deg) = direção = graus = ângulos, depois vem a combinação de cores podem ser utilizadas mais do que duas cores.\*/

    background-image: linear-gradient(to right, cores);

    /\*Direção para direita.\*/

    background-image: linear-gradient(to bottom right, cores);

    /\*De cima para baixo em diagonal.\*/

    background-image: linear-gradient(to top right, cores);

    /\*De baixo para cima a direita.\*/

    /\*Pode não possuir o valor da direção no inicio então vai ficar na direção padrão que do topo para baixo.\*/

    background-image: radial-gradient();

    /\*Faz o greadiente porém em circulo.\*/

    background-image: radial-gradient(red 10%, yellow 80%);

    /\*Não é preciso dizer a direção apenas as cores e opcional a porcentagem para as cores.\*/

/\*}

/\*PROPRIEDADE TEXT E BOX SHADOW.\*/

/\*TEXT SHADOW: efeito de sombras em textos.\*/

/\*BOX SHADOW: efeito de sombras em caixas.\*/

h1{

    text-shadow: 2px 2px 5px red;

    /\*2px na vertical, 2px na horzontal, 5px de (blur) = desfoque, espalhamento da sombra e a cor dessa sombra.\*/

    text-shadow: 2px 2px 5px red, 10px 10px 5px blue;

    /\*Pode ser adicionado mais de um efeito de sombra em um elemento.\*/

}

/\*Uma div.\*/

*.caixa*{

    width: 100%;

    height: 300px;

    box-shadow: 5px 5px 5px brown;

    background: grey;

}

/\*EFEITO DE CARD.\*/

HTML:

    <div *class*="caixa2">

    </div>

CSS:

/\*Uma div.\*/

*.caixa2*{

    background-color: white;

    width: 40%;

    height: 300px;

/\*height: auto; reconhece que é necessário mais tamanho para encaixar o conteúdo então se expande para encaixar todo o conteúdo.\*/

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0 6px 20px rgba(0, 0, 0, 0.19);

}

/\*EFEITO DE TEXTO.\*/

/\*Formatação de textos.\*/

h2{

    white-space: nowrap; /\*Faz com que um texto que tenha várias quebras de linhas fique tudo na mesma linha.\*/

    overflow: hidden;

    text-overflow: ellipsis;/\*Por exemplo eu tenho um texto que é muito grande e eu cortei ele usando o overflow: hidden; mas usando essa propriedade text-overflow: ellipsis; eu adiciono três pontos indicando que ali tem uma continuação.\*/

}

/\*MEDIA QUERIES.\*/

/\*Servem para deixar um site responsivo, ou seja, se ajustando aos tipos de telas. Propriedade que controla o que acontece quando temos diferentes resoluções de tela.\*/

/\* Smartphones (600px para baixo) \*/

@media only screen and (max-width: 600px) {

}

/\* Dispositivos um pouco maiores, mas ainda pequenos (Pequenos Tablets e Smartphones + largos, 600px para cima) \*/

@media only screen and (min-width: 600px) {

}

/\*Dispostivos Médios (Tablets deitados, 768px para cima) \*/

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

}

/\* Dispostivos Largos (laptops/desktops, 992px para cima) \*/

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

}

/\* Super Largos (Telas maiores laptops, desktops e até TVs, 1200px para cima) \*/

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

}

@media only screen and (min-width: 600px) and (max-width: 900px){

    /\*Caso tenhamos que estilizar entre um determinado espaço nesse caso é o espaço entre 600px e 900px de resolção.\*/

}

/\*PROPRIEDADE FLEXBOX.\*/

/\*FLEXIVEL.\*/

*.flex*{

    background: blue;

    display: flex;

    /\*O display: flex; é muito melhor de se trabalhar pois porque ele tem várias outras propriedades ligadas a ele. Como por exemplo o:\*/

    /\*Direção do flex.\*/

    flex-direction: column;/\*Coluna.\*/

    flex-direction: column-reverse;/\*Coluna invertida.\*/

    flex-direction: row;/\*Linha.\*/

    flex-direction: row-reverse;/\*Linha invertida.\*/

   /flex-wrap: ; /\*embrulho, ou seja, contido dentro de um elemento por exemplo uma div. Por padrão é o:\*/

   flex-wrap: nowrap; /\*Por exemplo temos uma div e dentro dessa div tem outras divs caso colocamos várias vai chegar uma hora que não vão couber mais nesse espaço então eles vão sair para fora.\*/

   flex-wrap: wrap;/\*Faz com que como já foi dito anteriormente ao invés de sair para fora da div, agora eles vão cair para baixo mas ainda coubendo detro da div pai que está contendo essas outras divs filhas.\*/

   /\*

  flex-direction: ;

  flex-wrap: ;

  Ao invés de passar essas duas propriedades separadamente podemos usar o:

  \*/

  flex-flow: row wrap;/\*Passamos as duas propriedades nessa única propriedade.\*/

  /\*Outra propriedade é o justify-content: ;\*/

  justify-content: center;/\*Os itens são alinhados ao centro.\*/

  justify-content: flex-end;/\*Os itens são alinhados ao fundo.\*/

  justify-content: flex-start;/\*Padrão, os itens são alinhados a esquerda.\*/

  justify-content: left;/\*Alinha os itens a esquerda.\*/

  justify-content: right;/\*Alinha os itens a direita.\*/

  justify-content: space-around;/\*Faz parecido com space-between porém esse espaço uniforme é em toda a volta do elemento.\*/

  justify-content: space-between;/\*Faz com que os itens se ajustem ao elemento em que estão contidos de forma que fique um espaço uniforme entre eles. Por a tradução é ESPAÇO ENTRE.\*/

  align-items: stretch;/\*Faz com que o itens ocupem a altura de 100% do elemento em que ele está contido.\*/

  align-items: center; /\*alinha os itens ao centro. Fica no centro em relação a altura.\*/

  align-items: flex-start;/\*Alinha na parte de cima do elemento em que ele está contido.\*/

  align-items: flex-end;/\*Alinha na parte de baixo do elemento em que ele está contido.\*/

  /\*align-content: ;Alinhamento de conteúdo.\*/

  align-content: flex-end;/\*Todo o conteúdo fica alinhado na parte de baixo.\*/

  align-content: flex-start;/\*Todo o conteúdo fica alinhado ao topo.\*/

  flex-grow: ; /\*Crescimento dos itens em relação a tela de forma automatizada\*/

  flex-grow: 1;/\*Cria uma largura/crescimento do item de forma que todos os elementos cresçam de forma uniforme uma em relação a outra.\*/

/\*FLEXIONAR.\*/

flex: 13%;

}

*.flex* div{

    background: white;

    border: 1px solid black;

    margin: 20px;

    padding: 30px;

    text-align: center;

}

***SASS.***

***CSS WITH SUPERPOWERS***

***Css com super-poderes***

***Sass:*** Sytactically Awesome Style Sheets.

***Sass:*** Folhas de estilo sintaticamente impressionantes.

***PHP***

***FUNÇÃO ISSET.***

Html:

<!DOCTYPE *html*>

<html *lang*="pt-br">

<head>

    <meta *charset*="UTF-8">

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

    <title>Curso de php</title>

</head>

<body>

<br><br>

<form *action*="index.php" *method*="post">

<label *for*="nome">Digite seu nome</label>

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

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

</form>

</body>

</html>

Php:

<?php

//FUNÇÃO ISSET.

//Serve para verificar se um certo dado existe/foi definido.

$nome = "Jeverson";

//Geralmente associada ao IF.

//Retorna um valor booleano, ou seja, True (caso o valor passado tenho sido definido) ou False (caso o valor passado não tenha sido definido).

if (isset($nome)) {

    echo "Variável nome foi definida <br><br>";

    //O resultado foi esse.

    //$nome = ""; mesmo que a variável não estivesse "recebendo nada" = uma string vazia ainda assim ela foi definida pois string vazia continua sendo um valor, diferente de NULL que ai sim não é valor nenhum então cai no ELSE.

}else {

    echo "Variável não definida <br><br>";

    //$nome = "Jeverson"; caso comentarmos a variável nome o resultado vai ser esse porque a variável não vai ser compilada, vai deixar de existir.

}

//Outra aplicação do ISSET seria verificar se os dados de um formulário ou o formulário foram enviados.

if (isset($\_POST['nome'])) {

    //IF (condição), ISSET (função), $\_POST (método), ['nome'] (conteúdo).

    //Caso o formulário tenha sido enviado.

    $vnome =  $\_POST['nome'];

    //$vnome = (recebe o conteúdo que foi enviado pelo formulário) = $\_POST o  valor['nome'].

    echo "$vnome <br>";

    //Quando clicarmos em ENVIAR vai cair nessa condição pois o formulário será submetido.

}else {

    echo "Formulário não enviado";

}

?>

***STRLEN.***

Html:

<!DOCTYPE *html*>

<html *lang*="pt-br">

<head>

    <meta *charset*="UTF-8">

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

    <title>Curso de php</title>

</head>

<body>

<form *action*="index.php" *method*="post">

<label *for*="nome">Digite seu nome</label>

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

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

</form>

</body>

</html>

Php:

<?php

//STRLEN.

//Serve para verificar a quantidade de caracteres por exemplo de um campo de digitação.

//Geralmente associado ao IF.

if (isset($\_POST['nome'])) {

    //Verifico se o formulário foi enviado. Se sim:

    if (strlen($\_POST['nome']) == 0) {

        //Verifico se os dados que vem do formulário pelo método POST que tem o conteúdo NOME tem sua quantidade de caracteres == 0, ou seja, não foi digitado nada no campo de digitação cai nessa condição.

        echo "Preencha seu nome corretamente.";

    }else {

        //Se não o campo teve digitações, teve preenchimento.

        echo "Seu nome está  correto!";

    }

}else{

    //Se os dados do formulário não foram enviados.

    echo "Esse dado não existe";

}

?>

***FUNÇÕES.***

<?php

//FUNÇÕES.

//Nada mais é do que um agrupamento de comandos/instruções que podemos chamar em um determinado tempo.

function aula(){

    //(FUNCTION) = declaração.

    //(AULA) = nome da função.

    //() = parametros.

    echo "<hr> OI <br>";

    echo "Estudo sobre funções <hr>";

}

//Chamada da função para que ela possa ser executada.

aula();

//Podemos chamar de uma vez.

//Não precisamos repetir os mesmos comandos apenas chamamos a função com os comandos.

aula();

aula();

aula();

aula();

function soma($n1, $n2){

    //$n1 e $n2 são os parâmetros da função.

    $resultado =  $n1 + $n2;

    echo "A soma de $n1 com $n2 é: $resultado";

}

//3 e 7 são os valores dos parâmetros dessa função.

//n1 = 3 e n2 = 7.

soma(3,7);

echo '<br>';

//As funções acima trabalharam com os valores mas elas apenas imprimiram esses valores na tela nada mais, agora vamos retornar esses valores.

function soma2 ($n1, $n2){

    $resultado = $n1 + $n2;

    return $resultado;

}

//Agora não basta apenas chamar a função, temos que chamar armazenando seu valor em algum lugar.

$res = soma2(10, 4);

echo "resultado = $res";

echo '<br>';

$valores = array(1,2,6,7,3,10,5,211);

function media($val){

    $tamanho = count($val);

    $soma =  0;

    for ($i=0; $i < $tamanho; $i++) {

        $soma = $soma + $val[$i];

    }

    return $soma/$tamanho;

}

$med =  media ($valores);

echo "$med";

?>

***ARRAYS.***

<?php

//ARRAYS.

//Um array é um conjunto de variáveis do mesmo tipo.

$vetor = array(5);

//array(5); quer dizer 5 posições, couberam 5 elementos.

$vetor[0] = "carro";

//Adicionando uma string na primeira posição do array.

$vetor[1] = "moto";

//Adicionando uma string na segunda posição e por ai vai até a posição 4 pois o número de elementos começa em 0.

echo "$vetor[0]".'<br>';

//Buscando um valor do array.

//$vetor = array(5); não precisamos fazer isso.

//Podemos fazer assim determinamos que o array tem três posições e os seus velores em cada uma das posições.

$vetor2 =  array(100,200,300);

//Array com indice.

//O indice é NOME e o seu valor é JEVERSON.

$vetor3 = array("nome"=>"Jeverson");

//Como armazenar os dados que vem de um formulário em um array.

HTML:

<body>

  <form *method*="post" *action*="index.php">

    <input *type*="text" *name*="nome" *placeholder*="Seu nome">

    <input *type*="email" *name*="email" *placeholder*="Seu e-mail">

    <textarea *name*="mensagem" *placeholder*="Sua mensagem"></textarea>

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

</form>

PHP:

// Recupera os dados do formulário

$nome = $\_POST["nome"];

$email = $\_POST["email"];

$mensagem = $\_POST["mensagem"];

// Cria um array associativo com os dados

$dados = [

    "nome" => $nome,

    "email" => $email,

    "mensagem" => $mensagem

];

// Agora você pode usar o array $dados para manipular os dados, como salvá-los em um banco de dados, etc.

// Por exemplo:

// - Inserir no banco de dados

// - Enviar por e-mail

// - Exibir na página

?>

***FUNÇÃO IMPLOLDE E FUNÇÃO EXPLODE.***

<?php

//EXPLODE.

//Você define o caractere, um ou mais caracteres e através desse caractere podemos transformar uma string em um array.

//Por exemplo.

$estado = "São Paulo/SP";

$arrayestado = explode('/', $estado);

//explode(1°(delimitador), 2°(a próprio valor  que no caso é a variável $estado.))

var\_dump($arrayestado);

//O resultado foi que ele transformou em um array com duas string diferentes.

/\*

1° = São Paulo.

2° =  SP.

\*/

$estado2 = "São Paulo/SP/Brasil/Terra";

$arrayestado2 = explode('/', $estado2);

var\_dump($arrayestado2);

//O resultado foi que ele transformou em um array com quatro string diferentes.

/\*

P0 = São Paulo.

P1 =  SP.

P2 = Brasil.

P3 = Terra.

\*/

//IMPLODE.

//Em quanto o EXPLODE transforma uma string em um array o IMPLODE transforma um array em um string.

//Exemplo.

$info = "São Paulo/SP/Brasil/Terra";

$arrayInfo = explode('/', $info);

echo implode('-', $arrayInfo) . '<br>';

//implode(1°(separaddor), 2°(conteúdo));

$arrayTest = [1,2,3,4,56];

echo implode('-', $arrayTest);

?>

***VAR\_DUMP.***

<?php

//VAR\_DUMP.

//Imprime não só o conteúdo de um varável por exemplo mais as suas informações.

//Exemplo.

$nome = "Jeverson";

var\_dump($nome);

/\*Resultado

C:\wamp64\www\curso\php\ "Local"

index.php "A pasta"

:9:string 'Jeverson' "A string"

(length=8) "A quantidade de caracteres"

\*/

$frutas = ["Laranja", "Banana", "Abacaxi", 1];

echo "<pre>";

var\_dump($frutas);

echo "<pre>";

/\*Resultado.

C:\wamp64\www\curso\php\

index.php:23:

array (size=4) "Array com 4 posições"

  0 => string 'Laranja' (length=7)

  1 => string 'Banana' (length=6)

  2 => string 'Abacaxi' (length=7)

  3 => int 1

  O var\_dump mostra o array com a sua quantidade de posições e o tipo do conteúdo que está armazenado em cada posição.

  Por exemplo :

  Laranja : string : 7 caracteres

  1 : inteiro

\*/

?>

***SESSÕES.***

<?php

/\*SESSÕES.

A super global $\_SESSION.

\*/

//Como trabalhamos com cookies esse comando dá algumas alterações para o cookie.

//session\_set\_cookie\_params();

//Como por exemplo.

//Sempre que trabalhamos com SESSÕES temos que iniciar com o session\_start(); de preferência em cima antes de tudo.

session\_start();

//Criar um SESSÃO.

$\_SESSION['usuario'] = "Jeverson";

/\*

$\_SESSION = super global.

['usuario'] = identificador.

"Jeverson" = conteúdo.

Para que usamos sessões?

Usamos sessões para pegar valores em qualquer para do nosso sistema desde que inicie com session\_start para reviver a sessão e todos os seus valores. Podemos simplesmente iniciar uma sessão em uma pasta e recuperar seus valores em outra. Pois uma sessão e válida entre pastas e enquanto o navegador estiver aberto.

Quando trabalhamos com sessões estamos trabalhando com COOKIES. Esse cookie tem um id identificador que é um sequência de números e letras e um nome.

nome = PHPSESSID

id =  (r52q4kfqrjuavoev75rnl1r4h4)

Para ver esse valor vamos em INPENCIONAR, APLICATIVO E COOKIES.

\*/

//Serve para mostrar o id da sessão

var\_dump(session\_id());

//sSeerve para fazer com que sempre que o navegador seja atualizado ou uma pessoa loga no sistema, seja gerado um novo id aleatório, para fins de proteção pois com javascript é possível roubar a sessão e com isso entrar na conta de uma pessoa no sistema.

session\_regenerate\_id(true);

//Serva para destruir uma sessão.

//sesssion\_destroy();

/\*

As sessões trabalham com arquivos e por isso que é possível recuperar os valores em qualquer parte do sistema.

\*/

session\_set\_cookie\_params(['lifetime' => 10]);

//Esse comando determina um tempo de vida de uma sessão em segundos, nesse nosso exemplo ela vai valer por apenas 10 segundo depois disso ela será destruida, já que por padrão uma sessão vale enquanto o navegador ou a guia não é fechada.

session\_set\_cookie\_params(['httponly' => true]);

//Para fins de segurança porque se irmos em INSPEIONAR, CONSOLE e digitar document.cookie lá vai estar sua sessão aberta para todos e com javascript possibilitando o roubo. Com esse parâmetro você esconde esse cookie

?>

***JAVASCRIPT.***

<body>

<!--INTERNO E EXTERNO.-->

<!--Diferente do css, que é referenciado dentro da tag (head), o javascript é geralmente referenciado dentro do (body), bem no final antes do seu fechamento, pois o javascript, afeta diretamente, de acordo com os códigos, o que vai aparecer no corpo do html então é importante que ele seja revernizado por último, para tenhamos certeza que as funções dentro dele vão funcionar e modificar o html que vem antes de sua chamada. O navegador lê as informações de cima para baixo por esse motivo, devemos colocar o javascript por último, pois geralmente é a parte mais pesada do site.-->

<!--INTERNO.-->

    <script>

        alert('Deu certo!');

    </script>

<!--EXTERNO.-->

    <script *src*="js/index.js"></script>

</body>

/\*TIPOS DE SAÍDAS.

São as maneiras que o javascript tem de exibir uma informação/dado na tela.

\*/

//Existem muitas formas de exibir uma informação na tela mas a PRIMEIRA delas é o innerHTML.

document.getElementById("texto").innerHTML="Meu primeiro texto asando javascript";

/\*

No nosso caso foi um paragráfo.

<p id="texto"></p>

[getElementById]: pegar um elemento por id.

("texto"): id do elemento.

[innerHTML="Meu primeiro texto asando javascript";]: informação a ser exibida.

\*/

<!--A SEGUNDA forma de exibir uma informação na tela, seria usando o  document.write mas é mais usando para fins de testes.-->

    <p>

        <script>

            document.write('Escrito usando document.write');

        </script>

    </p>

    <!--Ele também aceita funções.-->

    <p>

        <script>

            document.write(5 + 5);

        </script>

    </p>

//A TERCEIRA forma, seria usando o alert.

alert("Fala meu nobre!");

//Ele também aceita funções.

alert(10 + 5);

/\*A QUARTA forma seria, usando o console.log

Mas ele é mais silencioso, podemos dizer assim, pois se irmos na página html, aparentemente não vai acontecer nada mas se irmos em inspecionar e console, lá vai estar a mensagem.

\*/

console.log("Ablas!");

/\*CONCEITO DE DECLARAÇÃO DE JAVASCRIPT.

Um programa de computador, na verdade é uma lista de instruções e serem executadas pelo computador, passamos da forma que o ser humano entende dados/informações/comandos para a forma que a máquina condiga interpretar e essas instruções são chamadas de declarações.

\*/

alert("Declaração"); // isso é uma declaração. Em javascript o que separa uma declaração de outra é o ;

/\*O Javascipt ignora espaços em branco.

Abaixo ambos significam a mesma coisa.

\*/

let pessoa = "Jeverson";

let pessoa="Jeverson";

//O Javascript também aceita quebras de filhas livremente. Os exemplos abaixo, ambos vão funcionar normalmente, contanto que tenhamos a terminação da declaração, o ;, para que o javascript entenda que a declaração e continua mesmo tendo quebrado linhas. Temos que usar com cuidado.

document.getElementById("texto").innerHTML = "Meu site em javascript!";

document

.getElementById("texto").innerHTML =

"Meu site em javascript!";

//Blocos de códigos.

function minhaFuncao() {

}

//O Javascipt também é case-sensitive, o que significa que ele diferencia entre letras maiúsculas de minúsculas.

let pessoa = "Jeverson";

let = PESSOA = "Jeremias";

//Os exemplos acima são considerados variáveis diferentes, pois a diferença está na capitalização das letas e o javascript distingue entre essas duas formas de escrita e as trata como diferentes.

/\*SINTAXE.

É a maneira como devemos digitar os códigos em js, as regras, os comandos.

A regra mais básica é em relação a diferença entre textos e números.

\*/

var pessoa = "Jeverson"; //do tipo texto vai entre "".

var numero = 1;// do tipo número não vai entre "".

//Não podemos fazer isso:

 var primeiro-nome;

 //Pois o - já tem a sua função determinada para operações de subtração, para resolver esse problema podemos fazer das seguintes maneias:

 var primeiro\_nome;

 var primeiroNome;

/\*COMENTÁIOS EM JS.

Os comentários não serão lidos pelo navegador.

\*/

//Comentário de uma única linha.

/\*

Comentário de várias

Linhas.

\*/

/\*VARIÁVEIS.

Vamos entender sobre as variáveis.

Variáveis são "recipientes" onde podemos armazenar informações que podem variar, ou seja, podem ter qualquer tipo de valor.

No JS temos 3 palavras-chaves para declarar variáveis.

->var

->let

->const

\*/

/\* VAR de variável.

Essa palavra-chave pode ser usada em qualquer parte do código. Tanto que podemos fazer:

var a = 1;

var a = 2;

Retribuímos um novo valor de a.

\*/

//Declaração. Três variáveis.

var a,b,c;

//Valores.

a = 1;

b = 3;

c = a + b;

var nome, sobrenome, nome\_completo;

nome = "Jeverson";

sobrenome = "Fagundes";

nome\_completo = nome + " " + sobrenome;

//[+ " " + ]: nome mais espaço vazio mais sobrenome.

document.getElementById("texto").innerHTML = nome\_completo;

/\* LET.

Diferente do VAR, o LET não permite redeclararão de uma variável.

\*/

var x = 10;// aqui é 10.

{

//Bloco de código.

    var x = 2;// aqui é 2.

}

//aqui também vai ser 2. Porque o navegador estava lendo o código e percebeu que o var foi redeclarado.

document.getElementById("p1").innerHTML = x;

let x1 = 10;// aqui é 10.

{

//Bloco de código.

    let x1 = 2;// aqui é 2.

}

//aqui também vai ser 10. Porque o navegador estava lendo o código e como o LET não pode ser redeclarado então o que está dentro do bloco de código só vai funcionar lá dentro. Não temos o problema de subscrever uma variável.

document.getElementById("p2").innerHTML = x1;

/\*CONST.

Significa constante, algo que nunca muda, logo uma vez declarado uma constante dentro do código ela não vai mudar em lugar nenhum, não importa se está dentro de bloco de código nem nada disso.

\*/

/\*OPERADORES.

Vamos entender Operadores.

-> Os operadores em JS são usados para atribuir valores, comparar valores, executar operações aritméticas e muito mais.

São os sinais que usamos:

+ - \* / = ++ -- += -= && || etc...

São separados em 6 "categorias":

1) OPERADORES ARITIMÉTICOS (matemáticos): + - \* / (++ incremento) (-- decremento)

2) OPERADORES DE ATRIBUIÇÃO: = (+= é a mesma coisa que escrever valor1 = valor1 + valor2 fica valor1 += valor2) e o (-= é o inverso do +=) (/=) (\*=)

3) OPERADORES DE SEQUÊNCIA: concatenação. Texto + Texto, o operador + se torna uma concatenação, pois não é um cálculo matemático.

4) OPERADORES DE COMPARAÇÃO: == === != !== < > <= >=. Retornam true ou false.

5) OPERADOR CONDICIONAL (Ternário):

6) OPERADORES LÓGICOS: (&& = and) (|| = or)

\*/

/\*FUNÇÕES.

Uma função JavaScript é um bloco de código projetado para executar uma tarefa específica.

É como uma pequena "fábrica" onde você faz uma entrada e ele te dá uma saída.

Pode ser encarado como "mini-programas" projetados para fazer uma tarefa que vai contribuir para todo código.

Uma função JavaScript é executada quando "algo" a invoca (chama-a).

\*/

function soma (*valor1*, *valor2*) {

    return valor1 + valor2;

}

let total = soma(20, 30);

document.getElementById("texto").innerHTML = total;

/\*OBJETOS.

Objetos são basicamente variáveis com muitos valores dentro.

Ex: const carro = {marca:"ford", modelo:"ka", ano:2015}

Os valores dentro de um objeto são chamados de prropriedades.

Objetos também podem te métodos. Um Método é uma função colacada dentro de uma prropiedade.

\*/

let carro = "ford"; // Se pensarmos que um carro tem muito mais caracteristicas, do que apenas a marca, então podemos promover esse variável, para um objeto.

let carro1 = {

    //Esse exemplo, é chamado de objeto literal, porque somos nós que estamos definindo as propriedades, mas poderia ser dinâmico, ou seja, vindos do banco de dados.

    marca:"ford",

    modelo:"ka",

    ano:2015,

    cor:"cinza",

    placa:"ABC-1234",

    //Método é uma propriedade que tem, uma função dentro dela.

    buzina: function(){

        //Buzina é a propriedade.

        alert("Biiiiiiiiiiiiiiii");

    },

    completo: function(){

        return alert("A marca é " + *this*.marca + " e o modelo é: " + *this*.modelo)

    }

};

//Exibir uma objeto.

console.log(carro1);

//Exibir uma informação especifica de um objeto.

console.log(carro1.modelo);

//Também pode ser feito assim.

console.log(carro1["modelo"]);

//Acionando uma método.

carro1.completo();

/\*EVENTOS.

Eventos são ações disparadas pela interação dos usuários na página. É o correto manejo desses eventos que torna as páginas interativas e dinâmicas.

Existem muitos eventos. Veja os mais utilizados:

RELACIONADOS AO CLICK.

onclick -> Disparado quando recebe um click.

ondeblclick -> Disparado quando recebe click duplo.

RELACIONADOS AO MOUSE.

onmouseover -> Disparado quando o mouse está sobre.

onmouseout -> Disparado quando o mouse é movido para fora do elemento.

onmousemove -> Disparado quando o mouse é movido dentro do elemento.

onmousedown -> Disparado quando o click do botão foi pressionado.

onmouseup -> Disparado quando o click do botão é liberado.

RELACIONADOS AOS CAMPOS DE TEXTO.

onfocus -> Disparado quando o elemento recebe o foco. Válido para input,

onchange -> Disparado quando existe uma mudança no conteúdo. "Ao mudar".

onblur -> Disparado quando o elemento perde o foco.

RELACIONADOS AS TECLAS.

onkeydown  -> Disparado quando uma tecla é pressionada.

onkeypress  -> Disparado quando uma tecla é pressionada e solta.

onkeyup  -> Disparado quando uma tecla é solta sobre um elemento.

RELACIONADOS A PÁGINA.

onload  -> Disparado quando a página terminou de ser carregada. Body.

onresize  -> Disparado quando há um redimensionamento da janela.

\*/

ONCLICK.

<button *onclick*="eventoClick()">Clique aqui.</button>

function eventoClick() {

    alert("Bem baitola!!!");

}

// ONDBLCLICK.

<button *ondblclick*="eventoDblClick()">Clique aqui.</button>

function eventoDblClick() {

    alert("Receba!!!");

}

<p *id*="p"></p>

    <div *id*="test" *onmousemove*="adicionaTexto()"  *onmouseout*="corOriginal()"  *onmouseover*="vermelho()"  *style*="margin: auto;  width: 50%; height: 200px; background-color: cyan;"></div>

// ONMOUSEOVER.

function vermelho() {

    let div = document.getElementById("test");

    div.style.backgroundColor = "red";

}

//ONMOUSEOUT.

function corOriginal() {

    let div = document.getElementById("test");

    div.style.backgroundColor = "cyan";

}

//ONMOUSEMOVE.

function adicionaTexto() {

    let p = document.getElementById("p");

    p.append(' fala ai meu nobre<br>');

}

//ONMOUSEDOWN.

<body *onmousedown*="alert('Clicou na tela')">

//ONMOUSEUP.

<body *onmouseup*="alert('Clicou na tela')">

/\*ONFOCUS.

Quando clicamos em campo, ele fica com uma borda, então ele recebeu foco/atenção.

\*/

<select *onchange*="mudou()" *id*="test">

        <option *value*="1">Valor 1</option>

        <option *value*="2">Valor 2</option>

        <option *value*="3">Valor 3</option>

        <option *value*="4">Valor 4</option>

    </select>

function mudou() {

    console.log('Mudou');

}

/\*ONBLUR.

É disparado quando o elemento sai do foco.

\*/

 <input *onblur*="vermelho()" *type*="text" *id*="teste">

function vermelho() {

    let input = document.getElementById("teste");

    input.style.backgroundColor = "red";

}

//ONKEYPRESS.

<input *onkeypress*="teclaPress()" *type*="text" *id*="teste">

function teclaPress() {

    let input = document.getElementById("teste").value;

    console.log(input);

}

//ONKEYDOWN.

<input *onkeydown*="teclaPress()" *type*="text" *id*="teste">

function teclaPress() {

    let input = document.getElementById("teste").value;

    console.log(input);

}

//ONKEYUP.

 <input *onkeyup*="teclaPress()" *type*="text" *id*="teste">

function teclaPress() {

    let input = document.getElementById("teste").value;

    console.log(input);

}

/\*ONLOAD.

Dispara quando a página for carregada completamente.

\*/

<body *onload*="alert('Terminou de carregar!')">

/\*ONRESIZE.

Dispara quando há redimensionamento na página

\*/

<body *onresize*="console.log('Houve uma mudança nas dimensões da tela!!')">

/\*ARRAYS.

Os arrays JavaScript são usados para armazenar vários valores em uma única variável. Diferente dos objetos, que funcionam com propriedades ou "nomes" que você dá para os itens dentro dele, os arrays não possuem propriedades. O item dentro dele é encontrado pela posição.

Imagine um array como uma lista de itens, com controle, por ordem de posição dentro dele.

Ex: const lista = ["arroz", "feijão", "macarrão", "leite"];

A lista[0] (lista na posição 0) vai conter o valor de "arroz".

A lista[1] (lista na posição 1) vai conter o valor de "feijão".

E assim por diante.

\*/

const lista = [

    "arroz",

    "feijão",

    "macarrão",

    "leite"

];

//Outra forma de criar um array em JS, seria:

const lista = **new** Array(

    "arroz",

    "feijão",

    "macarrão",

    "leite"

);

//Mas usar o primeiro método é mais simples, por se parecer muito com outras linguagens de programação, como o php.

alert(lista[3]);

const pessoa = ["Jeverson", "Fagundes", 17];

alert(pessoa.length); //Vai mostrar quantos itens tem dentro do array.

alert(pessoa[pessoa.length -1]);// Vamos pegar o último valor do array.

pessoa.push("O melhor!");// vamos adicionar mais um item a lista do array.

//Outra forma de adicionar um item a lista do array seria:

pessoa[pessoa.length] = "Casado";

//Como saber se uma variável é mesmo um array.

Array.isArray(pessoa);// retorna tru ou false

//Métodos/funções para arrays (Matrizes) em JS.

/\*Método JOIN

Com o método join podemos colocar algo a mais entre cada um dos itens do array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método POP.

Com o método pop podemos remover o último item  do array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

pessoa.pop();

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método PUSH.

Com o método push podemos adicionar um item ao array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

pessoa.push("Qualquer coisa!");

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos ite

/\*Método SHIFT.

Com o método shift podemos remover o primeiro item do array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

pessoa.shift();

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método UNSHIFT.

Com o método unshift podemos adicionar um item no primeiro índice do array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

pessoa.unshift("Jeremias");

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método DELETE.

Com o método delete podemos remover um item do array. Embora delete o item naquela posição que foi especificada, aquela posição continuará "se ocupada" por isso não uma boa escolha.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

delete pessoa [0];

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método SPLICE.

Com o método splice podemos adicionar multiplos valores dentro de uma determinada posição dentro do array.

\*/

const pessoa = ["Jeverson", "Fagundes", 17, "Professor de Física"];

//splice(posição, quantos itens queremos que sejam removidos/sobescritos, adicionar itens separando por ,);

pessoa.splice(1, 0, "Item adicionado 1", "Item adicionado 2", "Item adicionado 3");

document.getElementById("teste").innerHTML = pessoa.join(" \* "); // trocou as vírgulas por \*. Troca o separador dos itens do array.

/\*Método CONCAT.

Com o método concat podemos concatenar arrays. Não podemos concatenar em um array que já exista, temos que criar uma nova variável que receba a concatenação.

\*/

const lista1 = ["leite", "macarrão", "carne"];

const lista2 = ["arroz", "café", "erva mate"]

const superLista = lista1.concat(lista2);

document.getElementById("teste").innerHTML = superLista.join(" \_\_ ");

/\*Método SLICE.

Com o método slice podemos "cortar uma parte do array", fazer ele valor a partir de uma determinada parte.

\*/

const jogadores = ["Jogador1", "Jogador2", "Jogador3"];

const craques = jogadores.slice(1);//vai aparecer os jogadores a partir dos jogadores 1 e 2.

document.getElementById("teste").innerHTML = craques.join(" \_\_ ");

const jogadores = [

    "Jogador1",

    "Jogador2",

    "Jogador3",

    "Jogador4",

    "Jogador5",

    "Jogador6",

];

const craques = jogadores.slice(1, 4);//vai aparecer os jogadores a partir dos jogadores 1 até o jogador 4.

/\*Método SORT.

Com o método sort podemos mostrar os itens em ordem alfabética.

\*/

const jogadores = [

    "Jogador1",

    "Jogador2",

    "Jogador3",

    "Jogador4",

    "Jogador5",

    "Jogador6",

];

const jogOrdem = jogadores.sort()

document.getElementById("teste").innerHTML = jogOrdem.join(" \_\_ ");

/\*Método REVERSE.

Com o método reverse podemos mostrar os itens em ordem do último para o primeiro.

\*/

const jogadores = [

    "Jogador1",

    "Jogador2",

    "Jogador3",

    "Jogador4",

    "Jogador5",

    "Jogador6",

];

jogadores.sort()

jogadores.reverse();

document.getElementById("teste").innerHTML = jogadores.join(" \_\_ ");

/\*Método para ordenar os números em ordem crescente.

\*/

const numeros = [

    2,

    5,

    0,

    75,

    100,

    200,

];

numeros.sort(function (*a*, *b*) {

    return a-b;

})

document.getElementById("teste").innerHTML = numeros.join(" \_\_ ");

/\*Método para ordenar os números em ordem decrescente.

\*/

const numeros = [

    2,

    5,

    0,

    75,

    100,

    200,

];

numeros.sort(function (*a*, *b*) {

    return b-a;

})

document.getElementById("teste").innerHTML = numeros.join(" \_\_ ");

/\*Método para pegar o maior número dentro do array.

\*/

const numeros = [

    2,

    5,

    0,

    75,

    100,

    200,

];

document.getElementById("teste").innerHTML = MaiorNumero(numeros);

function MaiorNumero(*array*) {

    return Math.max.apply(null, array);

}

/\*Método para pegar o menor número dentro do array.

\*/

const numeros = [

    2,

    5,

    0,

    75,

    100,

    200,

];

document.getElementById("teste").innerHTML = MenorNumero(numeros);

function MenorNumero(*array*) {

    return Math.min.apply(null, array);

}

/\*Método para pegar números que são maiores que 10 por exemplo.

\*/

const numeros = [

    2,

    5,

    0,

    75,

    100,

    200,

];

const maior10 = numeros.filter(filtragem);

document.getElementById("teste").innerHTML = maior10;

function filtragem(*value*, *index*, *array*) {

    return value > 10;

}

//Pega a hora atual.

    var hora = **new** Date().getHours();

    alert(hora);

var dia = **new** Date().getDay(); // pega a data atual.

var ano = **new** Date().getFullYear();//pega o ano atual.

/\*Estruturas condicionais.

IF e ELSE.

Sem esquema parecidas com o php.

\*/

/\*SWITCH.

É um tipo de declaração comparativa.

É usado para realizar diferentes ações com base em diferentes condições no mesmo bloco de verificação. Caso a condição não seja compatível não será executada e o valor padrão será acionado.

\*/

function verificarCor() {

    let cor = document.getElementById("cor").value;

    cor = cor.toLowerCase();//deixa tudo em minúsculo, pois o switch é muito literal, ou seja, tem que ser exatamente como está não case. Por exemplo, no case está "azul", se digitarmos "Azul", não vai funcionar.

    switch (cor) {

        //caso a cor digitada for azul, então essa condição é satisfeita.

    case "azul":

        //o que acontece.

        document.body.style.backgroundColor = "blue";

        break;

    case "vermelho":

        //o que acontece.

        document.body.style.backgroundColor = "red";

        break;

    case "amarelo":

        //o que acontece.

        document.body.style.backgroundColor = "yellow";

        break;

    default:

     //o que acontece.

     //defaut é o comportamento padrão se nenhum dos case forem atendidos.

     document.getElementById("teste").innerHTML = "Nenhuma cor disponível para: " + cor;

}

}

/\*Laços de repetição.

FOR.

Os laços de repetição oferecem um jeito fácil de executar uma ação repetidas vezes.

Sem esquema parecido com o php.

\*/

/\*EVENTOS DE TEMPO COM JS.

Os eventos de tempo permitem a execução do código em intervalos de tempo específicos. Esses intervalos de tempo são chamados de eventos de cronometragem.

Os dois métodos-chave para usar com JS são:

setTimeout(function, tempo em milisegundos)

-> Excuta uma função, depois de esperar um número especificado de milisegundos.

setInterval(function, milisegundos)

-> É o mesmo que setTimeout(), mas repete a execução da função continuamente.

\*/

//SETTIMEOUT.

<h2 *id*="tempo">0</h2>

   <button *onclick*="ativarContagem()" >Ativar contagem</button><br><br>

function ativarContagem() {

    document.getElementById("tempo").innerHTML = "Começou e contar!";

    //Excuta apenas uma vez a função após o tempo determinado.

    setTimeout(function () {

        document.body.style.backgroundColor = "cyan";

    }, 3000);

}

<h2 *id*="tempo">0</h2>

   <button *onclick*="ativarContagem()" >Ativar contagem</button><br><br>

   <button *onclick*="pararContagem()">Parar contagem</button>

function ativarContagem() {

    document.getElementById("tempo").innerHTML = "Começou e contar!";

    //Excuta apenas uma vez a função após o tempo determinado.

    tempo = setTimeout(function () {

        document.body.style.backgroundColor = "cyan";

    }, 3000);

}

function pararContagem() {

    clearTimeout(tempo);

    document.getElementById("tempo").innerHTML = "Parou a contagem do tempo!";

}

//SETINTERVAL.

<h2 *id*="tempo">0</h2>

   <button *onclick*="ativarContagem()" >Ativar contagem</button><br><br>

   <button *onclick*="pararContagem()">Parar contagem</button>

function ativarContagem() {

    tempo = setInterval(function(){

        var cronometro = document.getElementById("tempo").innerHTML;

        var soma = parseInt(cronometro) + 1;

        document.getElementById("tempo").innerHTML = soma;

    }, 1000);

}

function pararContagem() {

    clearInterval(tempo);

}

<h2 *id*="tempo">40</h2>

   <button *onclick*="ativarContagem()" >Ativar contagem</button><br><br>

   <button *onclick*="pararContagem()">Parar contagem</button>

function ativarContagem() {

    tempo = setInterval(function(){

        var cronometro = document.getElementById("tempo").innerHTML;

        var soma = parseInt(cronometro) - 1;

        if (soma == 0) {

            document.getElementById("tempo").innerHTML = "Tempo esgotado!";

            pararContagem();

        }else{

            document.getElementById("tempo").innerHTML = soma;

        }

    }, 1000);

}

function pararContagem() {

    clearInterval(tempo);

}

class Carro{

//class é uma palavra reservada no JS para criar uma classe, não tem nada a ver com a class em css.

    constructor(*valor1*, *valor2*, *valor3*){

    //o constructor é como se fosse a máquina que vai construir os nossos objetos, que nesse caso são os carros, e dentro () vão por assim dizer as matérias-primas para isso(valor1, valor2,valor3).

*this*.marca = valor1;

*this*.modelo = valor2;

*this*.ano = valor3;

    }

    buzina(){

        return *this*.modelo + " buzinou : Biiiiiiiiiiiiiii";

    }

}

const uno = **new** Carro("Fiat", "Uno",2001);

console.log(uno.buzina());

const uno = **new** Carro("Fiat", "Uno",2001);

//Modificação em uma propriedade.

console.log(uno);

uno.ano = 2014;

console.log(uno);

/\* MANIPULAÇÃO DE DATAS.

\*/

//Comando base para pagar a data.

let data  = **new** Date();

console.log(data);

//Pagar o ano atual com 4 dígitos.

let ano = data.getFullYear();

console.log(ano);

//Pegar o mês atual. De 0 até 11 sendo 0 Janeiro e 11 dezembro.

let mes =  data.getMonth();

//Mostrar o mês no formato escrito.

const mesesDoAno = ["Janeiro", "Fevereiro", "Março", "Abril", "Maio", "Junho", "Julho","Agosto", "Setembro", "Outubro", "Novembro"];

let mesEscrito = mesesDoAno[data.getMonth()];

console.log(mesEscrito);

//Pegar o dia do mês. De 1 até 31.

let diaDoMes = data.getDate();

console.log(diaDoMes);

//Mostrar dia da semana, "segunda-feira", "terça-feira".

let diaDaSemana = data.getDay();

const diasDaSemana = ["Domindo", "Segunda", "Terça", "Quarta", "Quinta", "Sexta", "Sábado"];

let diaSemanaEscrito = diasDaSemana[data.getDay()];

console.log(diaSemanaEscrito);

//Pegar a hora de 0 até 23 horas.

let hora = data.getHours();

console.log(hora);

//Pegar os minutos. De 0 até 59.

let minutos = data.getMinutes();

console.log(minutos);

//Pegar os segundos. De 0 até 59.

let segundos = data.getSeconds();

console.log(segundos);

//Pegar os milisegundos. De 0 até 999.

let milisegundos = data.getMilliseconds();

console.log(milisegundos);

//Como pegar a data no padrão brasileiro. DIA/MÊS/ANO.

let dataBR = data.toLocaleString('pt-BR',);

console.log(dataBR);

//Pegar os valores separadamente e concatenar.

d = **new** Date();

diaMes = d.getDate();

mes = d.getMonth() + 1;

ano = d.getFullYear();

function addZero(*x*) {

    return x < 10 ? '0' + x : '' + x;

}

let dataPadraoBR = addZero(diaMes) + "/" + addZero(mes) + "/" + ano;

console.log(dataPadraoBR);

//Comparar datas. Maior ou menor. EX VENCIMENTOS.

let hoje = **new** Date();

let vencimento = **new** Date(2025, 0, 15);

if (hoje > vencimento) {

    alert("Sua conta está vencida!");

}else{

    alert("Sua conta ainda não venceu!");

}

//Diferença entre duas datas em dias.

let dataInicial = **new** Date();

let dataFinal = **new** Date(2024, 11, 31);

let diferencaTempo = dataFinal.getTime() - dataInicial.getTime();

let diferencaDias = Math.ceil(diferencaTempo / (24 \* 60 \* 60 \* 1000));

console.log(diferencaDias + " dias");

/\*JSON.

JSON significa Javascript Object Notation que traduzido para o português fica algo como Notação de Objeto Javasscript.

Explicando de um modo simples o JSON é basicamente uma forma de converter um objeto em texto e o contrário também, um texto em um objeto.

Ele é usado principalmente para transmitir dados entre sistemas de forma simples, já que o formato de texto é lido por praticamente todas as linguagens de programação.

Para trabalhar com JSON no JS usamos dois métodos:

JSON.parse() -> Converte texto no padrão JSON em objetos.

JSON.stringify() -> Converte objetos em texto padrão JSON.

\*/

<p *id*="teste"></p>

const carro = {

    marca: "Fiat",

    modelo: "Uno",

    ano: 2001

};

let texto = JSON.stringify(carro);

document.getElementById("teste").innerHTML = texto;

let obj = JSON.parse(texto);

console.log(obj.marca);

const carro = {

    marca: "Fiat",

    modelo: "Uno",

    ano: 2001,

    motor: ["1.6", "1.8", "2.0"]

};

let texto = JSON.stringify(carro);

document.getElementById("teste").innerHTML = texto;

let obj = JSON.parse(texto);

console.log(obj.motor[0]);

<p *id*="teste"></p>

const ajax = **new** XMLHttpRequest();

ajax.open('GET', 'https://viacep.com.br/ws/01001000/json/');

ajax.send();

ajax.onload = function () {

    document.getElementById("teste").innerHTML = *this*.responseText;

}

***sweetAlert2.***

São alertas personalizados em JS.

Ainda não entendi muito bem!

Endereço do site: [SweetAlert2 – um substituto bonito, responsivo, personalizável e acessível (WAI-ARIA) para caixas pop-up do JavaScript](https://sweetalert2.github.io/#download)

Ao acessar o site, vamos em ***INSTALAÇÃO***. Ao chegar nesse passo, vamos ter este link:

<script src=”https://cdn.jsdelivr.net/npm/sweetalert2@11”>

</script>

Copiamos apenas o que está dentro das **“”** e antão colamos no navegador. Estando dentro da página no navegador copiamos **TUDO** que está lá dentro e colamos dentro de uma ***página.js***.

Feito tudo isso, vamos instalar na nossa página.

<body>

<button *id*=”btn”>SweetAlert</button>

<script *src*=”js/sweetAlert.js”></script>

<script>

const btnSweetAlert = document.querySelector(‘#btn’);

btnSweetAlert.addEventListener(‘click’, function () {

    Swal.fire({

    title: “Muito de bom!”,

    text: “Click neste botão!”,

    icon: “success”,

    showCancelButton: true

}). Then((*result*) => {

    if (result.isDismissed) {

    }else{

        if (result.isConfirmed) {

            window.location.href = “teste2.php”;

        }

    }

});

})

</script>

</body>

***BOOTSTRAP.***

O bootstrap é um framework que nos auxilia a criar sites responsivos. A responsabilidade do bootstrap é automática.

Link do site: [Bootstrap • A biblioteca HTML, CSS e JS mais popular do mundo. (getbootstrap.com)](https://getbootstrap.com/)

Vamos em **INTRODUÇÃO** ecopiamos o link a seguir:

<head>

    <link *href*=”https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css” *rel*=”stylesheet” *integrity*=”sha384-QWTKZyjpPEjISv5WaRU9OfeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH” *crossorigin*=”anonymous”>

</head>

Depois precisamos copiar e color este link, para que alguns componentes funcionem.

<body>

<script *src*=”https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js” *integrity*=”sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NdzOxhy9GkcIdslK1eN7N6jIeHz” *crossorigin*=”anonymous”></script>

</body>

Após isso vamos em **COMPONENTES** e escolhemos o elemento que queremos estilizar.

***DOMPDF.***

O dompdf é uma ferramenta do **php** que nos permite **imprimir um arquivo em pdf se baseando em html.**

Para instalar precisamos ter o **COMPOSER** instalado na máquina. Depois disso acessamos o link:

[GitHub - dompdf/dompdf: Conversor de HTML para PDF para PHP](https://github.com/dompdf/dompdf)

Após feito isso, temos que acessar o **CMD** e colar o caminho e acessar a pasta onde vamos trabalhar com o dompdf: ***cd C:\wamp64\www\curso\DomPDF***

Estando dentro da pasta, vamos colar o comando:

composer require dompdf/dompdf

Agora vem o arquivo.php com as informações de como usar. Essas são informações que o site nos dá!

<?php

//Incluir o composer.

include("vendor/autoload.php");

//Referenciar o namespace Dompdf.

use *Dompdf\Dompdf*;

//Instanciar e usar o Dompdf.

$dompdf = new *Dompdf*();

//Instânciar o método.

$dompdf->loadHtml('Eu sou o melhor de todos!!!');

//Configurar o tamanho e a orientação do papel.

//landscape - Imprimir no formato paisagem.

$dompdf->setPaper('A4', 'landscape');

//Redenrizar o html como pdf.

$dompdf->render();

//Gerar o pdf.

$dompdf->stream();

?>

**COMO RECEBER OS DADOS DO FORMALÁRIO E GERAR PDF EM PHP.**

**COMO GERAR UM PDF COM AS INFORMAÇÕES DO BANCO DE DADOS.**

**COMO GERAR O PDF COM IMAGENS.**

**COMO ESTILIZAR O PDF GERADO.**

***CRIPTOGRAFIA DE SENHAS COM PHP.***

Criptografar é pegar um dado, uma informação e transformar em algo que **não é humanamente legível**.

Para isso usaremos a função a seguir:

$senha = '12345';

$senhaCriptografada = password\_hash($senha, PASSWORD\_DEFAULT);

echo "$senhaCriptografada";

Cadastro de um usuário.

 include("conecta.php");

            $email = $\_POST['email'];

            $senha = password\_hash($\_POST['senha'], PASSWORD\_DEFAULT);

            $sql = "INSERT INTO teste (email, senha) VALUES ('$email','$senha')";

            mysqli\_query($mysql, $sql);

            if ($mysql->error) {

                echo "Falha ao cadastrar no banco de dados: " . $mysql->error;

            }else {

                header("location: login.php");

            }

        }

    }

Como cadastramos a senha criptografada no banco de dados, então usaremos a seguinte função para poder verificar se a senha digitada bate com a senha cadastrada.

if (password\_verify($senha, $usuario['senha'])) {

                echo "Deu certo e o usuário realizou o seu login com sucesso!";

            }else {

                echo "Falha ao logor, email ou senha incorretos!!";

            }

Verificando se deu certo a comparação das senhas.

 include("conecta.php");

            $email = $\_POST['email'];

            $senha = $\_POST['senha'];

            $sql = "SELECT \* FROM teste WHERE email = '$email'";

            $query = mysqli\_query($mysql, $sql);

            $usuario = $query->fetch\_assoc();

            if (password\_verify($senha, $usuario['senha'])) {

                echo "Deu certo e o usuário realizou o seu login com sucesso!";

            }else {

                echo "Falha ao logor, email ou senha incorretos!!";

            }

        }

    }

***UPLOAD DE ARQUIVOS.***

***REACT.***

O react é uma biblioteca JS desenvolvida pelo facebook , que permite a criação de interfaces de usuário (ui) de forma mais eficiente e escalável, ou seja, usamos react para criar o front-end do projeto, a parte visual.

Não é uma linguagem de programação. A linguagem de programação que usamos para trabalhar com react é o JS ou typescript.

É um padrão de desenvolvimento baseado em componentes.

Link do site para baixar: [Node.js — Run JavaScript Everywhere (nodejs.org)](https://nodejs.org/en)

Para usar o react temos que acessar via cmd a pasta onde vai estar o projeto que vamos criar: ***cd C:\wamp64\www\react***

Após isso acessamos este link: [Vite | Ferramentas front-end de última geração (vitejs.dev)](https://vitejs.dev/)

Vamos em começar, copiamos e colocar este comando no cmd:

***npm create vite@latest projeto1 -- --template react***

***SQL BANCO DE DADOS.***

***COMANDOS SQL.***

**CREATE DATABASE.**

Serve para criar um banco de dados.

Sintaxe:

CREATE DATABASE nome\_do\_bd ;

**CREATE TABLE.**

Serve para criar uma tabela no banco de dados.

Sintaxe:

CREATE TABLE nome\_da\_tabela (

idexemplo integer PRIMARY KEY,

exemplo integer,

quantidade integer

) ENGINE INNODB;

Acima está um exemplo de tabela e dentro dos parênteses estão os nomes das colunas com seu tipo correspondente.

**ALTER TABLE.**

Server para fazer alterações nas colunas de uma tabela no banco de dados.

Sintaxe.

ALTER TABLE nome\_da\_tabela (

XXXXXX

);

**DROP.**

Serve para excluir.

Sintaxe:

DROP TABLE nome\_da\_tabela; (exclui uma tabela)

DROP DATABASE; (exclui o banco de dados)

**ADICIONAR RESTRIÇÃO DE CHAVE ESTRANGEIRA.**

Sintaxe:

ALTER TABLE nome\_da\_tabela ADD CONSTRAINT fk\_nome\_da\_chave

FOREIGN KEY (chave estrangeira)

REFERENCES nome\_da\_tabela\_referênciada (chave primaria dessa tabela);

**INSERT INTO.**

Serve para adicionar informações as colunas de uma tabela no banco de dados.

Sintaxe:

INSERT INTO ’nome\_da\_tabela’ pessoa (

cpf,

endereco,

nome

)VALUES ('123.456.608-62', 'Rua Duque de Caxias', 'Jeremias');

**UPDATE.**

Serve para fazer alterações nas linhas de uma tabela no banco de dados.

Sintaxe:

UPDATE “nome\_da\_tabela”cliente SET nome = 'Verley', endereco = 'tedy', cpf = '123.456.789-62' WHERE cpf = '009. 438.111-78';

Acima eu basicamente disse que, eu quero (update) = modificar linhas na tabela cliente, (set) = nome, endereco, cpf onde (where) =o identificador “chave primária” for cpf ='009. 438.111-78' (where) é como se fosse um condicional, ou seja, o que precisa para que o comando seja executado.

**DELETE.**

Serve para apagar linhas de uma tabela no banco de dados.

Sintaxe:

DELETE FROM cliente WHERE cpf = '211.332.556.-76';

Eu basicamente disse que eu quero, que deletar = (delete) (from) = da tabela cliente (where) = onde cpf = '211.332.556.-76';

**TRUNCATE TEBLE.**

Serve para apagar todas as linhas de uma tabela no banco de dados, mas mantendo sua estrutura.

Sintaxe:

TRUNCATE TABLE nome\_da\_tabela;

**SELECT.**

Serve para fazer consultas no banco de dados.

SELECT <colunas>

FROM <tabelas>

WHERE <condições>

•<colunas>é a lista de colunas que devem ser exibidas pela consulta.

•<tabelas>é a lista de tabelas de onde as colunas serão recuperadas.

•<condições>são as condições que filtram quais linhas serão exibidas.

Sintaxes:

Nesse exemplo que exibir todas as colunas da tabela cidade.

SELECT \* FROM nome\_da\_tabela;

SELECT \* FROM cidade;

Podemos também dar apelidos fictícios a uma tabela n banco de dados.

SELECT cid.cidade\_id, cid.cidade FROM cidade cid;

Nesse caso (cid) é o apelido para a tabela cidade.

**WHERE.**

Caso seja necessário filtrar os dados a serem exibidos pelo select usamos o comando (where).

SELECT \* FROM ator WHERE primeiro\_nome = 'penelope';

Podemos ler assim (SELECT) (\* FROM) (ator) (WHERE) primeiro\_nome = ’Penélope’;

“Selecionar” “todas as colunas da tabela” “onde”

Nesse exemplo selecionamos todas as colunas da tabela ator mas filtramos para exibir apenas se o primeiro nome for Penélope.

**IN.**

SELECT \* FROM ator WHERE primeiro\_nome IN ('joe', 'ed');

IN(‘joe’, ‘ed’) busca na tabela ator todos os atores que tenham seu primeiro nome com os valor (‘joe’, ‘ed’).

**BETWEEN.**

SELECT \* FROM pagamento WHERE valor BETWEEN 2 AND 8;

O comando acima busca na tabela pagamento todos os valores que fiquem entre os valores 2 e 8 reais.

**LIKE.**

SELECT \* FROM ator WHERE primeiro\_nome LIKE 'a%';

LIKE 'a%' = todos os primeiro nomes que iniciem com a letra A;

LIKE '%a' = todos os primeiros nomes que terminem com a letra A;

LIKE '%a%' = todos os primeiros nomes que tenham a letra A em qualquer posição;

LIKE ‘A\_’ = String de dois caracteres que tenham a primeira letra A e o segundo caractere seja qualquer outro;

LIKE ‘\_A’ = String de dois caracteres cujo primeiro caractere seja qualquer um e a última letra seja A;

LIKE ‘\_A\_’ – String de três caracteres cuja segunda letra seja A, independentemente do primeiro ou do último caractere;

LIKE ‘%A\_’ – Todos que tenham a letra A na penúltima posição e a última seja qualquer outro caractere;

LIKE ‘\_A%’ – Todos que tenham a letra A na segunda posição e o primeiro caractere seja qualquer um.

**ORDER BY.**

É o comando para classificar em ordem a colunas a serem exibidas.

SELECT \* FROM pagamento ORDER BY valor;

Por padrão o (ORDER BY) ordena na ordem crescente.

Caso seja necessário ordenar em ordem decrescente colocamos assim:

SELECT \* FROM pagamento ORDER BY valor DESC;

(DESC) decrescente

INNER JOIN.

Inner joins ou junções servem para juntar informações entre tabelas diferentes.

Sintaxe:

SELECT \* FROM aluno a (tabela e o seu apelido).

INNER JOIN entrega\_atividade e

ON (condição de junção) a.id\_aluno = e.id\_aluno;

SELECT a.nome\_aluno, a.email, a.matricula, e.natureza, e.titulo\_certificado, e.carga\_horaria\_certificado, e.certificado, e.carga\_horaria\_aprovada, e.status FROM aluno a

INNER JOIN entrega\_atividade e

ON a.id\_aluno = e.id\_aluno;

***PRONPT DE COMANDOS DO WINDOWS***

***Comandos no cmd***

***1º comando:* cd**

O cd muda o diretório atual.

**Sintaxe:**

**cd “espaço” caminho** *“você quer mudar de diretório, então você tem que passar o caminho, para que você saia do diretório atual e entre no diretório que você quer acessar. Imagine que você quer ir na casa do seu amigo(a), para isso você precisa saber quais as ruas pegar para chegar lá, o mesmo acontece aqui.”*

**Caminho absoluto.**

O caminho absoluto é quando informamos todo o caminho até chegar onde queremos.

Exemplo:

Esse é todo o caminho até chegar na pasta de Donwloads

cd C:\Users\jever\Downloads

**Caminho relativo.**

O caminho relativo é quando não precisamos informar todo o caminho, apenas passamos o local que queremos chegar. Isso funcionado quando o arquivo ou pasta, está dentro do diretório atual em que estamos.

Exemplo:

cd Downloads

**cd ..**

O cd .. volta diretórios.

Exemplo:

Se eu na pasta Downloads cd C:\Users\jever\Downloads e eu quero voltar uma pasta, ou seja, a pasta anterior, poderíamos fazer assim (cd C:\Users\jever) ou assim (cd ..).

**dir**

O dir mostra todo o conteúdo do diretório em que estamos.

C:\>dir