

Treinamento Teknisa

HTML \ CSS \ JS
Rômulo A. Lousada

- I. IDE (VSCODE)
- II. Criando uma página HTML simples
- III. Adicionando lógica com Javascript
- IV. Adicionando estilos com CSS
- V. Ajustes finais

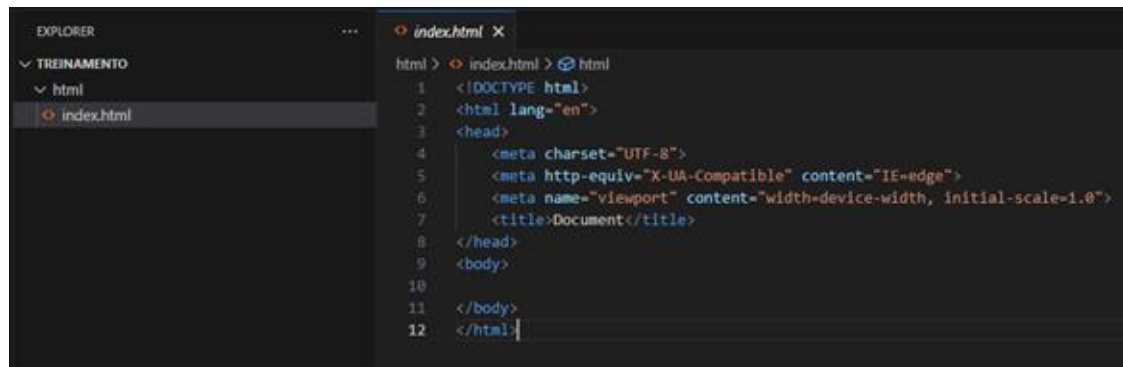
I. IDE (VSCODE)

IDE

VSCODE

IDE é um software criado para facilitar a vida dos programadores.

Ambiente para desenvolvimento com ferramentas que facilitam o processo de desenvolvimento, seja pela agilidade em programar, ou pela possibilidade de debugar o código dentro da própria IDE, assim como também poder verificar por falhas no código, alertar e instruir o desenvolvedor sobre formas para corrigir aquele problema.



The screenshot displays the Visual Studio Code interface. On the left, the 'EXPLORER' sidebar shows a project structure with a folder named 'TREINAMENTO' containing a subfolder 'html'. Inside 'html', the file 'index.html' is selected. The main editor area on the right shows the content of 'index.html', which is a basic HTML document structure. The code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Document</title>
8 </head>
9 <body>
10
11 </body>
12 </html>
```

VSCODE

Extensões

As extensões do Visual Studio Code (VSCode) são recursos adicionais que podem ser instalados para estender e aprimorar a funcionalidade da IDE. As extensões podem ser encontradas e instaladas diretamente na loja de extensões do VSCode.

Uma das extensões populares para o VSCode é a Live Server. Essa extensão é amplamente utilizada por desenvolvedores web, pois fornece uma maneira fácil de criar e executar um servidor local diretamente no editor.

VSCODE

Iniciando

Para dar início ao treinamento, defina um local no seu computador onde você vai salvar os arquivos do projeto que vamos criar.

Neste exemplo, vou criar uma pasta chamada Treinamento em C:

Após isso, ao iniciar o Visual Studio Code, vá em:
Arquivo -> Abrir Pasta...

Por fim, selecione o diretório criado para começar.

II. Criando uma página HTML simples

Criando uma página HTML simples

Para organizar melhor o projeto, crie uma pasta chamada **html** no diretório de Treinamento. Clique com o botão direito no explorador de arquivos e selecione a opção **Nova Pasta...**

Dentro desta pasta, crie um arquivo com o nome e extensão **index.html**. Da mesma forma, basta clicar com o botão direito na pasta **html** e selecionar a opção **Novo Arquivo...**

Abra o arquivo index.html que atualmente está em branco para utilizar de uma das funcionalidades da IDE chamada de Emmet Abbreviation.

Digite **html:5** e a IDE vai criar um esqueleto básico de um documento HTML5

Criando uma página HTML simples

Característica da tag **<head>**:

- Definir informações e configurações sobre o documento HTML.
- Contém elementos que fornecem metadados, como o título da página, links para arquivos de estilo CSS, scripts JavaScript, informações de codificação de caracteres.
- Usado pelos navegadores e mecanismos de busca para entender e exibir corretamente o conteúdo da página.
- Não são visíveis diretamente ao usuário no navegador.

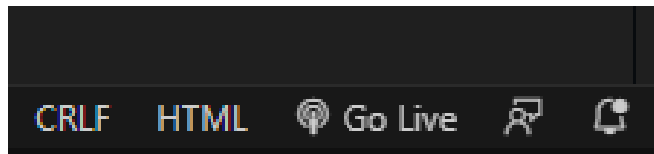
Características da tag **<body>**

- Contém todos os elementos que serão exibidos na janela do navegador, como texto, imagens, vídeos, links, formulários e outros elementos HTML.
- É o que os usuários vão ver e interagir ao visitar a página da web.

Criando uma página HTML simples

Para visualizar a página atual em branco e as eventuais alterações que ela vai sofrer, inicie um servidor local usando o Live Server.

No canto inferior direito da IDE, é possível encontrar a opção para iniciar o Live Server. Clique em **Go Live** e a extensão automaticamente vai subir o servidor na porta padrão 5500.



A página atual está em branco, mas ao começar a fazer alterações no arquivo o Live Server vai atualizar em tempo real esta página, à medida que o arquivo alterado for salvo.

Criando uma página HTML simples

- Dentro do <head>, altere o valor da tag <title> para **Treinamento**
- Dentro da tag <body>, adicione o seguinte trecho:

```
<body>
  <div class="container">
    <div class="header">
      <h3>Validador de CPF</h3>
    </div>
    <div class="body">
      <input type="text" id="cpf"
placeholder="Insira o seu CPF" maxlength="11"/>
      <button id="btValidar">Validar</button>
    </div>
    <div class="footer">
      <span id="result"></span>
    </div>
  </div>
</body>
```

Criando uma página HTML simples

Uma breve explicação das tags utilizadas:

<h> - Usado para criar cabeçalhos ou títulos em uma página da web.

<input> - Cria um campo de entrada interativo, onde os usuários podem inserir dados.

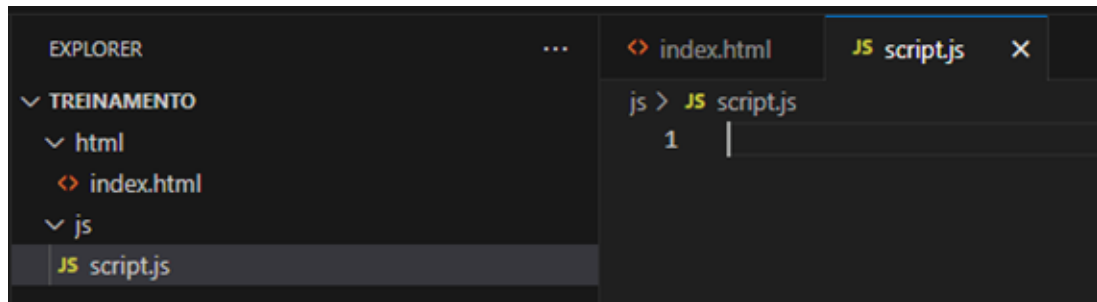
<button> - Cria um botão para interação do usuário e poder disparar eventos.

III. Adicionando lógica com Javascript

Adicionando lógica com Javascript

Para que o validador de cpf seja capaz de ler o que foi inserido pelo usuário e fazer a validação do valor no campo, é necessário utilizar a linguagem Javascript.

Crie uma pasta ao lado da pasta **html** chamada **js** e crie um arquivo com o nome **script.js**



Adicionando lógica com Javascript

É necessário criar um vínculo entre o arquivo html e o arquivo js, para que a lógica implementada no arquivo javascript seja executada na página.

Dentro da tag <head> do arquivo html, faça a importação do arquivo javascript da seguinte forma:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible"
content="IE=edge">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <script src="../../js/script.js" async></script>
  <title>Treinamento</title>
</head>
```

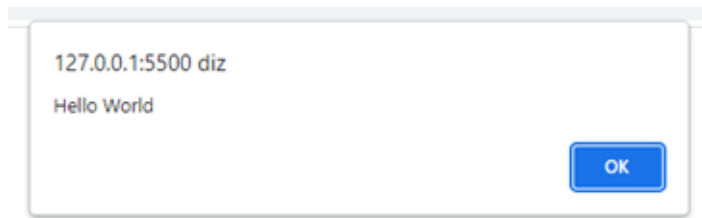
Adicionando lógica com Javascript

A importação do arquivo é feita com o caminho relativo do index.html até o script.js.

Para testar se a importação funcionou como esperado, adicione a seguinte linha de código ao arquivo script.js:

```
alert('Hello World');
```

Feito isso, ao recarregar a página, deve aparecer no topo da tela uma mensagem de alerta com o texto “Hello World”



Adicionando lógica com Javascript

Para não disparar a mensagem de alerta toda vez que a página for carregada, encapsule ele dentro de uma função chamada **validaCPF**.

Essa função deverá ser disparada ao clicar no botão Validar, e para isso, é necessário adicionar o atributo onclick nele, da seguinte forma:

```
function validaCPF() {  
    alert('Hello World');  
}
```

```
<button id="btValidar"  
onclick="validaCPF()">Validar</button>
```


Adicionando lógica com Javascript

Dando início ao procedimento para validar, é necessário buscar o valor informado no campo de CPF.

Logo em seguida, é feita uma verificação rápida para garantir que o valor possui 11 caracteres. Se ele não passar por esta validação, ele informa o usuário sobre o problema na validação e para a execução.

```
function validaCPF() {  
    const cpf = document.getElementById('cpf').value;  
  
    if (cpf.length !== 11) {  
        alert('O CPF informado deve conter 11 dígitos.');        return false;  
    }  
}
```

Adicionando lógica com Javascript

Crie a função `mostraResultado` para aproveitar um trecho de código que provavelmente será necessário em algumas partes do código.

Esta função recebe um texto e uma cor e fará a substituição do conteúdo atual do `span` de resultado no html.

```
function mostraResultado(text, color) {  
  const span = document.getElementById('result');  
  
  span.innerHTML = text;  
  span.style.color = color;  
}
```

Adicionando lógica com Javascript

Outra validação importante para fazer é impedir cpfs que possuem uma repetição do mesmo dígito.

Estes cpfs atendem a validação, mas são considerados inválidos:

111.111.111-11

222.222.222-22

```
function verificarDigitosRepetidos(cpf) {  
  return cpf.split('').every((d) => d !== cpf[0]);  
}
```

Adicionando lógica com Javascript

Esta função é a responsável por fazer a verificação de cada dígito verificador do CPF. É necessário executá-la duas vezes, informando qual dígito verificador do cpf será verificado.

Ela faz o somatório dos demais dígitos do cpf, acumula o resultado em uma variável e faz as operações necessárias, pegando o resto da divisão ao final e comparando com o dígito verificador.

```
function calcularDigitoVerificador(cpf, posicao) {  
  const sequencia = cpf.slice(0, 8 + posicao).split('');  
  
  let soma = 0;  
  let multiplicador = 9 + posicao;  
  
  for (const numero of sequencia) {  
    soma += multiplicador * Number(numero);  
    multiplicador--;  
  }  
  
  const restoDivisao = (soma * 10) % 11;  
  const digito = cpf.slice(8 + posicao, 9 + posicao);  
  
  return restoDivisao == digito;  
}
```

Adicionando lógica com Javascript

Por último, a função principal é incrementada, fazendo a chamada da função que calcula o dígito verificador, uma vez para cada dígito, e ao final verifica o resultado obtido.

CPFs para teste:

25933564052 (Válido)

42050323077 (Válido)

42050323076 (Inválido)

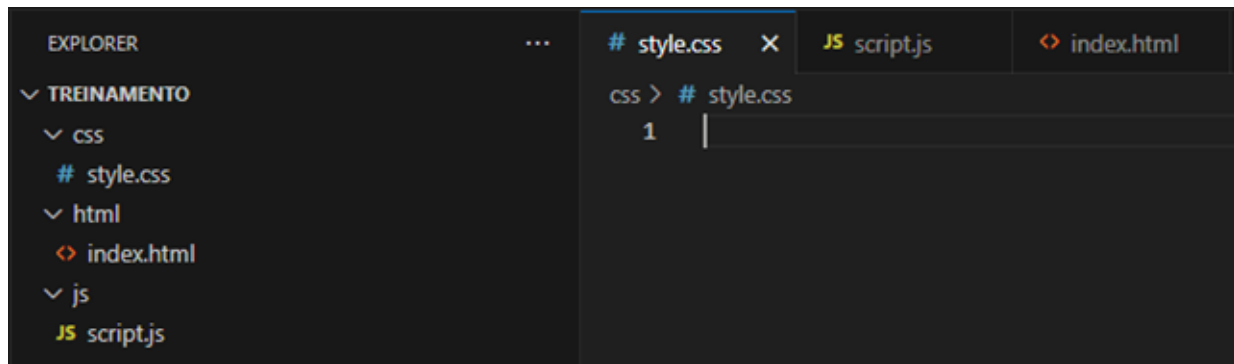
```
function validaCPF() {  
  const cpf = document.getElementById('cpf').value;  
  
  if (cpf.length !== 11) {  
    mostraResultado('CPF deve conter 11 dígitos.', 'red');  
    return false;  
  }  
  
  if (verificarDigitosRepetidos(cpf)) {  
    mostraResultado('CPF não pode conter repetição de dígitos.', 'red');  
    return false;  
  }  
  
  const digito1 = calculaDigitoVerificador(cpf, 1);  
  const digito2 = calculaDigitoVerificador(cpf, 2);  
  
  if (digito1 && digito2) {  
    mostraResultado(`CPF Válido - ${cpf}`, 'green');  
  } else {  
    mostraResultado(`CPF Inválido - ${cpf}`, 'red');  
  }  
}
```

IV. Adicionando estilos com CSS

Adicionando estilo com CSS

Para deixar a página web visualmente mais agradável aos usuários, é necessário incluir uma estilização através de CSS.

Crie uma pasta ao lado da pasta **html** e **js** chamada **css** e crie um arquivo com o nome **style.css**



Adicionando estilo com CSS

Como feito com o js, também é necessário criar um vínculo entre o arquivo html e o arquivo css, para que os estilos implementados no arquivo css sejam aplicados na página.

Dentro da tag <head> do arquivo html, faça a importação do arquivo css da seguinte forma:

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible"
content="IE=edge">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <script src="../../js/script.js" async></script>
  <link rel="stylesheet"
href="../../css/style.css"></style>
  <title>Treinamento</title>
</head>
```


Adicionando estilo com CSS

Para criar uma visualização um pouco mais agradável, vamos centralizar o conteúdo da página.

Para isso, adicione o seguinte código css ao arquivo e salve para ver as modificações.

```
.container {  
    text-align: center;  
}  
  
.body input {  
    padding: 8px;  
    margin-right: 8px;  
}  
  
.body button {  
    padding: 8px 16px;  
}  
  
.footer {  
    margin-top: 25px;  
    font-size: 20px;  
}
```

Adicionando estilo com CSS

Ao final, a exibição da página deve ter ficado assim:

Validador de CPF

CPF deve conter 11 dígitos.

V. Ajustes finais

Ajustes finais

Uma última alteração que é importante abordar é a importação de bibliotecas externas.

Para entender este conceito, importe a biblioteca da jquery e o inputmask, que serão utilizadas para criar uma máscara para o campo de cpf:

Validador de CPF

Validar

Ajustes finais

Além de incluir arquivos javascript locais, a tag script também permite referenciar arquivos disponibilizados na internet.

Para incluir as bibliotecas jquery e o plugin inputmask de maneira remota, é necessário colocar o link onde está disponível o arquivo javascript como referência.

```
<script  
src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3  
.1/jquery.min.js"></script>
```

```
<script  
src="https://cdnjs.cloudflare.com/ajax/libs/jquery.inp  
utmask/3.3.4/jquery.inputmask.bundle.min.js"></script>
```

Ajustes finais

Após a inclusão das bibliotecas externas, inclua no arquivo javascript o seguinte trecho:

É um evento jQuery que é acionado quando o documento HTML é carregado, garantindo que a função dentro do evento apenas seja disparada depois que todos os elementos da página estiverem disponíveis.

```
$(document).ready(function() {  
    $('#cpf').inputmask('999.999.999-99');  
});
```

Ajustes finais

Visto que agora a máscara ocupa alguns caracteres de espaço no input, será necessário aumentar a quantidade máxima de caracteres que podem ser inseridos nele.

Adicionando os pontos e o hífen, aumente para 14 o valor do atributo maxlength.

```
<input type="text" id="cpf" placeholder="Insira o seu  
CPF" maxlength="14"/>
```

Ajustes finais

E por último, é necessário remover toda a formatação do campo antes de repassar para as demais funções que já eram executadas.

Para isso, crie a função que recebe o texto formatado do cpf e retorna apenas os dígitos:

```
function limpaFormatacao(cpf) {  
  cpf = cpf.replace(/\D/g, '');  
  
  return cpf;  
}
```


Ajustes finais

Agora ajuste a função principal para que faça a formatação do valor antes de enviar para as demais funções que vão validar o cpf:

```
function validaCPF() {  
  const cpfFormatado = document.getElementById('cpf').value;  
  const cpf = limpaFormatacao(cpfFormatado);  
  
  if (cpf.length !== 11) {  
    mostraResultado('CPF deve conter 11 dígitos.', 'red');  
    return false;  
  }  
  
  if (verificarDigitosRepetidos(cpf)) {  
    mostraResultado('CPF não pode conter repetição de dígitos.', 'red');  
    return false;  
  }  
  
  const digito1 = calculaDigitoVerificador(cpf, 1);  
  const digito2 = calculaDigitoVerificador(cpf, 2);  
  
  if (digito1 && digito2) {  
    mostraResultado(`CPF Válido - ${cpfFormatado}`, 'green');  
  } else {  
    mostraResultado(`CPF Inválido - ${cpfFormatado}`, 'red');  
  }  
}
```

Dúvidas

