TRABALHANDO COM FORMULÁRIOS

Validando formulários com JS.

Uma das tarefas mais importantes no desenvolvimento web é a validação de dados por meio de formulários.

A validação pode ser feita quando os dados chegarem no servidor, porém, uma das práticas mais comuns para validação é com o uso de JavaScript, tendo em vista que, essa atividade vai ser realizada no navegador do cliente.

Vamos criar um formulário de controle de contatos, e nesse formulário vamos aplicar a programação para que não aceite dados em branco nesse formulário, caso o campo esteja em branco vamos emitir uma mensagem para o cliente, informando que tais informações são necessárias.

Nesse exemplo criado em sala de aula, efetuamos o desenvolvimento de um formulário em HTML e depois Utilizamos o CSS para estilizar o nosso formulário, e por ultimo desenvolvemos uma programação JS.

Segue nossa programação SCRIPT que deve ser incluída na página HTML.

```
  <script>
        function cadastro() {
            event.preventDefault(); // usado aqui para impedir que o
formulário seja enviado automaticamente, dando-nos controle para realizar
a validação personalizada antes de continuar com o envio.

            // Obtém os valores dos campos do formulário
            var nome = document.getElementById('nome').value;
            var telefone = document.getElementById('telefone').value;
            var email = document.getElementById('email').value;

            // Verifica se os campos estão vazios
            if (nome === '') {
                document.getElementById('mensagem-nome').textContent = 'O
campo nome é obrigatório';
                return;
            } else {
                document.getElementById('mensagem-nome').textContent =
'';
            }

            if (telefone === '') {
                document.getElementById('mensagem-telefone').textContent
= 'O campo telefone é obrigatório';
                return;
            } else {
                document.getElementById('mensagem-telefone').textContent
= '';
                formatarTelefone();
            }
```

```
            if (email === '') {
                document.getElementById('mensagem-email').textContent =
'O campo email é obrigatório';
                return;
            } else {
                document.getElementById('mensagem-email').textContent =
'';
            }

            // Se todos os campos estiverem preenchidos, faça o
processamento adicional ou envie o formulário
            // Aqui você pode adicionar o código para processar ou enviar
os dados do formulário
            // Exemplo:

        }

    </script>
```

Lembrando que após todos os arquivos digitados, teremos o seguinte efeito:

Ao clicar para enviar o formulário, se os campos estiverem em branco, uma mensagem deverá aparecer acima da caixa em branco alertando que tal informação é necessária.

EXPRESSÕES REGULARES

Expressões regulares, também conhecidas como regex ou regexp, são padrões de busca utilizados para encontrar e manipular texto em strings. Elas são uma sequência de caracteres que define um padrão de correspondência.

As expressões regulares são compostas por caracteres literais (como letras e números) e metacaracteres, que possuem um significado especial na sintaxe das expressões regulares. Alguns exemplos de metacaracteres são:

. (ponto): Representa qualquer caractere, exceto quebras de linha.

* (asterisco): Indica que o padrão anterior pode ocorrer zero ou mais vezes.

+ (sinal de adição): Indica que o padrão anterior pode ocorrer uma ou mais vezes.

? (interrogação): Indica que o padrão anterior é opcional, podendo ocorrer uma ou zero vezes.

[ ] (colchetes): Define uma classe de caracteres, onde qualquer caractere dentro dos colchetes é uma correspondência válida.

^ (circunflexo): Indica o início da string.

$ (cifrão): Indica o final da string.

As expressões regulares são amplamente utilizadas em linguagens de programação, como JavaScript, para realizar tarefas como validação de entrada, busca e substituição de texto, extração de informações específicas e formatação de dados.

É importante ressaltar que expressões regulares podem se tornar complexas e difíceis de ler ou entender, especialmente quando envolvem padrões mais avançados. Portanto, é recomendado estudar e praticar o uso de expressões regulares para se familiarizar com seus conceitos e aplicá-los de forma eficiente.

Apenas frisando que não é apenas a linguagem JavaScript que pode utilizar o recurso de expressões regulares. Outras linguagens como exemplo: PHP, PYTHON, C#, JAVA, e outras

Vamos dar um exemplo da utilização de expressões regulares na validação de campos de formulários, onde vamos criar máscaras de entrada de dados.

Observação:

Máscara de entrada do campo não é gravada no banco de dados, ela apresenta apenas o efeito visual do determinado campo.

Vejamos abaixo um exemplo de máscara de entrada para CPF.

Lembrando que abaixo temos apenas o código script de tal recurso, é necessário criar um campo de formulário, para entrada de dados do CPF, e sobre esse campo aplicar o recurso de máscara de entrada.

```
<script>
function formatarCPF() {
  let inputCPF = document.getElementById('cpf');
  let valor = inputCPF.value;

  // Remove todos os caracteres que não são números
  valor = valor.replace(/\D/g, '');

  // Adiciona os pontos e o traço conforme necessário
  valor = valor.replace(/(\d{3})(\d)/, '$1.$2');
  valor = valor.replace(/(\d{3})(\d)/, '$1.$2');
  valor = valor.replace(/(\d{3})(\d{1,2})$/, '$1-$2');

  // Atualiza o valor no campo de entrada
  inputCPF.value = valor;
}
</script>
```

CRIANDO ELEMENTOS DINAMICAMENTE

Criar elementos dinamicamente em JavaScript significa criar elementos HTML em tempo de execução, ou seja, durante a execução do seu código JavaScript. Isso é útil quando você deseja adicionar ou modificar elementos na página com base em determinadas ações do usuário ou eventos.

Para criar elementos dinamicamente, você pode usar o objeto document, que representa o documento HTML atual. O document fornece vários métodos que permitem manipular o DOM (Document Object Model), que é a representação em memória da estrutura HTML da página

Aqui está um exemplo básico de como criar um elemento de parágrafo <p> dinamicamente e adicioná-lo ao body da página:

```
    <script>
// Criando o elemento <p>
    var paragrafo = document.createElement("p");
    // Definindo o conteúdo do parágrafo
         paragrafo.textContent    =    "Este    é    o    parágrafo    criado
dinamicamente.<br>";
    // Adicionando o parágrafo ao body da página
    document.body.appendChild(paragrafo);
   </script>
```

Neste exemplo, usamos o método **createElement()** para criar um novo elemento <p>.

Em seguida, definimos o conteúdo desse parágrafo usando a propriedade **textContent.**

Por fim, usamos o método **appendChild()** para adicionar o parágrafo criado ao final do elemento body da página.

Você pode aplicar essa mesma lógica para criar e manipular qualquer elemento HTML, como divs, botões, inputs, etc. Basta substituir o nome do elemento e as propriedades relevantes de acordo com sua necessidade.

Essa capacidade de criar elementos dinamicamente é especialmente útil quando você precisa gerar conteúdo com base em dados obtidos de uma API, por exemplo, ou quando deseja criar interfaces interativas que respondam a eventos do usuário.

Vamos ver essa aplicação numa lista de compras, onde o item Listado será transferido para uma nova lista, onde diversos elementos serão criados.

Vamos práticar