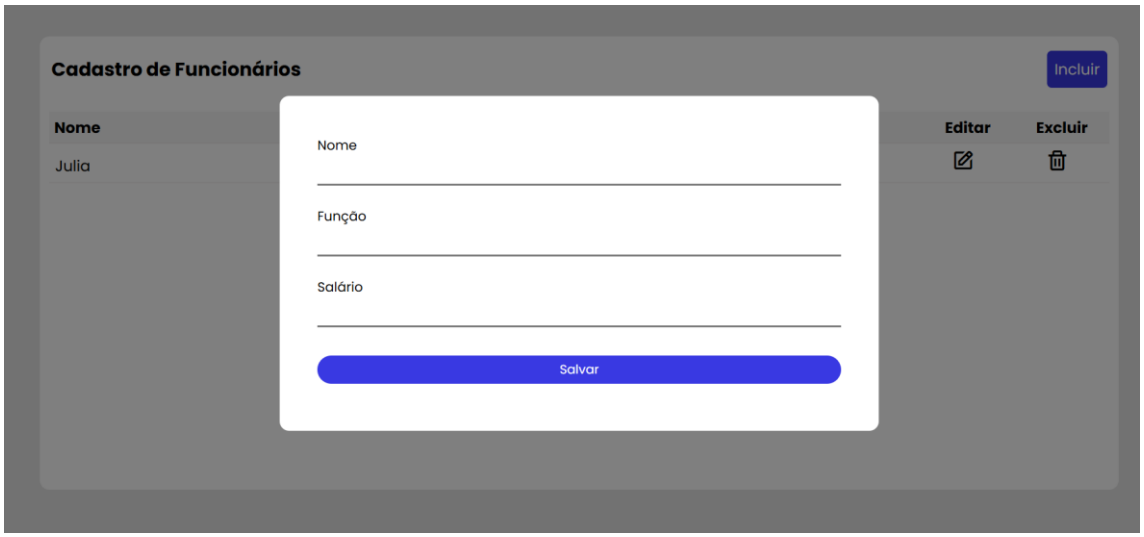
Aula 01 de setembro de 2025



Index.html

```
<!DOCTYPE html>
<html lang="en">

<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">
  <title>CRUD</title>
  <link rel="stylesheet" href="style.css">
  <link href='https://unpkg.com/boxicons@2.1.1/css/boxicons.min.css'
rel='stylesheet'>
</head>

<body>
  <div class="container">
    <div class="header">
      <span>Cadastro de Funcionários</span>
      <button onclick="openModal()" id="new">Incluir</i></button>
    </div>

    <div class="divTable">
      <table>
        <thead>
          <tr>
            <th>Nome</th>
            <th>Função</th>
            <th>Salário</th>
            <th class="acao">Editar</th>
            <th class="acao">Excluir</th>
          </tr>
        </thead>
```

```
        <tbody>
        </tbody>
      </table>
    </div>

    <div class="modal-container">
      <div class="modal">
        <form>
          <label for="m-nome">Nome</label>
          <input id="m-nome" type="text" required />

          <label for="m-funcao">Função</label>
          <input id="m-funcao" type="text" required />

          <label for="m-salario">Salário</label>
          <input id="m-salario" type="number" required />
          <button id="btnSalvar">Salvar</button>
        </form>
      </div>
    </div>

  </div>
  <script src="script.js"></script>
</body>

</html>
```

Sistema Java Script

script.js

```
const modal = document.querySelector('.modal-container')
const tbody = document.querySelector('tbody')
const sNome = document.querySelector('#m-nome')
const sFuncao = document.querySelector('#m-funcao')
const sSalario = document.querySelector('#m-salario')
const btnSalvar = document.querySelector('#btnSalvar')

let itens
let id

function openModal(edit = false, index = 0) {
  modal.classList.add('active')

  modal.onclick = e => {
    if (e.target.className.indexOf('modal-container') !== -1) {
```

```
      modal.classList.remove('active')
    }
  }

  if (edit) {
    sNome.value = itens[index].nome
    sFuncao.value = itens[index].funcao
    sSalario.value = itens[index].salario
    id = index
  } else {
    sNome.value = ''
    sFuncao.value = ''
    sSalario.value = ''
  }

}

function editItem(index) {

  openModal(true, index)
}

function deleteItem(index) {
  itens.splice(index, 1)
  setItensBD()
  loadItens()
}

function insertItem(item, index) {
  let tr = document.createElement('tr')

  tr.innerHTML = `
    <td>${item.nome}</td>
    <td>${item.funcao}</td>
    <td>R$ ${item.salario}</td>
    <td class="acao">
      <button onclick="editItem(${index})"><i class='bx bx-edit'
></i></button>
    </td>
    <td class="acao">
      <button onclick="deleteItem(${index})"><i class='bx bx-
trash'></i></button>
    </td>
  `
  tbody.appendChild(tr)
}

btnSalvar.onclick = e => {
```

```
  if (sNome.value == '' || sFuncao.value == '' || sSalario.value == '')
{
    return
  }

  e.preventDefault();

  if (id !== undefined) {
    itens[id].nome = sNome.value
    itens[id].funcao = sFuncao.value
    itens[id].salario = sSalario.value
  } else {
    itens.push({'nome': sNome.value, 'funcao': sFuncao.value,
'salario': sSalario.value})
  }

  setItensBD()

  modal.classList.remove('active')
  loadItens()
  id = undefined
}

function loadItens() {
  itens = getItensBD()
  tbody.innerHTML = ''
  itens.forEach((item, index) => {
    insertItem(item, index)
  })

}

const getItensBD = () => JSON.parse(localStorage.getItem('dbfunc')) ??
[]
const setItensBD = () => localStorage.setItem('dbfunc',
JSON.stringify(itens))

loadItens()
```

```
const modal = document.querySelector('.modal-container')
```

Aula 01 de setembro de 2025

O código const modal = document.querySelector('.modal-container') está selecionando um **elemento HTML** que possui a classe **CSS "modal-container**" e armazenando-o em uma constante chamada modal.

Essa linha de código é útil quando você deseja interagir com um modal em uma página da web. Modais são elementos que geralmente aparecem sobrepostos ao conteúdo principal da página, exibindo informações adicionais, formulários ou outros elementos interativos.

Por exemplo, se você tiver um modal em seu HTML definido com a classe CSS "modal-container", essa linha de código permite que você acesse e manipule esse modal usando JavaScript. Você pode alterar seu estilo, adicionar ou remover conteúdo, ou controlar seu comportamento (como abrir e fechar) por meio dessa referência modal.

```
function openModal(edit = false, index = 0) {
  modal.classList.add('active')

  modal.onclick = e => {
    if (e.target.className.indexOf('modal-container') !== -1) {
      modal.classList.remove('active')
    }
  }

  if (edit) {
    sNome.value = itens[index].nome
    sFuncao.value = itens[index].funcao
    sSalario.value = itens[index].salario
    id = index
  } else {
    sNome.value = ''
    sFuncao.value = ''
    sSalario.value = ''
  }

}
```

modal.classList.add('active'): Adiciona a classe CSS "active" ao elemento HTML selecionado anteriormente com querySelector('.modal-container'). Geralmente, essa classe é usada para exibir ou ativar o modal, aplicando estilos ou comportamentos específicos para torná-lo visível ou interativo.

modal.onclick = e => { ... }: Define um evento de clique para o elemento modal. Quando o modal é clicado, o código dentro da função de callback é executado.

`if (e.target.className.indexOf('modal-container') !== -1) { ... }:` Verifica se o elemento clicado (representado por `e.target`) tem a classe CSS "modal-

container". Se o elemento clicado for o próprio modal (ou seja, não um de seus elementos filhos), o código dentro do bloco `if` é executado.

`modal.classList.remove('active')`: Remove a classe CSS "active" do elemento modal. Isso geralmente é feito para fechar ou desativar o modal, ocultando-o da visualização ou desativando seu comportamento interativo.

if (edit) { ... } else { ... }: Verifica se o parâmetro edit é verdadeiro ou falso. Se for verdadeiro, significa que o modal está sendo aberto para edição de um item existente. Nesse caso, os campos do modal são preenchidos com as informações do item correspondente no índice index do array itens. Se for falso, significa que o modal está sendo aberto para adicionar um novo item e os campos do modal são limpos.

id = index: Atribui o valor do índice index à variável id, que provavelmente é usada em algum outro lugar do código.

Em resumo, quando a função editItem é chamada com um índice específico, ela abre o modal com o formulário preenchido com as informações desse item, permitindo que o usuário edite esses dados.

```
function deleteItem(index) {
  itens.splice(index, 1)
  setItensBD()
  loadItens()
}
```

A função editItem(index) tem a finalidade de chamar a função openModal com o parâmetro edit definido como true e o índice do item a ser editado passado como argumento.

Vamos analisar o que esse código faz:

function editItem(index) { ... }: Esta é a declaração da função editItem que aceita um parâmetro index, que provavelmente indica o índice do item que será editado.

openModal(true, index): Dentro da função editItem, chama a função openModal passando dois argumentos: true e index. O primeiro argumento, true, indica que o modal será aberto para edição (como visto na função openModal, isso resultará no preenchimento dos campos do modal com as informações do item correspondente ao índice index). O segundo argumento é o índice do item que será editado.

Em resumo, quando a função editItem é chamada com um índice específico, ela abre o modal com o formulário preenchido com as informações desse item, permitindo que o usuário edite esses dados.

Aula 01 de setembro de 2025

```
function insertItem(item, index) {
  let tr = document.createElement('tr')

  tr.innerHTML = `
    <td>${item.nome}</td>
    <td>${item.funcao}</td>
    <td>R$ ${item.salario}</td>
    <td class="acao">
      <button onclick="editItem(${index})"><i class='bx bx-edit'
></i></button>
    </td>
    <td class="acao">
      <button onclick="deleteItem(${index})"><i class='bx bx-
trash'></i></button>
    </td>
  `
  tbody.appendChild(tr)
}
```

let tr = document.createElement('tr'): Cria um novo elemento HTML <tr> (linha da tabela) e o armazena na variável tr.

tr.innerHTML = ...: Define o conteúdo HTML da linha da tabela. Dentro das marcas de crase (), está sendo utilizado o template string, que permite incorporar variáveis e expressões JavaScript diretamente em uma string.

${item.nome}, ${item.funcao}, ${item.salario}: São as propriedades do objeto item que contêm as informações a serem exibidas na tabela.

<button onclick="editItem(${index})">: Cria um botão que, quando clicado, chama a função editItem passando o índice do item como argumento. Isso permite que o usuário edite o item correspondente a esta linha da tabela.

<button onclick="deleteItem(${index})">: Cria um botão que, quando clicado, chama a função deleteItem passando o índice do item como argumento. Isso permite que o usuário exclua o item correspondente a esta linha da tabela.

tbody.appendChild(tr): Adiciona a linha criada à tabela (<tbody>), inserindo-a como um novo filho do elemento tbody.

Resumindo, esta função cria dinamicamente uma nova linha na tabela com base nas informações do objeto item, e inclui botões de edição e exclusão para interação com os itens da tabela.

Resumindo, esta função cria dinamicamente uma nova linha na tabela com base nas informações do objeto item, e inclui botões de edição e exclusão para interação com os itens da tabela.

Aula 01 de setembro de 2025

```
btnSalvar.onclick = e => {

  if (sNome.value == '' || sFuncao.value == '' || sSalario.value == '')
{
    return
  }
```

if (sNome.value == '' || sFuncao.value == '' || sSalario.value == '') { ... }: Esta linha verifica se algum dos campos de entrada (sNome, sFuncao e sSalario) está vazio. Se algum deles estiver vazio, isso significa que não foram fornecidos dados suficientes para salvar o item, então a função para por aí e não executa mais nada.

return: Quando os campos não estão preenchidos corretamente, a função para por aqui, ou seja, ela não executa o restante do código abaixo.

Em resumo, o código dentro do evento de clique do botão btnSalvar verifica se todos os campos necessários estão preenchidos antes de prosseguir com o salvamento dos dados. Se algum campo estiver vazio, a função para de executar e não faz mais nada. Isso ajuda a garantir que o usuário forneça todas as informações necessárias antes de salvar os dados.

```
 e.preventDefault();
```

O comando e.preventDefault() é usado para evitar o comportamento padrão de um evento em JavaScript.

Quando um evento ocorre em um elemento HTML, como um clique em um link (<a>), envio de um formulário (<form>), ou envio de uma tecla em um campo de formulário, o navegador normalmente executa uma ação padrão associada a esse evento. Por exemplo, ao clicar em um link, o navegador redireciona para o URL especificado no atributo href do link.

Uma função de callback é uma função que é passada como argumento para outra função e é executada após um determinado evento ou operação. Em outras palavras, é uma função que é "chamada de volta" (daí o nome) quando uma determinada ação é concluída ou quando ocorre um determinado evento.

As funções de callback são amplamente usadas em JavaScript, especialmente em situações assíncronas, como manipulação de eventos, operações de E/S (entrada/saída) e chamadas de API.

```
if (id !== undefined) {
    itens[id].nome = sNome.value
    itens[id].funcao = sFuncao.value
    itens[id].salario = sSalario.value
  } else {
    itens.push({'nome': sNome.value, 'funcao': sFuncao.value,
'salario': sSalario.value})
  }
```

Esse trecho de código está verificando se a variável `id` não é `undefined`. Se não for `undefined`, isso significa que já existe um item no array `itens` que está sendo editado, então ele atualiza as propriedades desse item com os valores dos campos do formulário. Caso contrário, se `id` for `undefined`, isso significa que um novo item está sendo adicionado ao array `itens`, então um novo objeto é criado com as propriedades nome, função e salário obtidos dos campos do formulário, e esse objeto é adicionado ao final do array `itens` usando o método `push`.

Vamos analisar em detalhes:

1. `if (id !== undefined) { ... }`: Verifica se a variável `id` não é `undefined`. Se `id` for diferente de `undefined`, isso significa que já existe um item no array `itens` que está sendo editado.

   a. `itens[id].nome = sNome.value`: Atualiza a propriedade `nome` do item no índice `id` do array `itens` com o valor do campo `sNome`.

   b. `itens[id].funcao = sFuncao.value`: Atualiza a propriedade `funcao` do item no índice `id` do array `itens` com o valor do campo `sFuncao`.

   c. `itens[id].salario = sSalario.value`: Atualiza a propriedade `salario` do item no índice `id` do array `itens` com o valor do campo `sSalario`.

2. `else { ... }`: Se a condição do `if` não for atendida (ou seja, se `id` for `undefined`), isso significa que um novo item está sendo adicionado ao array `itens`.

   a. `itens.push({'nome': sNome.value, 'funcao': sFuncao.value, 'salario': sSalario.value})`: Adiciona um novo objeto ao array `itens`. Esse objeto tem as propriedades `nome`, `funcao` e `salario` definidas com os

valores obtidos dos campos do formulário `sNome`, `sFuncao` e `sSalario`, respectivamente.

Essencialmente, este código está manipulando a atualização ou adição de itens em um array com base na existência ou não do `id`. Se o `id` já estiver definido, significa que um item existente está sendo editado. Caso contrário, um novo item está sendo adicionado ao array.

```
 setItensBD()
```

O comando setItensBD() provavelmente chama uma função que salva ou atualiza os itens em um banco de dados (BD), comumente usado em aplicações web para armazenar dados de forma permanente.

```
modal.classList.remove('active')
  loadItens()
  id = undefined
}
```

Este bloco de código realiza uma série de ações quando chamado:

modal.classList.remove('active'): Remove a classe "active" do elemento modal. Isso provavelmente faz com que o modal seja ocultado ou desativado na interface do usuário.

loadItens(): Chama uma função chamada loadItens(). Presumivelmente, essa função é responsável por carregar ou recarregar os itens que estão sendo exibidos na interface do usuário. Por exemplo, ela pode recuperar os itens do banco de dados ou atualizar a lista de itens de acordo com as alterações feitas.

id = undefined: Define a variável id como undefined. Isso é comumente usado para limpar o estado da variável id, indicando que nenhum item está atualmente em modo de edição.

Em resumo, este bloco de código é executado quando um modal é fechado. Ele remove a classe "active" do modal, recarrega a lista de itens exibidos na interface do usuário e limpa o estado da variável id, que provavelmente é usada para rastrear o item em edição (ou inserção) quando o modal está aberto.

```
function loadItens() {
  itens = getItensBD()
```

```
  tbody.innerHTML = ''
  itens.forEach((item, index) => {
    insertItem(item, index)
  })

}
```

Essa função loadItens() realiza o carregamento dos itens a partir do banco de dados (assumindo que a função getItensBD() faça isso), limpa o conteúdo atual da tabela (<tbody>) e insere os itens carregados na tabela.

Vamos analisar o que cada linha faz:

itens = getItensBD(): Chama a função getItensBD() para obter os itens do banco de dados e os armazena na variável itens.

tbody.innerHTML = '': Limpa o conteúdo atual da tabela (<tbody>) atribuindo uma string vazia ao seu conteúdo HTML. Isso remove todos os elementos filhos da tabela.

itens.forEach((item, index) => { ... }): Itera sobre todos os itens no array itens. Para cada item, executa uma função de callback.

insertItem(item, index): Chama a função insertItem() para inserir o item na tabela. Esta função provavelmente adiciona uma nova linha à tabela com as informações do item.

Em resumo, a função loadItens() carrega os itens do banco de dados, limpa a tabela e, em seguida, insere os itens carregados na tabela novamente. Isso é útil para atualizar a interface do usuário com os dados mais recentes do banco de dados.

```
const getItensBD = () => JSON.parse(localStorage.getItem('dbfunc')) ??
[]
const setItensBD = () => localStorage.setItem('dbfunc',
JSON.stringify(itens))
```

Esses comandos estão relacionados à persistência de dados usando o armazenamento local (localStorage) do navegador. Eles permitem armazenar e recuperar itens de uma forma simples e eficiente.

Vamos analisar cada comando:

const getItensBD = () => JSON.parse(localStorage.getItem('dbfunc')) ?? []:

localStorage.getItem('dbfunc'): Recupera os dados armazenados no armazenamento local do navegador sob a chave 'dbfunc'.
JSON.parse(...): Converte a string recuperada do armazenamento local em um objeto JavaScript.
?? []: Usa o operador de coalescência nula para retornar um array vazio ([]) caso não haja dados salvos para a chave 'dbfunc'.

Aula 01 de setembro de 2025

const getItensBD = () => ...: Define uma função de flecha (getItensBD) que encapsula essa operação.
Em resumo, esta função getItensBD retorna os itens armazenados no armazenamento local sob a chave 'dbfunc', ou um array vazio se não houver itens armazenados.

const setItensBD = () => localStorage.setItem('dbfunc', JSON.stringify(itens)):

JSON.stringify(itens): Converte o array itens em uma string JSON para ser armazenada no armazenamento local.
localStorage.setItem('dbfunc', ...): Armazena a string JSON no armazenamento local do navegador sob a chave 'dbfunc'.
const setItensBD = () => ...: Define uma função de flecha (setItensBD) que encapsula essa operação.
Em resumo, esta função setItensBD armazena o array itens no armazenamento local sob a chave 'dbfunc', substituindo qualquer dado anteriormente armazenado sob essa chave.

Essas funções getItensBD e setItensBD são úteis para lidar com a persistência de dados no navegador, permitindo armazenar e recuperar itens de forma simples e eficiente. Eles usam o localStorage, que oferece uma maneira fácil de armazenar dados localmente no navegador do usuário.

```
loadItens()
```

O comando loadItens() provavelmente é uma chamada para a função que carrega os itens em algum tipo de interface de usuário ou aplicação.

Baseado no contexto dos comandos anteriores que você forneceu, essa função provavelmente executa as seguintes ações:

Recuperação dos Itens: Chama uma função para obter os itens do banco de dados ou de alguma fonte de dados. Esta função pode ser getItensBD() ou outra função semelhante.

Limpeza da Tabela: Remove todos os itens existentes da interface do usuário, geralmente uma tabela, para preparar a exibição dos itens carregados.

Inserção dos Itens: Para cada item recuperado na etapa 1, insere esse item na interface do usuário. Isso geralmente envolve adicionar linhas em uma tabela HTML ou atualizar outros elementos da interface com os dados dos itens.

Sem ver o código específico da função loadItens(), é difícil fornecer detalhes precisos sobre o que exatamente ela faz. No entanto, com base nos padrões de nomenclatura e na lógica típica de um sistema que manipula itens, os pontos acima são comuns em uma função com esse nome.

Aula 01 de setembro de 2025