**1º Passo**

**Foi criado uma função ligada a um evento onclick para recuperar o valor dos campos que serão preenchidos.**

**function cadastrarDespesa(){**

**let ano = document.getElementById('ano')**

**let mes = document.getElementById('mes')**

**let dia = document.getElementById('dia')**

**let tipo = document.getElementById('tipo')**

**let descricao = document.getElementById('descricao')**

**let valor = document.getElementById('valor')**

**console.log(ano.value, mes.value, dia.value, tipo.value, descricao.value, valor.value)**

**}**

**2º Passo**

**Foi criado um objeto de classe que recebe como parâmetro os valores passados pela função**

**class Despesa {**

**constructor(ano, mes, dia, tipo, descricao, valor){**

**this.ano = ano**

**this.mes = mes**

**this.dia = dia**

**this.tipo = tipo**

**this.descricao = descricao**

**this.valor = valor**

**}**

**}**

**function cadastrarDespesa(){**

**let ano = document.getElementById('ano')**

**let mes = document.getElementById('mes')**

**let dia = document.getElementById('dia')**

**let tipo = document.getElementById('tipo')**

**let descricao = document.getElementById('descricao')**

**let valor = document.getElementById('valor')**

**let despesa = new Despesa(ano.value, mes.value, dia.value, tipo.value, descricao.value, valor.value)**

**console.log(despesa)**

**}**

**3º Passo**

**Foi criado uma função para gravar as informações do objeto em localStorage, e inserido a um start para está função dentro da função “cadastrarDespesa” já criada antes.**

**function cadastrarDespesa(){**

**let ano = document.getElementById('ano')**

**let mes = document.getElementById('mes')**

**let dia = document.getElementById('dia')**

**let tipo = document.getElementById('tipo')**

**let descricao = document.getElementById('descricao')**

**let valor = document.getElementById('valor')**

**let despesa = new Despesa(ano.value, mes.value, dia.value, tipo.value, descricao.value, valor.value)**

**gravar(despesa)**

**}**

**function gravar(d){**

**localStorage.setItem('despesa', JSON.stringify(d))**

**}**

**Como usamos o recurso setItem, toda vez que enviamos dados, eles sobrepõem um ao outro ao invés de adicionar mais um, devemos mudar isso com a inserção de uma lógica.**

**4º Iremos**

**Iremos implementar um objeto Bd que recebe a função gravar agora, e inserimos um índice dinamico**

**class Bd{**

**constructor() {**

**let id = localStorage.getItem('id')**

**if(id === null) {**

**localStorage.setItem('id', 0)**

**}**

**}**

**getProximoId() {**

**let proximoId = localStorage.getItem('id')**

**return parseInt(proximoId) + 1**

**}**

**gravar(d) {**

**let id = this.getProximoId()**

**localStorage.setItem(id, JSON.stringify(d))**

**localStorage.setItem('id', id)**

**}**

**}**

**let bd = new Bd()**

**Foi construído uma lógica de gravação onde se recebe uma lógica de indexação de id e um código onde se converte strings para JSON e após isso atualiza. Depois disso foi criado a lógica de indexação “getProximoId”**

**I – Retorna os índices de um determinado Array() ou os atribustos de um determinado objeto.**

**5º Iremos implementar uma lógica para validação das informações inseridas**

**Foi inserido dentro da Class “Despesa” uma função que contem uma lógica para identificar se dentre as informações enviadas pelo usuário contem algum campo undefined, null ou vazio.**

**class Despesa{**

**constructor(ano, mes, dia, tipo, descricao, valor){**

**this.ano = ano**

**this.mes = mes**

**this.dia = dia**

**this.tipo = dia**

**this.descricao = descricao**

**this.valor = valor**

**}**

**validarDados(){**

**for(let i in this){**

**if(this[i] == undefined || this[i] == '' || this[i] == null){**

**return false**

**}**

**}**

**return true**

**}**

**}**

**Dentro da função “cadastrarDespesas()” foi inserido um condicional para retornar informação para o usuário se a operação foi completa ou não, porem por enquanto está funcionando como debug.**

**function cadastrarDespesa() {**

**let ano = document.getElementById('ano')**

**let mes = document.getElementById('mes')**

**let dia = document.getElementById('dia')**

**let tipo = document.getElementById('tipo')**

**let descricao = document.getElementById('descricao')**

**let valor = document.getElementById('valor')**

**let despesa = new Despesa(ano.value, mes.value, dia.value, tipo.value, descricao.value, valor.value)**

**despesa.validarDados()**

**if(despesa.validarDados()){**

**//bd.gravar(despesa)**

**//Dialogo de Sucesso**

**console.log('Dados válidos')**

**}else{**

**//Dialogo de Erro**

**console.log('Dados invalidos')**

**}**

**}**

**5º Iremos implementar um aviso**

**Continuaremos o passo anterior que estava em fase de Debug de criar uma condicional, e nela faremos um incremento de inserir um aviso caso os dados estejam incompletos e caso os dados sejam armazenados com sucesso.**

**if(despesa.validarDados()){**

**bd.gravar(despesa)**

**$('#sucessoGravacao').modal('show')**

**}else{**

**//Dialogo de Erro**

**$('#erroGravacao').modal('show')**

**}**

**Esses Id que são chamados são Modais que foram inseridos no arquivo Html**

**6º Iremos modificar os modais inseridos na página HTML**

**O novo modal vai funcionar com uma lógica onde se a operação for sucesso terá um conteúdo especifico e se a operação der erro terá outro conteúdo.**

**Foi adicionado o conceito de identificador único para que ao seguir uma lógica pré-estipulada, seja inserido textos ou até mesmo class especificas no elemento HTML.**

**if(despesa.validarDados()){**

**bd.gravar(despesa)**

**document.getElementById('modal\_titulo').innerHTML = 'Registro Incluido com Sucesso'**

**document.getElementById('modal\_titulo\_div').className = 'modal-header text-sucess'**

**document.getElementById('modal\_conteudo').innerHTML = 'Despesa foi cadastrada com sucesso'**

**document.getElementById('modal\_btn').innerHTML = 'Voltar'**

**document.getElementById('modal\_btn').className = 'btn btn-success'**

**$('#modalRegistraDespesa').modal('show')**

**}else{**

**//Dialogo de Erro**

**document.getElementById('modal\_titulo').innerHTML = 'Erro na inclusão do registro'**

**document.getElementById('modal\_titulo\_div').className = 'modal-header text-danger'**

**document.getElementById('modal\_conteudo').innerHTML = 'Erro na gravação, verifique se todos os campos foram preenchidos com sucesso'**

**document.getElementById('modal\_btn').innerHTML = 'Voltar e Corrigir'**

**document.getElementById('modal\_btn').className = 'btn btn-danger'**

**$('#modalRegistraDespesa').modal('show')**

**}**

**Foi realizado para que as modificações não sejam feitas de modo hard-code, é importante que haja uma reutilização, para que não seja necessário o retrabalho.**

**7º Iremos começar a construir uma lógica para que possamos ver as despesas cadastradas em localStorage.**

**Foi criado uma função chamada “carregaListaDespesas()” que é rodada toda vez que a página de consulta é rodada, essa função por sua vez chama uma outra função que se encontra dentro da class Bd.**

**recuperarTodosRegistros(){**

**let id = localStorage.getItem('id')**

**//recuperar todas as despesas cadastradas em localStorage**

**for(let i = 1; i<= id; i++){**

**let despesa = JSON.parse(localStorage.getItem(i))**

**console.log(i, despesa)**

**}**

**A função ira pegar os id que contem em localStorage, após ele ira usar uma estrutura de repetição para armazenar os id, dentro da mesma estrutura de repetição é recuperados as informações do localStorage e convertidos de JSON para objetos literais. Foi inserido também a criação de uma Array() para armazenar as informações e uma lógica para que não seja armazenado índices nulos/escluidos.**

**recuperarTodosRegistros(){**

**//array de despesas**

**let despesas = Array()**

**let id = localStorage.getItem('id')**

**//recuperar todas as despesas cadastradas em localStorage**

**for(let i = 1; i<= id; i++){**

**let despesa = JSON.parse(localStorage.getItem(i))**

**//existe a possibilidades de ter itens que foram removidos??**

**if(despesa === null){**

**continue**

**}**

**despesas.push(despesa)**

**}**

**return despesas**

**}**

**function carregaListaDespesas(){**

**//Aqui criamos uma variavel chamada despesas que recebe as informações que vem da função**

**let despesas = Array()**

**despesas = bd.recuperarTodosRegistros()**

**console.log(despesas)**

**}**

**8º Iremos pegar as informações dos itens que já estão registrados no Array() e vamos passar isso para a tabela HTML de forma dinamica.**

**function carregaListaDespesas(){**

**//Aqui criamos uma variavel chamada despesas que recebe as informações que vem da função**

**let despesas = Array()**

**despesas = bd.recuperarTodosRegistros()**

**console.log(despesas)**

**//selecionando o elemento Tbody da tabela**

**var listaDespesas = document.getElementById('listaDespesas')**

**/\***

**<tr>**

**<td>15/03/2018</td>**

**<td>Alimentação</td>**

**<td>Compras</td>**

**<td>500</td>**

**</tr>**

**\*/**

**//percorrer o Array() despesas, listando cada despesa de forma dinamica**

**despesas.forEach(function(d){**

**console.log(d)**

**//criando as linhas <tr>**

**let linha = listaDespesas.insertRow()**

**//cria as colunas <td>**

**linha.insertCell(0).innerHTML = `${d.dia}/${d.mes}/${d.ano}`**

**//ajustar o tipo**

**switch(d.tipo){**

**case '1': d.tipo = 'Alimentação'**

**break**

**case '2': d.tipo = 'Educação'**

**break**

**case '3': d.tipo = 'Lazer'**

**break**

**case '4': d.tipo = 'Saúde'**

**break**

**case '5': d.tipo = 'Transporte'**

**break**

**}**

**linha.insertCell(1).innerHTML = d.tipo**

**linha.insertCell(2).innerHTML = d.descricao**

**linha.insertCell(3).innerHTML = d.valor**

**})**

**}**

**9º Iremos complementar a lógica de registro de sucesso para que ao armazenar com êxito, as caixas de inserção se limpem sozinhas para que não haja necessidade de fazer manualmente.**

**ano.value = ''**

**mes.value = ''**

**dia .value = ''**

**tipo.value = ''**

**descricao.value = ''**

**valor.value = ''**

**//Formas de armazenamento de Dados em Browers**

**Storage**

**-Local Storage – Ficam armazenados no navegador**

**-Session Storage - Ficam armazenados no navegador enquanto o brower estiver ativo;**

**-IndexedDB**

**-Web SQL**

**-Cookies - funcionam como dados que são armazenados do lado do cliente, de modo que o servidor possa extrair e trabalhar com esses dados;**

**-JSON.parse() – converte de Json para objeto literal**

**-JSON.stringify() – converte de objeto literal para JSON**