Como funciona a aplicação?

No main.js file a gente importa uma função, que se chama FavoritesView, e logo após, criamos um objeto com o new FavoritesView(valor)

import {FavoritesView} from "./favorites.js"

**new** FavoritesView('#app')

Então, iremos inicializar o objeto FavoritesView com a div do HTML que tem ID #APP

A FavoritesView irá ter, até então, 4 principais métodos

1 – MÉTODO CONSTRUTOR

O método construtor é um tipo especial de método que vai obrigar o usuário a colocar as informações (Que eu coloquei como valor) dentro da classe para ser inicializada.

constructor(*root*) { //constructor da classe favoritesview

*super*(*root*)

*this*.tbody = document.querySelector('table tbody') //LINHA 3

*this*.update() //LINHA4

 }

Este método, que no momento está com o construtor com a root sendo #app, realizará um super( ), essa keyword irá chamar o construtor da classe Favorites

constructor(*root*) { //constructor da classe favorites

*this*.root = document.querySelector(*root*)

*this*.load()

    }

Perceba que uma variável está sendo criada, chamada de this.root que será a DOM do #app

. Neste momento, this.root é a div #app.

Além disso, foi inicializada o método load(), que é descrito a seguir:

 load() {

      const entries=

      [

         {

           login: 'kinhoreis2000',

           name: 'Luiz Henrique',

           public\_repos:'76',

           followers: '120000'

         },

         {

           login: 'kinhoreis2000',

           name: 'Luiz Henrique',

           public\_repos:'76',

           followers: '120000'

         }

       ]

*this*.entries = entries

    }

No momento, o método load() está criando uma constante que se chama entries e colocando um objeto dentro do array com os dados de cada usuário. Então, é criada uma variável this.entries que contenha tudo que está em entries

Com isso, voltaremos a LINHA3 com o construtor da Favoritesview que irá, neste momento, criar uma variável que se chama this.tbody que será o seletor da DOM do tbody da tabela do HTML. E irá rodar o this.update() que é o método que descreveremos a seguir

2- MÉTODO UPDATE

update() {

*this*.removeAllTr()

*this*.entries.forEach(*user* => {const row = *this*.createRow()

      row.querySelector('.user img').src = `https://github.com/${user.login}.png`

      row.querySelector('.user img').alt = `Imagem de ${*user*.name}`

      row.querySelector('.User-name').textContent = *user*.name

      row.querySelector('.repo').textContent = *user*.public\_repos

*this*.tbody.append(row)

    })

O método update() irá começar inicializando um método removeAllTr() que será explicado no (4 MÉTODO REMOVEALLTR)

Após isso, ele irá pegar a this.entries que foi criada no construtor da classe Favorites, e para cada elemento uma função com parâmetro user vai rodar. Esta função irá criar uma constante que se chama ROW, sendo ela um this.createRow() (3 – MÉTODOS CREATE ROW). A constante row é um tr que contém um HTML interno.

A partir daí, é alterado o HTML interno do tr é alterado utilizando várias propriedades diferentes.

3- MÉTODO CREATEROW

Esse método será para criar linhas das tabelas

Primeiro, uma constante chamda ‘tr’ é criada e, com a DOM, é criado um elemento HTML de div ‘tr’ pra ela. Após isso, é alterado o HTML de dentro da tr com a propriedade innerHTML.

    createRow() {

      const tr = document.createElement('tr')

      tr.innerHTML = `<td class="user">

          <img src="https://github.com/kinhoreis2000.png" alt="Imagem do henrique">

          <a href="https://github.com/kinhoreis2000" target="\_blank">

          <p class="User-name" >Luiz Henrique Reis</p>

          </a>

        </td>

        <td class="repo">

          76

        </td>

        <td class="year">

          2011-2019

        </td>

        <td>

          <button class="btn-remove">remove</button>

        </td>`

      return tr

 }

O método createRow irá retornar o tr criado, já com o innerHTML inserido

4-MÉTODO REMOVE ALL TR

Este método será responsável por retirar todos os TR da tabela, ele será descrito da seguinte forma:

*this*.tbody.querySelectorAll('tr')

      .forEach(

        function (*tr*) {

*tr*.remove()

        }

        )

Primeiro ele vai selecionar todos os ‘tr’ do this.tbody criado no construtor. Então, para cada (método forEach) TR selecionado, uma função que tem como parâmetro **tr** (que neste momento é o tr HTML da tabela) irá ser removida pelo método remove()