**jQuery**

**Introdução**

Biblioteca JavaScript com a proposta de: escreva menos e faça mais;

Recursos:

Seleção e manipulação de elementos HTML,

Navegação pelo DOM,

Manipulação de CSS,

Eventos: click sobre botões, entrada de texto no input,

Efeitos e animações: efeitos já prontos; ex.: ocultar ou exibir elemento HTML,

Ajax: realização de requisições assíncronas,

Plugins: podem ser reaproveitados dentro das aplicações; por exemplo: ao invés de criar uma galeria de imagens, pode ser utilizado um plugin que já tem essa finalidade.

**Configurando e testando o jQuery**

No terminal: **npm i jquery;**

**jquery.com > https://code.jquery.com/jquery-3.7.1.min.js(salvar link no projeto em que o jquery será utilizado).**

**CDN** – conectar com o link do j Query**:**

[https://releases.jquery.com](https://releases.jquery.com/) > minified > CTRL + C > Copia na tag script

**Seletores – Selecionando elementos HTML parte 1**

Dependendo do objetivo para modificar elemento, eles precisam ser selecionados.

Selecionando elementos por tag - **$(tag)**

// // // classe - **$(.classe)**

// // // id - **$(#id)**

**$** - permite selecionar elementos, acesso aos recursos da biblioteca jQuery;

- permite executar ações da biblioteca jQuery;

- permite o acesso aos recursos da biblioteca jQuery.

**Seletores – Selecionando elementos HTML parte 2**

Praticando a seleção de elementos de um modo um pouco mais avançado.

**-** Combinação de seletores do jQuery;

**-** Como atuar sobre o array de retorno da seleção de elemento.

  //selecionar elementos por classe de tag específica

      console.log($("li.exemplo1"));

retorna a classe **exemplo1** quando ela estiver dentro de um elemento **li**, caso exista **mais** **de** **uma li** com a mesma classe, **retorna** também;

      //selecionar elementos por classe de classe específica

      console.log($(".lista1 .exemplo2"));

retorna os elementos de **qualquer tag** **com** a classe **lista1** que **possui elementos** **com** a **classe** **exemplo2**

      console.log($(".lista1 div.exemplo2"));

retorna os elementos de **qualquer tag com** a classe **lista1** que **possui** elementos **div com** a **classe exemplo2**

//atuação sobre o array de retorno dos elementos selecionados

**:first 🡪** retorna primeiro elemento da seleção

**:last 🡪** retorna último elemento da seleção

**:even 🡪** retorna elementos contidos nos índices pares

**:odd 🡪** retorna elementos contidos nos índices ímpares

Alguns elementos que permite **reaplicar filtro** **no retorno de elementos já filtrados**.

**Sequência de execução de código e $(document).ready()**

*Formas tradicionais com JS puro*

<body onload =”funcao()” />

windows.onload = function() { ... }

*Utilizando jQuery*

$(document).ready(function() { ... })

$(function() { ... })

- Essas, são 2 formas de implementar o controle do momento em que o código JS entra em ação.

$(document).ready(function () {

        //'construi' código javascript no momento em que a página está carregada

        function teste() {

          console.log($("#exemplo"));

        }});

      $(teste); //nao foi colocado o parênteses após a função, pois não será passado nenhum parâmetro, a função está apenas sendo referenciada

$(function () {

        //passando direto a função como parâmetro

        console.log($("#exemplo"));

      });

**Recuperando e manipulando os atributos dos elementos HTML**

*<tag attr1=”alguma coisa” attr2=”alguma coisa”></tag>*

**attr – atributos**

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

recuperando atributo ‘**src**’ da tag **img.**

- Um segundo valor, é um valor que iremos atribuir para aquele atributo

Forma

Descrição gerada automaticamente com confiança média

Nesse exemplo, estava sendo exibida a imagem1 e mudamos para imagem2.

        $("div").attr("style", "background-color:blue", "width:100");//recuperando atributo style da tag div e atribuindo 2 valores a ela

**Recuperando e manipulando o conteúdo interno de elementos HTML**

- Elemento interno, diferente de value

- .html 🡪 método que define ou retorna o conteúdo dos elementos selecionados

*<tag attr=”alguma\_coisa”></tag>*

*<tag attr=”alguma\_coisa”>... conteúdo inerno ...</tag>*

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

Modificando ***valor* *interno***da div1.

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

**Tags** também **podem ser adicionadas**, nesse exemplo é a tag strong.

Forma

Descrição gerada automaticamente

Adicionando style.

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

Método **text** recebe string e mesmo que ela contenha uma tag HTML válida, **atribui o texto puro**.

**Recuperando e manipulando valores de inputs (text) e selects**

.val() – método que definir ou obtém valor de elementos de formulário

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

Recuperando valor por id

Padrão do plano de fundo

Descrição gerada automaticamente com confiança baixa

Modificando valor pelo Id

**Recuperando e manipulando valores checkboxs/radios e Loop com $.each()**

Forma

Descrição gerada automaticamente

Recuperando classe ‘sexo’ que está dentro do input 🡪 ‘Feminino’;

Recuperando valor checked (verificado, que foi selecionado) da classe ‘sexo’.

Texto

Descrição gerada automaticamenterecuperando valor contido dentro do elemento ‘radio’ que está checked.

Para um melhor entendimento sobreo valor (value) e nome (name) do elemento:

Tela preta com letras brancas

Descrição gerada automaticamente

Recuperando elementos que contém a classe ‘interesse’:

Texto

Descrição gerada automaticamente

Recuperando apenas os elementos que contém a classe ‘interesse’ que estão selecionados:

Texto

Descrição gerada automaticamente

A diferença é que enquanto o **radio só permite** a seleçãode **uma opção** do conjunto, o **checkbox permite** a seleção **de 1 ou mais opções**.

Caso haja a tentativa de retornar **valores do checkbox**, **só retorna o primeiro**:



Método $.each() retorna 1 array e 1 função de call-back e o índice e o valor de cada elemento do array

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenterecuperando diretamente o atributo value.

**Inserindo e removendo elementos HTML**

Feito dentro de:

      $(document).ready(() => {});

**Inserindo** como último item da ‘lista1’ 🡪 ‘li – Item 4’

      $("#lista1").append("<li>Item 4</li>");

**Inserindo** como último item da ‘lista1’ 🡪 ‘li – Item 0’

        $("#lista1").prepend("<li>Item 0</li>");

**Inserindo** como 1° item da lista 2 uma li com o conteúdo interno ‘Item novo’

          $("#lista2").prepend("<li>Item novo</ li>");

A diferença entre after() e prepend(), é que os elementos adicionados com o **.after** estão no **mesmo nível** hierárquico **que o elemento alvo**, já os elementos adicionados com **.prepend** é ‘**filho’ do elemento alvo**.

Ex.:

É adicionado um elemento HTML com o **.after**;

Esse elemento será adicionado após a tag que contém a classe utilizada.

Se esse elemento é adicionado com o **.prepend**;

Esse elemento ficará após os elementos já existentes, porém, dentro da tag que contém a classe.

Diagrama

Descrição gerada automaticamente

        $("#lista2").remove(); //removendo item

**Navegando entre elementos HTML**

**parent ()** 🡪 elemento pai

Texto

Descrição gerada automaticamenterecuperando elemento pai da tag que contém a classe ‘secao1’:

<div id="pagina">

**closest ()** 🡪 procura por elementos pais

seleciona o primeiro elemento que ele encontra debaixo para cima

Texto

Descrição gerada automaticamenteretorna 1° elemento pai de ‘.subSecao’ que contém o Id ‘#pagina’.

Texto

Descrição gerada automaticamenteacessando elementos pai de ‘.item’ que tem a tag ‘ul’.

Diferença entre **parent()** e **closest()**, é que enquanto closest procura até achar a referência solicitada, parent vai direto no elemento pai.

**find ()** 🡪 procura por elementos filhos

Texto

Descrição gerada automaticamenterecupera elementos pais de ‘#topo’ que contém a tag ‘li’.

Texto

Descrição gerada automaticamente recupera elementos pais de ‘#rodape’ que contém a tag ‘p’.

Texto

Descrição gerada automaticamente’secao1’ sobe um nível e recupera elemento filho ‘h1’

Texto

Descrição gerada automaticamenteseleciona elemento pai de ’secao1’ que contém ‘#pagina’ e seleciona elemento filho ‘h1’

**Manipulando CSS**

- Manipular estilos visuais dos elementos HTML

- .css() 🡪 adiciona propriedades de estilo inline:

**$(‘<elemento-estilizado>’).css(‘atributo-estilo’, ‘valor’)**

Essa **estilização pode ser feita** através do ***id****,* ***class*** e até mesmo a***tag.***

Ex.: estilizando topo



Adicionando mais de um atributo de estilo:

Texto

Descrição gerada automaticamente

Adicionando mais de um valor no mesmo método:

Basta adicionar um objeto literal: $(‘’).css(**{**‘ ’**:**’ ’**,** ‘ ’**:**’ ’**}**);

Texto

Descrição gerada automaticamente

modificando item através da tag ‘h1’

.addClass – adiciona uma classe

adicionando classe ‘.campo’ aos inputs

Texto

Descrição gerada automaticamenteatribuindo 2 classes de uma só vez

hasClass() 🡪 verifica a existência de uma classe no elemento html

Interface gráfica do usuário, Texto

Descrição gerada automaticamenteverificando se dentro de ‘textarea’, existe as classes ‘campo’ e ‘erro’

removeClass() 🡪 remove classes

Texto

Descrição gerada automaticamente

**Eventos**

- Envio de formulário;

- Entrada de texto em algum campo;

- Scrow (rolagem) sobre a janela;

- Movimento do mouse;

- Qualquer interação do usuário com a página web.

**Eventos – browser (navegador)**