JAVASCRIPT

Nós já vimos sobre lógica da programação, já nos aprofundamos em HTML e CSS, então o que nos falta da trindade do front-end seria o JavaScript, para nos ajudar a dar uma maior dinâmica em nosso site. Dito isso é importante dizer que o básico dessa linguagem, vem de conceitos dos cursos de lógica da programação, como function, if, else, var, etc. Então, vamos falar de coisas novas, como:

Document.

Quando queremos implementar recursos do javascript para o HTML, precisamos “linkar” o elemento do mundo do HTML para o do javascript. Para isso usamos o document.querySelector(“”), em que em seu parâmetro ficará a classe, o ID, ou até mesmo a tag que sofrerá alterações do JS. Porém nós não queremos apenas transporta-la de um mundo para o outro, queremos também alterar seu conteúdo e para isso usamos o “nomeDaVariavelDoElemento”.textContent = “(novo conteúdo)” em que podemos, por exemplos colocar outro texto ou número.

Operadores lógicos

Dentro de if’s, para validarmos se uma coisa OU outra são verdadeiras ou falsas ao mesmo tempo, podemos usar o operador || entre as duas opções. Ex:

If (peso < 0 || peso > 0) {

...;

}

Agora, se quisermos validar se mais de uma coisa é valida ao mesmo tempo, podemos usar os &&, dessa forma:

If (peso < 0 && peso > 0) {

...;

}

Bem, descobrimos como mudar o conteúdo de UM único elemento, traze-lo para o javascript realizar uma série de if’s, operadores lógicos e etc. Mas e se quisermos, por exemplo, calcular o imc de 5 pacientes? Não podemos selecionar apenas um, logo teríamos que trazer todos para o javascript como se fosse uma lista, um array. Para fazermos isso usamos o document.querySelectorAll(“classe/id do elemento-pai”), criando assim uma lista com todos os elementos filhos desse elemento pai.

Ok, trouxemos esses elementos para o mundo do JS, mas e agora?

Podemos usar o for(), que cria um sistema de loop verificando se algo é correto ou não, se a ação pedida foi feita ou não e etc.

E para trazermos a lista para esse for, basta nós colocarmos o nome da variável que contem ela da seguinte forma:

For (i = 0; i < nomeDaVariavel.length; i++){

}

Assim nós estaremos trazendo os 5 pacientes, por exemplo, para dentro do for.

Outra coisita, se queremos realizar alguma alteração CSS no nosso for ou qlq outro elemento no mundo do JS, podemos fazer essa mudança dentro do próprio JS, ou fazer o mais indicado que seria mexer e criar uma classe no JS e adiciona-la ao elemento que foi mexido no JS por lá mesmo. Nós conseguimos fazer isso da seguinte forma:

Podemos usar o nomeDaVariavelDoElementoQueSeráMudado.classList.add(“nomeDaClasseCriadaNoCSS”).

Assim nós podemos estilizar no CSS e a modificação será feita já no JS.

Temos também o toFixed(númerosDeCasasDecimais), que serve para fixar um número de casas decimais de um número.

Aprendemos a mexer na tabela e tals, mais na prática é provável que o nosso cliente vá querer adicionar novos pacientes conforme for passando o tempo. Sabemos que existem os formulários criados através do html, mas como fazemos para bota-los para funcionar?

É aí que o JavaScript entra, para isso podemos usar o

“elementoQueVaiSofrerAAção”.addEventListerner (“nome da ação(Ex: click)”, (pode colocar uma função nomeada ou anônima), etc).

Com ele poderemos dizer que, por exemplo, ao botão submit, que tem nos formulários que criamos, ser clicado irá ser adicionado um novo paciente.

Descobrimos como fazer o formulário funcionar, mas como implementaremos os dados na tabela?

Para isso podemos trazer o formulário como um todo para o Js e ir trazendo cada valor dos dados individualmente da seguinte forma:

var nome = form.nome.value

A parte form.nome, diz ao programa que é para pegar o dado “nome” do formulário e o .value informa que o valor desse dado “nome” será passado para a variável “nome” criada agora.

Feito isso, precisamos criar a linha e as colunas do paciente que será adicionado. Fazemos isso pelo próprio Js através do document.createElement(“nome da tag HTML”). Fazendo isso criamos qualquer tag do HTML, como “h1”, “p”, “tr”, “td” e etc.

Conseguimos cria-los, porém se criarmos 5 colunas(“td”) e 1 linha (“tr”), ficará uma bagunça. Como no HTML a “tr” é o elemento pai da “td”, precisamos colocar as “trd’s” como filhas da “tr”. Fazemos isso através do uso da: “elementoPai”.appendChild(“nome Da Variável Que Representa o elemento filho”).

Assim como já vimos no CSS, é uma boa prática nós dividirmos os nossos códigos JS em vários arquivos menores em que cada um terá sua própria funcionalidade.

Também vimos que é sempre bom deixar nosso código o mais legível possível, mesmo que acabe ficando um pouco mais extenso. Conseguimos fazer isso com o uso de functions, assim podendo dividir a responsabilidade de um bloco de códigos em várias funções menores com cada uma sua própria responsabilidade.

No JS existem os objetos, eles podem ser considerados e vistos como entidades da realidade, como um copo, uma pessoa, um animal dos quais terão suas próprias características como altura, peso, cor, etc. Para criarmos esses objetos podemos faze-los da seguinte forma:

var Gabriel = {

nome: Gabriel,

peso: 72,

altura: 1.79,

gordura: 20

}

Assim criamos o objeto Gabriel que contém algumas características que o representam.

Uma coisinha rápida sobre os formulários mesmo, é que para os dados desaparecerem assim que o botão for clicado, é usado a propriedade .reset();. Só isso mesmo ;)

Nas nossas séries de If’s para não precisarmos ficar colocando if () { } else { }, e tals, no próprio if podemos colocar um “!” antes do elemento que vai passar para a validação, ai nós leríamos mais ou menos assim:

If (!ValidaPeso == 0) {...}

Se meu peso não for válido tal coisa acontece.

Nós já sabemos sobre os arrays, que servem para nós armazenarmos informações, como se fosse uma grande cesta. Mas e se quisermos inserir mais itens nesse array de forma externa? Tipo em um função chamada incrementarArray? Para isso podemos chamar a variável que contém esse array e usarmos a propriedade “.push(“o que queremos adicionar na array”)”.

Além disso temos uma outra propriedade chamada .forEach que nos ajuda a percorrer pelos itens do array, dizendo o que eles fazem. Para isso usamos:

nomeDaVarQueContémArray.forEach(function(“parâmetro”) {

Bloco de código da função.

})

Para finalizar essa parte, temos a propriedade .innerHTML que com ele podemos editar/modificar o conteúdo interno do elemento do HTML escolhido. Nos retornando uma string com tudo que esse elemento tem, como classes, id’s, texto, números e etc. Basicamente podemos modificar o elemento como um todo.

Acredito que vcs lembrem de como fazer o programa escutar quando algum evento é acionado, por exemplo com o evento click, que faz com que quando algum elemento que tiver a propriedade for clicado o programa reconheça a ação. Agora mais um de vários eventos seria o double click, que para você se referir no JS seria o dblclick.

Aprendemos como criar/adicionar elementos no nosso HTML através do JS, mas e como remover? Para isso podemos usar a propriedade nomeDoElemento.remove(). Assim podemos remover o elemento desejado.

Uma propriedadizinha bem interessante seria o “this” do qual é bastante usado para se referir à algum elemento que tenha sofrido uma ação. Por exemplo, temos uma array que contém 5 linhas de uma tabela e queremos que a linha que for clicada 2 vezes seja excluída. Para fazermos isso, como vimos podemos usar o .remove() e para selecionarmos o elemento que havia sofrido a ação recentemente usamos o “this”, ficando então this.remove().

Vimos esse esquema de remover elementos usando o .remove e a questão do “this”, do qual seleciona quem estiver sofrendo a ação que esteja dentro de um elemento pai já carregado na página(“isso é muito importante”), e por conta disso elementos que são inseridos com um formulário, por exemplo, dos quais não são lidos e executamos antes mesmo de iniciar a página, se deve se atentar e usar outras maneiras de selecionar os mesmos.

Para isso podemos usar o event.target.parentNode

No caso, o event representa o evento escutado pelo elemento pai. O target simboliza exatamente aonde o evento foi acionado e o parentNode deixa o elemento pai como o responsável por escutar e “dizer” ao programa qual de seus elementos-filhos sofreu o evento.

Uma outra propriedade do JS deverás interessante seria o setTimeOut(“”) do qual através deles podemos “pedir” ao JS que espere um determinado tempo antes de executar uma ação.

Uma outra propriedade além do click e double-click seria a input, nessa, geralmente usada em input’s, reconhece quando algo é digitado no elemento selecionado. Exemplo: temos um campo de texto (input) e nós escrevemos dentro dele “Douglas”. Com esse input atribuído com o .addEventListener(“input”), será possível identificar cada caractere que seja digitado dentro do campo de texto.

Expressões regulares

Algo bem impressionante das linguagens de programação é a existência das “expressões regulares” que servem para facilitar realizarmos alguma ação, como buscar os nomes de uma tabela, filtrando letra por letra. Para essa ação citada, podemos usar o new RegExp(“”, ) que recebe como parâmetro o local que vai receber a análise e alguma característica dessa análise, por exemplo: o “i”, irá representar que a análise não precisa se preocupar se as letras, palavras e etc são maiúsculas ou minúsculas.

Com essa expressão, dizemos onde e como procurar, porém para manda-la realmente fazer a pesquisa, é preciso usar o .test(“elemento que vai ser analisado o que está sendo escrito”), assim o elemento vai passar pelas as análises e consequentemente pela filtração.

Tudo o que a gente fez de colocar e remover dados em uma tabela, também considerado como um “banco de dados”, pode ser feito incorporando dados de um outro banco de dados externo ao seu computador pelo JS, ou seja você adicionar informações em sua tabela por exemplo advindas de um terceiro.

Para isso você poderá usar algumas propriedades como:

new XMLHttpRequest()

Esta basicamente irá fazer o JS requisitar a entrada em um site selecionado, porém apenas isso. É como se você abrisse uma nova aba do google sem colocar o link de um site, apenas deixando essa aba aberta

Agora, para fazermos, por exemplo, o nosso programa entrar em um site e retirar os dados do mesmo incorporando ao nosso, usamos o seguinte:

xhr.open("GET", "https://api-pacientes.herokuapp.com/pacientes");

O xhr é o nome da variável, o .open é a propriedade que fará o programa abrir o site e dizer para ele que tipo de ação você quer que ele execute, como o tipo “GET”, que diz para ele pegar os elementos desse local, e no segundo parâmetro é o link que queremos que ele abra.

Lembra daqueles “escutadores de ações” como o “click”, “dblclick” e etc? Para podermos realmente trazer as informações de outro site precisaremos usar o “load”, que vai literalmente carregar as informações desse site ao nosso.

Outra propriedadezinha interessante é se quisermos saber se ocorreu algum erro ao programa abrir o site desejado, como o erro 404.

Para isso usamos o .status e o .responseText. O .status dirá qual erro, e o .responseText um textinho mostrando o pq do erro.

Para finalizar temos o JSON.parse(); que serve para transformar uma string em um array por exemplo. Facilitando o controle das informações do site.