**JavaScript\_Programando na linguagem da web – Micro-certificação Tecnologia WEB - ALURA**

**Aula 01 - Consolidando o seu conhecimento**

## O primeiro arquivo .js

Para dar início ao nosso projeto, vamos começar criando nosso primeiro arquivo Javascript, o principal.js, que vai conter o código inicial que faremos neste primeiro capítulo. Para **manter uma boa organização de código** crie o arquivo principal.js dentro da pasta **/js**. Sua estrutra de pastas deve ficar assim:

introducao-javascript

├── css

│ ├── index.css

│ └── reset.css

├── js

│ └── principal.js

├── img

├── favicon.ico

└── index.html

## Importando o arquivo

Claro que após criar um novo arquivo Javascript, não pode esquecer de importá-lo ao final da tag <body> em nosso HTML, utilizando a tag <script>:

// Restante do código HTML

</section>

</main>

<!-- Importação dos Javascripts AQUI -->

<script src="js/principal.js"></script>

</body>

</html>

Repare que colocamos no atributo src o caminho relativo para nosso arquivo principal.js, ou seja a partir da pasta atual, vamos descer para a pasta js/ e em seguida linkar com arquivo.

## Trocando o título da página

Para dar início aos nossos conhecimentos com o Javascript, vamos trocar o título da nossa página de Aparecida Nutrição para Aparecida Nutricionista, para praticar. Em nosso principal.js vamos:

1- O primeiro passo é selecionar o elemento da página que queremos alterar, neste caso o :

<h1>Aparecida Nutrição</h1>

Sabemos que é possível selecionar um elemento através de sua tag, e como só temos este h1 na página poderíamos facilmente utilizá-lo, porém também vimos que não é uma boa prática utilizar um seletor de tag quando estamos selecionando um elemento do HTML, pois este h1 poderia facilmente ser trocado por um h2 e nosso código quebraria.

Vamos adicionar uma classe no h1 para que a seleção do elemento sempre seja feita atráves desta classe:

<h1 class="titulo">Aparecida Nutrição</h1>

2- Agora que o h1 já tem uma classe, podemos utilizar a **função seletora** do Javascript que é responsável de trazer um elemento do mundo do HTML para o mundo do Javascript que é a querySelector(). Sabemos que queremos buscar na nossa página pelo h1, então vamos chamar a querySelector() na variável que representa a nossa página para o Javascript , o document:

document.querySelector(".titulo");

Como queremos manipular este elemento selecionado, vamos colocá-lo em uma variável de nome semântico para poder ter acesso a ele depois:

var titulo = document.querySelector(".titulo");

3- Por último, como queremos trocar o texto do h1, vamos acessar a propriedade textContent do elemento selecionado, que é a propriedade que tem o contéudo de texto daquela tag. Como queremos **alterar** o contéudo de texto daquela tag, vamos **trocar** o valor da propriedade textContent para um novo texto, que é o nosso novo título Aparecida Nutricionista:

titulo.textContent = "Aparecida Nutricionista"

Ao recarregar a página você deve ver que assim que a página é carregada o novo título já aparece no topo!

Apesar deste capítulo inicial ter sido curto, aprendemos pontos importantes como:

* Separar a organização de nosso código em arquivos .js
* Importar arquivos .js sempre ao final do body
* A representação do HTML pela variável document
* A função que faz busca querySelector()
* Preferir selecionar por #id ou .classe

No próximo avançaremos mais em nossos conhecimentos de Javascript.

**Aula 02 - Consolidando o seu conhecimento**

## Buscando os dados do primeiro paciente

Neste capítulo vamos calcular o IMC do primeiro paciente , buscando os seus dados de altura e peso, fazendo as validações necessárias e em seguida vamos preencher o IMC na tabela.

Em seu arquivo principal.js, vá executando o passo a passo abaixo:

1- Vamos primeiro selecionar o primeiro paciente, que é uma <tr> com a nossa conhecida função document.querySelector() e salvar na variável paciente. Vamos usar o seletor de #id. Se você ainda não colocou, adicione o id primeiro-paciente no primeiro <tr> de nossa tabela:

<!-- restante do HTML -->

<tr class="paciente" id="primeiro-paciente">

<td class="info-nome">Paulo</td>

<td class="info-peso">100</td>

<td class="info-altura">2.00</td>

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>

Em seguida no seu principal.js, selecione o paciente:

var paciente = document.querySelector("#primeiro-paciente");

2- Agora com o paciente em mãos, podemos buscar dentro dele as informações que precisamos que são a altura e o peso da paciente. Vamos utilizar a função querySelector() desta vez para buscar dentro da váriavel paciente, pelo <td> que contêm a altura e o peso:

var tdPeso = paciente.querySelector(".info-peso");

var tdAltura = paciente.querySelector(".info-altura");

3- Mas sabemos que não estamos interessados nos <td> e sim no conteudo de texto de cada um deles, então vamos extraí-los:

var tdPeso = paciente.querySelector(".info-peso");

var tdAltura = paciente.querySelector(".info-altura");

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

4- Com os valores de peso e altura em mãos, podemos calcular o IMC. Não vamos nos esquecer dos parênteses também para que nossa conta seja feita como esperamos:

var imc = peso / (altura \* altura);

5- Por último, com o IMC calculado devemos preencher a coluna de IMC do paciente selecionado com novo valor calculado. Vamos primeiro pegar o <td> que irá guardar o IMC:

var tdImc = paciente.querySelector(".info-imc");

6- Agora vamos colocar o valor calculado dentro do tdImc:

tdImc.textContent = imc;

Seu código deve estar assim:

var paciente = document.querySelector("#primeiro-paciente");

var tdPeso = paciente.querySelector(".info-peso");

var tdAltura = paciente.querySelector(".info-altura");

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

var imc = peso / (altura \* altura);

var tdImc = paciente.querySelector(".info-imc");

tdImc.textContent = imc;

Agora, ao recarregar a página, você verá que o IMC do primeiro paciente substituiu o valor apresentado no HTML.

Já estamos conseguindo calcular o IMC, porém falta validar os dados que vêm da tabela. Vamos lá:

1- O primeiro passo é fazer uma verificação do peso, vamos estabelecer que um peso é inválido se ele for menor ou igual a 0 quilos **OU** maior que 1000 quilos. Para isto, vamos utilizar um *if* para checar o peso, e a condição lógica de **OU** (||):

if(peso <= 0 || peso > 1000){

console.log("Peso inválido");

}

2- Não basta apenas exibir no console a validação. Vamos escrever na coluna do IMC também o erro de peso inválido:

if(peso <= 0 || peso > 1000){

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

}

3- Vamos repetir a mesma lógica para a altura, só que considerando inválido uma altura menor ou igual a 0 metros ou maior ou igual a 3 metros.

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

}

4- Agora não basta verificar se a altura ou o peso estão errados, só devemos calcular o IMC se ambos passarem na validação. Para isto, vamos criar duas variáveis, **pesoEhValido** e **alturaEhValida**, iniciar seus valores como **true**. Caso as validações falhem, trocaremos este valor para **false**:

var alturaEhValida = true;

var pesoEhValido = true;

if(peso <= 0 || peso > 1000){

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}

5- Agora vamos fazer uma lógica para apenas calcular o IMC se o peso **E\*\* a altura forem válidos. Vamos utilizar o operador lógico de \*\*E** (&&):

var alturaEhValida = true;

var pesoEhValido = true;

if(peso <= 0 || peso > 1000){

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}

if(pesoEhValido && alturaEhValida){

var imc = peso / ( altura \* altura);

tdImc.textContent = imc;

}

O seu código final deve estar assim:

var paciente = document.querySelector("#primeiro-paciente");

var tdAltura = paciente.querySelector(".info-altura");

var tdPeso = paciente.querySelector(".info-peso");

var tdImc = paciente.querySelector(".info-imc");

var altura = tdAltura.textContent;

var peso = tdPeso.textContent;

var alturaEhValida = true;

var pesoEhValido = true;

if (peso <= 0 || peso > 1000) {

console.log("Peso inválido!");

tdPeso.textContent = "Peso inválido!";

pesoEhValido = false;

}

if (altura <= 0 || altura >= 3) {

console.log("Altura inválida!");

tdAltura.textContent = "Altura inválida!";

alturaEhValida = false;

}

if (alturaEhValida && pesoEhValido) {

var imc = peso / (altura \* altura);

tdImc.textContent = imc;

} else {

tdImc.textContent = "Altura e/ou peso inválidos!"

}

Agora estamos validando antes de realizar o cálculo do IMC!

Chegamos ao fim de mais um capítulo, e neste aqui utilizamos um pouco da lógica de programação e das funções do Javascript para validar nosso primeiro paciente.

Neste capítulo vimos:

* O operador lógico de OU ( || )
* O operador lógico de E ( && )
* Busca de elementos

**Aula 03 - Consolidando o seu conhecimento**

Por enquanto nosso código só calcula e valida o IMC de **um** único paciente. Vamos atacar este problema neste capítulo para que o IMC de todos os pacientes sejam carregados:

1- Primeiro passo então é deixar de selecionar apenas um paciente e passar a selecionar **todos** os pacientes. Faremos isto buscando por todas as <tr> que tem a classe paciente. Claro, devemos também utilizar outra função, a função querySelectorAll() que é capaz de retornar mais de um elemento quando faz a busca no HTML. Faça a substituição no seu código para trocar o nome da variável paciente para pacientes e também faça a substituição da função e do seletor:

var pacientes = document.querySelectorAll(".paciente");

console.log(paciente);

2- Recarregando a página agora, veremos que o console dará um erro, mas é por que trocamos o nome da variável e ainda não adaptamos nosso código para trabalhar com um *array* de pacientes em vez de um único. Para validar as informações e calcular o IMC de cada paciente do array precisamos **percorrer** este array utilizando um clássico **for**:

for(var i = 0; i < pacientes.length ; i++){

var paciente = pacientes[i];

console.log(paciente);

}

3- Agora que conseguimos percorrer a lista de pacientes, basta mover o nosso código que validava e calculava o IMC do paciente para dentro da tabela. O código final deve ficar assim:

var pacientes = document.querySelectorAll(".paciente");

for(var i = 0; i < pacientes.length ; i++){

var paciente = pacientes[i];

var tdPeso = paciente.querySelector(".info-peso");

var tdAltura = paciente.querySelector(".info-altura");

var tdImc = paciente.querySelector(".info-imc");

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

var alturaEhValida = true;

var pesoEhValido = true;

if(peso <= 0 || peso >= 1000){

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

if(altura <= 0 || altura >= 3.0){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}

if(pesoEhValido && alturaEhValida){

var imc = peso / ( altura \* altura);

tdImc.textContent = imc;

}

}

4- Por último, podemos limitar o número de casas decimas do IMC utilizando a função .toFixed(). Faça a substuição para imprimir o IMC na tabela com apenas duas casas decimais:

// Restante do código

if(pesoEhValido && alturaEhValida){

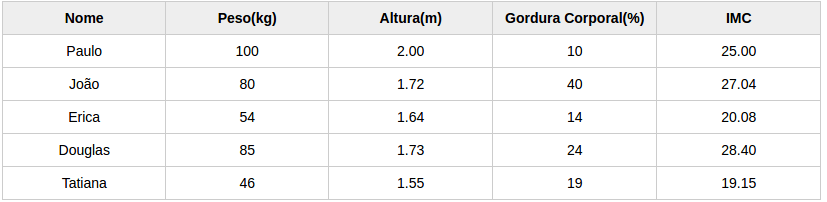
var imc = peso / ( altura \* altura);

//ALTERAÇÃO AQUI...

tdImc.textContent = imc.toFixed(2);

}

}



Agora cada paciente deve estar com IMC sendo cálculado!

Vamos melhorar um pouco a visualização de nossa tabela, para que os erros pintem a linha de vermelho além escrever na coluna de IMC quando ocorrer um erro.

1- Vamos trocar a cor da linha colocando uma nova classe na mesma, que será a classe paciente-invalido. Como primeiro passo crie uma nova classe no seu arquivo index.css como a classe abaixo:

.paciente-invalido{

background-color: lightcoral;

}

Nossa classe troca o background-color do elemento para a cor lightcoral, que é um vermelho claro.

2- Agora que temos uma classe com o estilo que queremos, devemos aplicá-la quando acontecer um erro, e os erros são detectados dentro de cada if. No if de peso, adicione a classe paciente-invalido a linha da tabela que isto ocorreu ( ou seja, na variável paciente):

if(peso <= 0 || peso >= 1000){

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

paciente.classList.add("paciente-invalido");

}

Estamos alterando o estilo colocando uma classe para facilitar a manutenção de nosso código, afinal se precisarmos alterar a cor, basta alterar o valor da propriedade background-color no CSS.

2- Faça a mesma coisa agora para quando a altura for inválida:

if(altura <= 0 || altura >= 3.00){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

paciente.classList.add("paciente-invalido");

}

Pronto, agora temos um visual melhor quando ocorrer um erro em nossa tabela!

Neste capítulo aprendemos a lidar uma coleção de elementos do HTML e vimos:

* Como selecionar diversos elementos com a função querySelectorAll()
* Relembramos um método clássico de iteração com o **for**
* Vimos como manipular estilos com a propriedade .style
* Vimos que devemos utilizar camelCase quando queremos modificar uma propriedade que tenha duas palavras, como a background-color
* A boa prática de não alterar o estilo diretamente no Javascript e sim modificar as classes dos elementos com a propriedade .classList

**Aula 04 - Consolidando o seu conhecimento**

Neste capítulo vamos focar em adicionar um novo paciente na tabela através de um formulário.

1- O primeiro passo é criar o nosso formulário no HTML. Adicione o seguinte <form> abaixo da tag <main> :

<section class="container">

<h2 id="titulo-form">Adicionar novo paciente</h2>

<form id="form-adiciona">

<div class="grupo">

<label for="nome">Nome:</label>

<input id="nome" name="nome" type="text" placeholder="digite o nome do seu paciente" class="campo">

</div>

<div class="grupo">

<label for="peso">Peso:</label>

<input id="peso" name="peso" type="text" placeholder="digite o peso do seu paciente" class="campo campo-medio">

</div>

<div class="grupo">

<label for="altura">Altura:</label>

<input id="altura" name="altura" type="text" placeholder="digite a altura do seu paciente" class="campo campo-medio">

</div>

<div class="grupo">

<label for="gordura">% de Gordura:</label>

<input id="gordura" name="gordura" type="text" placeholder="digite a porcentagem de gordura do seu paciente" class="campo campo-medio">

</div>

<button id="adicionar-paciente" class="botao bto-principal">Adicionar</button>

</form>

</section>

2- Com o formulário adicionado, vamos começar adicionando um event listener no botão de adicionar ao formulário. Ao final do principal.js, selecione o botão e pendure o listener:

var botaoAdicionar = document.querySelector("#adicionar-paciente");

botaoAdicionar.addEventListener("click", function() {

console.log("Fui clicado!");

});

Teste se o click está funcionando no botão.

3- Se você testou o código acima, deve reparar que a mensagem do console.log não surge no console. Ou se aparece, ela apenas pisca e some rapidamente. Como estamos submetendo um <form> ,por padrão a página é recarregada, logo devemos impedir que isto ocorra. Adicione o parâmetro *event* na função anônima e dentro dela chame o event.preventDefault():

var botaoAdicionar = document.querySelector("#adicionar-paciente");

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

console.log("Fui clicado!");

});

Agora sim ao clicar vemos a mensagem "Fui clicado" surgir no console.

Com nosso formulário criado, podemos implementar o código Javascript responsável por criar um novo paciente na tabela.

Com o form criado e já escutando nossos eventos de click, vamos começar a implementar a adição de um novo paciente na tabela:

1- Nosso primeiro passo é selecionar o <form> para que tenhamos acesso aos campos de input com os dados do paciente. Vamos trazê-lo do HTML:

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

});

2- Depois de selecionado o form, precisamos ter acesso a cada um dos inputs. Para fazer isto podemos nos apoiar numa característica do form, que é o acesso que temos aos seus inputs como se fossem propriedades do form, bastando usar como nome da propriedade os atributos *name* dos inputs. Como temos inputs com os name's com valor de nome,altura,peso e gordura faremos assim:

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

});

Lembrando que para pegar o valor digitado dentro de um input, devemos acessar a sua propriedade value.

3- Agora com os valores extraídos do formulário HTML para o Javascript, podemos começar a criar um novo paciente na tabela. Sabemos que pacientes são representados por <tr>'s na tabela, então precisamos criar um elemento <tr> dentro do Javascript. Para isto, vamos utilizar a função document.createElement():

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

var pacienteTr = document.createElement("tr");

});

4- Agora precisamos criar cada uma das <td>'s que ficarão dentro da <tr>. Como um paciente tem 5 atributos (nome,peso,altura,gordura e imc).Vamos utilizar a mesma função para criar estas <td's>:

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

var pacienteTr = document.createElement("tr");

var nomeTd = document.createElement("td");

var pesoTd = document.createElement("td");

var alturaTd = document.createElement("td");

var gorduraTd = document.createElement("td");

var imcTd = document.createElement("td");

});

5- Temos que colocar os valores que extraímos do form dentro de cada uma das <td's>, então vamos preênche-las com nossa conhecida .textContent:

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

var pacienteTr = document.createElement("tr");

var nomeTd = document.createElement("td");

var pesoTd = document.createElement("td");

var alturaTd = document.createElement("td");

var gorduraTd = document.createElement("td");

var imcTd = document.createElement("td");

nomeTd.textContent = nome;

pesoTd.textContent = peso;

alturaTd.textContent = altura;

gorduraTd.textContent = gordura;

});

6- Agora temos 5 <td>'s e 1 <tr> , mas elas não estão conectadas. O certo é posicionar as <td>'s dentro de cada <tr>'s. Vamos fazer isso através da função appendChild():

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

var pacienteTr = document.createElement("tr");

var nomeTd = document.createElement("td");

var pesoTd = document.createElement("td");

var alturaTd = document.createElement("td");

var gorduraTd = document.createElement("td");

var imcTd = document.createElement("td");

nomeTd.textContent = nome;

pesoTd.textContent = peso;

alturaTd.textContent = altura;

gorduraTd.textContent = gordura;

pacienteTr.appendChild(nomeTd);

pacienteTr.appendChild(pesoTd);

pacienteTr.appendChild(alturaTd);

pacienteTr.appendChild(gorduraTd);

});

7- Por último precisamos adicionar esta <tr> na tabela. Para isto vamos usar uma tática parecida, selecionar a tabela em seguida usar a função .appendChild():

var botaoAdicionar = document.querySelector("#adicionar-paciente");

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var nome = form.nome.value;

var peso = form.peso.value;

var altura = form.altura.value;

var gordura = form.gordura.value;

var pacienteTr = document.createElement("tr");

var nomeTd = document.createElement("td");

var pesoTd = document.createElement("td");

var alturaTd = document.createElement("td");

var gorduraTd = document.createElement("td");

var imcTd = document.createElement("td");

nomeTd.textContent = nome;

pesoTd.textContent = peso;

alturaTd.textContent = altura;

gorduraTd.textContent = gordura;

pacienteTr.appendChild(nomeTd);

pacienteTr.appendChild(pesoTd);

pacienteTr.appendChild(alturaTd);

pacienteTr.appendChild(gorduraTd);

var tabela = document.querySelector("#tabela-pacientes");

tabela.appendChild(pacienteTr);

});

Agora sim estamos adicionando o paciente a partir do formulário, mas veja que ele ainda está sem a coluna de IMC. Vamos resolver isto no próximo capítulo com boas práticas de reuso de código, até lá!

Neste capítulo implementamos a criação de usuários a partir de um formulário, e com isso aprendemos:

* A diferença entre as funções nomeadas e as funções anônimas
* A escutar eventos do browser com a função addEventListener()
* Que a função criadora de elementos é .createElement()
* A pegar o valor de um input por meio da propriedade .value
* A acessar os input de um form por meio da propriedade \_name\_.
* A adicionar elementos na página e dentro de outros elementos com a função appendChild()