JavaScript

Parte 5

Boas Práticas com Javascript

Exercícios

1 Organizando

Neste capítulo vamos organizar nosso código para que ele fique mais fácil de manter e para que sigamos as boas práticas.

1- Nosso arquivo `principal.js` estava começando a ficar muito grande, logo vamos separá-lo em arquivos Javascript individuais, onde cada arquivo ficará responsável por uma funcionalidade do sistema. Primeiro, crie um novo arquivo chamado `form.js`, e importe-o no fim no seu HTML.

2- Com o arquivo criado e importado, mova tudo que estiver relacionado com nosso formulário de adicionar paciente para lá. Basicamente tudo dentro do event listener do click do botão.

3- Aproveite e renomeie o seu arquivo `principal.js` para um nome mais semântico, como `calcula-imc.js` , que diz melhor o que aquele código faz.

4- Agora que já fizemos esta separação em arquivos, vamos começar a melhorar mais ainda o código. É a hora de exportarmos certas partes do código para funções, assim fica mais fácil reutilizá-las. Um bom exemplo é o código responsável por calcular um imc, que deve ser usado tanto na função cálcula IMC quanto quando criamos um novo paciente através do formulário. Crie a função`calculaImc` em seu arquivo `calcula-imc.js` .

5- Agora substitua o antigo cálculo na mão que fazíamos dentro do *if* por uma chamada a nossa recém criada função `calculaImc`, passando os parâmetros de acordo.

6- Vamos também chamar a função `calculaImc` no nosso `form.js`, para que o IMC do paciente também seja calculado quando ele for inserido na tabela. Adicione junto dos outros `<td's>` para que o contéudo do tdImc seja o retorno da função `calculaImc`.

7- E não vamos esquecer de colocar o tdImc também dentro do `<tr>` paciente.

Agora quando o paciente for adicionado na tabela, seu IMC também será calculado e inserido automaticamente!

2 Extraindo mais códigos

Dando continuidade as boas práticas de organização de código, uma outra que devemos atacar é que nossa função anônima do `form.js` está com muitas responsabilidades, ela sozinha está fazendo muitas coisas. Ela obtém um paciente do formulário, cria a `<tr>` paciente, cria diversos `<td>`, coloca um dentro do outro e depois ainda adiciona-os na tabela!

São muitas funcionalidades para uma única função, vamos quebrá-la em funções menores para melhorar a legibilidade de nosso código:

1- O primeiro passo é extrair a responsabilidade de obter os dados do formulário para uma nova função. Crie a função `obtemPacienteDoFormulario`, que irá cuidar disto. Está função deve receber o formulário e retornar todos os dados do paciente, e para isto vamos salvar estes dados dentro de um objeto do Javascript, var paciente = {}.

2- Agora vamos chamar esta função no local aonde criávamos várias variáveis com peso, altura,etc..:

```
// form.js
botaoAdicionar.addEventListener("click", function(event) {
    event.preventDefault();

    var form = document.querySelector("#form-adiciona");
    // Remova a criação das variaveis individuais e deixe apenas o
objeto paciente
    var paciente = obtemPacienteDoFormulario(form);

    // Restante do código
    ...
});
```

3- A próxima parte que iremos atacar é a criação da `<tr>` paciente. Vamos criar uma função para criar a `<tr>` e uma para as `<td>`. Vamos começar pela função `montaTd`, que deve receber o dado que vai ser colocado dentro da `td` e a classe, e nos retornar o objeto montado.

4- Agora aproveitando a função `montaTd`, vamos criar a `montaTr`, que vai receber um objeto paciente, criar cada uma das `td`, e colocar dentro da `tr` .

5- Agora vamos fazer as substituições no event listener:

```
var botaoAdicionar = document.querySelector("#adicionar-paciente");
botaoAdicionar.addEventListener("click", function(event) {
    event.preventDefault();

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

    var paciente = obtemPacienteDoFormulario(form);

    var pacienteTr = montaTr(paciente);

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

    tabela.appendChild(pacienteTr);

});
```

6- Por último vamos fazer que o form seja limpo após adicionar um paciente. Utilize a função `.reset()` como último comando.