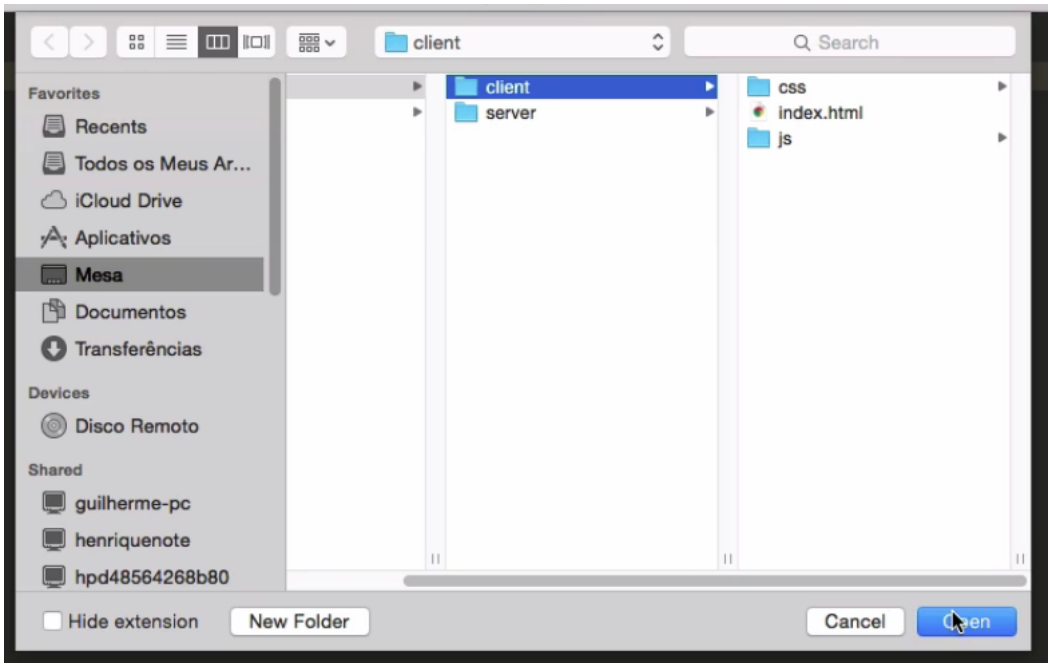
03

Prólogo: regras, código e manutenção

Transcrição

Este vídeo é um prólogo do que está por vir. A trama que se desenrolará será a implementação a funcionalidade de inclusão de uma nova linha na tabela com base na entrada do usuário, usando a manipulação de DOM e outras ferramentas que aprendemos em JavaScript. Se você já sente bastante seguro sobre esta parte, pode avançar para o último vídeo da aula 1 e conferir os desdobramentos finais. Caso contrário, você pode recordar o conteúdo agora, com explicações detalhadas sobre o que está sendo feito. Assim teremos a chance de caminharmos juntos.

Começaremos abrindo o Visual Studio Code - mas você pode usar o seu editor favorito. Abriremos a pasta "client". Trabalharemos com a pasta "server" apenas no fim do curso.



Dentro de "client", encontraremos diversas subpastas que ainda estão vazias, mas o arquivo mais importante será o `index.html`, no qual encontraremos os `input`s. Sabemos que para capturarmos as informações na página, teremos que identificar os elementos no DOM. Por isso, adicionamos um `id` em cada `input`, como no exemplo do seguinte trecho:

```
<form class="form">

      <div class="form-group">
          <label for="data">Data</label>
          <input type="date" id="data" class="form-control" required autofocus/>
      </div>
  //...
```

Então, conseguimos obter facilmente cada input desse formulário usando o `id` de cada um. Em seguida, vamos criar o arquivo aluraframe/client/js/index.js. E para não corrermos o risco de esquecermos de importá-lo, abaixo da tag `<tfoot>`, criaremos a tag `script` que importará o arquivo `index.js`.

```
<script src="js/index.js"></script>
```

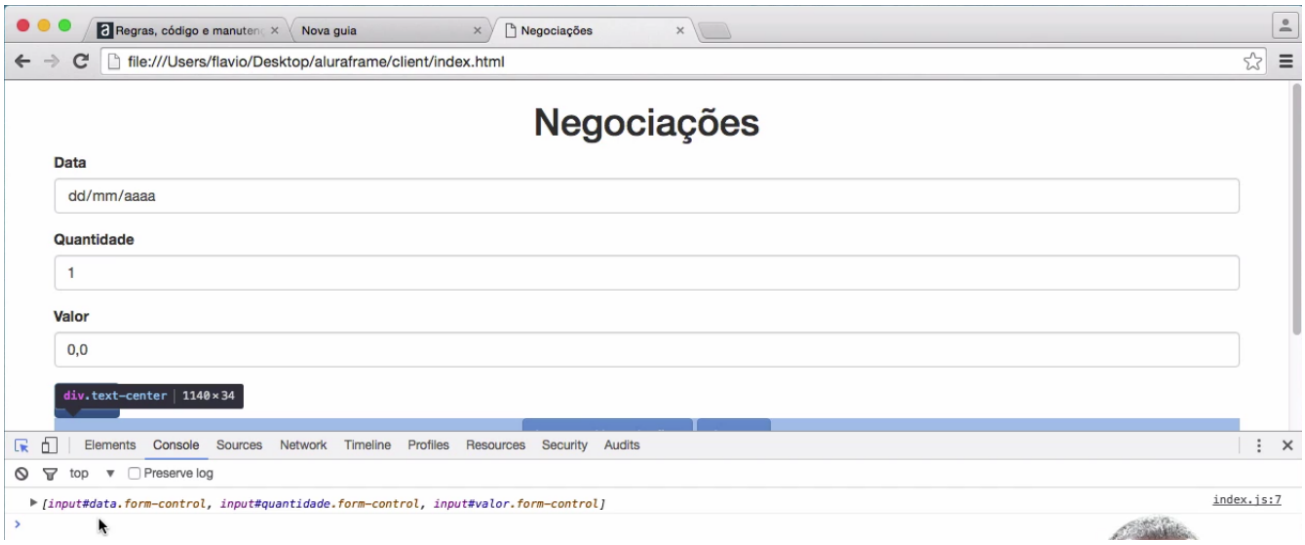
Uma das vantagens do Visual Studio Code é que ele já possui o Emmet, assim não precisaremos instalar nada. O plugin transformará a sintaxe automaticamente na importação.

Dentro do arquivo `index.js`, adicionaremos uma lista com todos os campos que temos na nossa página. Desta forma, poderemos varrer cada um deles e perguntar "qual é o seu valor?". Assim poderemos montar as colunas da nossa `tr`. Vamos criar uma variável que receberá o nome do `campos`, que será um *array*. Cada campo será o resultado de um `document.querySelector()`, uma API do DOM que nos permite buscar um elemento usando um seletor CSS ou id do elemento.

```
var campos = [
    document.querySelector('#data'),
    document.querySelector('#quantidade'),
    document.querySelector('#valor')
];

console.log(campos);
```

No fim, adicionamos o `console.log`.

Vamos recarregar a página no navegador e ver como está o formulário agora.



A ordem do elementos no nosso array será: `data`, `quantidade` e `valor`. Já conseguimos imprimir os campos, agora, precisamos que ao clicarmos no botão "Incluir", o formulário seja submetido. Durante a submissão pegaremos o valor de cada `input` e montaremos dinamicamente a `tr` com cada coluna. Lembrando que como estamos usando HTML5, enquanto não preenchermos os campos, os dados do formulário não serão submetidos. Em seguida, no editor, pediremos para o `document.querySelector()` selecionar a classe `.form`, porque encontraremos a mesma no nosso formulário. Também adicionaremos um evento `submit`, usando o `addEventListener`. Quando o evento for disparado, executaremos uma função de callback que será chamada quando alguém clicar no formulário.

```
document.querySelector('.form').addEventListener('submit', function(event) {

  alert('oi');
});
```

Observe que adicionamos o `alert` que exibirá a mensagem `oi`.

Nosso formulário está funcionando corretamente. Agora, sempre que formos submeter o nosso formulário, iremos submeter `input` do array e criar um `tr` com os valores dos mesmos. Em seguida, adicionaremos um variável chamada `tr`. Também criaremos um novo elemento, usando o `document.createElement`.

```
document.querySelector('.form').addEventListener('submit', function(event) {

  var tr = document.createElement('tr');
});
```

Estamos vendo como fazer isso com JavaScript. Agora, adicionaremos as `td` s da `tr`. Vamos usar o `forEach()` do JavaScript, uma maneira funcional de você iterar um array sem precisar do loop `for`:

```
document.querySelector('.form').addEventListener('submit', function(event) {

  var tr = document.createElement('tr');

  campos.forEach(function(campo) {

    })
});
```

A primeira vez que passarmos o `forEach`, teremos acesso ao primeiro elemento (`#data`), nas vezes seguintes acessaremos a `#quantidade` e o `#valor`. Depois, criaremos um `td` dinamicamente que não conterá nenhuma informação e informaremos que o conteúdo do mesmo será `campo.value`. Com o `appendChild`, adicionaremos a `td` como filho.

```
document.querySelector('.form').addEventListener('submit', function(event) {

  var tr = document.createElement('tr');

  campos.forEach(function(campo) {

      var td = document.createElement('td');
      td.textContent = campo.value;
      tr.appendChild(td);
```

```
    })
});
```

Até aqui, não fizemos nada muito avançado.

Então, quando chegarmos no fim do `forEach()`, ele terá criado uma `td` para cada campo. O que precisamos fazer é incluir uma `tr` na tabela.

Em seguida, abaixo do `appendChild()`, adicionaremos a variável `tdVolume`:

```
var tdVolume = document.createElement('td');
tdVolume.textContent = campos[1].value * campos[2].value;

tr.appendChild(tdVolume);
```

Agora, teremos que incluir a `tr` no arquivo `index.html` como filha da tag `<tbody>`. Começaremos adicionando o `<tbody>` no arquivo JS, `index.js`.

```
var tbody = document.querySelector('table tbody');

document.querySelector('.form').addEventListener('submit', function(event) {

    var tr = document.createElement('tr');

    campos.forEach(function(campo) {
        var td = document.createElement('td');
        td.textContent = campo.value;
        tr.appendChild(td);
    });
```

O valor da `td` será o `campo` na posição `1`, referente a `#quantidade`, multiplicado pelo `campo` na posição `2`, referente ao `#valor`. Incluímos também a `tr` do `appendChild`. Por que não adicionamos o `tbody` dentro do `querySelector()`? Porque teríamos que buscar o `tbody` a cada submissão. Da forma como escrevemos o código, não teremos que refazer a busca no DOM.

Agora, no fim do arquivo adicionaremos o `tbody` seguido pelo `appendChild()`:

```
tbody.appendChild(tr);
```

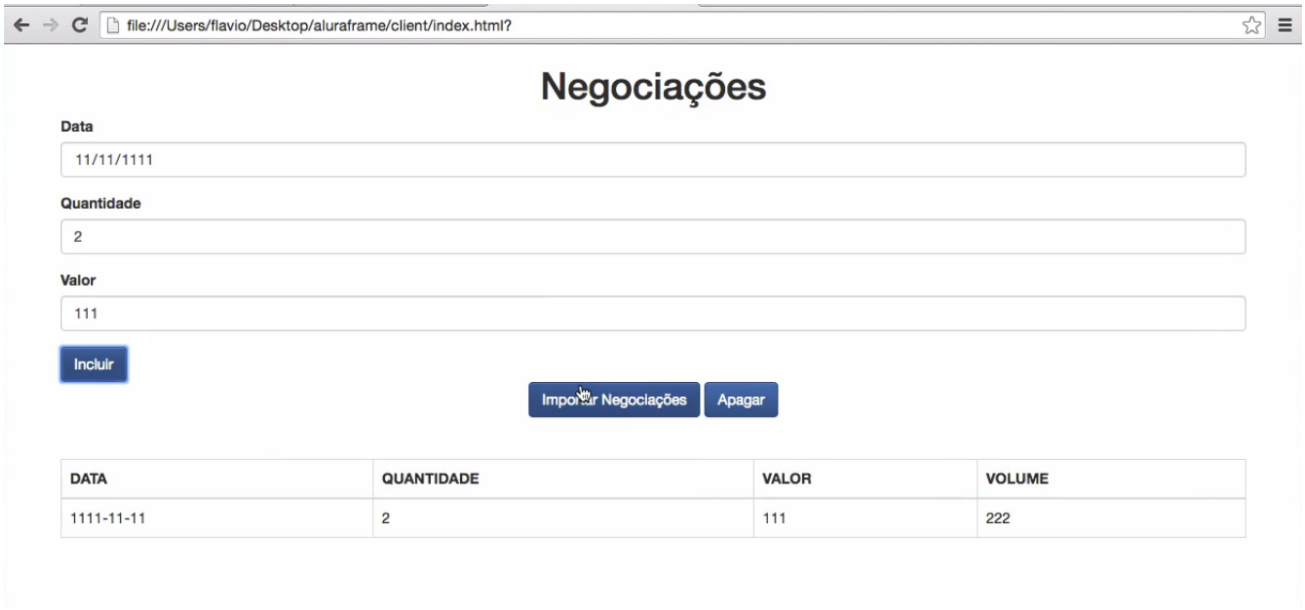
O código ficou assim:

```
var campos = [
    document.querySelector('#data'),
    document.querySelector('#quantidade'),
    document.querySelector('#valor')

];

var tbody = document.querySelector('table tbody');
```

```
document.querySelector('.form').addEventListener('submit', function(event) {

    var tr = document.createElement('tr');

    campos.forEach(function(campo) {
        var td = document.createElement('td');
        td.textContent = campo.value;
        tr.appendChild(td);
    });

    var tdVolume = document.createElement('td');
    tdVolume.textContent = campos[1].value * campos[2].value;

    tr.appendChild(tdVolume);

    tbody.appendChild(tr);

});
```

Vamos recarregar a página e abrir o console. Ao preenchermos os campos com uma data aleatória, veremos que não aparecerá nada no console. Isso acontece, porque quando recarregamos a página, perdemos a `tr`. Por isso, adicionaremos o `event.preventDefault()`, assim indicaremos para o JavaScript que ele não submeta o formulário e evite que a página seja recarregada. Vemos que os valores aparecerão na tabela abaixo:

```
var campos = [
    document.querySelector('#data'),
    document.querySelector('#quantidade'),
    document.querySelector('#valor')
];

var tbody = document.querySelector('table tbody');

document.querySelector('.form').addEventListener('submit', function(event) {

    event.preventDefault();

    var tr = document.createElement('tr');

    campos.forEach(function(campo) {
        var td = document.createElement('td');
        td.textContent = campo.value;
        tr.appendChild(td);
    });

    var tdVolume = document.createElement('td');
    tdVolume.textContent = campos[1].value * campos[2].value;

    tr.appendChild(tdVolume);

    tbody.appendChild(tr);

});
```



Porém, assim que terminarmos de nos cadastrar, teremos que limpar os campos. E para melhorar a experiência do usuário, faremos com que o campo "Data" ganhe foco. Adicionaremos o `campos[]`, indicando cada posição e no primeiro, vamos colocar o `focus`:

```
campos[0].value = '';
campos[1].value = 1;
campos[2].value = 0;

campos[0].focus();
```

Até agora nosso código ficou assim:

```
var tdVolume = document.createElement('td');
tdVolume.textContent = campos[1].value * campos[2].value;

tr.appendChild(tdVolume);

tbody.appendChild(tr);

campos[0].value = '';
campos[1].value = 1;
campos[2].value = 0;

campos[0].focus();

});
```

Faremos um novo teste, usando um data aleatória. Vemos que os dados aparecerão corretamente na tabela final e os campos do formulário estão limpos novamente. Aparentemente, tudo está funcionando corretamente, mas o código ainda tem pontos que precisam ser analisados.