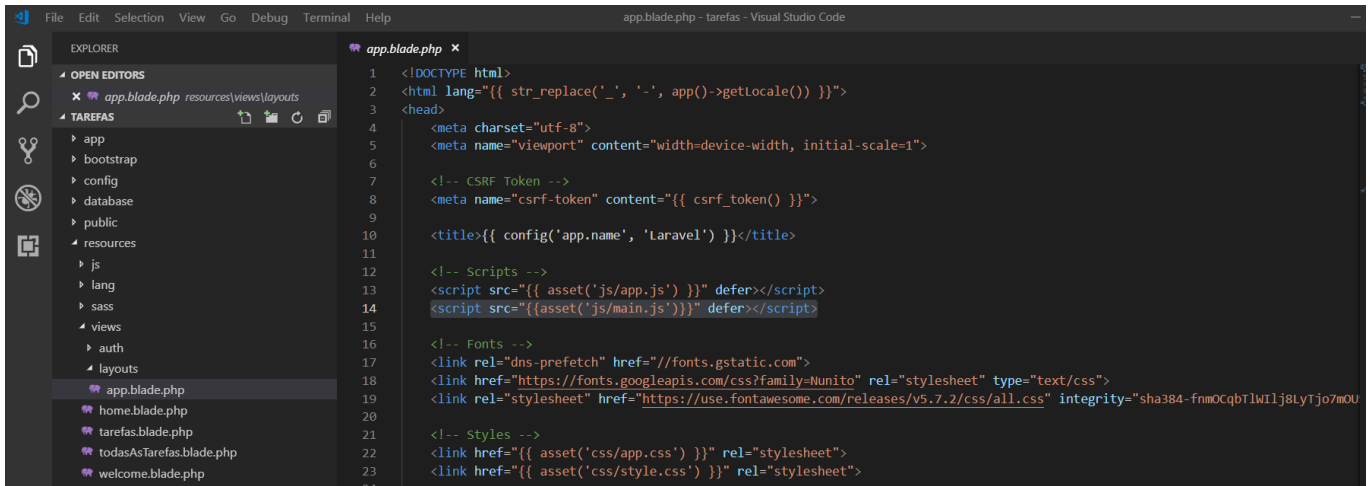


JavaScript Tutorial

Utilizando o projeto Laravel Tarefas, vamos melhorar a experiência do usuário com um pouco de **JavaScript**

Primeiro devemos incluir a tag `<script></script>` na nossa *view* principal do projeto, localizada na pasta `\tarefas\resources\views`



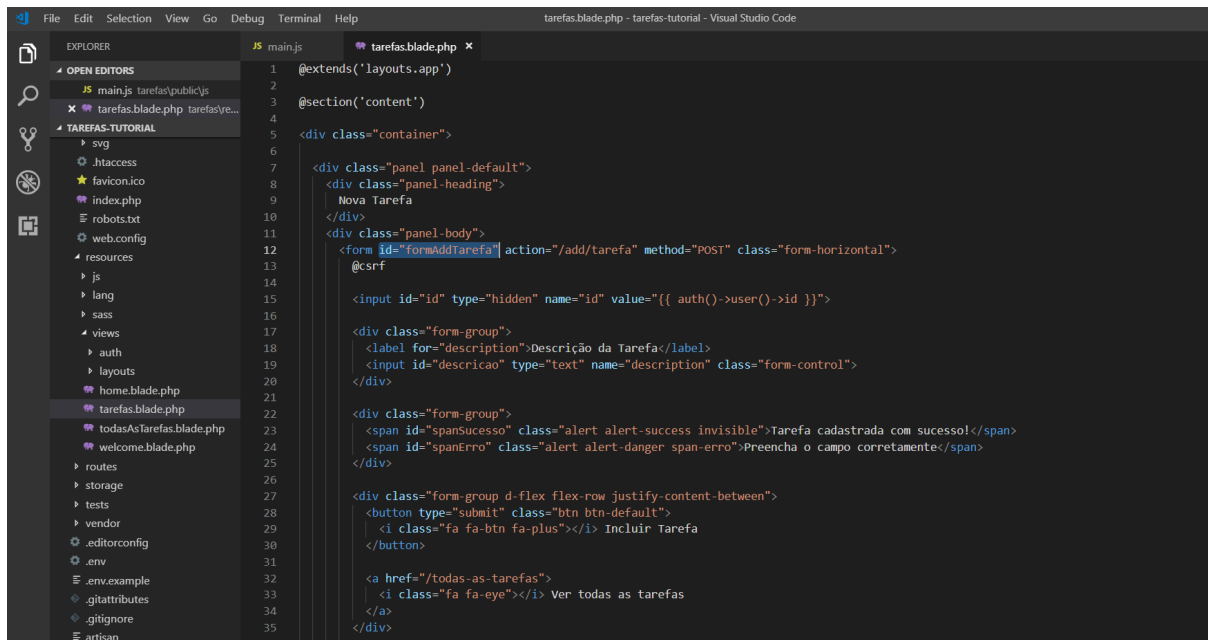
```
1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6
7     <!-- CSRF Token -->
8     <meta name="csrf-token" content="{{ csrf_token() }}">
9
10    <title>{{ config('app.name', 'Laravel') }}</title>
11
12    <!-- Scripts -->
13    <script src="{{ asset('js/app.js') }}" defer></script>
14    <script src="{{ asset('js/main.js') }}" defer></script>
15
16    <!-- Fonts -->
17    <link rel="dns-prefetch" href="//fonts.gstatic.com">
18    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet" type="text/css">
19    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.2/css/all.css" integrity="sha384-fnmOCqBTLIj8LyTj07mOU"
20
21    <!-- Styles -->
22    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
23    <link href="{{ asset('css/style.css') }}" rel="stylesheet">
```

Dentro da tag `<script></script>` declarar na propriedade `src` o caminho do arquivo JavaScript que vamos desenvolver. Perceba que estamos adicionando a propriedade `defer`, ela é utilizada para que o Navegador não espere a execução completa do script para carregar o restante da página. Outra forma de conseguir o mesmo resultado é incluindo a tag ao final do HTML.

Crie um novo arquivo **main.js** no caminho `public\js\`

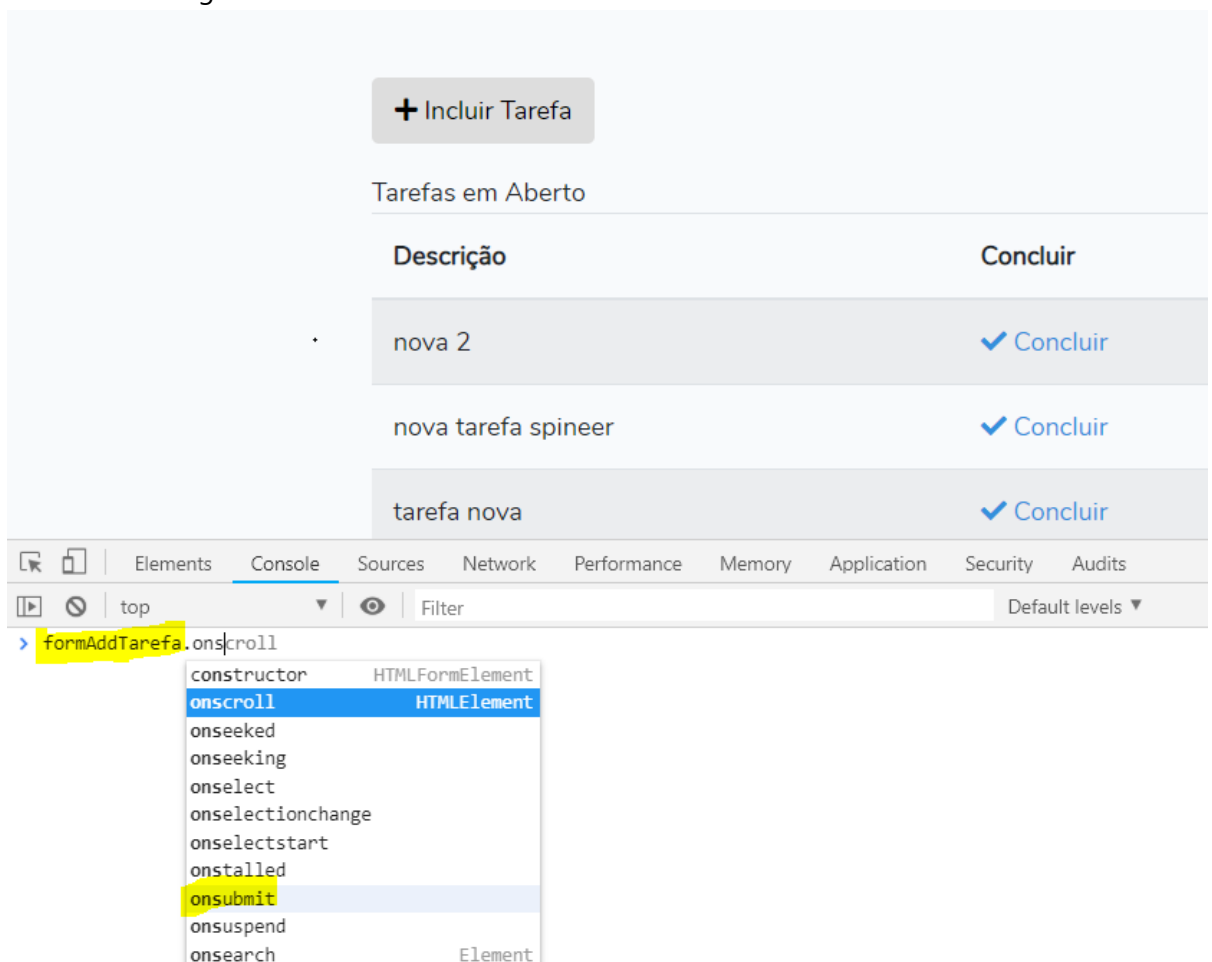
Para adicionar algum tipo de interação em uma página, precisamos basicamente selecionar o elemento HTML, adicionar um evento e "interceptar" a ação desse evento, para que possamos adicionar uma função.

- Primeiro vamos selecionar o formulário da página:
 - Incluir a propriedade `id` na tag `<form></form>`, seguindo o padrão camel-case de nomenclatura, sem espaços ou traços:



```
1 @extends('layouts.app')
2
3 @section('content')
4
5 <div class="container">
6
7 <div class="panel panel-default">
8   <div class="panel-heading">
9     Nova Tarefa
10  </div>
11  <div class="panel-body">
12    <form id="formAddTarefa" action="/add/tarefa" method="POST" class="form-horizontal">
13      @csrf
14
15      <input id="id" type="hidden" name="id" value="{{ auth()->user()->id }}">
16
17      <div class="form-group">
18        <label for="description">Descrição da Tarefa</label>
19        <input id="descricao" type="text" name="description" class="form-control">
20      </div>
21
22      <div class="form-group">
23        <span id="spanSucesso" class="alert alert-success invisible">Tarefa cadastrada com sucesso!</span>
24        <span id="spanErro" class="alert alert-danger span-erro">Preencha o campo corretamente</span>
25      </div>
26
27      <div class="form-group d-flex flex-row justify-content-between">
28        <button type="submit" class="btn btn-default">
29          <i class="fa fa-btn fa-plus"></i> Incluir Tarefa
30        </button>
31
32        <a href="/todas-as-tarefas">
33          <i class="fa fa-eye"></i> Ver todas as tarefas
34        </a>
35      </div>
36    </form>
37  </div>
38 </div>
```

- Na versão ES06 o JavaScript é capaz de selecionar um elemento utilizando apenas o `id` declarado na tag:



- Adionar o evento:

`public\js\main.js`

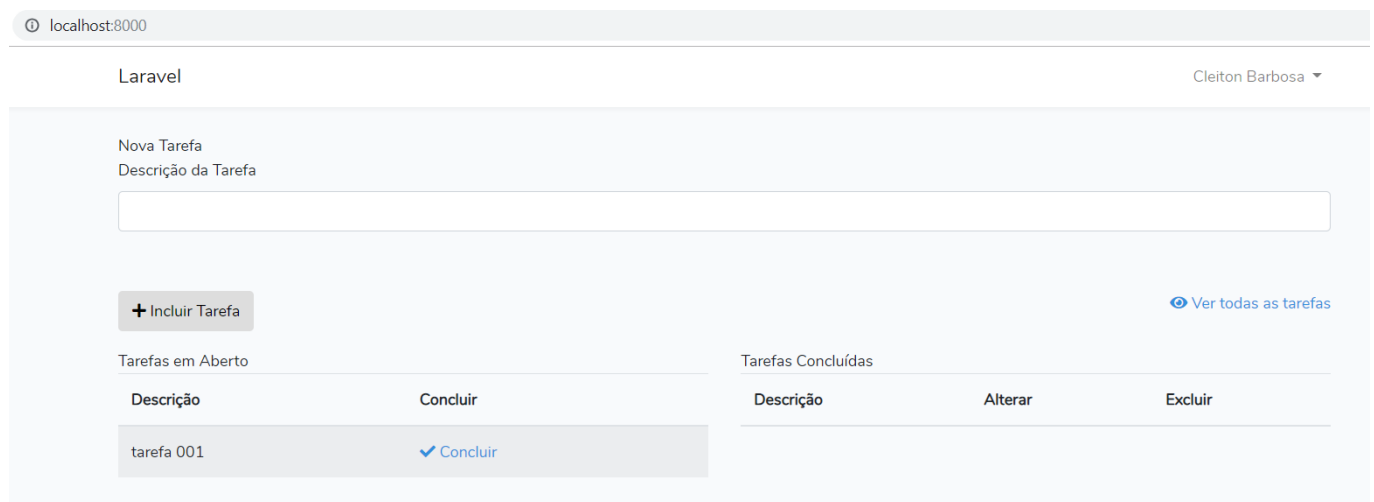
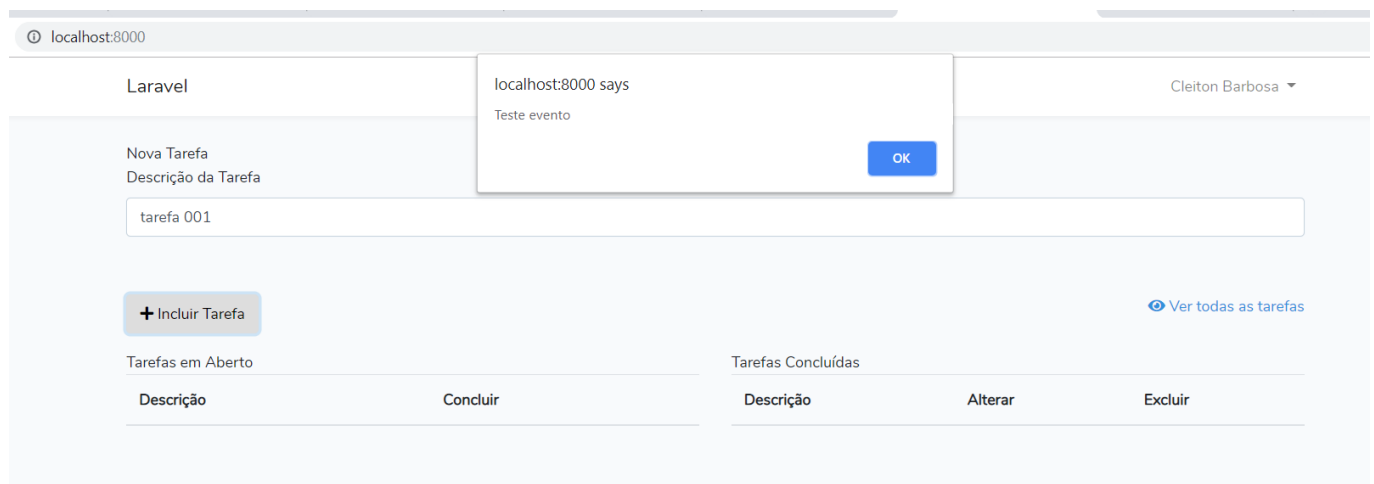
```
formAddTarefa.onSubmit
```

- Interceptar a ação do evento:

public\js\main.js:

```
formAddTarefa.onSubmit = function (event) {  
  event.preventDefault()  
  
  alert('Teste evento')  
  
  formAddTarefa.submit()  
}
```

No trecho de código acima, estamos atribuindo uma função "anônima" para o evento que criamos no formulário. Quando o usuário submeter o formulário, nosso script vai passar o *Objeto evento* para nossa função que através da função `preventDefault()` vai *interceptar* o envio do formulário para a rota definida na tag `<form></form>`, em seguida um alerta é disparado pela função `alert()`, e finalmente o formulário será submetido utilizando a função `submit()`.



Em algumas situações, o elemento HTML que precisamos selecionar é gerado dinamicamente múltiplas vezes, sendo assim não podemos utilizar o `id`, pois não será único. Para contornar este problema precisamos

selecionar o elemento através da classe

A lista abaixo, é gerada dinamicamente pelo `@foreach` do PHP:

Tarefas em Aberto	
Descrição	Concluir
tarefa 004	✓ Concluir
tarefa 003	✓ Concluir
tarefa 002	✓ Concluir
tarefa 001	✓ Concluir

resources\views\tarefas.blade.php

```
<table class="table table-striped task-table">
<thead>
  <th>Descrição</th>
  <th>Concluir</th>
</thead>
<tbody id="tbody">
  @foreach($tarefasInc as $tarefa)
    <tr>
      <td>
        {{ $tarefa->description }}
      </td>
      <td>
        <a href="concluir/tarefa/{{ $tarefa->id }}">
          <i class="fa fa-check"></i> Concluir
        </a>
      </td>
    </tr>
  @endforeach
</tbody>
</table>
```

- Incluir o atributo `class` na tag `<a>` e atribuir um nome para classe que será utilizado no script para selecionar o elemento.

resources\views\tarefas.blade.php

```
<a class="btnConcluir" href="concluir/tarefa/{ $tarefa->id }">
  <i class="fa fa-check"></i> Concluir
</a>
```

- Criar um seletor para os elementos da classe `btnConcluir`

public\js\main.js:

```
var btnsConcluir = document.querySelectorAll('.btnConcluir')
```

O objeto `document` possui várias propriedades para interagir com o DOM (Document Object Model), que representa a página HTML e todos os seus elementos. Com a função `querySelectorAll()` buscar e selecionar elementos através de `id` ou `class`, basta passar como parâmetro o valor da classe precedido de um `"."` ou do id precedido de um `"#"`.

A função `querySelectorAll()` retorna um Array com todos elementos encontrados para a classe ou id informado, portanto a variável `btnsConcluir` será um Array que podemos iterar, note que o nome da variável está no plural, justamente por ser uma lista com vários `btnConcluir`.

- Adicionar um evento para cada botão

Para adicionar o evento para cada botão do Array precisamos iterar a lista utilizando a função `forEach()`

public\js\main.js:

```
btnsConcluir.forEach(function (btnConcluir){
  btnConcluir.onclick
})
```

A função `forEach()` aceita como parâmetro uma função anônima que por sua vez recebe para cada iteração um item do Array, estamos declarando que o nome de cada item é a própria classe do elemento `btnConcluir`, porém poderia ser qualquer nome, essa forma é apenas para facilitar o entendimento.

Dentro da função anônima, estamos atribuindo o evento `onclick` para cada item da lista.

- Interceptar a ação do evento

public\js\main.js:

```
btnsConcluir.forEach(function (btnConcluir){
  btnConcluir.onclick = function (event){
    event.preventDefault()
  }
})
```