WEB ACADEMY

Programação Avançada Front-end

Daniel Augusto Nunes da Silva

Apresentação

Ementa

- Pré-compiladores CSS: sintaxe, recursos, integração com frameworks front-end, Scss e Sass. Frameworks CSS: estilização, bibliotecas de componentes visuais reutilizáveis e grid systems. Validação no front-end. Autenticação e controle de acesso no front-end.

Objetivos

- **Geral:** Capacitar os alunos com habilidades avançadas no desenvolvimento front-end, através da exploração de ferramentas e técnicas, com o intuito de aproximá-los das exigências e realidades do ambiente profissional.

- **Específicos:**
  - Explorar os recursos avançados do ecossistema front-end para compreender sua dinâmica e potencialidades no desenvolvimento web.
  - Utilizar técnicas avançadas de estilização, com foco em layouts e componentes visuais, através do CSS em conjunto com frameworks e pré-processadores.
  - Compreender os conceitos fundamentais de autenticação e controle de acesso em aplicações front-end, visando a implementação de mecanismos de segurança eficazes.

Conteúdo programático

CSS e Acessibilidade

- Introdução ao Sass/Scss;
- Sintaxe e recursos do SCSS;
- Migrando para SCSS no Angular;
- Frameworks CSS;
- Resolução de problemas de Acessibilidade.

Validação

- Formulários Reativos;
- Validação no front-end.

Segurança

- Autenticação de usuários e autorização de acesso no front-end;
- Autenticação de usuários no Angular;
- Controle de acesso por rotas no Angular.

Sites de referência

- Angular Docs: https://v19.angular.dev/docs
- Bootstrap: https://getbootstrap.com/docs/5.3/getting-started/introduction/
- MDN Web Docs: https://developer.mozilla.org/pt-BR/docs/Learn
- Sass Documentation: https://sass-lang.com/documentation/
- TypeScript Documentation: https://www.typescriptlang.org/docs/

Contato

https://github.com/danielnsilva

CSS e Acessbilidade

Pré-processadores CSS

- **Estendem a funcionalidade padrão do CSS** com recursos adicionais como variáveis, *mixins* (blocos de código reutilizáveis), funções, herança e aninhamento de regras.
  - Pré-processadores resolvem certas limitações do CSS, projetada para ser simples e declarativa. No entanto, **atualizações mais recentes já incorporam recursos inspirados nos pré-processadores, como o suporte a variáveis**.
- Exemplos: **Sass/Scss**, Less e Stylus.

- **O código precisa ser compilado** para gerar o CSS (suportado pelo navegador).

Sass/Scss

SCSS

```scss
$cor: darkred;

.button {
  background-color: $cor;
  &:hover {
    background-color: lighten($cor, 25%);
  }
}
```

CSS

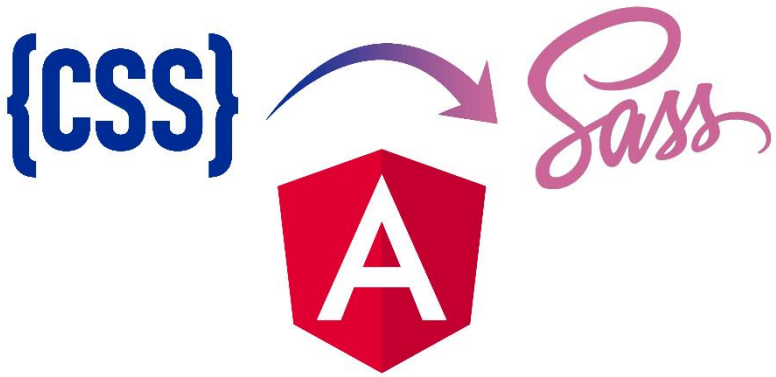
```css
.button {
    background-color: darkred;
}

.button:hover {
    background-color: #ff0c0c;
}
```

https://sass-lang.com/playground

Migrando para SCSS no Angular

1. Renomear arquivos CSS no diretório `src/`.
2. Alterar o parâmetro `styleUrls` nos componentes.
3. Atualiza a configuração de estilo padrão para novos componentes no arquivo `angular.json`
4. Renomear todas as referências ao arquivo `styles.css` para `styles.scss` no arquivo `angular.json`.

- SCSS-Migrate:
  - ng add schematics-scss-migrate



Frameworks CSS

- São ferramentas que **oferecem conjuntos de regras de estilização pré-definidas**, além de **componentes** como tabelas, layouts, botões e formulários **prontos para uso**, acelerando o desenvolvimento e garantindo **consistência visual**.

- Exemplos:
  - Materialize: baseado no Material Design do Google.
  - Tailwind CSS: oferece muito recursos de customização e controle granular.
  - **Bootstrap**: um dos mais populares, como em layout e componentes.

Bootstrap

- Framework de código aberto desenvolvido pelo Twitter para **criação rápida de sites responsivos com estilos e componentes predefinidos**.

- Principais características incluem uma biblioteca de componentes e **suporte a customização com CSS e SASS**.
- Pode ser instalado via npm, CDN ou fazendo download e inserindo diretamente com conteúdo estático no projeto.
- Referência: https://getbootstrap.com/docs/5.3/getting-started/introduction/

Bootstrap no Angular

- Instalação via npm:
  - npm install bootstrap
  - Adicionar ao angular.json:

    "styles": ["node_modules/bootstrap/dist/css/bootstrap.min.css“]
- Alternativa (ng-bootstrap):
  - ng add @ng-bootstrap/ng-bootstrap
  - Vantagens: oferece componentes do Bootstrap adaptados para o Angular, garantindo integração nativa.

Acessibilidade na web

- **Acessibilidade** na web significa permitir que o maior número possível de pessoas possa usar a web, independente da sua limitação, **removendo barreiras que possam impedir esse acesso**. Exemplos de barreiras:
  - Imagens que não possuem texto alternativo.
  - Formulários que não podem ser navegados em uma sequência lógica ou que não estão rotulados.
  - Páginas com tamanhos de fontes absoluta, que não podem ser aumentadas ou reduzidas facilmente.
- Os **padrões web** representam o básico para uma página web acessível.
- É também importante acrescentar aos padrões web as técnicas de acessibilidade associadas ao **WCAG** e suas recomendações.

WCAG

- **WCAG** (**W**eb **C**ontent **A**ccessibility **G**uidelines) são padrões internacionais definidos pelo W3C para tornar o conteúdo web mais acessível, sobretudo para pessoas com algum tipo de limitação.
- Possui 3 níveis para representar o grau de acessibilidade:
  - Nível **A** (básico): Requisitos mínimos para ser acessível.
  - Nível **AA** (intermediário): Padrão recomendado para a maioria dos casos.
  - Nível **AAA** (avançado): O padrão mais elevado, mas nem sempre prático para todos os conteúdos.
- Referência: https://www.w3c.br/traducoes/wcag/wcag22-pt-BR

Resolução de problemas de Acessibilidade

- Identificar e resolver problemas de acessibilidade é um processo que envolve uma combinação de recursos:
  - Ferramentas de avaliação automática;
  - Testes manuais;
  - Envolvimento do público alvo (usuários com diversos tipos de limitações).
- Ferramentas: Siteimprove Accessibility Checker (Extensão Chrome), Lighthouse (Extensão Chrome), NVDA.

Validação

Formulários reativos

- Um tipo de formulário no Angular, baseado em Observables, que **permite controlar seus campos e validações de maneira mais direta e organizada**, na própria classe do componente (**.component.ts*).

- Diferente dos formulários baseados em template, formulários reativos oferecem mais controle, o que pode ser **útil para formulários mais complexos**.

```
const form = new FormGroup({
  nome: new FormControl(""),
  email: new FormControl("")
});
```

Formulários reativos

Principais Diferenças entre Formulários Reativos e Baseados em Template:

Características	Formulários Reativos	Formulários Baseados em Template
Configuração do Modelo	Explícita, criada na classe do componente	Implícita, criada por diretivas
Modelo de Dados	Estruturado	Não estruturado
Fluxo de Dados	Síncrono	Assíncrono
Validação do Formulário	Funções	Diretivas

Referência: https://angular.dev/guide/forms

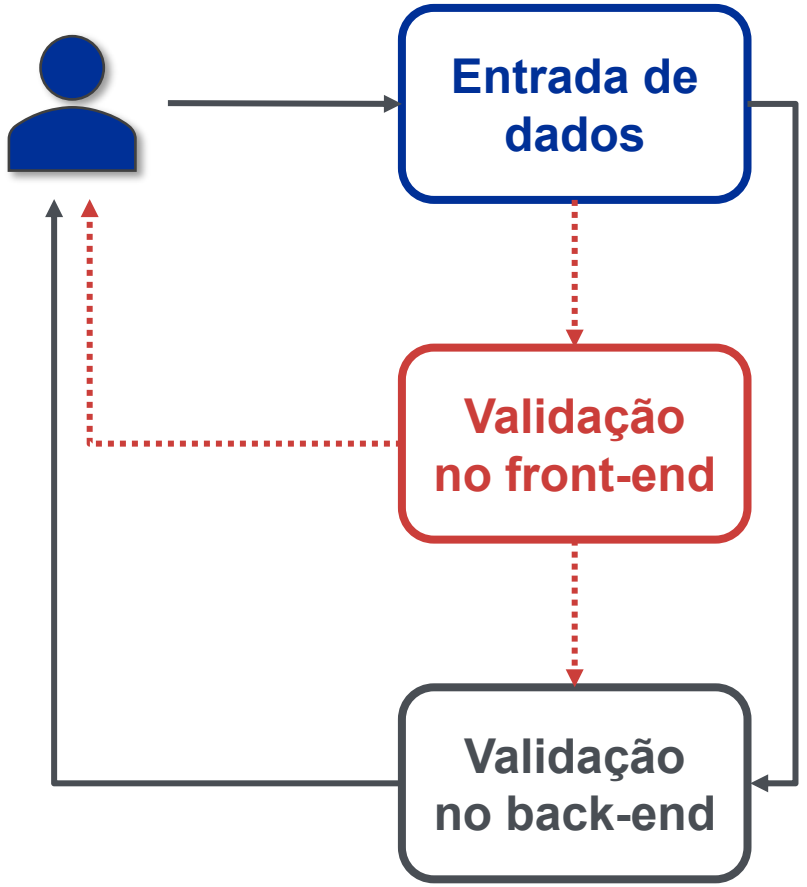
Formulários reativos

```
import {
  FormGroup, FormControl, Validators
} from '@angular/forms';

const formCadastro = new FormGroup({
  nome: new FormControl("",
    Validators.required),
  email: new FormControl("",
    [Validators.required,
     Validators.email])
});
```

```
<form [formGroup]="formCadastro">
  <label for="nome">Nome:</label>
  <input id="nome"
         formControlName="nome">
  <label for="email">Email:</label>
  <input id="email"
         formControlName="email">
  <button type="submit"
    [disabled]="formCadastro.invalid">
    Submit</button>
</form>
```

Validação de dados no front-end

- Objetivo principal: **melhorar a experiência do usuário** ao fornecer validação imediata sobre os dados inseridos.
- Não garante segurança e integridade dos dados (o que é feito no back-end), mas pode funcionar de forma complementar, **ajudando o usuário a corrigir erros antes de submeter os dados**.



Validação de dados no front-end

- **HTML 5** introduziu recursos de validação em formulários por meio dos **tipos de entrada semânticos** (ex: *email*) e **atributos de validação** (ex: *required*).
- O Angular oferece **funções de validação nativas** por meio da classe Validators (apenas em formulários reativos).
- Também é possível criar funções de validação customizadas.

```
const form = new FormGroup({
  nome: new FormControl("", [
    Validators.required,
    notBlankValidator()]),
  email: new FormControl("",
    Validators.email),
});
```

Continua…

Referências

- DUCKETT, Jon. **HTML e CSS: projete e construa websites**. 1. ed. [S. l.]: Alta Books, 2016. 512 p.
- DUCKETT, Jon. **Javascript e Jquery: desenvolvimento de interfaces web interativas**. 1. ed. [S. l.]: Alta Books, 2016. 640 p.
- GOOGLE (ed.). **Angular Docs**. [S. l.], 2025. Disponível em: https://v19.angular.dev/docs.
- LETT, Jacob. **Bootstrap 4 Quick Start**. 1. ed. [S. l.]: Bootstrap Creative, 2018. 196 p.

Referências

- MICROSOFT (ed.). **The TypeScript Handbook**. [S. l.], 2025. Disponível em: https://www.typescriptlang.org/docs/handbook/intro.html.
- MOZILLA (ed.). **MDN Web Docs: Aprendendo desenvolvimento web**. [S. l.], 2025. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn.
- W3SCHOOLS (ed.). **W3Schools Online Web Tutorials**. [S. l.], 2025. Disponível em: https://www.w3schools.com/.