Desenvolvimento web com Angular

Jackson Gomes \ jgomes@ceulp.edu.br

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Códigos-fontes

Prefácio

Este é um livro open-source, tanto no conteúdo quanto no código-fonte associado. O conteúdo é resultado de algumas práticas com o **Framework Angular** e segue uma abordagem prática, com foco no entendimento de conceitos e tecnologias no contexto do desenvolvimento de um software web.

Um *framework* representa um modelo, uma forma de resolver um problema. Em termos de desenvolvimento de software para a web um framework fornece ferramentas (ie. código) para o desenvolvimento de aplicações. Geralmente o propósito de um framework é agilizar as atividades de desenvolvimento de software, inclusive, fornecendo código pronto (componentes, bibliotecas etc.) para resolver problemas comuns, como uma interface de cadastro.

O objetivo deste livro é fornecer uma ferramenta para o desenvolvimento de habilidades de desenvolvimento web com Angular, com a expectativa de que você comece aprendendo o básico (o "hello world") e conclua com habilidades necessárias para o desenvolvimento de software que consome dados e interage com uma API HTTP REST, por exemplo.

0.1 Fonte de referência e versão do Angular

Parte do conteúdo do livro é baseada na documentação oficial do Angular, disponível em https://angular.io.

Como o Angular é um projeto em constante desenvolvimento (pelo menos até agora) serão publicadas atualizações no conteúdo do livro sempre que possível, para refletir novos recursos e funcionalidades. No momento, o conteúdo do livro é baseado na versão **6.0.0**.

0.2 Convenções

Os trechos de código apresentados no livro seguem o seguinte padrão:

- **comandos**: devem ser executados no prompt; começam com o símbolo `$`
- **códigos-fontes**: trechos de códigos-fontes de arquivos

A seguir, um exemplo de comando:

```
$ mkdir hello-world
```

O exemplo indica que o comando `mkdir`, com a opção `hello-world`, deve ser executado no prompt para criar uma pasta com o nome `hello-world`.

A seguir, um exemplo de código-fonte:

```
1 class Pessoa:
2     pass
```

O exemplo apresenta o código-fonte da classe `Pessoa`. Em algumas situações, trechos de código podem ser omitidos ou serem apresentados de forma incompleta, usando os símbolos `...` e `#`, como no exemplo a seguir:

```
1 class Pessoa:
2     def __init__(self, nome):
3         self.nome = nome
4 
5     def salvar(self):
6         # executa validação dos dados
7         ...
8         # salva
9         return ModelManager.save(self)
```

0.3 Conhecimentos desejáveis

Este livro aborda o desenvolvimento de software front-end para a web do ponto-de-vista do Angular. Isso quer dizer que não trata de conceitos iniciais de HTML, CSS, JavaScript, TypeScript e Bootstrap. Entretanto, os conceitos fundamentais dessas tecnologias vão sendo apresentados no decorrer dos capítulos, conforme surge a necessidade deles.

Para aprender mais sobre essas tecnologias recomendo essas fontes:

- **TypeScript**: Documentação oficial do TypeScript - Microsoft, TypeScript Deep Dive
- **HTML, CSS e JavaScript:** W3Schools
- **Boostrap**: Documentação oficial do Bootstrap

Este livro não leva em consideração o Sistema Operacional do seu ambiente de desenvolvimento, mas é importante que você se acostume a certos detalhes e a certas ferramentas, como o **prompt** ou **prompt de comando**.

Além destas ferramentas também são utilizadas:

- **Node.js**: disponível em https://nodejs.org representa um ambiente de execução do JavaScript fora do browser e também inclui o **npm**, um gerenciador de pacotes

- **Editor de textos ou IDE**: atualmente há muitas opções, mas destaco o **VisualStudio-Code**, disponível em https://code.visualstudio.com/
- **Git**
- **Heroku**

O **Git** é um gerenciador de repositórios com recursos de versionamento de código. É uma ferramenta essencial para o gerenciamento de código fonte de qualquer software.

O **Heroku** é um serviço de **PaaS** (de *Platform-as-a-Service*). PaaS é um modelo de negócio fornece um ambiente de execução conforme uma plataforma de programação, como o Python, um tecnologia de banco de dados, como MySQL e PostgreSQL e ainda outros recursos, como cache usando Redis.

> **Calma!** Não pira! (In)Felizmente você não vai usar todas as tecnologias lendo o conteúdo desse livro. Fica para outra oportunidade.

Para utilizar o Heroku você precisa criar uma conta de usuário. Acesse https://www.heroku.com/ e crie uma conta de usuário.

Depois que tiver criado e validado sua conta de usuário instale o **Heroku CLI**, uma ferramenta de linha de comando (prompt) que fornece uma interface de texto para criar e gerenciar aplicativos Heroku. Detalhes da instalação dessa ferramenta não são tratados aqui, mas comece acessando https://devcenter.heroku.com/articles/heroku-cli.

Capítulo 1

Introdução

1.1 Servidor web

Um **servidor web** é um programa que fornece um serviço de rede que funciona recebendo e atendendo requisições de clientes. Um **cliente**, por exemplo, é o browser.

Um **cliente** solicita um arquivo ao **servidor web**, que recebe a solicitação, atende a solicitação e retorna uma resposta para o cliente.

Esse modelo é chamado **cliente-servidor** e, na web, utiliza o protocolo **HTTP** (de *Hypertext Transfer Protocol*). O protocolo HTTP determina as regras da comunicação no modelo cliente-servidor:

- como o cliente deve enviar uma solicitação para o servidor
- como o servidor deve interpretar a solicitação
- como o servidor deve enviar uma resposta para o cliente
- como o cliente deve interpretar a resposta do servidor

Para ilustrar esse processo a Figura 1.1 demonstra a comunicação entre cliente e servidor.

Como a Figura 1.1 apresenta, quem inicia a comunicação é o cliente. O servidor recebe a solicitação e retorna uma resposta. A resposta pode ser interpretada como sucesso ou erro. No caso da figura, se o servidor encontrar o arquivo, ele retorna um código de resposta do HTTP com o número 200 e o conteúdo HTML do arquivo `index.html`, caso contrário ele retorna um código de resposta HTTP com o número 404, indicando que o arquivo não foi encontrado.

1.2 Desenvolvimento front-end

O termo **front-end** no contexto do desenvolvimento do software tem relação com a utilização de tecnologias e ferramentas para o desenvolvimento de software que, geralmente, executa em um cliente. Considerando o cenário anterior, da comunicação **cliente-servidor**, estamos falando

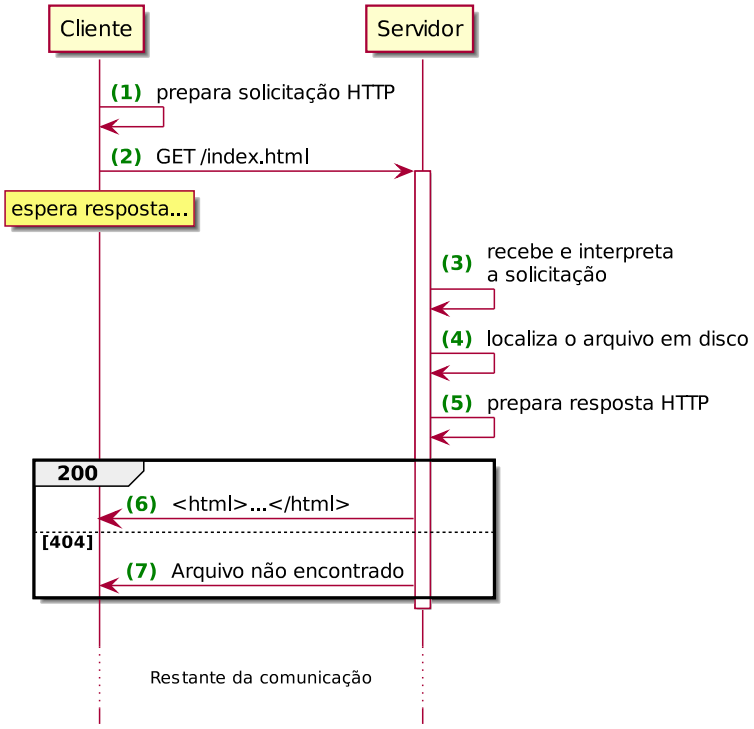


Figura 1.1: Exemplo de comunicação cliente-servidor

justamente do **browser**. O **browser** se torna uma peça fundamental nesse tipo de desenvolvimento de software.

Grande parte do desenvolvimento front-end se direciona para a tríade composta por HTML, CSS e JavaScript:

- **HTML** sendo utilizada como linguagem de marcação
- **CSS** sendo utilizada como linguagem de formatação
- **JavaScript** sendo utilizada como linguagem para adicionar interação (lógica de interface e lógica de negócio)

Para exemplificar, veja o projeto **noticias-js**. **noticias-js** é um software de gerenciamento de notícias com repositório em https://github.com/jacksongomesbr/webdevbook-noticias-js, desenvolvido em HTML, CSS e JavaScript e possui as seguintes funcionalidades:

- cadastrar notícia (título e conteúdo)
- ver a lista de notícias (título)
- ver o conteúdo de uma notícia (clicando no título)

Figuras 1.2, 1.3 ilustram o software e essas funcionalidades.

Esse comportamento já não é novidade em software web: a interface com o usuário permite a entrada de dados e a interação por meio de cliques. Os detalhes para fazer esse comportamento estão na utilização de JavaScript. Primeiro, a estrutura do software é baseada em três partes:

- `index.html`: contém o HTML para a marcação
- `main.css`: contém o CSS para a formatação

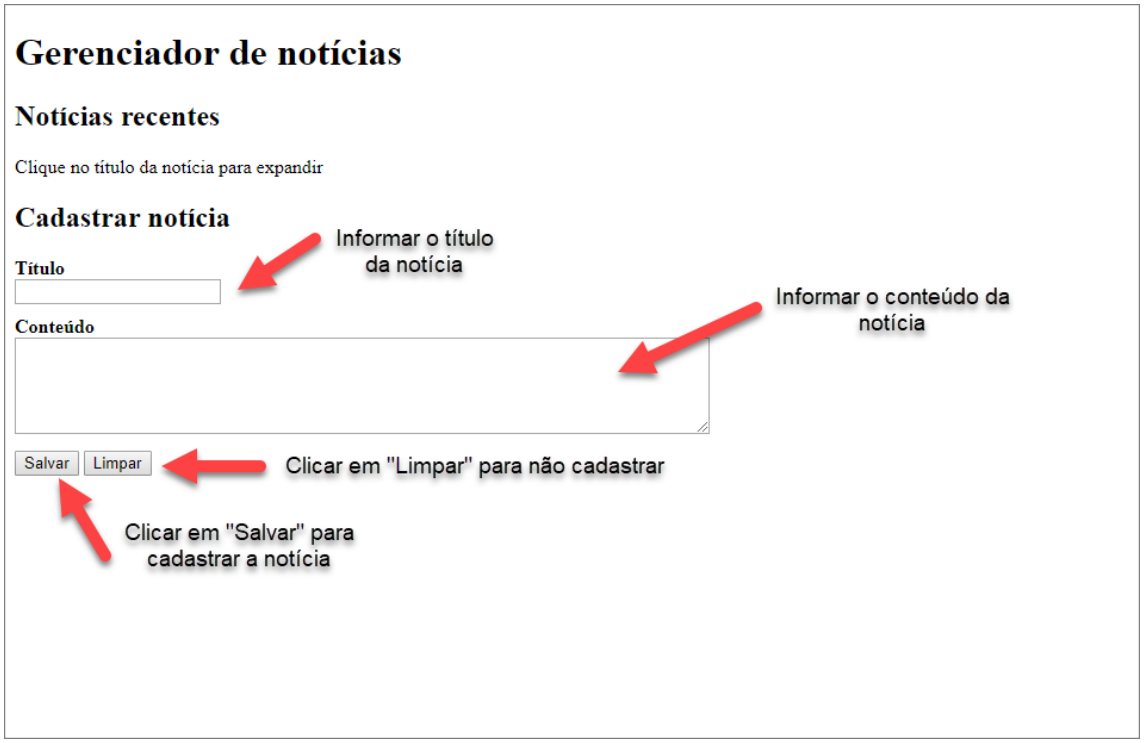


Figura 1.2: Tela de cadastro de notícias no software noticias-js

Figura 1.3: Tela de lista de notícias no software noticias-js

- `main.js`: contém o JavaScript para implementação da lógica da interface

A seguir, as seções demonstram detalhes dessa estrutura.

1.2.1 HTML para a marcação

Código-fonte 1.1 apresenta um trecho do arquivo `index.html`.

Código-fonte 1.1: Trecho do arquivo index.html do software noticias-js

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      ...
5      <link rel="stylesheet" type="text/css" media="screen" href="main.css"
           />
6      <script src="main.js"></script>
7  </head>
8  <body>
9      <h1>Gerenciador de notícias</h1>
10     <h2>Notícias recentes</h2>
11     <p>Clique no título da notícia para expandir</p>
12     <div id="noticias-recentes">
13         <ul id="noticias-recentes-list"></ul>
14     </div>
15
16     <h2>Cadastrar notícia</h2>
17     <form onsubmit="salvar(this); return false;">
18         <div>
19             <label for="frm-titulo">Título</label>
20             <input type="text" id="frm-titulo" name="titulo" required>
21         </div>
22         <div>
23             <label for="frm-conteudo">Conteúdo</label>
24             <textarea id="frm-conteudo" name="conteudo" cols="80" rows="5"
                   required></textarea>
25         </div>
26         <div>
27             <button type="submit">Salvar</button>
28             <button type="reset" formnovalidate>Limpar</button>
29         </div>
30     </form>
31 </body>
32 </html>
```

A primeira parte importante é o elemento `ul` com identificador (atributo `id`) `noticias-recentes-list`. A importância se dá para o fato de que esse identificador será utilizado no código JavaScript

para adicionar elementos `li`, um recurso chamado de **manipulação do HTML DOM**. Outra parte importante é em relação ao formulário de cadastro. Primeiro, o elemento `form` possui o atributo `onsubmit` com um valor que é um código JavaScript. Depois, cada campo do formulário está declarado para receber entrada do usuário:

- `input` com identificador `frm-titulo` é usado para o título da notícia
- `textarea` com identificador `frm-conteudo` é usado para o conteúdo da notícia

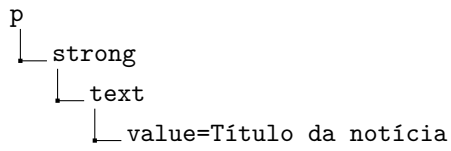
Todos os campos são de preenchimento obrigatório, então está sendo utilizado um recurso de validação diretamente no HTML por meio do atributo `required`. Por fim, o formulário tem dois botões (elemento `button`): "Salvar", que tem o atributo `type` com valor `submit` e "Limpar", que além de ter `type` com valor `reset` tem o atributo `formnovalidate`, que é utilizado para desabilitar a validação do formulário no clique do botão (é o comportamento padrão).

1.2.2 Manipulação do DOM

O DOM (de *Document Object Model*) é uma representação em memória de um documento HTML, na forma de uma **árvore** composta por nós que correspondem aos elementos do documento HTML. Por exemplo, para um trecho HTML como o seguinte:

```
<p><strong>Título da notícia</strong></p>
```

a árvore teria a seguinte estrutura:



Por causa da estrutura em árvore é possível identificar relações entre os elementos, por exemplo:

- a raiz da árvore é o nó `p`
- o nó `p` tem um filho, o nó `strong`
- o nó `strong` tem um pai, o nó `p`
- o nó `strong` tem um filho, o nó `text`
- o nó `text` tem um filho, o valor `Titulo da notícia`

Dessa forma grande parte da responsabilidade de **manipulação do DOM** recai sobre tarefas como encontrar um nó, percorrer filhos, adicionar filho em um nó e remover um nó. Para isso o DOM fornece objetos (princpalmente o `document`), métodos e propriedades, que serão apresentados a seguir.

1.2.3 CSS para formatação

Usar CSS para formatação corresponde a criar **regras CSS** e definir como elas serão aplicadas a determinados elementos do documento HTMl por meio dos **seletores**. Uma regra CSS é um

conjunto composto por pares **propriedade**:**valor**. O seletor informa para o browser como encontrar elementos para aplicar as propriedades. Há seletores: de elemento, de id e de classe. O seletor de elemento contém o nome do elemento. O setor de id contém o símbolo `#` seguido de um identificador de elemento (valor do atributo `id`). O seletor de classe contém o símbolo `.` seguido de uma classe de elemento (um dos valores do atributo `class`).

A tela do software usa formatação em CSS, conforme mostra Código-fonte 1.2.

Código-fonte 1.2: Trecho do arquivo main.css do software noticias-js

```
1  label {
2      display: block;
3      font-weight: bold;
4  }
5
6  form div {
7      margin-bottom: 10px;
8  }
9
10 .noticia .titulo {
11     font-weight: bold;
12 }
13
14 .noticia .conteudo {
15     display: none;
16 }
```

Há quatro grupos de regras, com seletores diferentes:

- `label`: aplica propriedades `display` e `font-height` para todos os elementos `label`
- `form div`: aplica propriedade `margin-bottom` para todos os elementos `div` dentro de elementos `form`
- `.noticia .titulo`: aplica propriedade `font-weight` para todos os elementos que tenham atributo `class` contendo `titulo` dentro de elementos que tenham atributo `class` contendo `noticia`
- `.noticia .conteudo`: aplica propriedade `display` para todos os elementos que tenham atributo `class` contendo `noticia` dentro de elementos que tenham atributo `class` contendo `noticia`

A propriedade `display` com valor `none` é importante porque é utilizada para ocultar o conteúdo da notícia.

As regras CSS são aplicadas **em cascata** o que significa que há uma ordem de prioridade que o browser considera para resolver conflitos de estilos:

1. estilo **in-line** (definido no atributo `style` do elemento em questão)
2. estilo definido no elemento `style`
3. estilo definido em um arquivo `.css` externo (obtido por meio do elemento `link`)

Aprender a utilizar os seletores é uma parte importante do trabalho com CSS.

1.2.4 JavaScript para lógica

O código JavaScript é parcialmente ilustrado por Código-fonte 1.3.

Código-fonte 1.3: Trecho do arquivo main.js do software noticias-js

```
1  var noticias = [];
2
3  function atualizarLista(noticia) {
4  }
5
6  function salvar(form) {
7  }
8
9  function mostrarNoticia(id) {
10 }
11
12 function ocultarNoticia(id) {
13 }
```

O código foi apresentado parcialmente para um entendimento inicial da sua estrutura. A variável `noticias` é um `Array`, utilizado para armazenar objetos que representam as notícias cadastradas. Dessa forma o conteúdo está **apenas em memória** ou **em tempo de execução**. Quando a página é recarregada, o conteúdo é perdido. Na sequência são declaradas quatro funções: `atualizarLista()`, `salvar()`, `mostrarNoticia()` e `ocultarNoticia()`.

A função `salvar()` é chamada por meio de um tratador de evento. No código HTML ([lst:noticias-js-html]), no elemento `form` o atributo `onsubmit` representa um tratador de evento, que é ativado quando algum botão dentro do formulário é clicado (nesse caso, queremos que o botão "Salvar" ative esse tratador de evento). O conteúdo de um tratador de evento é um código JavaScript e, nesse caso, há duas instruções:

- chamar a função `salvar()` passando como argumento `this` (que é uma referência ao objeto DOM que representa o formulário HTML)
- cancelar o evento ao chamar `return false`

A seguir, o código completo da função `salvar()`:

```
function salvar(form) {
    var titulo = document.getElementById('frm-titulo').value;
    var conteudo = document.getElementById('frm-conteudo').value;
    var noticia = {
        id: noticias.length,
        titulo: titulo,
        conteudo: conteudo
    };
    noticias.push(noticia);
```

```
10      atualizarLista(noticia);
11      form.reset();
12  }
```

O função `salvar()` tem o parâmetro `form`, que recebe o argumento usado na chamada da função, no tratador de evento `onsubmit`. O interior do código tem duas linhas importantes, que interagem com o HTML DOM para obter valores dos campos do formulário. Isso é feito por meio do método `getElementById()` do objeto `document`, que procura um elemento no documento HTML cujo identificador seja igual ao argumento (`frm-titulo`, por exemplo) e retorna um objeto do DOM que representa o elemento. Por ser um campo de formluário, a propriedade `value` retorna o valor digitado pelo usuário.

Na sequência o código cria um objeto `noticia` com três atributos:

- `id`: que representa um identificador numérico da notícia (começando em zero)
- `titulo`: representa o título da notícia
- `conteudo`: representa o contéudo da notícia

Depois, a sequência continua:

- o objeto `noticia` é adicionado no `Array noticias` por meio de uma chamada ao método `push()`
- chama a função `atualizarNoticia()` (descrita a seguir), informando como argumento o objeto `noticia` para que a notícia que acaba de ser cadastrada seja apresentada na lista
- chama o método `reset()` do objeto `form`, que é utilizado para redefinir os valores dos campos do fomulário

A seguir, o código completo da função `atualizarList()`:

```
1  function atualizarLista(noticia) {
2      var lista = document.getElementById('noticias-recentes-list');
3      var li = document.createElement('li');
4      li.setAttribute('id', 'noticia-' + noticia.id);
5      li.setAttribute('class', 'noticia');
6      li.innerHTML = '<p class="titulo" onclick="mostrarNoticia(' + noticia.
           id + ')">'
7          + noticia.titulo
8          + '</p>'
9          + '<p class="conteudo">'
10         + noticia.conteudo
11         + '<br>'
12         + '<span>------------------</span>'
13         + '<br>'
14         + '<button onclick="ocultarNoticia(' + noticia.id + ')">Fechar</
               button>';
15         + '</p>';
16     lista.appendChild(li);
17 }
```

O código utiliza o método `getElementById()` para obter uma referência para o objeto com iden-

tificador `noticias-recentes-list`, que representa o elemento `ul` que contém elementos `li` para apresentar a lista de notícias. A partir de então o objetivo do código é criar um elemento `li` e adicioná-lo ao elemento `ul`. Para isso, começa criando um elemento no DOM por meio do método `createElement()`, cujo argumento `"li"` representa o nome do elemento criado. Essa referência é mantida na variável `li` para o código da sequência:

- utiliza o método `setAttribute()` para definir o valor do atributo `id` (baseado no identificador da notícia)
- utiliza o método `setAttribute()` para definir o valor do atributo `class`
- utiliza a propriedade `innerHTML` para definir o restante do conteúdo HTML

Essa última parte, do valor de `innerHTML` merece destaque. A manipulação do DOM do HTML pode ser feita utilizando métodos (como `getElementById()` e `createElement()`) e também fazendo um **parser** de um conteúdo HTML. Nesse caso, por se tratar de um contéudo mais longo, o código utiliza a segunda opção. Perceba que o conteúdo da propriedade, uma `string`, é conteúdo HTML, que é interpretado pelo **browser** para modificar o DOM do HTML.

Outra parte importante desse trecho de HTML representado na `string` é sua estrutura:

- elemento `p` com atributo `class` contendo `titulo` e atributo `onclick` (tratador de evento para clique)
- título da notícia
- elemento `p` com atributo `class` contendo `conteudo`
- conteúdo da notícia
- elemento `br`
- elemento `span` contendo traços
- elemento `br`
- elemento `button` com rótulo “Fechar” e atributo `onclick`

O atributo `onclick` representa o tratador de evento para clique. Nesse caso, o elemento `p` que contém o título da notícia tem um tratador de evento que chama a função `mostrarNoticia()`. O botão “Fechar” tem o tratador de evento que chama a função `ocultarNoticia()`. Por fim, o elemento `li` é adicionado na lista de filhos do objeto `lista` por meio do método `appendChild()`.

A seguir, o código da função `mostrarNoticia()`:

```
1  function mostrarNoticia(id) {
2      var li = document.getElementById('noticia-' + id);
3      for (var i = 0; i < li.childNodes.length; i++) {
4          var node = li.childNodes[i];
5          if (node.getAttribute('class') == 'conteudo') {
6              node.setAttribute('style', 'display:inline');
7          }
8      }
9  }
```

A função `mostrarNoticia()` recebe o parâmetro `id`, que representa o identificador da notícia que cujo conteúdo deve ser apresentado. O código opera da seguinte forma:

- encontra o elemento `li` cujo identificador corresponde ao parâmetro `id`
- para cada nó filho do elemento `li` (usa a propriedade `childNodes`):
  - se o nó filho (objeto `node`) tiver atributo `class` com o valor `'conteudo'` (usa o método `getAttribute()`) então
    - define o valor do atributo `style` com `'display:inline'`, o que faz com que ele se torne visível (contrário de `display:none`)

De forma semelhante, a função `ocultarNoticia()` recebe o parâmetro `id`, que representa o identificador da notícia cujo conteúdo deve ser ocultado:

```
1  function ocultarNoticia(id) {
2      var li = document.getElementById('noticia-' + id);
3      for (var i = 0; i < li.childNodes.length; i++) {
4          var node = li.childNodes[i];
5          if (node.getAttribute('class') == 'conteudo') {
6              node.setAttribute('style', 'display:none');
7          }
8      }
9  }
```

A principal diferença para a função `mostrarNoticia()` é que a a função `ocultarNoticia()` modifica o atributo `style` para o valor `display:none`, o que torna o conteúdo invisível novamente, completando, assim, a interação com o usuário.

Certamente esse não é um software simples para quem tem a primeira experiência com esse tipo de programação, mas é importante destacar esses aspectos:

- a estrutura do HTML é criada tendo em vista possibilitar a **manipulação do DOM** com o JavaScript (por isso o uso de valores controlados para os atributos `id` e `class`)
- o atributo `onclick` é um tratador de evento para clique
- o atributo `onsubmit` é um tratador de evento para o envio do formulário
- o atributo `formnovalidate` impede a validação do formulário
- o objeto `document` dá acesso ao DOM do HTML e permite usar as funções para manipulação do DOM
- o método `getElementById()` encontra um nó do DOM com base em um identificador (atributo `id`)
- o método `setAttribute()` cria ou altera o valor de um atributo de um nó
- o método `getAttribute()` retorna o valor de um atributo de um nó
- a propriedade `innerHTML` permite fazer parser de um trecho de HTML e inserir o resultado na árvore DOM
- o método `appendChild()` adiciona um nó na lista de nós filhos do nó pai
- a propriedade `childNodes` contém a lista de nós filhos do nó pai (é um `Array`)

1.3 jQuery

O **jQuery** é uma das primeiras **bibliotecas JavaScript** e foi criada para evitar uma quantidade enorme de retrabalho e verificações de suporte de diferentes versões e tipos de browser e também inclui funções para manipulação do DOM (THE JQUERY FOUNDATION, [s.d.]).

O repositório no **noticias-js** tem um branch `jquery`, que contém a implementação utilizando a biblioteca jQuery. Uma lista completa das diferenças entre o branch `master` e o `jquery` pode ser obtida em https://github.com/jacksongomesbr/webdevbook-noticias-js/compare/jquery. Na prática, as principais modificações estão no arquivo `main.js`, com detalhes para as implementações das funções. Começando pela função `salvar()` temos o seguinte:

```
function salvar(form) {
    var titulo = $('#frm-titulo').val();
    var conteudo = $('#frm-conteudo').val();
    ...
}
```

O código em ... não muda em relação ao branch `master`. As variáveis `titulo` e `conteudo` continuam recebendo os valores informados pelo usuário no formulário, mas agora utilizam a função `$()`, que é a principal função do jQuery e, nesse caso, acessa a árvore DOM em busca de elementos com is identificadores indicados por seletores CSS de id: `#frm-titulo` e `#frm-conteudo` encontram, respectivamente, os elementos com identificador `frm-titulo` e `frm-conteudo`. O valor dos campos é obtido pela função `val()`.

Já a função `atualizarLista()` muda bastante:

```
function atualizarLista(noticia) {
    var lista = $('#noticias-recentes-list');
    var li = $('<li id="noticia-' + noticia.id + '">');
    li.addClass('noticia');
    var p_titulo = $('<p>');
    p_titulo.addClass('titulo');
    p_titulo.attr('onclick', 'mostrarNoticia(' + noticia.id + ')');
    p_titulo.html(noticia.titulo);
    var p_conteudo = $('<p>');
    p_conteudo.addClass('conteudo');
    p_conteudo.html(noticia.conteudo
        + '<br>'
        + '<span>------------------</span>'
        + '<br>'
        + '<button onclick="ocultarNoticia(' + noticia.id + ')">Fechar</
            button>');
    li.append(p_titulo, p_conteudo);
    p_conteudo.hide();
    lista.append(li);
}
```

A variável `lista` representa o elemento do DOM com identificador `noticias-recentes-list`. A variável `li` recebe a chamada da função `$()` com uma string HTML como parâmetro (linha 3). Nesse caso, o jQuery cria uma árvore parcial do DOM fazendo **parser** do argumento (como acontece com a propriedade `innerHTML`). Uma classe CSS é adiciona no nó por meio do método `addClass()` (linha 4). Um atributo é adicionado ou alterado por meio do método `attr()` (linha 7). O conteúdo de um nó pode ser definido usando o método `html()` (como com a propriedade `innerHTML`), na linha 8. O método `append()` é utilizado para adicionar um nó na lista de filhos de um pai (linha 15). Por fim, o jQuery tem um modo próprio de esconder e mostrar elementos usando, respectivamente, os métodos `hide()` e `show()`. Esses métodos também são usados nas implementações das funções `ocultarNoticia()` e `mostrarNoticia()`, que se tornam:

```
1 function mostrarNoticia(id) {
2     $('.conteudo', '#noticia-' + id).show();
3 }
4
5 function ocultarNoticia(id) {
6     $('.conteudo', '#noticia-' + id).hide();
7 }
```

A parte importante fica por conta da chamada da função `$()`. Nesse caso há dois argumentos:

1. o seletor de classe `.conteudo`
2. o contexto, que usa um seletor de id (`#noticia-` seguido do identificador da notícia)

Na prática, o jQuery fornece novas possibilidades de manipulação do DOM e, nesse caso, é utilizado para encontrar um elemento que tenha a classe CSS `conteudo` e esteja dentro de um elemento cujo identificador combina com o da notícia em questão (para ter o conteúdo apresentado ou ocultado).

Apêndice A

Configuração do ambiente de desenvolvimento

A.1 Node.js

O **Node.js** é um ambiente de execução do JavaScript independente do browser e multiplataforma (NODE.JS FOUNDATION, [s.d.]). Nos projetos desse livro é necessário utilizar **Node.js** e também a ferramenta **npm**, um gerenciador de pacotes JavaScript para o **Node.js** (NPM, INC., [s.d.]).

A instalação do **Node.js** é simples, bastando acessar https://nodejs.org/en/download/ para obter os binários de instalação conforme a plataforma desejada. O **npm** também é fornecido junto com a instalação do **Node.js**.

Para verificar se seu ambiente de execução do **Node.js** está operando normalmente, execute os comandos a seguir em um prompt:

```
$ node -v
$ npm -v
```

A saída dos programas apresenta, respectivamente, as versões do **Node.js** e do **npm** instaladas, como:

```
v10.5.0
6.2.0
```

A.2 Angular CLI

O **Angular CLI** é fornecido como um pacote **npm**, então deve ser instalado da seguinte forma:

```
$ npm install -g @angular/cli
```

O comando `install` seguido da opção `-g` faz uma *instalação global* do **Angular CLI**, o que significa que ele estará disponível para qualquer usuário.

Referências

GIT COMMUNITY. **Git**, [s.d.]. Disponível em: <https://git-scm.com/>. Acesso em: 22 jul. 2018

GOOGLE. **Angular**, [s.d.]. Disponível em: <https://angular.io/>. Acesso em: 22 jul. 2018a

GOOGLE. **Angular CLI**, [s.d.]. Disponível em: <https://cli.angular.io/>. Acesso em: 22 jul. 2018b

MICROSOFT. **Visual Studio Code - Code Editing. Redefined**, [s.d.]. Disponível em: <https://code.visualstudio.com/>. Acesso em: 22 jul. 2018

NODE.JS FOUNDATION. **Node.js**, [s.d.]. Disponível em: <https://nodejs.org>. Acesso em: 23 jul. 2018

NPM, INC. **npm**, [s.d.]. Disponível em: <https://www.npmjs.com/>. Acesso em: 23 jul. 2018

THE JQUERY FOUNDATION. **jQuery**, [s.d.]. Disponível em: <http://jquery.com/>. Acesso em: 22 jul. 2018

W3SCHOOLS. **JavaScript Tutorial**, [s.d.]. Disponível em: <https://www.w3schools.com/js/default.asp>. Acesso em: 22 jul. 2018a

W3SCHOOLS. **JavaScript and HTML DOM Reference**, [s.d.]. Disponível em: <https://www.w3schools.com/jsref/default.asp>. Acesso em: 22 jul. 2018b

W3SCHOOLS. **HTML5 Tutorial**, [s.d.]. Disponível em: <https://www.w3schools.com/html/default.asp>. Acesso em: 22 jul. 2018c

W3SCHOOLS. **CSS Tutorial**, [s.d.]. Disponível em: <https://www.w3schools.com/css/default.asp>. Acesso em: 22 jul. 2018d

W3SCHOOLS. **CSS Reference**, [s.d.]. Disponível em: <https://www.w3schools.com/cssref/default.asp>. Acesso em: 22 jul. 2018e

W3SCHOOLS. **HTML Element Reference**, [s.d.]. Disponível em: <https://www.w3schools.com/tags/default.asp>. Acesso em: 22 jul. 2018f

W3SCHOOLS. **jQuery Tutorial**, [s.d.]. Disponível em: <https://www.w3schools.com/jquery/default.asp>. Acesso em: 22 jul. 2018g

W3SCHOOLS. **jQuery Reference**, [s.d.]. Disponível em: <https://www.w3schools.com/jquery/

jquery_ref_overview.asp>. Acesso em: 22 jul. 2018h