- JavaScript é uma linguagem de destaque no cenário atual. É muito valorizada por empresas de desenvolvimento Web pela capacidade de adicionar inúmeros recursos nas páginas, visando, por exemplo, à interação com os usuários e à criação de layouts profissionais.
- Existem diversas opções de editores de código JavaScript. Alguns estão disponíveis online e são úteis para realizarmos pequenos testes. Para criar programas maiores, instale um editor profissional que contém diversos recursos visando auxiliar o trabalho do programador. Há ótimas alternativas de editores gratuitos na internet. O Visual Studio Code é um deles.
- A linguagem JavaScript dispõe dos comandos (métodos) `prompt()` e `alert()` para realizar pequenas interações com os usuários. Eles nos permitem praticar as etapas de entrada e saída de dados de um algoritmo.
- Variável é um conceito fundamental para a criação de programas. São as variáveis que permitem guardar os dados de entrada, armazenar um cálculo ou outro processamento e, a partir delas, exibir dados de saída personalizados para cada interação de um usuário do sistema.

Para nos tornarmos desenvolvedores de sistemas, é preciso construir um conhecimento sólido sobre a base, assim como na construção de um grande edifício, em que o alicerce é fundamental para as demais estruturas. Construir e entender os exemplos e exercícios deste capítulo são igualmente fundamentais para que você avance nos estudos sobre lógica de programação.

CAPÍTULO 2
Integração com HTML

Para desenvolver uma página web, devemos criar um arquivo HTML (HiperText Markup Language) contendo as tags (comandos) HTML que definem o conteúdo e a semântica dos elementos que constituem a página. Depois de salvar o arquivo, ele deverá ser aberto em um navegador web que vai renderizar (visualizar) esse documento. Nenhum processo adicional é necessário. Os códigos de programas JavaScript são desenvolvidos para adicionar um comportamento à página. Igualmente, não é preciso compilar o programa ou outra ação adicional. O próprio navegador web contém um interpretador para os programas JavaScript. Eles são inseridos nas páginas web em uma seção delimitada pelas tags `<script>` e `</script>` ou em um arquivo `.js` que deve ser referenciado pelo documento HTML.

Com JavaScript, podemos interagir de diversas formas com os usuários de páginas web. Os conceitos vistos no Capítulo 1 serão agora empregados para recuperar informações digitadas em campos de formulário de uma página e para exibir os resultados em parágrafos do documento HTML.

Nosso objetivo no livro não é produzir páginas bonitas, mas abordar os conceitos de lógica de programação utilizando a linguagem JavaScript. Caso você já possua conhecimentos em HTML e deseje melhorar o visual dos exemplos, ótimo. Caso você ainda não possua esses conhecimentos, fique tranquilo. Entenda que deve dar um passo de cada vez e que o passo que você está dando ao ler este livro é muito importante para construir uma carreira profissional na área de TI (Tecnologia da Informação).

Vamos acrescentar diversos comandos HTML. Alguns servem para definir a estrutura básica de uma página web. Outros, para exibir textos e campos de formulário. Novos comandos JavaScript também serão abordados. Neste contexto, é importante o uso de um editor profissional, como o Visual Studio Code, cujo processo de instalação foi apresentado no Capítulo 1. Um dos benefícios de um editor profissional é nos auxiliar na digitação dos comandos, que exigem cuidados quanto a sua sintaxe. Ao digitar as letras iniciais de um comando, o editor apresenta as opções de comandos que há àquelas letras. Ao pressionar a tecla **Tab** ou **Enter**, o

editor conclui a digitação do comando selecionado. Além disso, há diversos atalhos que inserem um conjunto de códigos, como o que define a estrutura básica de um documento HTML de uma página web.

Vamos agora criar uma nova pasta, chamada `cap2`, dentro da pasta `livrojs`, a fim de manter organizados os exemplos e exercícios deste livro. Em seguida, abra o Visual Studio Code.

2.1 Estrutura básica de um documento HTML

Para criar um novo arquivo HTML, devem ser inseridas algumas tags que definem as seções e configurações básicas do documento. Você pode digitá-las uma vez, salvar o documento e recorrer a ele para copiar e colar essas linhas. Mas também pode fazer isso de uma forma bem mais simples.

No Visual Studio Code, inicie um novo arquivo (**File / New File**), salve o documento como sendo do tipo HTML (dentro da pasta `cap2`, com o nome `ex2_1.html`) e depois digite !. O editor vai apresentar um recurso que permite inserir um modelo de códigos no documento (Emmet Abbreviation), conforme ilustra a Figura 2.1. Pressione **Tab** ou **Enter** para que os comandos básicos de uma página HTML sejam inseridos, como mostra a Figura 2.2. Há outros "atalhos" para facilitar a digitação de códigos que serão vistos no decorrer do livro.

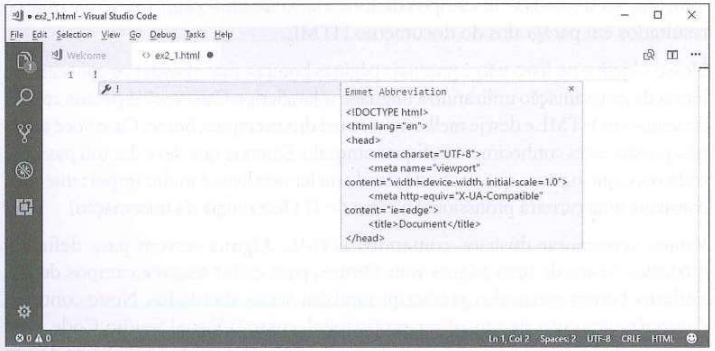

Figura 2.1 – Depois de salvar o documento como tipo HTML, digite !.

Observe que na linha 2 é especificado o idioma da sua página. Troque o `"en"` (English) para `"pt-br"` (Português do Brasil). Definir corretamente o idioma

do documento é importante por diversos aspectos, como permitir uma melhor pronúncia por um software de leitura de tela (para portadores de necessidades especiais) e indicar ao browser o dicionário a ser utilizado para a correção gramatical de textos digitados em campos de formulário.

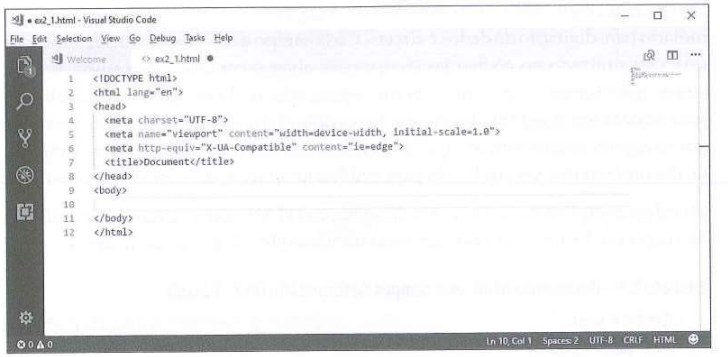

Figura 2.2 – Ao pressionar Tab ou Enter, os comandos básicos de um documento HTML são inseridos.

Outro detalhe sobre as tags HTML é que elas geralmente são declaradas aos pares. Há `<html>` e `</html>`, `<head>` e `</head>`, `<body>` e `</body>`. As tags `<head>` e `<body>` definem as seções principais da página. Na seção de cabeçalho (head), foram inseridas três metatags e o título do documento que você pode alterar conforme o exemplo. O título define o texto a ser exibido em uma aba na barra superior do navegador. `<meta charset="utf-8">` já foi utilizada nos exemplos do Capítulo 1 e serve para definir a página de códigos do documento. A metatag `<meta name="viewport" ...>` está relacionada ao processo de criação de páginas responsivas, ou seja, que respondem adequadamente aos diversos tipos de dispositivos utilizados pelos usuários, como computadores, tablets e smartphones. Já a metatag `<meta http-equiv...>` tem relação com os aspectos de compatibilidade entre navegadores.

2.2 Cabeçalhos, parágrafos e campos de formulário

Vamos acrescentar outras tags HTML ao corpo (body) do documento. Digite as seguintes linhas:

```
<h1> Programa Olá Você! </h1>
<p> Nome: <input type="text" id="nome">
```

```
    <input type="button" value="Mostrar"></p>
  <p id="resposta"></p>
```

A tag `<h1>` serve para destacar um texto com um conteúdo relevante no site. Para realizar a entrada de dados, vamos criar campos de formulário, que são a principal forma de interagir com os usuários do site. A tag HTML que cria um campo de formulário para digitação de dados é `<input>`. Cada campo deve possuir um identificador (id) a ser utilizado no código JavaScript para obter o conteúdo do campo. E a tag `<input type="button" ...>`, como o nome sugere, cria um botão geralmente utilizado para acionar um programa JavaScript. Esses dois últimos comandos estão dentro de um parágrafo criado com as tags `<p>` e `</p>`. A última linha cria um novo parágrafo no documento, que será utilizado para exibir a mensagem de resposta do programa.

O código completo desse primeiro documento HTML com o acréscimo das linhas do corpo do documento pode ser visto no Exemplo 2.1 destacado a seguir.

Exemplo 2.1 – Documento HTML com campos de formulário (ex2_1.html)

```
<!DOCTYPE html>
<html lang="pt-br">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Exemplo 2.1</title>
</head>

<body>
  <h1> Programa Olá Você! </h1>
  <p> Nome: <input type="text" id="nome">
     <input type="button" value="Mostrar"> </p>
    <p id="resposta"></p>
</body>

</html>
```

Depois de concluir a digitação, salve o arquivo. Da mesma forma como visto no Capítulo 1, o próximo passo é renderizar o documento no browser de sua preferência. Para isso, abra o navegador e digite na barra de endereços o caminho do arquivo. Ou, então, vá até a pasta em que você salvou o arquivo, selecione-o, clique com o botão direito do mouse, encontre a opção **Abrir com** e escolha o seu

navegador preferido. A Figura 2.3 apresenta a tela do documento HTML, criada no Exemplo 2.1, aberta no navegador Google Chrome.

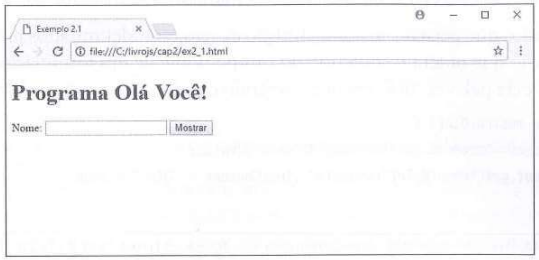

Figura 2.3 – Renderização do código HTML com campos de formulário.

2.3 Introdução a eventos e funções

Ao carregar a página HTML do Exemplo 2.1, o browser apresentou o nome do programa, um campo de texto precedido pela palavra Nome e um botão. A página está ali estática, esperando que você digite um nome e clique no botão. Ao clicar sobre o botão, uma função deve ser executada. Muito da programação JavaScript construída em páginas web é desenvolvida desta forma: elas são acionadas a partir da ocorrência de um evento. Quando o usuário executa uma ação, o programa responde ao evento do usuário com uma ou mais ações programadas em uma função. O evento mais comum de ser programado é o clique no botão. Mas há diversos outros, como modificar o conteúdo de um campo, enviar os dados de um formulário, sair de um campo, carregar a página, entre outros. Mais detalhes sobre os eventos serão apresentados no Capítulo 7.

Para criar um evento e definir qual função será acionada quando esse evento ocorrer, deve-se utilizar uma palavra reservada para indicar para qual evento a linguagem ficará "na escuta", seguido do nome da função a ser acionada. A palavra reservada pode ser, por exemplo, `(on)click`, `(on)change`, `(on)submit`, `(on)blur` ou `(on)load`. O uso do "`on`" para preceder o evento vai depender da forma a ser utilizada para criar o link entre o HTML e o JavaScript (visto nas próximas seções).

Já as funções JavaScript são declaradas a partir da palavra-chave `function` seguida do nome da função e dos parênteses `( )`. Uma função contém um conjunto de comandos que realizam uma ação. Os exemplos do Capítulo 1, de calcular o dobro de um número, a média de notas de um aluno ou o valor total de um jantar com a

taxa do garçom, serão agora inseridos dentro de uma função e executados quando um evento ocorrer. Não utilizaremos caixas para solicitar dados e exibir respostas, mas campos de formulário e mensagens em parágrafos do documento.

Os comandos que pertencem a uma função devem estar delimitados pelas chaves { }. A função que obtém o conteúdo do campo nome de um formulário e a exibe precedida pela palavra "Olá" em um parágrafo com id="resposta" é a seguinte:

```
function mostrarOla() {
  var nome = document.getElementById("nome").value;
  document.getElementById("resposta").textContent = "Olá " + nome;
}
```

Como uma função executa um conjunto de ações, é uma boa prática de programação dar um nome para a função começando por um verbo. O uso do camelcase (com letras maiúsculas no meio do nome para indicar uma nova palavra) é também um padrão recomendado e facilita a compreensão da leitura do nome.

2.4 Método getElementById()

Para referenciar um elemento HTML identificado no documento, deve-se utilizar o método `getElementById()`. Esse método permite referenciar qualquer elemento da página, como um campo de formulário, um parágrafo, um botão, uma imagem, entre outros. Para que um elemento HTML seja referenciado, ele precisa conter um atributo `id`.

Podemos armazenar a referência a um elemento em uma variável e depois obter a sua propriedade, como no exemplo a seguir:

```
var inputNome = document.getElementById("nome");
var nome = inputNome.value;
```

Ou, então, utilizar um único comando, acessando diretamente a propriedade que queremos obter ou alterar, como a seguir.

```
var nome = document.getElementById("nome").value;
```

Se o programa trabalhar com o mesmo elemento mais de uma vez, é recomendado armazenar a localização dele em uma variável (exemplo de duas linhas). Caso contrário, você pode fazer uma referência direta a sua propriedade (exemplo de uma linha). Nos exemplos do livro, daremos preferência pelo uso dos comandos em duas linhas, a fim de padronizar um formato (facilita o aprendizado) e também para evitar linhas longas que (no livro) dificultariam a leitura e a compreensão do código.

Neste capítulo, novos termos foram utilizados na descrição dos programas, como objeto, método e propriedade. Observe a seguir, de uma forma resumida, o que cada um deles representa em um programa:

- **Objeto** – representa uma instância de uma classe.
- **Método** – representa uma instrução ou um conjunto de instruções que executam uma tarefa.
- **Propriedade** – representa uma característica (atributo) de um objeto.

No Exemplo 2.1 visto anteriormente, utilizamos o objeto `document`, que a partir da execução do método `getElementById()` pode referenciar os elementos identificados pelos ids "`nome`" e "`resposta`". Já a propriedade `value` é utilizada para obter o conteúdo digitado no campo de formulário. A propriedade `textContent`, por sua vez, altera um atributo do documento, que é o conteúdo do parágrafo identificado por "`resposta`".

2.5 Propriedades textContent, innerHTML e value

Na função `mostrarOla()`, são utilizadas as propriedades `value` e `textContent`. Elas serão utilizadas em praticamente todos os programas desenvolvidos neste e nos demais capítulos do livro. A propriedade `value` obtém ou altera o conteúdo de um campo de formulário HTML. Portanto, para obter o nome do usuário informado no Exemplo 2.1, é preciso utilizar essa propriedade junto com o método `getElementById()` que faz uma referência a um campo de formulário identificado no código HTML.

Já a propriedade `textContent` serve para alterar ou obter o conteúdo de elementos de texto do documento identificados no código HTML. É possível, portanto, alterar o texto de qualquer parágrafo ou texto de cabeçalho em uma página web utilizando essa propriedade. Há também a propriedade `innerHTML`, semelhante a `textContent` quanto aos elementos em que atua, porém renderiza os códigos HTML existentes no seu conteúdo. A Tabela 2.1 destaca a diferença entre as propriedades `textContent`, `innerHTML` e `value`.

Tabela 2.1 – Comparativo entre as propriedades textContent, innerHTML e value

`textContent`	Consulta ou altera o texto exibido por elementos HTML como parágrafos (`p`), cabeçalhos (`h1, h2,...`) ou containers (`span, div`).
`innerHTML`	Consulta ou altera o conteúdo de elementos HTML como parágrafos (`p`), cabeçalhos (`h1, h2,...`) ou containers (`span, div`). Códigos HTML presentes no conteúdo são renderizados pelo navegador.
`value`	Consulta ou altera o conteúdo de campos de formulário.

A propriedade `innerHTML` pode apresentar algum risco relacionado à segurança na construção de páginas web em um tipo de ataque denominado XSS (Cross-Site Scripting). Essa vulnerabilidade explora a exibição de dados contendo códigos que poderiam ser enviados por usuários maliciosos. Para evitar esse problema, é necessário filtrar os dados de entrada de um site. Nos exemplos do livro, utilizaremos as propriedades `value` e `textContent`. O inconveniente da `textContext` é que, no Internet Explorer, ela só passou a ser suportada na versão 9. Caso necessite rodar nossos exemplos nas versões antigas do Internet Explorer, substitua a `textContent` por `innerHTML`.

2.6 Formas de adicionar JavaScript ao HTML

Nas seções anteriores, vimos como criar uma página HTML básica e uma função JavaScript. Veremos agora como criar um "link" entre elas.

2.6.1 Uso de rotinas de tratamento de eventos HTML

Uma das formas de indicar qual função JavaScript será executada quando um determinado evento ocorrer é informar isso no próprio elemento HTML que vai acionar a function. Assim, a tag HTML programada para chamar o script deve conter a palavra-chave que identifica o evento (`onclick`, `onchange`, `onmouseover`) associado ao nome da função a ser executada. No Exemplo 2.1, a tag `<input type="button"...>` ficaria da seguinte forma:

```
<input type="button" value="Mostrar" onclick="mostrarOla()">
```

Nesse modelo, os códigos JavaScript seriam inseridos no documento HTML a partir do acréscimo das tags `<script>` e `</script>` e da `function mostrarOla()`, vista na seção anterior.

O uso desse tipo de vinculação do JavaScript com o documento HTML não é considerado uma boa prática de programação, visto que o papel do HTML (conteúdo e semântica) e do JavaScript (comportamento) são diferentes no processo de construção de sites web. E misturá-los num mesmo arquivo com referências definidas nas próprias tags HTML remete a uma prática antiga de construir sites de uma forma desorganizada, em que HTML, CSS e JavaScript ocupavam um mesmo espaço. Dar manutenção em um sistema web construído dessa forma causa "arrepios" em qualquer equipe de desenvolvimento...

2.6.2 Uso de rotinas de tratamento de eventos DOM

Separar os arquivos contendo programas JavaScript dos arquivos contendo as tags HTML é um modo melhor de organizar os documentos de um site. Para criar o vínculo entre os arquivos, deve-se acrescentar no documento HTML a seguinte tag:

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

E adicionar um identificador para o botão que vai controlar o evento `onClick` para que ele seja referenciado.

```
<input type="button" value="Mostrar" id="mostrar">
```

No arquivo `.js`, as funções JavaScript devem estar no início do arquivo. Em seguida, é necessário referenciar o elemento programado para acionar o script. Observe no exemplo a seguir as regras de sintaxe do arquivo `.js` desse modelo.

```
function mostrarOla() {
  var nome = document.getElementById("nome").value;
  document.getElementById("resposta").innerHTML = "Olá " + nome;
}
var mostrar = document.getElementById("mostrar");
mostrar.onclick = mostrarOla;
```

Essa abordagem é conhecida como rotina de tratamento de eventos DOM. A DOM (Document Object Model) permite acessar cada elemento de uma página HTML como uma estrutura hierárquica – semelhante à árvore genealógica de uma família. No Capítulo 9, vamos realizar operações de inserção, acesso e remoção de elementos de um site utilizando a DOM.

Esse modo de vinculação entre os documentos HTML e JavaScript é suportado pelos principais navegadores web. Possui apenas um inconveniente, que é a impossibilidade de anexar mais de uma função a um mesmo evento.

2.6.3 Uso dos listeners (ouvintes) de eventos

A abordagem recomendada para vincular um arquivo `.js` ao documento HTML é a utilização dos chamados listeners (ouvintes) de eventos ou modelos de eventos DOM nível 2. Nesse modelo, é possível registrar múltiplas funções para um mesmo elemento HTML em um mesmo evento. Essa será a forma utilizada nos exemplos do livro.

Para criar um listener, deve-se utilizar o método `addEventListener`, cuja sintaxe é:

```
elemento.addEventListener('evento', função);
```

O nome do evento não necessita ser precedido pela sigla 'on', como nos modelos anteriores. Após o nome da função, pode ainda ser informado um terceiro parâmetro. Ele é opcional e indica a forma de propagação do evento.

No Internet Explorer, esse método passou a ser suportado na versão 9. Caso necessite desenvolver para versões antigas desse navegador, pesquise sobre o método `attachEvent()`. Com ele, é possível criar uma condição para verificar aspectos de compatibilidade entre os navegadores.

Vamos então criar esse arquivo para depois rodar o nosso programa. Para melhor organização, vamos colocar os arquivos JavaScript dentro de uma nova pasta chamada `js` dentro de `cap2`. Crie a pasta. Inicie um novo programa no Visual Studio Code, salve o arquivo dentro da pasta `js` com o nome `ex2_1.js`. Lembre-se de indicar que o arquivo deve ser do tipo JavaScript. Digite o código a seguir, sendo que as linhas que iniciam pelas `//` são comentários e, portanto, opcionais para o funcionamento do programa.

Programa JavaScript que exibe o nome informado pelo usuário no campo de edição (js/ex2_1.js)

```
// declara a função mostrarOla
function mostrarOla() {
  // obtém o conteúdo do campo (com id=) nome
  var nome = document.getElementById("nome").value;
  // exibe no parágrafo (resposta): "Olá " e o nome informado
  document.getElementById("resposta").textContent = "Olá " + nome;
}
// cria uma referência ao botão (com id=) mostrar
var mostrar = document.getElementById("mostrar");
// registra para o botão "mostrar" um ouvinte para o evento click,
// que ao ser clicado irá chamar a função mostrarOla
mostrar.addEventListener("click", mostrarOla);
```

Neste capítulo, foram adicionados novos conceitos e instruções de programação. Não se preocupe... eles serão revisados em diversos exercícios neste e nos demais capítulos do livro. Lembre-se de explorar os recursos do editor. Observe que, ao iniciar a digitação de um comando, o Visual Studio Code apresenta uma caixa com sugestões de comandos ou métodos contendo as letras já digitadas. Na Figura 2.4, é apresentado o funcionamento desse recurso denominado IntelliSense na digitação do método `getElementById()`. Para concluir a digitação do método, pressione **Tab** ou **Enter**. É possível também obter informações sobre o método ao clicar no ícone ao final de cada linha.

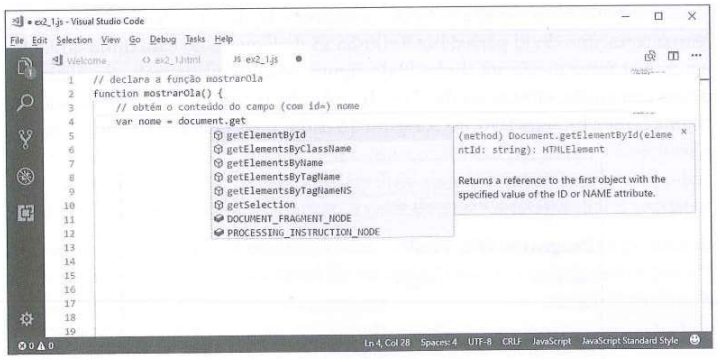

Figura 2.4 – IntelliSense no Visual Studio Code.

Outro auxílio importante proporcionado pelo uso de um editor profissional como o Visual Studio Code refere-se à formatação (indentação) do documento. Esta também é considerada uma boa prática de programação, pois facilita a compreensão das estruturas utilizadas no programa – seja no arquivo HTML, seja no programa JavaScript. O atalho do Visual Studio Code para aplicar essa formatação aos comandos do seu programa é **Alt + Shift + F**. Você pode obter um resumo com as teclas de atalho disponíveis no editor acessando o menu **Help / Keyboard Shortcuts Reference**. Na Figura 2.5, é exibida a tela, com os comandos do arquivo HTML formatados pelo editor.

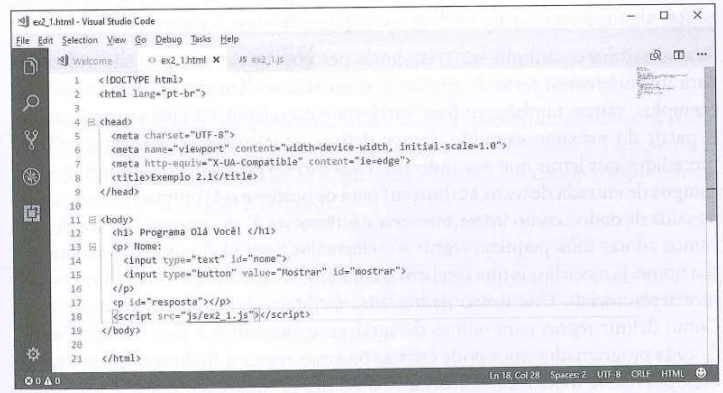

Figura 2.5 – Indentação do código facilita a compreensão e manutenção de seus programas.

Vamos testar o nosso programa! Não se esqueça de alterar o documento HTML com o acréscimo do id para o botão (linha 15 da Figura 2.5) e da linha que referencia o arquivo JavaScript (linha 18 da Figura 2.5). Depois de realizar a alteração desses comandos, salve o arquivo HTML (e também o arquivo `.js`) e recarregue a página no seu browser favorito. A Figura 2.6 exibe a tela com a execução do script.

Figura 2.6 – Ao clicar no botão Mostrar, a mensagem Olá seguida do nome é exibida.