Introdução ao HTML e CSS: Crie sites incríveis e responsivos

Autor: Eduardo Marques

Capítulo 1: Introdução as ferramentas

Ferramentas necessárias

Para começar a programar com HTML5, você precisará de algumas ferramentas básicas. O primeiro passo é ter um editor de código. Um editor de código é um programa que permite escrever e editar o código-fonte de um programa. Existem muitos editores de código diferentes disponíveis, mas alguns dos mais populares para trabalhar com HTML5 incluem o Notepad++, Sublime Text, Atom, Visual Studio Code, entre outros.

Especificamente, o Notepad++ é um editor de código gratuito e de código aberto para o sistema operacional Windows. Ele destaca a sintaxe de vários idiomas, incluindo HTML, e possui recursos como a capacidade de abrir vários arquivos ao mesmo tempo em guias separadas. Já o Sublime Text é um editor de código pago, mas possui uma versão gratuita de avaliação, e está disponível para Mac, Windows e Linux. Ele também destaca a sintaxe de vários idiomas e possui recursos como a capacidade de pesquisar e substituir em vários arquivos ao mesmo tempo. O Atom é outro editor de código gratuito e de código aberto, desenvolvido pela GitHub, e está disponível para Mac, Windows e Linux. Ele também destaca a sintaxe de vários idiomas e possui recursos como a capacidade de personalizar a aparência e as funcionalidades através de pacotes.

Além disso, você também precisará de um navegador web para visualizar suas páginas HTML5. Os navegadores mais comuns incluem o Google Chrome, Mozilla Firefox, Safari e Microsoft Edge. Especificamente, o Google Chrome é um navegador web gratuito e de código aberto desenvolvido pela Google. Ele está disponível para Windows, Mac e Linux e possui recursos como a capacidade de sincronizar suas preferências e marcadores entre dispositivos. O Mozilla Firefox é outro navegador web gratuito e de código aberto desenvolvido pela Mozilla. Ele está disponível para Windows, Mac e Linux e possui recursos como a capacidade de personalizar a aparência e as funcionalidades através de extensões. O Safari é o navegador web padrão para o sistema operacional Mac da Apple e possui recursos como a capacidade de bloquear rastreamento de terceiros. E o Microsoft Edge é o navegador web padrão para o sistema operacional Windows da Microsoft e possui recursos como a capacidade de salvar seus arquivos e leituras para acessá-los offline.

Nos exemplos citados aqui, utilizaremos o editor de código Visual Studio Code e o navegador Google Chrome. Antes de iniciar os estudos, você precisará fazer o download e instalar essas ferramentas em seu computador. É importante notar que essas são apenas algumas das opções disponíveis e que você pode escolher outras ferramentas que sejam mais adequadas às suas necessidades. É recomendável testar várias opções e escolher a que você se sentir mais confortável. Ao usar essas ferramentas, você estará pronto para começar a escrever seu código HTML5 e criar suas próprias páginas web.

Capítulo 2: Introdução ao HTML

O que é HTML e por que é importante

HTML é uma linguagem de marcação usada para estruturar o conteúdo de uma página web. Ele permite que você crie diferentes elementos, como headings, paragraphs, links, images e lists, e os organize de maneira lógica na sua página. A partir desses elementos básicos, é possível criar estruturas mais complexas, como tabelas e formulários, e adicionar interatividade com JavaScript.

HTML é importante porque é a base do desenvolvimento de páginas web. É como um esqueleto que sustenta o conteúdo visual de um site. Sem HTML, não seria possível criar páginas web com estrutura e significado. Além disso, como HTML é uma linguagem universal e suportada por todos os navegadores, é possível criar conteúdo que é acessível e funciona em todos os dispositivos, desde computadores até smartphones.

Uma visão geral das principais versões do HTML

HTML 1.0: A primeira versão do HTML, lançada em 1991. Possui uma pequena lista de tags e não possui suporte para estilos CSS ou JavaScript.

HTML 2.0: Lançado em 1995, essa versão introduziu suporte para tabelas e formulários.

HTML 3.2: Lançado em 1997, essa versão introduziu suporte para imagens e cabeçalhos.

HTML 4.01: Lançado em 1999, essa versão introduziu suporte para objetos embutidos, como plugins de mídia.

HTML 5: Lançado em 2014, HTML5 é a versão mais recente do HTML. Ele introduziu uma série de novos recursos, como suporte para vídeo e áudio sem a necessidade de plug-ins, uma estrutura mais clara para a criação de páginas, e suporte para formulários e semântica de elementos.

Em geral, ao longo do tempo, as versões do HTML foram adicionando novos recursos e melhorias para tornar a criação de páginas web mais fácil e flexível. Além disso, as novas versões tendem a ser mais eficientes e estar mais alinhadas com as necessidades e tendências atuais da web. Então recomenda-se sempre usar a versão mais atualizada para beneficiar do melhor desempenho e funcionalidade.

Capítulo 3: Estrutura básica de uma página HTML

Criando a estrutura básica de uma página HTML com as tags doctype, head e body

A tag doctype declara qual versão do HTML está sendo usada na página. Ela deve ser a primeira coisa a ser escrita na página e geralmente se parece com isso: <!DOCTYPE html>. Isso garante que o navegador entenda qual versão do HTML está sendo usada e como interpretar a página.

A tag head contém informações sobre a página, como o título (que é exibido na aba do navegador), metadados e links para folhas de estilo CSS. O conteúdo dentro da tag head não é exibido na página, mas é usado pelos navegadores e mecanismos de busca para entender o conteúdo da página.

A tag body contém o conteúdo que será exibido na página. É dentro dela que você irá escrever seus textos, imagens e outros elementos que compõem a sua página.

A estrutura básica de uma página HTML é como segue :

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
    </head>
    <body>
        Minha primeira página HTML
    </body>
</html>
```

Após criar a estrutura básica no seu editor de código, salve o arquivo com o nome “index.html”. Ao nomear dessa forma, você informa ao navegador que esse é seu “index”, ou seja, sua página principal no formato HTML. Salve no seu computador e, em seguida, abra o arquivo salvo com o navegador Google Chrome (ou outro de sua preferência).

OBS: Para formatar os espaçamentos igual ao exemplo, utilize a tecla TAB. Essa forma de organizar os espaçamentos se chama **"indentação"** e serve para manter seu código organizado e fácil de entender onde começa e onde termina cada tag.

É importante que você entenda a estrutura básica de uma página HTML, pois é a base para todo o desenvolvimento de sites. Ao entender como as tags doctype, head e body funcionam juntas, você terá uma base sólida para continuar a construir páginas mais complexas e interativas.

Capítulo 4: Elementos básicos de HTML

Criando headings, paragraphs, links, images, lists

Headings: As tags de heading (h1, h2, h3, etc.) são usadas para criar títulos e subtítulos na sua página. A tag h1 é usada para o título principal da página, enquanto as tags h2 e h3 são usadas para subtítulos. É importante usar as tags de heading de maneira hierárquica, começando com h1 e depois usando h2 e h3 para subtítulos. Isso ajudaria os navegadores e mecanismos de busca a entender a estrutura do conteúdo.

Paragraphs: A tag p é usada para criar parágrafos de texto. Qualquer texto escrito dentro da tag p será exibido como um parágrafo.

Links: A tag a é usada para criar links. Para criar um link, você precisará especificar o endereço do link (URL) usando o atributo href. Por exemplo, *Exemplo* cria um link para o site "example.com" com o texto "Exemplo" exibido como o link.

Images: A tag img é usada para inserir imagens na página. Você precisará especificar o caminho para a imagem usando o atributo src. Por exemplo, *<img src="image.jpg" alt="Descrição da imagem">* exibiria uma imagem chamada "image.jpg" na página e "Descrição da imagem" seria exibida caso a imagem não possa ser carregada.

Lists: As tags ul, ol e li são usadas para criar listas. A tag ul é usada para criar listas não ordenadas (com bullet points), enquanto a tag ol é usada para criar listas ordenadas (com números). A tag li é usada para criar itens de lista. Por exemplo:

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

Aí vão alguns exercícios para você praticar:

1. Crie um título (heading) de nível 1 com o título "Meus Hobbies"
2. Crie três parágrafos descrevendo seus hobbies favoritos
3. Adicione um link para um site relacionado a um dos seus hobbies
4. Adicione uma imagem relacionada a um dos seus hobbies
5. Crie uma lista numerada com 3 atividades relacionadas a um dos seus hobbies
6. Crie uma lista com marcadores com 3 equipamentos necessários para uma atividade relacionada a um dos seus hobbies

7. Utilize a tag "abbr" para abreviar uma palavra relacionada a um dos seus hobbies e adicione uma tag "title" para mostrar a palavra completa quando o mouse passar sobre a abreviação.

Utilizando as principais tags de HTML

Semântica: As tags semânticas como header, nav, article, section e footer são usadas para dar significado ao conteúdo e dizer ao navegador e mecanismo de busca como ele está organizado. Isso ajudaria a melhorar a acessibilidade e otimização do mecanismo de busca.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
    </head>
    <body>
        <header>
            <h1>Meu Site</h1>
            <nav>
                <ul>
                    <li><a href="#home">Home</a></li>
                    <li><a href="#about">Sobre</a></li>
                    <li><a href="#services">Serviços</a></li>
                    <li><a href="#contact">Contato</a></li>
                </ul>
            </nav>
        </header>
        <main>
            <article>
                <h2>Bem-vindo ao Meu Site</h2>
                <p>Este é o meu site pessoal, onde compartilho meus
hobbies e interesses.</p>
            </article>
            <section>
                <h2>Sobre mim</h2>
                <p>Meu nome é João e sou apaixonado por esportes
radicais e viagens.</p>
            </section>
            <section>
                <h2>Meus Serviços</h2>
```

```
                <ul>
                    <li>Fotografia aérea</li>
                    <li>Guia de viagem</li>
                    <li>Treinamento em esportes radicais</li>
                </ul>
            </section>
        </main>
        <footer>
            <p>Meu Site</p>
        </footer>
    </body>
</html>
```

Texto: As tags de texto como p, h1-h6, strong, em, a e br são usadas para criar e formatar o conteúdo de texto. Por exemplo, <p> é usada para criar parágrafos, é usada para enfatizar o texto, <a> é usada para criar links, etc.

Tag <p> para criar um parágrafo:

```
<p>Este é um exemplo de parágrafo. Ele pode conter qualquer tipo de
texto, incluindo links e estilos.</p>
```

Tags <h1> a <h6> para criar títulos:

```
<h1>Título principal</h1>
<h2>Título secundário</h2>
<h3>Título terciário</h3>
```

Tag para deixar o texto em negrito:

```
<p>Este é um <strong>texto em negrito</strong> dentro de um
parágrafo.</p>
```

Tag para deixar o texto em itálico:

```
<p>Este é um <em>texto em itálico</em> dentro de um parágrafo.</p>
```

Tag <a> para criar um link:

```
<p>Este é um <a href="https://www.google.com">link</a> para o
```

```
Google.</p>
```

Tag
 para criar uma quebra de linha:

```
<p>Este é um parágrafo com<br>uma quebra de linha no meio.</p>
```

Listas: As tags ul, ol e li são usadas para criar listas.

Tag <ul> para criar uma lista não ordenada (com marcadores de bullet):

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

Tag <ol> para criar uma lista ordenada (com números):

```
<ol>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ol>
```

Tag <li> para criar um item dentro de uma lista:

```
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ul>
```

É possível também aninhar as listas, onde uma lista é incluida dentro de outra lista:

```
<ul>
    <li>Item 1</li>
    <li>Item 2
        <ul>
            <li>sub-item 1</li>
```

```
            <li>sub-item 2</li>
        </ul>
    </li>
    <li>Item 3</li>
</ul>
```

Imagem: A tag img é usada para inserir imagens em uma página.

Uma imagem simples:

```
<img src="caminho/para/imagem.jpg" alt="Descrição da imagem">
```

Uma imagem com largura e altura definidas:

```
<img src="caminho/para/imagem.jpg" alt="Descrição da imagem"
width="200" height="100">
```

Uma imagem com uma descrição longa:

```
<img src="caminho/para/imagem.jpg" alt="Esta é uma imagem de uma
paisagem bonita com montanhas verdes e céu azul">
```

Uma imagem como link:

```
<a href="https://www.google.com">
  <img src="caminho/para/imagem.jpg" alt="Descrição da imagem">
</a>
```

A tag <img> é usada para incluir imagens em uma página web. O atributo src especifica o caminho para a imagem, e o atributo alt fornece uma descrição da imagem para usuários com dificuldade de visão ou dispositivos que não conseguem carregar a imagem. Os atributos width e height são usados para definir a largura e altura da imagem. É importante ter em mente que a imagem deve estar no servidor ou na internet para ser carregada.

Tabelas: As tags table, thead, tbody, tfoot, tr, td e th são usadas para criar tabelas com dados e estilizá-las.

Uma tabela simples com cabeçalho, corpo e rodapé:

```
<table>
    <thead>
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Data 1</td>
            <td>Data 2</td>
        </tr>
        <tr>
            <td>Data 3</td>
            <td>Data 4</td>
        </tr>
    </tbody>
    <tfoot>
        <tr>
            <td>Footer 1</td>
            <td>Footer 2</td>
        </tr>
    </tfoot>
</table>
```

A tag <table> é usada para criar uma tabela, e as tags <thead>, <tbody> e <tfoot> são usadas para agrupar o conteúdo da tabela em seções específicas. As tags <tr> são usadas para criar linhas na tabela, as <td> tags para criar células de dados e as <th> tags para criar células de cabeçalho. É importante notar que as tags <thead>, <tbody> e <tfoot> são opcionais, mas é uma boa prática usá-las para organizar o conteúdo da tabela e facilitar a estilização.

Formulários: As tags form, input, label, select, option e textarea são usadas para criar formulários e campos de entrada.

Um formulário simples com um campo de texto e um botão de envio:

```
<form>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name">
    <input type="submit" value="Submit">
</form>
```

Um formulário com radio button:

```
<form>
    <label for="gender">Gender:</label>
    <input type="radio" id="gender" name="gender" value="male"> Male
    <input type="radio" id="gender" name="gender" value="female">
Female
    <input type="submit" value="Submit">
</form>
```

Um formulário com um campo de seleção:

```
<form>
    <label for="car">Choose a car:</label>
    <select id="car" name="car">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="fiat">Fiat</option>
        <option value="audi">Audi</option>
    </select>
    <input type="submit" value="Submit">
</form>
```

Um formulário com uma área de texto:

```
<form>
    <label for="message">Message:</label>
    <textarea id="message" name="message"></textarea>
    <input type="submit" value="Submit">
</form>
```

A tag <form> é usada para criar um formulário, e as tags <input>, <label>, <select>, <option> e <textarea> são usadas para criar diferentes tipos de campos de formulário, como campos de texto, botões de rádio, campos de seleção e áreas de texto. As tags <label> são usadas para associar um rótulo a um campo de formulário específico, para que os usuários saibam o que é esperado nesse campo. Os atributos id e name são usados para identificar os campos de formulário, e os atributos value e for são usados para vincular os rótulos aos campos correspondentes.

Div e Span: A tag é usada para criar uma divisão ou seção na página, enquanto que a tag é usada para marcar um pequeno trecho de texto dentro de um elemento

pai, como um parágrafo. As tags e são geralmente utilizadas como contêineres para aplicar estilos CSS e para agrupar elementos em conjunto para fins de estilização e layout. É importante notar que as tags e não possuem nenhum significado semântico específico, ou seja, não transmitem nenhuma informação sobre o conteúdo que elas estão contidas, por isso, é importante utilizá-las em conjunto.

Uma div com conteúdo e atributos HTML:

```
<div align="center" bgcolor="lightgray" style="padding: 10px;">
    Conteúdo da div
</div>
```

Um span com conteúdo e atributos HTML:

```
<p>Este é um parágrafo com <span style="font-weight:bold; background-color: yellow;">texto destacado</span> dentro dele.</p>
```

Nesses exemplos, os atributos align, bgcolor e style foram usados para aplicar estilos às tags e . O atributo align é usado para alinhar o conteúdo dentro da div, o atributo bgcolor é usado para definir a cor de fundo, e o atributo style é usado para aplicar estilos inline ao elemento. É importante notar que esses atributos são considerados como uma prática antiga e não são mais recomendados, pois eles misturam conteúdo e estilos e dificultam a manutenção do código. A recomendação é usar CSS para estilizar a página, pois separa o conteúdo da estilização.

Mídia: As tags audio e video são usadas para incluir mídia como áudio e vídeo na página sem precisar de plugins.

Um arquivo de áudio com controles de reprodução:

```
<audio controls>
    <source src="audio.mp3" type="audio/mpeg">
    Your browser does not support the audio element.
</audio>
```

Um arquivo de vídeo com controles de reprodução e com legenda:

```
<video width="320" height="240" controls>
    <source src="video.mp4" type="video/mp4">
    <track src="legendas.vtt" kind="subtitles" srclang="pt-BR" label="Português">
    Your browser does not support the video tag.
</video>
```

A tag <audio> é usada para adicionar um arquivo de áudio à página, enquanto a tag <video> é usada para adicionar um arquivo de vídeo. O atributo controls é usado para adicionar os controles de reprodução (play/pause, volume, etc.) ao arquivo de áudio ou vídeo. As tags <source> são usadas para especificar o arquivo de áudio ou vídeo e o tipo de arquivo, enquanto a tag <track> é usada para adicionar legenda ao vídeo. Caso o navegador não suporte as tags, o conteúdo dentro das tags <audio> ou <video> será exibido.

Vamos praticar:

1. Crie uma página HTML com um título (h1) no topo da página e seis subtítulos (h6) abaixo dele, cada um com um conteúdo diferente.
2. Adicione três parágrafos (p) abaixo dos subtítulos, cada um com um conteúdo diferente.
3. Inclua uma imagem (img) no meio dos parágrafos e adicione uma legenda com a tag "figcaption"
4. Crie uma tabela (table) com três colunas e cinco linhas, preencha as células com conteúdo fictício.
5. Adicione um formulário (form) com três campos de entrada (input) para nome, email e senha e um botão de envio (input type="submit")
6. Adicione uma divisão (div) em volta do formulário e adicione uma classe para estilizar com CSS
7. Adicione um span em volta de uma palavra dentro de um dos parágrafos para dar destaque a esta palavra
8. Adicione um áudio (audio) e um vídeo (video) na página, cada um com um controle de reprodução (play/pause)

Capítulo 5: Formatando o conteúdo com CSS

Adicionando estilo e formato ao seu site com CSS

CSS (Cascading Style Sheets) é um conjunto de regras que permitem controlar a aparência dos elementos em uma página HTML. Usando CSS, você pode controlar coisas como cor de fundo, tamanho de fonte, espaçamento, posicionamento e muito mais.

Adicionar estilo CSS em um site pode ser feito de duas maneiras: através de um arquivo externo ou incluindo as regras dentro da tag head.

Adicionando uma folha de estilo externa:

Primeiro, crie um arquivo com a extensão “.css”. Por exemplo, "estilo.css". Em seguida, adicione a tag <link> dentro da tag <head> do seu arquivo HTML, com o atributo "href" apontando para o arquivo CSS criado.

Exemplo do HTML no arquivo index.html:

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="estilo.css">
        <title>Meu Site</title>
    </head>
    <body>
        <!-- conteúdo do site -->
    </body>
</html>
```

Exemplo do CSS no arquivo estilo.css:

```
/* Define as cores e tipos de fonte padrão para o site */
body {
    font-family: Arial, sans-serif;
    color: #333;
    background-color: #f5f5f5;
}

/* Define a cor e a fonte dos títulos */
h1, h2, h3 {
    color: #333;
    font-weight: bold;
}
```

Incluindo regras CSS dentro da tag head:

Da mesma forma que é possível escrever o CSS em um arquivo separado, também é possível escrever dentro do arquivo HTML. Para isso, dentro da tag <head> do seu arquivo HTML, adicione a tag <style> e escreva o código css dentro dessa tag. Exemplo:

```
<head>
    <style>
        /* suas regras CSS aqui */
        h1 {
          color: blue;
        }
    </style>
</head>
```

Agora que você já sabe onde colocar o CSS, vamos explicar como utilizá-lo.

Seletores e regras: Os seletores CSS são usados para selecionar elementos na página e aplicar estilos a eles. Existem vários tipos de seletores, incluindo seletores de tag, classe e ID.

1. **Seletores de tag:** São usados para selecionar elementos específicos baseados na tag HTML. Por exemplo, para selecionar todas as tags <h1> e aplicar uma determinada cor, você poderia usar o seguinte código CSS:

```
h1 {
    color: blue;
}
```

2. **Seletores de classe:** São usados para selecionar elementos que possuem uma classe específica. As classes são definidas usando o atributo "class" no HTML. Por exemplo, para selecionar todos os elementos <h1> com a classe "destaque" e aplicar uma determinada cor de fundo, você poderia usar o seguinte código CSS:

```
.destaque {
    background-color: yellow;
}
```

Perceba que há um ponto antes do nome da palavra “destaque”. Assim selecionamos todos os elementos com a classe “destaque”.

3. **Seletores de ID:** São usados para selecionar um único elemento que possui um ID específico. Os IDs são definidos usando o atributo "id" no HTML. Por exemplo, para

selecionar um elemento com o ID "cabecalho" e aplicar uma determinada cor de fundo, você poderia usar o “#” seguido do nome do id. Exemplo:

```
#cabecalho {
    background-color: green;
}
```

É importante notar que os seletores de ID têm uma prioridade maior do que os seletores de classe, e os seletores de classe têm uma prioridade maior do que os seletores de tag. Isso significa que se você tiver regras CSS conflitantes, a regra com o seletor de ID será aplicada, seguida pela regra com o seletor de classe e, finalmente, pela regra com o seletor de tag.

Em resumo, os seletores de tag, classe e ID são os principais seletores CSS e são usados para selecionar elementos na página e aplicar estilos a eles. Cada um deles tem sua própria prioridade e pode ser usado de acordo com as necessidades do projeto.

Comandos CSS

Alguns comandos básicos do CSS incluem font-size, color, margin, padding, width, e height. Eles são usados para controlar a aparência e formatação dos elementos na página.

1. **font-size:** Controla o tamanho da fonte de um elemento. Por exemplo, para aumentar o tamanho da fonte dos títulos <h1> para 36px, você poderia usar o seguinte código:

```
h1 {
    font-size: 36px;
}
```

2. **color:** Controla a cor da fonte de um elemento. Por exemplo, para mudar a cor dos links para vermelho, você poderia usar o seguinte código:

```
a {
    color: red;
}
```

3. **margin:** Controla o espaço externo de um elemento. Por exemplo, para adicionar 20px de margem à direita de um elemento, você poderia usar o seguinte código:

```
.destaque {
    margin-right: 20px;
```

```
}
```

4. **padding:** Controla o espaço interno de um elemento. Por exemplo, para adicionar 10px de espaço interno à todas as bordas de um elemento, você poderia usar o seguinte código:

```
header {
    padding: 10px;
}
```

5. **width:** Controla a largura de um elemento. Por exemplo, para definir a largura de uma imagem para 300px, você poderia usar o seguinte código:

```
img {
    width: 300px;
}
```

6. **height:** Controla a altura de um elemento. Por exemplo, para definir a altura de um botão para 50px, você poderia usar o seguinte código:

```
button {
    height: 50px;
}
```

Usando CSS para controlar layout, cores e fontes

Layout: CSS oferece uma variedade de propriedades para controlar o layout de elementos na página. Essas propriedades incluem display, position, e float que permitem controlar como os elementos são exibidos e organizados. Outras propriedades como width, height, margin e padding, permitem controlar o tamanho e espaçamento dos elementos.

Segue abaixo um exemplo simples para cada propriedade mencionada.

Propriedade Display:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade display</title>
```

```
        <style>
            .box {
                display: inline-block;
                width: 100px;
                height: 100px;
                background-color: red;
                color: white;
                text-align: center;
                line-height: 100px;
            }
        </style>
    </head>
    <body>
        <div class="box">Box 1</div>
        <div class="box">Box 2</div>
        <div class="box">Box 3</div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **display** para definir a caixa como um **inline-block**. Isso faz com que as caixas sejam exibidas uma ao lado da outra, em vez de em linhas separadas. As caixas têm um tamanho de 100x100 pixels, um fundo vermelho, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Propriedade Position:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade position</title>
        <style>
            .box {
                position: absolute;
                top: 50%;
                left: 50%;
                transform: translate(-50%, -50%);
                width: 200px;
                height: 200px;
                background-color: blue;
                color: white;
                text-align: center;
                line-height: 200px;
```

```
            }
        </style>
    </head>
    <body>
        <div class="box">Box 1</div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **position** para posicionar a caixa absolutamente no centro da página. As propriedades **top** e **left** especificam a posição da caixa em relação à parte superior e esquerda do elemento pai, que é o corpo do documento neste caso. O **transform** é usado para centralizar verticalmente e horizontalmente a caixa, movendo-a para a esquerda e para cima em metade da largura e altura da caixa. A caixa tem um tamanho de 200x200 pixels, um fundo azul, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Propriedade Float:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade float</title>
        <style>
            .box1 {
                float: left;
                width: 50%;
                height: 200px;
                background-color: green;
                color: white;
                text-align: center;
                line-height: 200px;
            }

            .box2 {
                float: right;
                width: 50%;
                height: 200px;
                background-color: orange;
                color: white;
                text-align: center;
                line-height: 200px;
            }
```

```
            .clear {
                clear: both;
            }
        </style>
    </head>
    <body>
        <div class="box1">Box 1</div>
        <div class="box2">Box 2</div>
        <div class="clear"></div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **float** para definir as caixas como flutuantes, permitindo que elas fiquem lado a lado. A primeira caixa é definida como flutuante à esquerda e a segunda caixa é definida como flutuante à direita. Cada caixa tem um tamanho de 50% da largura da janela do navegador, uma altura de 200 pixels, fundo verde ou laranja, texto branco centralizado e altura de linha correspondente ao seu tamanho.

A classe **clear** é adicionada para limpar o float e garantir que o conteúdo posterior à caixa flutuante fique abaixo delas. Essa técnica é chamada de **"clearfix"**.

Propriedade Width:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade width</title>
        <style>
            .box {
                width: 50%;
                height: 200px;
                background-color: purple;
                color: white;
                text-align: center;
                line-height: 200px;
            }
        </style>
    </head>
    <body>
        <div class="box">Box com largura de 50%</div>
    </body>
```

```
</html>
```

Nesse exemplo, usamos a propriedade **width** para definir a largura da caixa como 50% da largura da janela do navegador. A caixa tem uma altura de 200 pixels, um fundo roxo, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Propriedade height:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade height</title>
        <style>
            .box {
                width: 200px;
                height: 400px;
                background-color: pink;
                color: white;
                text-align: center;
                line-height: 400px;
            }
        </style>
    </head>
    <body>
        <div class="box">Box com altura de 400px</div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **height** para definir a altura da caixa como 400 pixels. A caixa tem uma largura de 200 pixels, um fundo rosa, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Propriedade Margin:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade margin</title>
        <style>
            .box {
```

```
                width: 200px;
                height: 200px;
                background-color: yellow;
                color: white;
                text-align: center;
                line-height: 200px;
                margin: 50px;
            }
        </style>
    </head>
    <body>
        <div class="box">Box com margem de 50px</div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **margin** para definir a margem da caixa como 50 pixels. A caixa tem um tamanho de 200x200 pixels, um fundo amarelo, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Propriedade Padding:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Exemplo de uso da propriedade padding</title>
        <style>
            .box {
                width: 200px;
                height: 200px;
                background-color: gray;
                color: white;
                text-align: center;
                line-height: 200px;
                padding: 50px;
            }
        </style>
    </head>
    <body>
        <div class="box">Box com preenchimento de 50px</div>
    </body>
</html>
```

Nesse exemplo, usamos a propriedade **padding** para definir o preenchimento interno da caixa como 50 pixels. A caixa tem um tamanho de 200x200 pixels, um fundo cinza, texto branco centralizado e uma altura de linha correspondente ao seu tamanho.

Cores: as propriedades **color** e **background-color** permitem controlar as cores dos elementos e de seus backgrounds.

Background-color é utilizada para definir a cor de fundo de um elemento.

Color é utilizada para definir a cor do texto dentro de um elemento tornando-o mais legível e destacado dentro da página.

Segue um exemplo de uso dessas duas propriedades:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Exemplo de color e background-color</title>
    <style>
      .caixa, .caixa2 {
        background-color: #41c5ab;
        width: 200px;
        height: 200px;
        border: 1px solid black;
      }
      .caixa p{
        color: black;
      }
      .caixa2 p {
        color: red;
      }
    </style>
  </head>
  <body>
    <div class="caixa">
        <p>Este é um exemplo de parágrafo com cor preta.</p>
    </div>
    <div class="caixa2">
        <p>Este é um exemplo de parágrafo com cor vermelha.</p>
    </div>
  </body>
</html>
```

No CSS, as cores podem ser especificadas usando dois formatos diferentes: nome da cor e código da cor.

O nome da cor é um valor pré-definido que representa uma determinada cor. Por exemplo, a palavra "red" representa a cor vermelha e a palavra "green" representa a cor verde. Existem várias palavras-chave de cores disponíveis no CSS, como "blue", "yellow", "black", "white", entre outras. O uso do nome da cor é muito útil para trabalhar com cores comuns, que são fáceis de memorizar e aplicar.

Já o código da cor é uma representação numérica da cor em hexadecimal ou RGB. O código hexadecimal é um valor de 6 dígitos que representa a quantidade de vermelho, verde e azul (RGB) na cor. O código RGB é um valor numérico que representa a quantidade de vermelho, verde e azul em uma escala de 0 a 255. O uso do código da cor é útil quando você precisa usar uma cor específica que não está disponível como palavra-chave, ou quando você deseja aplicar uma cor personalizada.

Segue abaixo um exemplo de código que usa ambos os formatos para especificar a cor de um elemento:

```
.seletor {
  background-color: red; /* Usando o nome da cor */
  color: #00FF00; /* Usando o código hexadecimal */
}
```

Neste exemplo, a propriedade **background-color** usa o nome da cor "red" para definir a cor de fundo do elemento, enquanto a propriedade **color** usa o código hexadecimal "#00FF00" para definir a cor do texto. Note que ambos os formatos são válidos e podem ser usados para especificar as cores no CSS.

Fontes: CSS também oferece uma variedade de propriedades para controlar as fontes usadas em seus elementos. Propriedades como font-size, font-family, font-style e font-weight permitem controlar o tamanho, estilo e espessura da fonte. Seguem alguns exemplos:

font-size: controla o tamanho da fonte. É possível especificar um valor numérico, em pixels, em pontos ou em porcentagem. Por exemplo:

```
.seletor {
  font-size: 16px; /* define o tamanho da fonte como 16 pixels */
}
```

font-family: define a família de fontes a ser usada. É possível especificar várias fontes em ordem de preferência. Por exemplo:

```
.seletor {
  font-family: Arial, sans-serif; /* define Arial como fonte
preferencial e sans-serif como fonte alternativa */
}
```

font-style: controla o estilo da fonte. Pode ser definido como normal, italic ou oblique. Por exemplo:

```
.seletor {
  font-style: italic; /* define o estilo da fonte como itálico */
}
```

font-weight: controla a espessura da fonte. Pode ser definido como normal, bold ou um valor numérico específico. Por exemplo:

```
.seletor {
  font-weight: bold; /* define a espessura da fonte como negrito */
}
```

Lembrando que essas propriedades podem ser combinadas em uma única regra CSS, para definir o estilo completo de um elemento. Por exemplo:

```
.seletor {
  font-size: 16px;
  font-family: Arial, sans-serif;
  font-style: italic;
  font-weight: bold;
}
```

Neste exemplo, o elemento com a classe ".seletor" terá um tamanho de fonte de 16 pixels, a fonte Arial será usada preferencialmente, com sans-serif como fonte alternativa, o estilo da fonte será itálico e a espessura da fonte será negrito.

Há também a possibilidade de combinar a propriedade de fontes em uma única linha utilizando o **font**. Segue um exemplo:

```
.seletor {
  font: italic bold 16px/1.5 Arial, sans-serif;
}
```

Neste exemplo, a propriedade **font** combina todas as propriedades de fonte em uma única regra.

A ordem dos valores é importante e deve seguir o seguinte padrão:

1. **font-style:** normal, italic ou oblique;
2. **font-weight:** normal, bold, bolder, lighter, ou um valor numérico específico;
3. **font-size:** valor numérico, em pixels, em pontos, em em ou em rem;
4. **line-height:** valor numérico que define a altura da linha;
5. **font-family:** família de fontes, separadas por vírgulas.

No exemplo, a propriedade **font** define que o texto terá estilo itálico (italic) e espessura negrito (bold), tamanho de fonte de 16px e uma altura de linha de 1.5. Em seguida, define que a fonte a ser usada preferencialmente é Arial, e caso ela não esteja disponível, a fonte sans-serif será usada como alternativa.

Controlando posicionamentos e criando animações com CSS

Grid e Flexbox: O Grid e o Flexbox são técnicas modernas de layout em CSS que permitem organizar os elementos HTML de uma forma mais flexível e adaptável. Essas técnicas tornam possível criar layouts complexos e sofisticados que antes eram difíceis de realizar com CSS puro.

O Flexbox é uma técnica de layout unidimensional que trabalha com uma única direção, geralmente horizontal ou vertical. É ótimo para criar layouts responsivos que se adaptam a diferentes tamanhos de tela. Com o Flexbox, é possível controlar o posicionamento, o tamanho e o alinhamento dos elementos de forma muito mais fácil.

O CSS Grid, por outro lado, é uma técnica de layout bidimensional que permite controlar o layout de uma página da web em duas direções: linha e coluna. O Grid é ótimo para layouts complexos e sofisticados que requerem controle granular sobre o posicionamento dos elementos.

Vamos a um exemplo de uso do Flexbox para posicionar o menu e o conteúdo em uma linha horizontal. O menu é alinhado à esquerda com margens entre os itens, enquanto o conteúdo é alinhado à direita com o botão abaixo do conteúdo:

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            .container {
              display: flex;
```

```
            flex-direction: row;
            justify-content: space-between;
            align-items: center;
          }

          .menu {
            display: flex;
            flex-direction: row;
          }

          .menu li {
            margin-right: 20px;
          }

          .content {
            display: flex;
            flex-direction: column;
            align-items: flex-end;
          }

          .content button {
            margin-top: 20px;
          }
        </style>
    </head>
    <body>
        <div class="container">
        <div class="menu">
            <ul>
            <li><a href="#">Item 1</a></li>
            <li><a href="#">Item 2</a></li>
            <li><a href="#">Item 3</a></li>
            </ul>
        </div>
        <div class="content">
            <h1>Título</h1>
            <p>Conteúdo</p>
            <button>Botão</button>
        </div>
        </div>
    </body>
</html>
```

Vamos agora a um exemplo de uso do Grid para criar um layout de página complexo com um cabeçalho, barra lateral, conteúdo principal e rodapé. Usamos o “grid-template-columns” para especificar a largura das colunas, “grid-template-rows” para especificar a altura das linhas e “grid-template-areas” para especificar a área que cada elemento deve ocupar. Também adicionamos uma margem entre os elementos com “grid-gap”:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
        <style>
            .container {
              display: grid;
              grid-template-columns: 1fr 2fr;
              grid-template-rows: auto 1fr auto;
              grid-template-areas:
                "header header"
                "sidebar content"
                "footer footer";
              grid-gap: 20px;
            }

            .header {
              grid-area: header;
            }

            .sidebar {
              grid-area: sidebar;
            }

            .content {
              grid-area: content;
            }

            .footer {
              grid-area: footer;
            }
        </style>
    </head>
    <body>
        <div class="container">
            <div class="header">Cabeçalho</div>
```

```
            <div class="sidebar">Barra lateral</div>
            <div class="content">Conteúdo principal</div>
            <div class="footer">Rodapé</div>
        </div>
    </body>
</html>
```

Transições e animações: Transições e animações são recursos poderosos que permitem adicionar interatividade e dinamismo ao site, tornando-o mais atraente e envolvente para os usuários. Com essas propriedades, é possível controlar como os elementos mudam de estado, seja por meio de transições suaves entre valores de propriedade diferentes, ou por meio de animações mais elaboradas que criam movimento e efeitos visuais.

A propriedade “transition” permite definir uma transição suave entre dois estados de um elemento quando uma propriedade é alterada. Para usar essa propriedade, é necessário especificar as propriedades que serão afetadas pela transição, a duração da transição, a curva de aceleração e o atraso. Por exemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
        <style>
          button {
              background-color: rgb(83, 83, 255);
              transition: background-color 0.3s ease-in-out;
          }

          button:hover {
              background-color: rgb(250, 96, 96);
          }
        </style>
    </head>
    <body>
        <button>Botão</button>
    </body>
</html>
```

Nesse exemplo, a propriedade “background-color” será afetada pela transição, que terá uma duração de 0.3 segundos e uma curva de aceleração ease-in-out. Quando o mouse estiver sobre o botão, a cor de fundo mudará suavemente de azul para vermelho.

A propriedade “animation”, por outro lado, permite criar animações mais elaboradas e complexas, que podem incluir movimento, transformações, rotações, efeitos visuais, entre outros. Para usar essa propriedade, é necessário definir uma série de quadros-chave que descrevem a animação em diferentes momentos, bem como a duração, a curva de aceleração, o atraso e o número de repetições. Por exemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
        <style>
            @keyframes pulse {
                0% { transform: scale(1); }
                50% { transform: scale(1.1); }
                100% { transform: scale(1); }
            }

            button {
                animation: pulse 1s infinite;
            }
        </style>
    </head>
    <body>
        <button>Botão</button>
    </body>
</html>
```

Nesse exemplo, definimos uma animação chamada “pulse” que faz o botão crescer e encolher suavemente em um ciclo contínuo. A animação é definida usando a regra “@keyframes”, que especifica a transformação do botão em diferentes momentos. Depois, adicionamos a animação ao botão usando a propriedade “animation”, que define a duração, o número de repetições e a curva de aceleração.

Em resumo, as propriedades “transition” e “animation” são recursos poderosos para adicionar interatividade e dinamismo ao site. Com elas, é possível controlar como os elementos mudam de estado, seja por meio de transições suaves ou animações elaboradas. Com um pouco de criatividade, é possível criar efeitos impressionantes que vão surpreender e encantar os usuários.

Media Queries: Media queries são uma técnica de CSS que permite definir estilos diferentes para diferentes dispositivos, baseando-se nas características da tela, como a largura, a altura e a orientação.

A sintaxe básica de uma media query é a seguinte:

```
@media (condição) {
  /* estilos a serem aplicados se a condição for verdadeira */
}
```

A condição dentro dos parênteses pode ser uma variedade de características da tela, incluindo a largura da tela (**width**), a altura da tela (**height**), a orientação (**orientation**) e a densidade de pixels (**resolution**), entre outras.

A seguir, veremos um exemplo simples de código que define uma media query para dispositivos com largura de tela de até **600px** e aplica uma cor de fundo diferente:

```
body {
  background-color: blue;
}

@media (max-width: 600px) {
  body {
    background-color: red;
  }
}
```

Nesse exemplo, a cor de fundo padrão do corpo do documento é definida como azul. A media query **@media (max-width: 600px)** especifica que o estilo de fundo vermelho deve ser aplicado ao corpo do documento quando a largura da tela for menor ou igual a **600px**.

Além disso, a media query também pode ser usada para adaptar outros estilos e propriedades de layout, como mudanças na posição, tamanho e cor do texto, fontes, margens e preenchimentos, etc.

Por fim, é importante lembrar que o CSS é uma ferramenta poderosa para estilizar e dar vida aos elementos de uma página web. Combinando as propriedades corretas e usando a criatividade, é possível criar layouts e efeitos que tornam a experiência do usuário mais agradável e interativa.

Capítulo 6: Criando tabelas e formulários

Criando tabelas e formulários com HTML

Tabelas: As tags table, tr, td e th são usadas para criar tabelas e estruturar dados. A tag table é usada para criar uma tabela, tr para criar linhas, td para criar células de dados e th para criar células de cabeçalho. É importante que você mostre como usar essas tags de forma hierárquica correta, bem como mostrar como aplicar estilos para formatar e estilizar tabelas usando CSS.

Exemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
    </head>
    <body>
      <table>
        <tr>
          <th>Cabeçalho 1</th>
          <th>Cabeçalho 2</th>
          <th>Cabeçalho 3</th>
        </tr>
        <tr>
          <td>Informação 1</td>
          <td>Informação 2</td>
          <td>Informação 3</td>
        </tr>
        <tr>
          <td>Informação 4</td>
          <td>Informação 5</td>
          <td>Informação 6</td>
        </tr>
      </table>
    </body>
</html>
```

Formulários: As tags form, input, label, select, option e textarea são usadas para criar formulários e campos de entrada. A tag form é usada para criar um formulário e a tag input é

usada para criar diferentes tipos de campos de entrada, como texto, caixa de seleção, botão de rádio e botão de envio. A tag label é usada para criar rótulos para os campos de entrada, a tag select é usada para criar menus suspenso e a tag option é usada para criar opções para o menu suspenso. A tag textarea é usada para criar uma área de texto com múltiplas linhas.

Exemplo:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
    </head>
    <body>
        <form action="#" method="post">
            <label for="name">Nome:</label>
            <input type="text" id="name" name="name"
placeholder="Nome">

            <label for="email">Email:</label>
            <input type="email" id="email" name="email"
placeholder="Email">

            <label for="password">Senha:</label>
            <input type="password" id="password" name="password"
placeholder="Senha">

            <label for="gender">Gênero:</label>
            <select id="gender" name="gender">
              <option value="masculino">Masculino</option>
              <option value="feminino">Feminino</option>
              <option value="outro">Outro</option>
            </select>

            <input type="submit" value="Enviar">
        </form>
    </body>
</html>
```

Capítulo 7: Responsividade e design adaptável

Fazendo com que o seu site se adapte a diferentes tamanhos de tela

A responsividade é a capacidade de adaptar a exibição de um site para diferentes tamanhos de tela, garantindo que os usuários tenham uma boa experiência independentemente do dispositivo que estão usando.

Existem várias técnicas para fazer com que um site seja responsivo, algumas das quais incluem:

Usando porcentagens em vez de valores absolutos para definir tamanhos de elementos e espaçamentos. Isso garante que os elementos sejam proporcionalmente dimensionados em relação à tela do usuário.

```
<style>
    img {
        width: 100%;
        /* Usando percentual no lugar de pixels (px) */
    }
</style>
```

Utilizando Media Queries para aplicar estilos específicos baseado nas dimensões da tela do usuário:

```
<style>
    @media screen and (max-width: 600px) {
        /* Aplica os estilos para dispositivos com telas com largura menor que 600px */
    }
</style>
```

Utilizando Flexbox ou Grid para criar layouts flexíveis que se adaptam a diferentes tamanhos de tela:

```
<style>
    .container {
        display: flex;
```

```
        /* Ou display: grid */
    }
</style>
```

Além disso, é importante lembrar que a responsividade é apenas uma parte da adaptação para diferentes tamanhos de tela, e é preciso levar em conta outras considerações, como acessibilidade e usabilidade. Portanto, é importante fazer testes em diferentes dispositivos e navegadores, e se certificar de que o seu site é acessível e fácil de usar para todos os usuários, independentemente do dispositivo que estão usando.

Usando media queries e viewports para criar design responsivo

Media Queries: É uma técnica de CSS que permite aplicar estilos diferentes baseado em características do dispositivo do usuário, como largura e altura da tela.

Viewport: É uma meta tag HTML que informa ao navegador como renderizar o site em diferentes tamanhos de tela. Ele pode ser usado para definir a largura inicial da visualização e controlar a escala do site. Por exemplo, você pode usar uma viewport para assegurar que seu site é renderizado em 100% da largura da tela em dispositivos móveis:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

Aqui estão alguns exemplos adicionais de como usar media queries e viewports para criar design responsivo:

Exemplo de Media Queries:

```
<style>
    /* Estilo para telas com largura menor que 480px */
    @media (max-width: 480px) {
        h1 {
            font-size: 24px;
        }
        .container {
            flex-wrap: wrap;
        }
    }
```

```
    /* Estilo para telas com largura maior que 1024px */
    @media (min-width: 1024px) {
        .container {
            display: grid;
            grid-template-columns: 1fr 1fr;
        }
    }
</style>
```

Perceba que no exemplo acima, aplicamos estilos que já vimos antes. Mas, dessa vez, colocamos dentro dos limites dos medias queries. Ou seja, para cada tamanho de tela definimos características diferentes aos elementos do nosso site.

Exemplo de Viewport:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

No exemplo acima, a tag <meta> é utilizada para definir o viewport com as seguintes propriedades:

- **width=device-width:** Faz com que o viewport tenha a largura do dispositivo (como um smartphone ou tablet);
- **initial-scale=1.0:** Define o nível de zoom inicial para 1.0 (sem zoom).

Além dessas propriedades, o viewport pode ser configurado com outros atributos, como:

- **minimum-scale:** Define o menor nível de zoom permitido;
- **maximum-scale:** Define o maior nível de zoom permitido;
- **user-scalable:** Permite ou impede o usuário de dar zoom na página.

Segue um exemplo de como utilizar o viewport com todos os atributos:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Meu Site</title>
        <meta name="viewport" content="width=device-width, initial-
scale=1.0, minimum-scale=1.0, maximum-scale=2.0, user-scalable=yes">
    </head>
    <body>
        <h1>Exemplo de utilização do Viewport com atributos</h1>
        <p>Este é um exemplo de como utilizar o viewport no HTML5 com
todos os atributos.</p>
```

```
    </body>
</html>
```

Neste exemplo, o viewport é configurado com as seguintes propriedades:

- **width=device-width:** Faz com que o viewport tenha a largura do dispositivo;
- **initial-scale=1.0:** Define o nível de zoom inicial para 1.0;
- **minimum-scale=1.0:** Define o menor nível de zoom permitido como 1.0;
- **maximum-scale=2.0:** Define o maior nível de zoom permitido como 2.0;
- **user-scalable=yes:** Permite ao usuário dar zoom na página.

Conclusão

Em resumo, este livro fornece uma introdução abrangente às ferramentas necessárias para criar um site moderno e responsivo. Você foi apresentado ao HTML, suas versões e estrutura básica, bem como aos elementos mais comuns usados em uma página HTML, como headings, paragraphs, links e imagens. Além disso, este livro ensina como adicionar estilo e formato ao seu site com CSS e controlar posicionamentos e animações com esta linguagem. Também são explorados os recursos para criação de tabelas e formulários com HTML, bem como a criação de designs responsivos, usando media queries e viewports.

Com este conhecimento em mãos, você está pronto para começar a criar seus próprios sites incríveis e adaptáveis a diferentes dispositivos e tamanhos de tela.